



Kivy

Kivy: Building GUI and Mobile apps with Python

Thomas Hansen - fresk - [@hansent](#)

Mathieu Virbel - Melting Rocks - [@mathieuvirbel](#)

PyCon 2013

Show & Tell



2013

- Some things have changed...
- Powerful GPU's everywhere
- Input / Sensors much more diverse

Issues

- GPU's have a different computing paradigm
- More / Higher Bandwidth Inputs increase complexity
- Composition / Authoring of Interfaces is hard.

Kivy

- Use GPU power, but keep simple things simple
- Events are still good, but async / callbacks suck
- Zen of Python

Kivy - Graphics

- Everything is GPU accelerated (GLES)
- 3D and Shaders! (if you know what you're doing...power!)
- Optimized rendering using Cython/C
- Graphics Instructions for sequential programming & "JIT graphics compiler"
- Also eases porting, if there is a GPU, GLES isn't far away...

Kivy - Input

- Fundamentally based on concept of multi-input
- Conceptual tools / approaches to event handling / interpretation
- Provider based library design / architecture

Kivy - Core Providers

Providers / Interfaces for:

- Image, Font Rendering, Window, Video, Audio, Camera, Clipboard, Spelling, etc.
- Load/process/store/render accessible to Python and/or GL
- Allows swapping out Dependencies
- Makes porting / supporting new platforms easier

Kivy - The Zen of Kivy

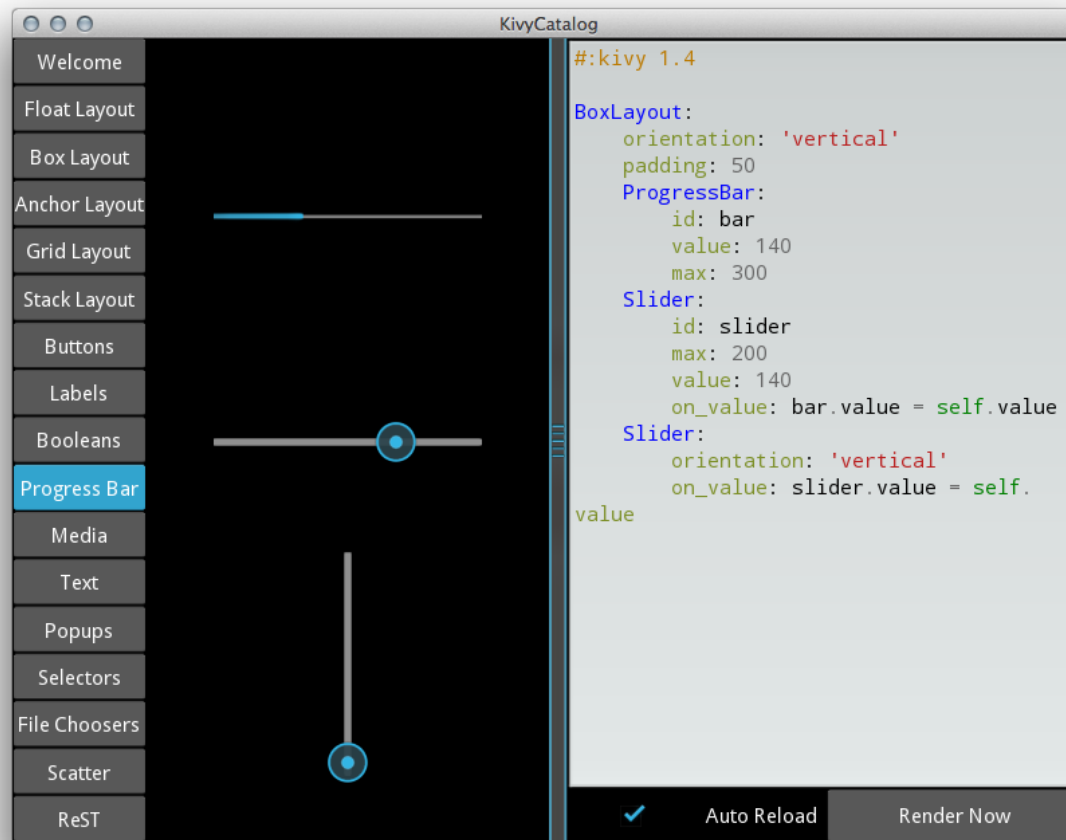
- Zen of Python
- KV language. UI composition/styling DSL
- Indentation based
- Declarative, concise, auto-binding
- Works well with graphics instructions approach

Kivy - Batteries Included!

Extensive Widget collection

- Labels, Buttons, Sliders, Images
- Layouts, ListViews, Screens, Transitions
- Interactions like scatter, zoom, pan, rotate
- Support different input modalities out of box (e.g. Virtual Keyboards).

UIX Catalog



The screenshot shows the KivyCatalog application interface. On the left is a sidebar with a list of widget categories: Welcome, Float Layout, Box Layout, Anchor Layout, Grid Layout, Stack Layout, Buttons, Labels, Booleans, Progress Bar (highlighted), Media, Text, Popups, Selectors, File Choosers, Scatter, and ReST. The main area displays three widgets: a horizontal progress bar at the top, a horizontal slider in the middle, and a vertical slider at the bottom. On the right is a code editor showing the Kivy code for the selected widget. The code is as follows:

```
#:kivy 1.4
BoxLayout:
    orientation: 'vertical'
    padding: 50
    ProgressBar:
        id: bar
        value: 140
        max: 300
    Slider:
        id: slider
        max: 200
        value: 140
        on_value: bar.value = self.value
    Slider:
        orientation: 'vertical'
        on_value: slider.value = self.
value
```

At the bottom right of the code editor, there are two buttons: "Auto Reload" (with a checkmark icon) and "Render Now".

Kivy - Develop once...

- Linux, OSX, Windows
- Android, iOS, Meego
- Raspberry Pi ?

Develop & test on workstation, deploy to mobile...(not emulated)

Kivy Organization

- 7 core developers
- 13 public projects
- 2 mailing lists (kivy-users, kivy-dev), ~ 760 users
- IRC channel #kivy

All the projects are available at github.com/kivy

History

PyMT

- Created in 2007
- Mainly focused for NUI on custom hardware
- Place for experimentation
- Direct GL API, no way to run it on mobile.

Kivy

- born in 2010
- API renamed and stabilized
- New graphics pipeline, built on OpenGL ES 2
- Android and iOS added during 2011

Building for mobile: toolchain

We provide toolchain to compile app for Android and iOS

Python-for-android

- Cross compilation toolchain for ARM/Android
- Python, SDL, pygame, and many others libraries are shipped
- One tool for packaging into APK
- ! Our project is at <http://github.com/kivy/python-for-android/>

Kivy iOS

- Cross compilation toolchain for ARM/iOS
- Not modular, few libraries are available
- **Goal:** redo it with the same architecture as Python for Android

Building for mobile: buildozer

Buildozer

- One definition file called "buildozer.spec"
- Create application packages for Android and iOS

```
$ buildozer init  
$ vim buildozer.spec  
$ buildozer android debug deploy run
```

SHELL

- It hides the complexity of packaging

Building for mobile: buildozer

SPEC

```
[app]
title = Logotouch
package.name = logotouch
package.domain = org.erasme
source.dir = .
source.include_exts = ini,py,png,jpg,ttf,kv,mo,atlas
version.regex = __version__ = '(.*)'
version.filename = %(source.dir)s/main.py
requirements = pika,kivy
presplash.filename = %(source.dir)s/data/presplash.png
icon.filename = %(source.dir)s/data/icon.png

# android specific
android.permissions = INTERNET

# ios specific
ios.codesign.debug = "iPhone Developer: Mathieu Virbel (PSW92G0TX8)"
```

Accessing mobile API

Mobile have Accelerometer, Camera, Compass, Contacts, Geolocation (GPS, Wifi), Notification etc.

We don't provide abstraction for it (yet).

What if we could have access to anything in Java from Python ?

Accessing mobile API

PyJNIus - Access Java classes from Python

- Works on Desktop and Android
- It uses JNI / Java reflection

PyOBJus - Access Objective-C from Python

- It uses Objective C reflection
- (Still in progress, doesn't work)

Accessing mobile API: PyJNIus

Using Text-To-Speech: Make your phone say "Hello World"

```
from jnius import autoclass

Locale = autoclass('java.util.Locale')
PythonActivity = autoclass('org.renpy.android.PythonActivity')
TextToSpeech = autoclass('android.speech.tts.TextToSpeech')

tts = TextToSpeech(PythonActivity.mActivity, None)
tts.setLanguage(Locale.US)
tts.speak('Hello World.', TextToSpeech.QUEUE_FLUSH, None)
```

PYTHON

Accessing mobile API: PyJNIus

One step further: implement Java class in Python! (wip).

PYTHON

```
class GpsListener(PythonJavaClass):
    __javainterfaces__ = ['android/location/LocationListener']

    @java_method('(Landroid/location/Location;)')
    def onLocationChanged(self, location):
        print 'lat', location.getLatitude()
        print 'lon', location.getLongitude()

LocationManager = autoclass('android.location.LocationManager')
PythonActivity = autoclass('org.renpy.android.PythonActivity')
Context = autoclass('android.content.Context')

locationManager = PythonActivity.mActivity.getSystemService(
    Context.LOCATION_SERVICE)
locationManager.requestLocationUpdates(
    LocationManager.GPS_PROVIDER, 10000, 10,
    GpsListener())
```

Accessing mobile API: Plyer

Platform-independent wrapper for python, for platform-dependant apis

Not started at the moment, but GSOC is coming :)

Funding

PSF granted \$5,000 USD to Kivy project for porting Kivy and all the subprojects to Python 3.

Thank you PSF!

Funding

We asked \$600 USD to run Kivy on Raspberry Pi from Bounty Source

We got funded in [5 days](#) !

The next step is \$1,000 USD for doing a POC with ANGLE on Windows.

Thank you backers!

Thank you

Thank you for
coming!