KIVY - A Framework for **N**atural **U**ser **I**nterfaces

Nik Klever
Faculty of Computer Sciences
University of Applied Sciences Augsburg

Source of all Slides adopted from http://www.kivy.org

Kivy - Open Source Library

Kivy is an

- **Open Source Python library** for
- **rapid development** of applications that make use of
- **innovative user interfaces**, such as multi-touch apps.

Cross Platform

Kivy is running on

- **Linux,**
- **Windows,**
- **MacOSX,**
- **Android and**
- **IOS**

You can run the same code on all supported platforms.

It can use natively most input protocols and devices like

- WM_Touch, WM_Pen on **Windows**,
- Mac OS X Trackpad and Magic Mouse on **MacOSX**,
- mtdev, Linux Kernel HID, TUIO on **Linux**.

A multi-touch mouse simulator is included.

Business Friendly

Kivy is 100% **free to use**, under LGPL 3 licence. The toolkit is professionally developed, backed and used. You can use it in a product and sell your product.

The **framework is stable** and has a **documented API**, plus a programming guide to help for in the first step.

GPU accelerated

The graphics engine is built over **OpenGL ES 2,** using modern and fast way of doing graphics.

The toolkit is coming with more than 20 widgets designed to be extensible. Many parts are written in C using **Cython**, tested with regression tests.

Philosophy - 1

Fresh

Kivy is made for today and tomorrow. **Novel input methods** such as Multi-Touch have become increasingly important. Kivy is created from scratch, specifically for this kind of interaction.

Fast

Kivy is fast. This applies to both: **application development** and **application execution speeds**. Time-critical functionality in Kivy is implemented on the **C level** to leverage the power of existing compilers. Most importantly, we use the **GPU** wherever it makes sense in our context.

Flexible

Kivy is flexible. This means it can be run on a variety of different devices, including Android and iOS powered smartphones and tablets. We support all major operating systems (Windows, Linux, OS X). Kivy supports TUIO (Tangible User Interface Objects) and a number of other input sources.

Philosophy - 2

Focused

Kivy is focused. You can write a simple application with a **few lines of code**. Kivy programs are created using the Python programming language, which is incredibly versatile and powerful, yet easy to use. In addition, we created our own description language, the **Kivy Language**, for creating sophisticated user interfaces. This language allows you to set up, connect and arrange your application elements quickly.

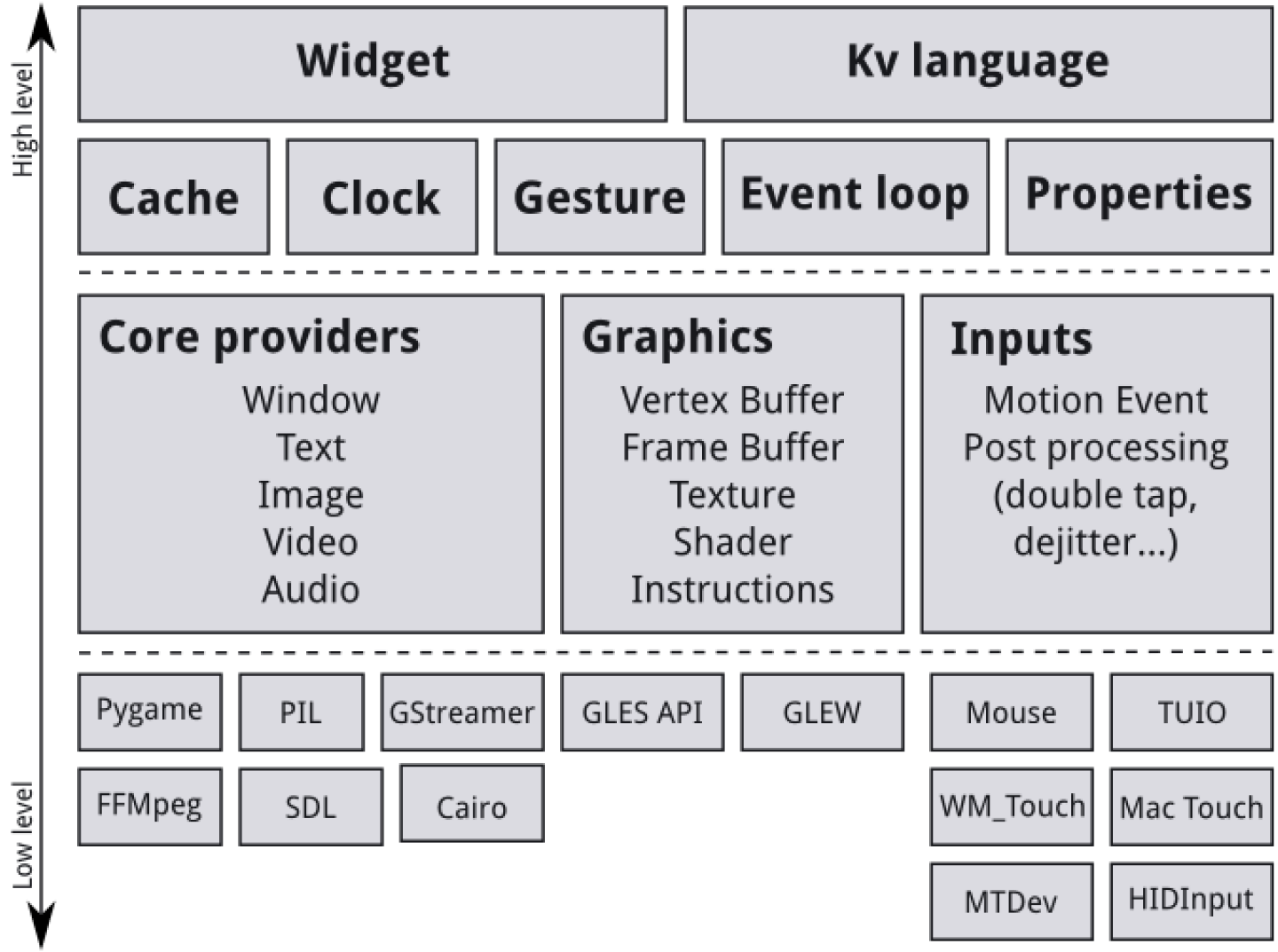
Funded

Kivy is actively **developed by professionals** in their field. Kivy is a community-influenced, professionally developed and commercially backed solution.

Free

Kivy is **free to use**. You don’t have to pay for it. You don’t even have to pay for it if you’re making money out of selling an application that uses Kivy.

Kivy Architecture



Core

The code in the core package provides commonly used features, such as:

- **Clock**
  You can use the clock to schedule timer events. Both one-shot timers and periodic timers are supported
- **Cache**
  If you need to cache something that you use often, you can use our class for that instead of writing your own.
- **Gesture Detection**
  We ship a simple gesture recognizer that you can use to detect various kinds of strokes, such as circles or rectangles. You can train it to detect your own strokes.
- **Kivy Language**
  The kivy language is used to easily and efficiently describe user interfaces.
- **Properties**
  These are not the normal properties that you may know from python. They are our own property classes that link your widget code with the user interface description.

UIX (Widgets & Layouts)

The UIX module contains commonly used widgets and layouts that you can reuse to quickly create a user interface.

- **Widgets**
  Widgets are user interface elements that you add to your program to provide some kind of functionality. They may or may not be visible. Examples would be a file browser, buttons, sliders, lists and so on. Widgets receive MotionEvents.
- **Layouts**
  You use layouts to arrange widgets. It is of course possible to calculate your widgets' positions yourself, but often it is more convenient to use one of our ready made layouts. Examples would be Grid Layouts or Box Layouts. You can also nest layouts.

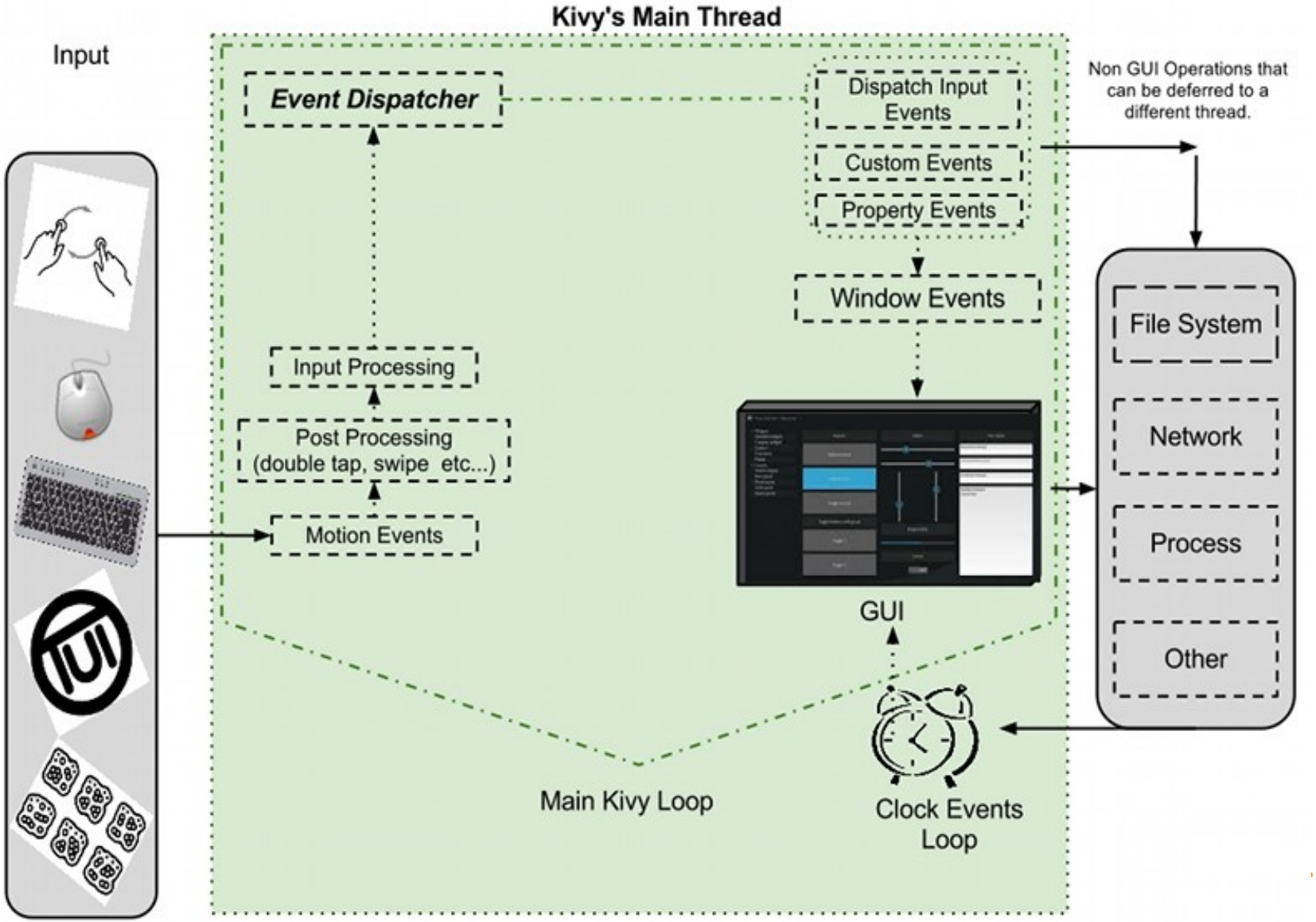
Input Events (Touches)

Kivy **abstracts different input types** and sources such as touches, mice, TUIO or similar. What all of these input types have in common is that you can associate a 2D onscreen-position with any individual input event.

All of these input types are represented by instances of the Touch() class. A touch instance, or object, can be in one of three states. When a touch enters one of these states, your program is informed that the event occurred. The three states a touch can be in are:

- **Down**
  A touch is down only once, at the very moment where it first appears.
- **Move**
  A touch can be in this state for a potentially unlimited time. A touch does not have to be in this state during its lifetime. A ‘Move’ happens whenever the 2D position of a touch changes.
- **Up**
  A touch goes up at most once, or never. In practice you will almost always receive an up event because nobody is going to hold a finger on the screen for all eternity, but it is not guaranteed. If you know the input sources your users will be using, you will know whether or not you can rely on this state being entered.

Events and Properties



Tutorial Example **Pong**

Pong is an easy, small and very common example of the old Ping-Pong Computer Game – it is introduced as a tutorial for Kivy on the site

http://kivy.org/docs/tutorials/pong.html

What is needed ? Download Kivy as described on the Download Site

HowTo-Example on Android:

1. Download the Kivy Launcher App from the Google Play Store
2. Create a directory named **kivy** in the root-directory on the storage of your Android smartphone
3. Create a directory named **pong** in this directory
4. Copy the files **main.py** and **pong.kv** into this directory pong
5. Create a file name **android.txt** with the following content:

```
title=Pong Tutorial
author=KivyTeam
orientation=landscape
```

6. Run the application **Kivy Launcher** and select **Pong Tutorial**
7. Change pong.kv for your special needs (e.g. introduce a Color)

Questions ?

Kivy is well documented on it's Website

http://www.kivy.org

Thanks to all contributors of Kivy, especially the core team

- Mathieu Virbel
- Thomas Hansen
- Gabriel Pettier