INTRODUCCIÓN A KIVY

PROGRAMACIÓN III

Abdel G. Martínez L.

CONCEPTO Y DEFINICIÓN

- Librería Python de código abierto para el rápido desarrollo de aplicaciones que hacen uso de interfaces de usuario modernas, como aplicaciones multi-touch.

CARACTERÍSTICAS

Múltiples Plataformas

- Se ejecuta en Linux, Windows, Mac OS X, Android y iOS.
- El mismo código en todas las plataformas.
- Usa nativamente muchas entradas, protocolos y dispositivos.
- Se incluye un simulador de mouse multi-toques.

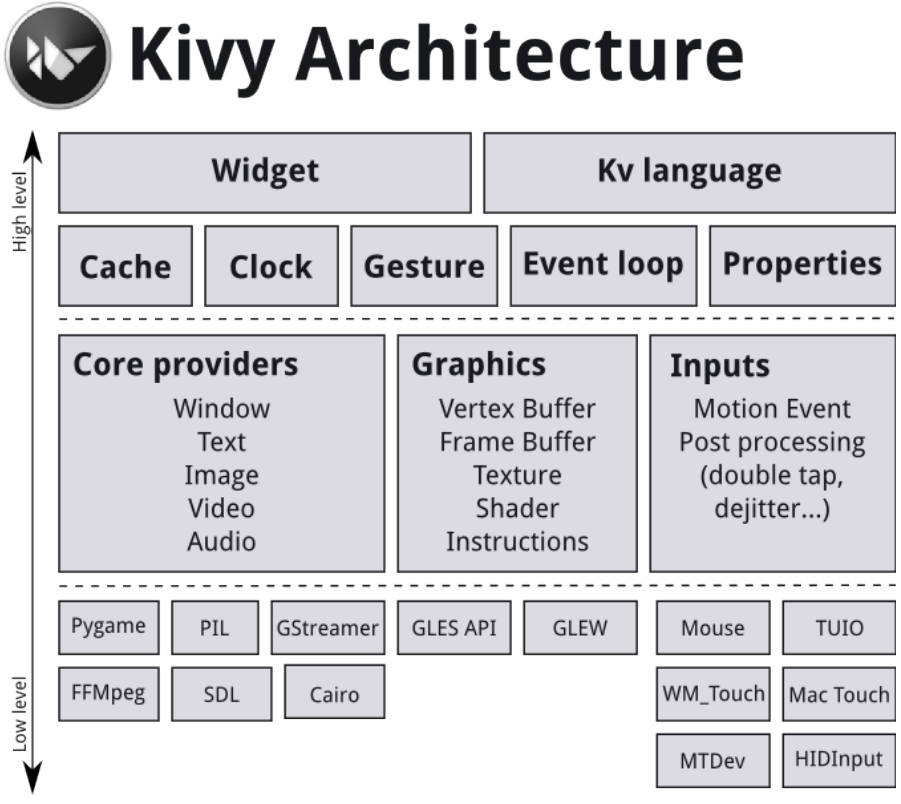
Amigable a Negocios

- Es 100% libre, bajo licencia MIT.
- Profesionalmente desarrollado, empacado y utilizado.
- Puede ser utilizado en un producto comercial.
- Este framework es estable y está bien documentado.
- Posee un amplio API.

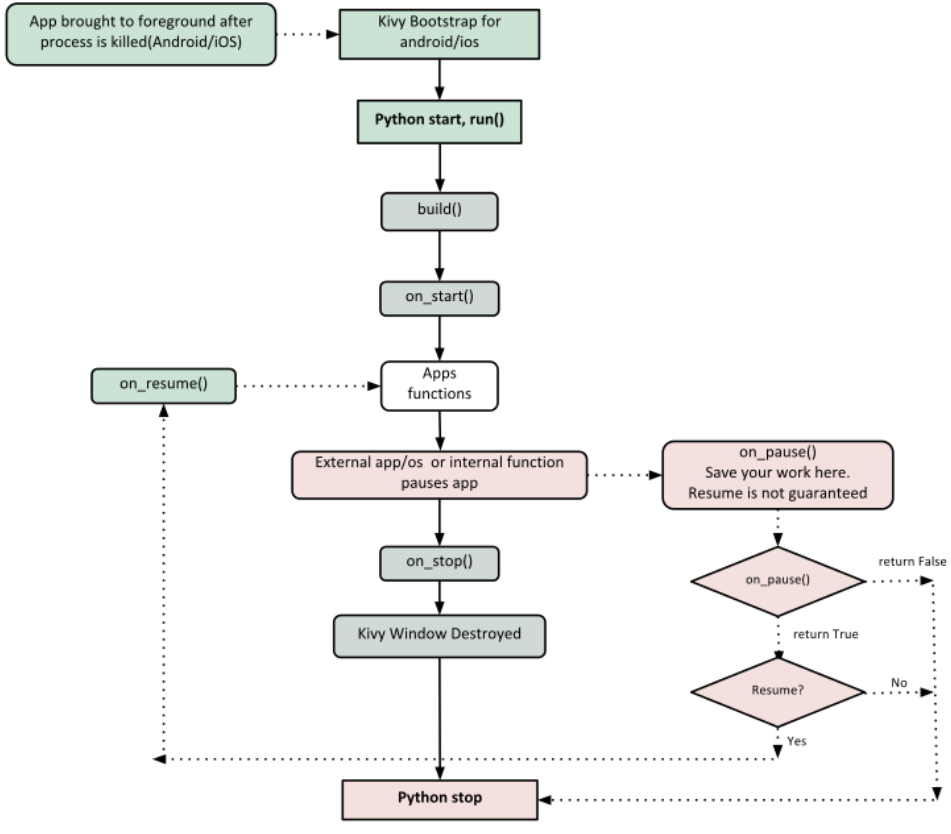
Acelerado por GPU

- El motor gráfico está construido sobre OpenGL ES 2, utilizando una línea gráfica moderna y rápida.
- Viene con más de 20 widgets, altamente extensibles.
- Muchas partes fueron escritas en C y Cython, probadas con pruebas de regresión.

ARQUITECTURA

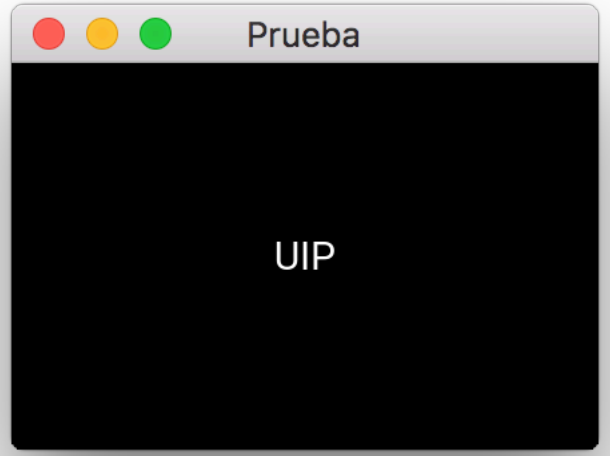


CICLO DE VIDA DE APLICACIÓN EN KIVY



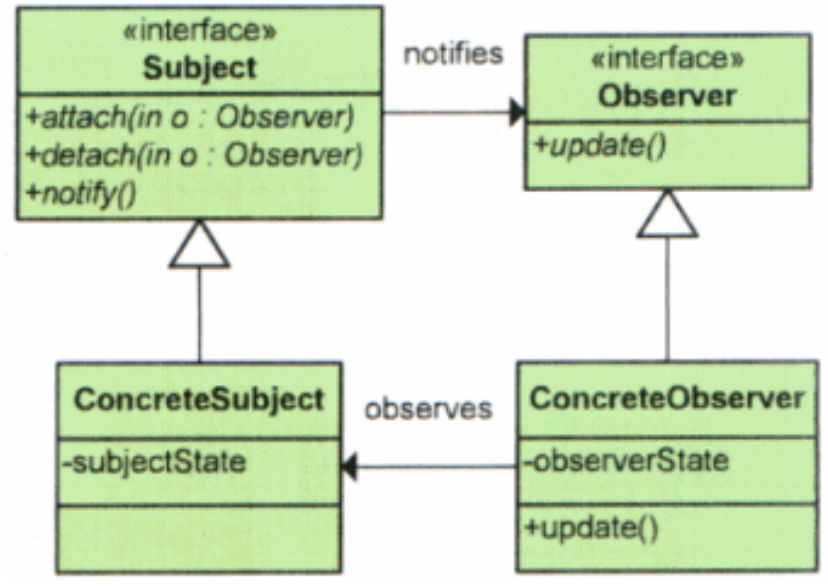
PRIMER PROGRAMA

```
from kivy.app import App
from kivy.uix.label import Label

class PruebaApp(App):
    def build(self):
        return Label(text='UIP')

PruebaApp().run()
```

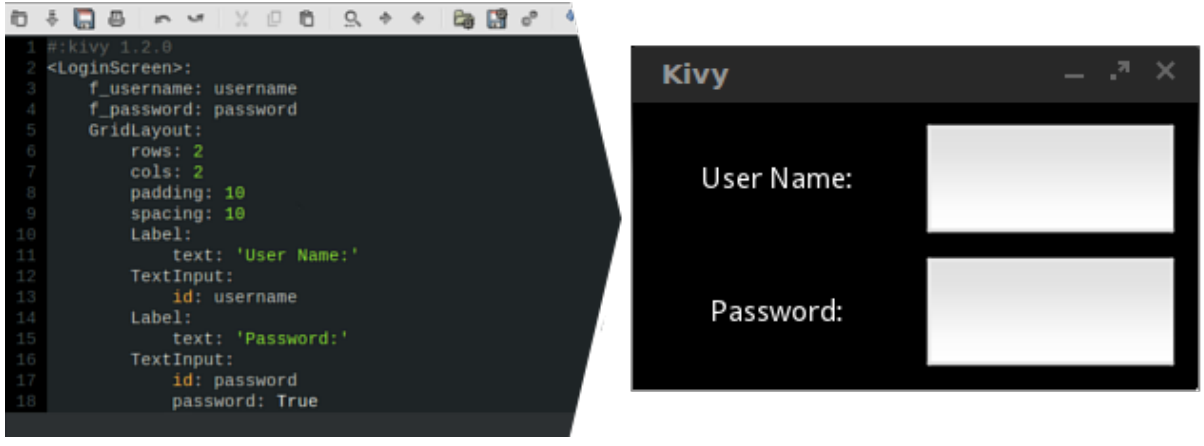


OBSERVER (PATRÓN DE DISEÑO)

- Define dependencia uno a muchos entre objetos.
- Cuando un objeto cambia su estado, notifica este cambio a todos los dependientes.
- Este patrón es la clave de la arquitectura MVC.
- Implementado en GUI.



LENGUAJE DE DISEÑO KV



- Kivy provee un lenguaje de diseño específico para GUI.
- Este lenguaje simplifica la separación entre la interfaz de usuario y la lógica de la aplicación.

PRIMER PROGRAMA (KV)

```
# Archivo: prueba.py
from kivy.app import App
from kivy.uix.label import Label

class PruebaApp(App):
    def build(self):
        return Label()

PruebaApp().run()
```

```
# Archivo: prueba.kv
#:kivy 1.9.0
<Label>:
    text: 'UIP'
```

EVENTOS

Eventos de Reloj

- Permite agendar una llamada a una función como un evento único con schedule_once(), o como un evento repetitivo con schedule_interval().

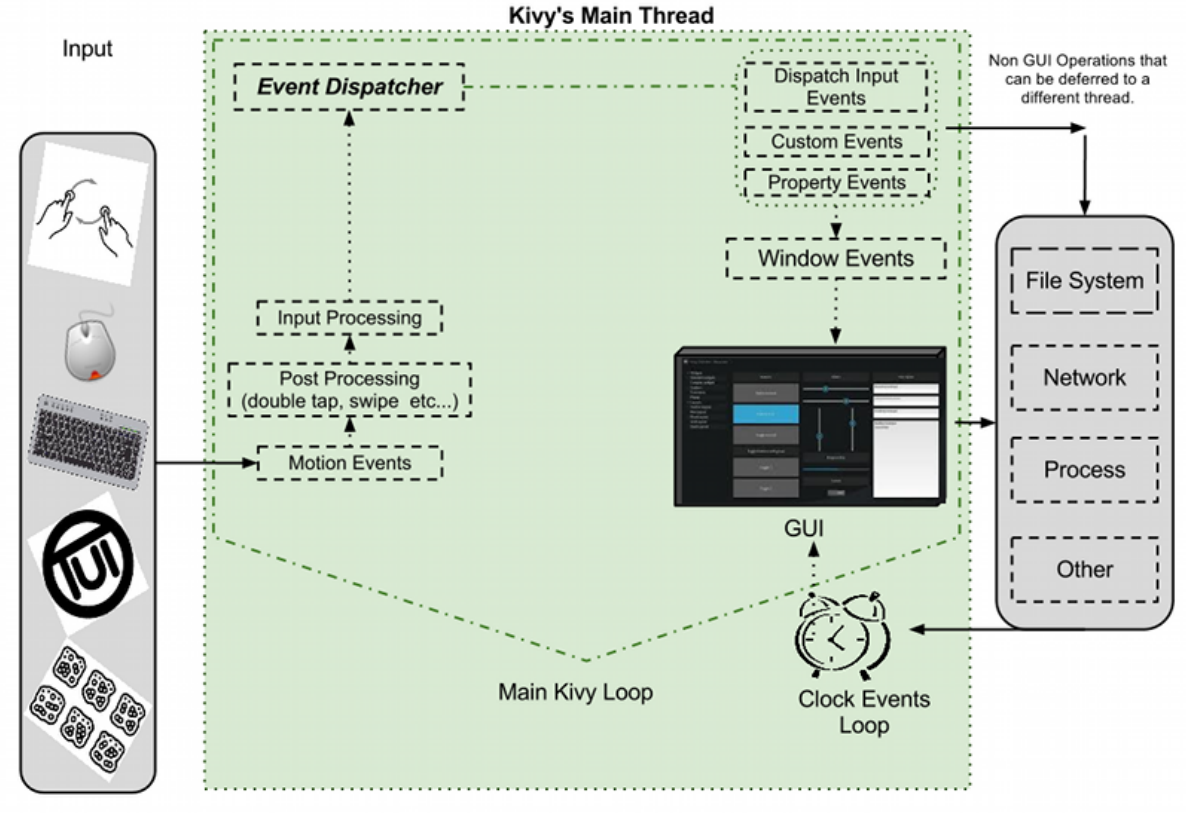
Eventos de Entrada

- Todos los eventos de clic del ratón y desplazamiento se despachan por el evento on_motion.
- Genera on_touch_down(), on_touch_move(), on_touch_up().

Eventos de Clases

- Cuando un widget cambia su posición o tamaño.
- Da la oportunidad de crear nuestros propios eventos y sobreescribir los existentes.

EVENTOS


Kivy's Main Thread
Input
Event Dispatcher
Dispatch Input Events
Custom Events
Property Events
Window Events
Non GUI Operations that can be deferred to a different thread.
File System
Network
Process
Other
Input Processing
Post Processing (double tap, swipe etc...)
Motion Events
GUI
Main Kivy Loop
Clock Events Loop

DISEÑO

AnchorLayout	• Widgets pueden ser anclados a arriba, abajo, izquierda, derecha o centro.
BoxLayout	• Widgets son acomodados secuencialmente en una orientación vertical u horizontal.
FloatLayout	• Widgets no siguen restricciones.
RelativeLayout	• Los widgets hijos se posicionan de acuerdo al diseño.

DISEÑO

GridLayout	• Widgets se acomodan en una grilla definida por filas y columnas.
PageLayout	• Utilizado para crear diseños multipáginas.
ScatterLayout	• Similar a RelativeLayout, pero los widgets pueden ser movidos, rotados o escalados.
StackLayout	• Widgets se apilan en lr-tb o tb-lr.

CANVAS

```python
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.graphics import *

class MyWidget(Widget):

    def __init__(self, **kwargs):
        super(MyWidget, self).__init__(**kwargs)
        self.bind(pos=self.update_canvas)
        self.bind(size=self.update_canvas)
        self.update_canvas()

    def update_canvas(self, *args):
        # need to reset everything
        self.canvas.clear()
        with self.canvas:
            # context instruction
            Color(0.5, 0.5, 0.5, 0.5)
            # vertex instruction
            Ellipse(pos = self.pos, size = self.size)

class MyApp(App):
    def build(self):
        return MyWidget()

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

VS

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

class MyWidget(Widget):
    pass

class MyApp(App):
    def build(self):
        return MyWidget()

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

my.kv

```
#kivy 1.0
<MyWidget>
    canvas:
        Color:
            rgba: .5, .5, .5, .5
        Ellipse:
            size: self.size
            pos: self.pos
```

EMPAQUETADO

PyInstaller

Buildozer

python-for-android

kivy-ios

PLYER

Platform	Android	iOS	Windows	OS X	Linux
Accelerometer	X	X		X	X
Call	X				
Camera (taking picture)	X	X			
GPS	X	X			
Notifications	X		X	X	X
Text to speech	X	X	X	X	X
Email (open mail client)	X	X	X	X	X
Vibrator	X	X			
Sms (send messages)	X				
Compass	X	X			
Unique ID	X	X	X	X	X
Gyroscope	X	X			
Battery	X	X	X	X	X
Native file chooser			X	X	X
Orientation	X				
Audio recording	X				
Flash	X	X			

RECURSOS ADICIONALES

- **API:** http://kivy.org/docs/api-kivy.html
- **Wiki:** https://github.com/kivy/kivy/wiki
- **Documentación:** https://kivy.org/docs/gettingstarted/intro.html
- **Google Groups:** #kivy-users
- **IRC:** #kivy on irc.freenode.net