A Quick Introduction to

kivy

Toronto Python Meetup
August 12, 2015
Christopher Trudeau (@cltrudeau)

kivy

- python based GUI toolkit
- multi-platform: Windows, Linux, OS X, Raspberry PI, Android, iOS
- current version 1.9
- 2.7 <= python < 3
- SDL2 or PyGame Rendering

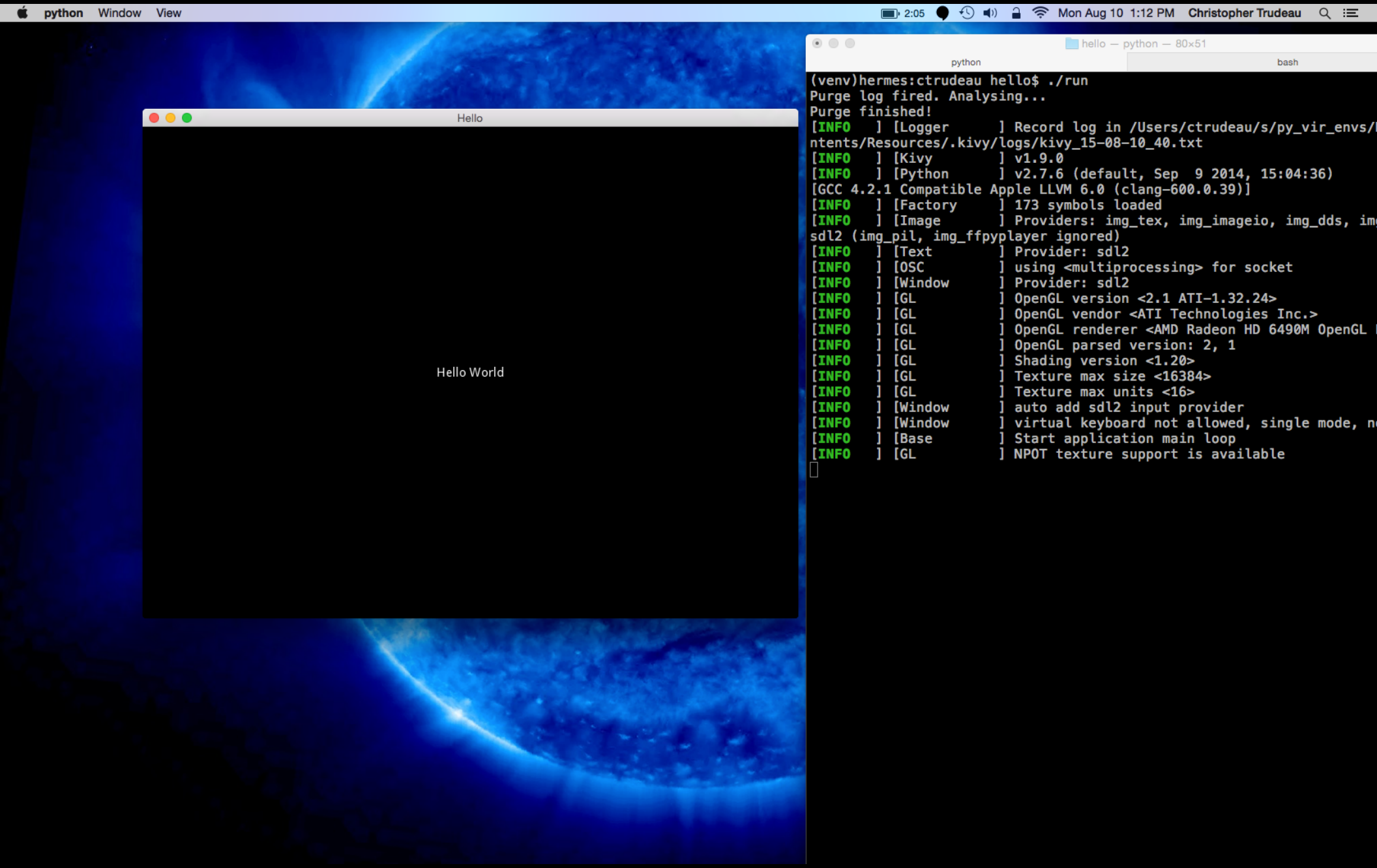
Hello World

hello.py

```
import kivy
kivy.require('1.9.0')
from kivy.app import App
from kivy.uix.label import Label

class HelloLabel(Label):
    pass

class HelloApp(App):
    def build(self):
        return HelloLabel()

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

hello.kv

```
<HelloLabel>:
    text: 'Hello World'
```


python Window View
2:05
Mon Aug 10 1:12 PM
Christopher Trudeau
Hello
Hello World
hello — python — 80×51
python
bash
(venv)hermes:ctrudeau hello$ ./run
Purge log fired. Analysing...
Purge finished!
[INFO] [Logger] Record log in /Users/ctrudeau/s/py_vir_envs/
ntents/Resources/.kivy/logs/kivy_15-08-10_40.txt
[INFO] [Kivy] v1.9.0
[INFO] [Python] v2.7.6 (default, Sep 9 2014, 15:04:36)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.39)]
[INFO] [Factory] 173 symbols loaded
[INFO] [Image] Providers: img_tex, img_imageio, img_dds, im
sdl2 (img_pil, img_ffpyplayer ignored)
[INFO] [Text] Provider: sdl2
[INFO] [OSC] using <multiprocessing> for socket
[INFO] [Window] Provider: sdl2
[INFO] [GL] OpenGL version <2.1 ATI-1.32.24>
[INFO] [GL] OpenGL vendor <ATI Technologies Inc.>
[INFO] [GL] OpenGL renderer <AMD Radeon HD 6490M OpenGL
[INFO] [GL] OpenGL parsed version: 2, 1
[INFO] [GL] Shading version <1.20>
[INFO] [GL] Texture max size <16384>
[INFO] [GL] Texture max units <16>
[INFO] [Window] auto add sdl2 input provider
[INFO] [Window] virtual keyboard not allowed, single mode, n
[INFO] [Base] Start application main loop
[INFO] [GL] NPOT texture support is available

Kivy Chat

- main: App, Rootbox
- chatbox: Chatbox, MessageBox, MessageTextInput
- chatclient: implements SleekXMPP
- friends: Friend, FriendLabel, FriendRow, FriendBox
- tabbox: TabLabel, Tab, TabBox
- utils: DoubleClickBehavior, Menu, TextBoxLabel, ConfirmPopUp, load_kv_files

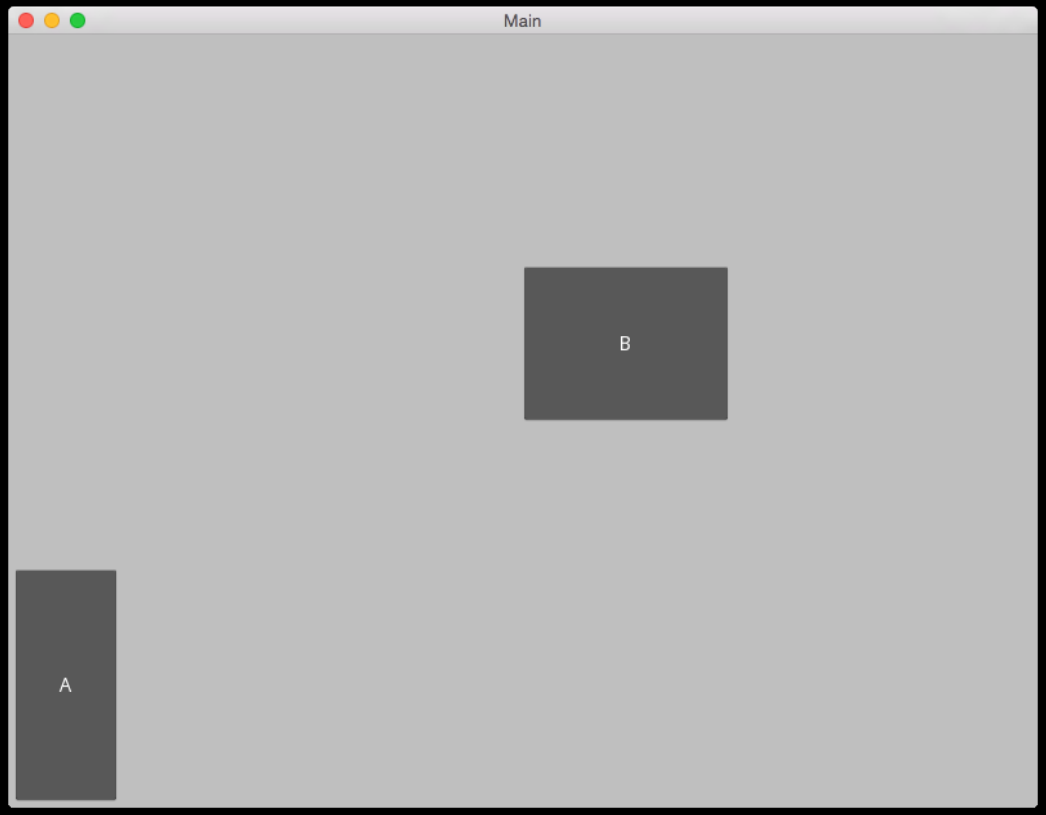
Layouts

- Anchor: widget sticks to positions: top, bottom, left, right or centre
- Box: horizontal or vertical sequence
- Float: anywhere
- Relative: widget position relative to layout container
- Grid: rows and columns of widgets
- Page: multiple screens to flip between
- Scatter: like Relative but with transformations
- Stack: left -> right then wrap, or top -> bottom wrap

Layout Attributes

- widgets have standard: x, y, width, height
- size_hint, size_hint_x, size_hint_y: percentage of parent's size to take up as a floating point
- pos_hint, pos_hint_x, pos_hint_y: percentage of margin
- respects resize

```
<MainBox>:
    Button:
        text:'A'
        size_hint:(.1, .3)
        pos:(5, 5)

    Button:
        text:'B'
        size_hint_x:.2
        size_hint_y:.2
        pos_hint:{'x':.5, 'y':.5}
```



Layouts and Text

- Text layout behaviour is a bit painful
- Separation of concept of size of widget, size of text and the texture being drawn

```
<TextBoxLabel>:
    size_hint_y: None
    text_size: self.width, None
    height: self.texture_size[1]
    markup: True
```

Events

- Everything is Event based
- Bind events as python functions

```
class KivyChatApp(App):
    def build(self):
        self.root_box = RootBox()
        Window.bind(on_close=self.pushed_close)

        return self.root_box

    def pushed_close(self, *args):
        print 'Disconnecting...'
        if self.root_box.chat_client:
            self.root_box.chat_client.disconnect(wait=True)
```

Thread Safety

- `MyApp().run()` spawns its own thread
- Kivy GUI is NOT thread safe!
- Use `Clock` and `Queue`
- `Clock.create_trigger()` — creates a single call-back to be triggered once at the end of the next redraw

```
class ChatClient(ClientXMPP):
    def __init__(self, root_box, userid, password):
        # <snip>

        # register the callback trigger for when we receive a chat (processing
        # of chat has to happen on kivy's thread)
        self._chat_trigger = Clock.create_trigger(self._sync_receive_chat)

    def receive_chat(self, msg):
        """This call will happen on SleekXMPP's thread"""
        #print '==> rcvd:', msg
        if msg['type'] in ('chat', 'normal', ):
            self.root_box.receive_queue.put(msg)
            self._chat_trigger()

    def _sync_receive_chat(self, delta_time):
        """Should only be called by triggered event on kivy's thread to deal
        with any received messages."""
        try:
            while(True):
                # not an infinite loop, get() throws Empty
                msg = self.root_box.receive_queue.get(block=False)
                self._handle_chat_message(msg)
        except Empty:
            # nothing left to do
            pass
```

Behavio(u)rs

- Mix-in classes that manage certain kinds of events
- Easy way of adding click handling, etc. to your own widgets

```
class DoubleClickBehavior(ButtonBehavior):
    def on_touch_down(self, touch):
        if not touch.is_double_tap:
            return False

        super(DoubleClickBehavior, self).on_touch_down(touch)



class FriendLabel(DoubleClickBehavior, TextBoxLabel):
    def pushed(self, *args):
        friend = self.parent.friend
        if friend.status != 'online':
            return

        # click means start/switch to a conversation
        self.friend.message_count = 0
        app = App.get_running_app()
        app.root_box.menu.show_item('Chats', select='Chats')
        app.root_box.ids.screen_manager.current = 'Chats'
        app.root_box.chat_box.add_chat(friend)
```

The Good

- Easy rapid prototyping
- multi-platform
- robust toolkit
- responsive community

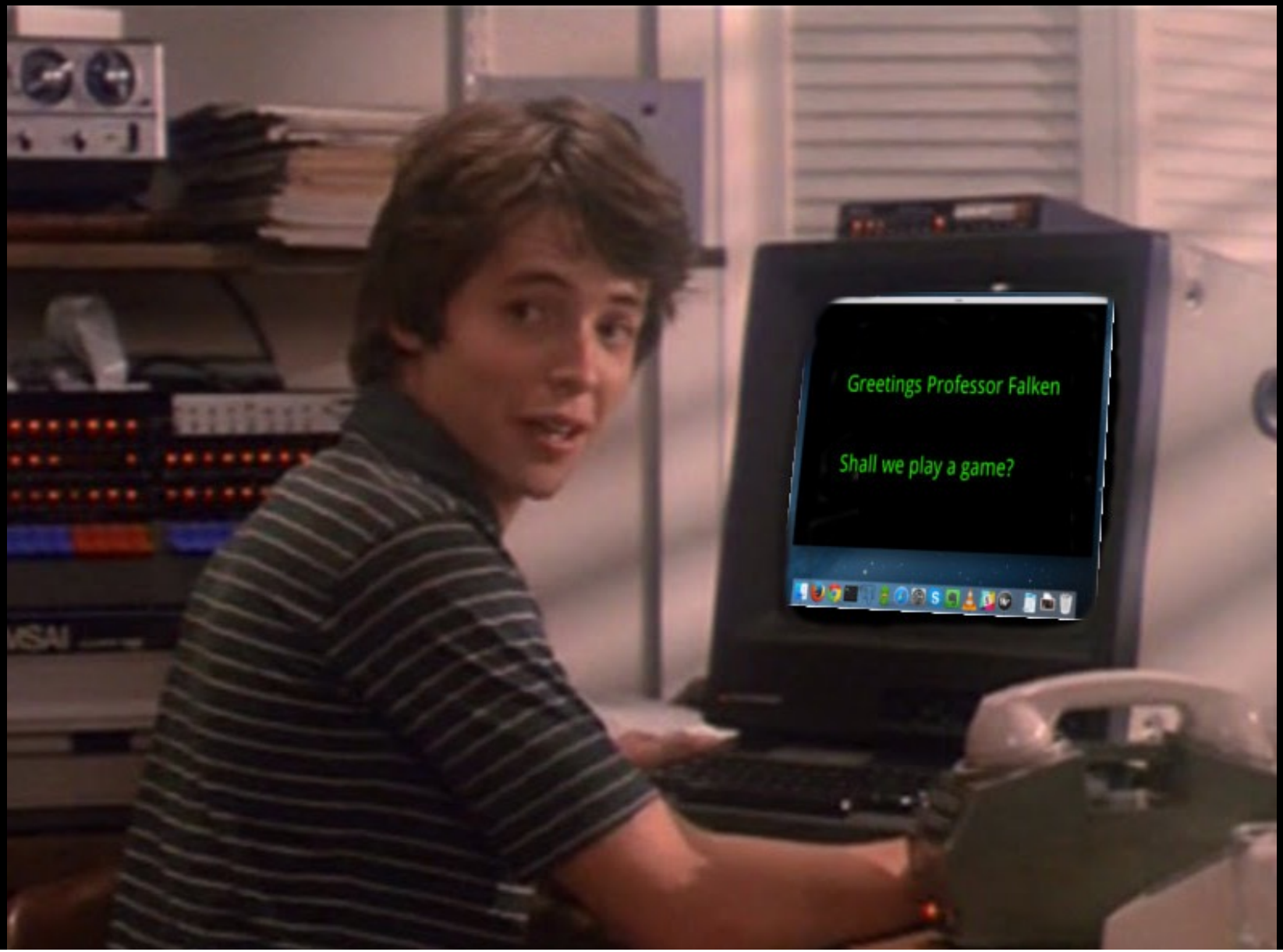
The Bad

- Doesn't install with "pip"
- Under v1.8 install as system and use

```
virtualenv –system-site-packages
```

- Under v1.9 it has its own virtualenv, so copy the whole thing and install to it

The Ugly



Thanks

Twitter: @cltrudeau

Presentation & Code:

https://github.com/Aziiri-dev/kivychat