Screenflow Documentation

Release 1.0.0

Félix Voituret

Apr 07, 2017

Screenflow

1 Getting Started **1**
- 1.1 ScreenFlow is based on Screen . . . 1
- 1.2 Installation . . . 1
- 1.3 First screenflow . . . 1

2 Screen **3**
- 2.1 Rendering . . . 3
- 2.2 Event handling . . . 3

3 ScreenFlow **5**
- 3.1 Screen binding . . . 5
- 3.2 XML loading . . . 5
- 3.3 Custom screen . . . 6

4 MessageScreen **7**
- 4.1 XML definition . . . 7
- 4.2 Callback binding . . . 7

5 SelectScreen **9**
- 5.1 XML definition . . . 9
- 5.2 Callback binding . . . 9

Python Module Index **11**

CHAPTER 1

Getting Started

ScreenFlow is a light UI engine built on top of Pygame. It is primarily designed for building configuration interface for IoT devices like RaspberryPi.

ScreenFlow is based on Screen

There is no screen flow without screen obviously, so an application built with **ScreenFlow** requires to create and connect together a serie of screens. The application switch from one screen to another by sliding horizontally, keeping the navigation state into an internal stack.

A **Screen** aims to be dedicated to a single task, for exemple :

- Input text
- Allow user to select option
- View a list or a grid of items
- And so on ...

That is why lot of basic **Screen** implementation are already available as in.

Installation

Screenflow is available on **PyPi** repository and can then be installed through *pip* :

```
pip install screenflow
```

First screenflow

Simple hello world with **screenflow** looks like this :

```
from screenflow import ScreenFlow
from screenflow.screens import MessageScreen

screenflow = ScreenFlow()
message = MessageScreen('intro', 'Hello screenflow !')

@screenflow.intro.on_touch()
def on_message_touch():
    screenflow.quit()

screenflow.add_screen(message)
screenflow.run(message)
```

CHAPTER 2

Screen

A **Screen** is a basic unit which is manipulated by a **Screenflow**.

Rendering

Text rendering

In order to draw text, a **Screen** use a **FontManager**, which manages.

Background

Event handling

CHAPTER 3

ScreenFlow

ScreenFlow class is the library entry point, which implements all the logic to manages event, handles screen creation, and so on.

Screen binding

Each screen belonging to your screenflow can be retrieved using attribute binding. Let say you added a screen with name *foo* in a given screenflow instance, such screen could be accessed as following :

```
screenflow.foo
```

XML loading

ScreenFlow allows you to define your screens by using XML format using following convention :

```
<?xml version="1.0"?>
<screenflow>
    <!-- Your screens here-->
</screenflow>
```

Where each screen is defined as following :

```
<screen name="screen_name" type="screen_type">
    <!-- Your screen specific parameter here -->
</screen>
```

The *name* attribute will be used for attribute binding. Checkout available screens documentation to know what parameter can be settled.

Custom screen

You can implements your own screen by extending the Screen base class. Although for screenflow to recognize your screen implementation when parsing an XML file, you should register a factory function.

Such factory function should match the following signature :

```
def my_factory(screen_def):
    my_screen = ... // Create your screen instance here.
    return my_screen
```

Where *screen_def* parameter is a dictionary from xmltodict parsing library. Then registering such function is as easy as following :

```
screenflow = ScreenFlow()
screenflow.register_factory('type_name', my_factory)
```

CHAPTER 4

MessageScreen

MessageScreen is the most basic screen implementation that just aims to display a message to the user and trigger callback when a mouse event occurs at any position in the screen.

XML definition

```
<screen name="foo" type="message">
    <message>displayed message</message>
</screen>
```

Callback binding

Given a *screenflow* instance, with registered *foo* message screen, callback binding can be achieved using `on_touch` decorator:

```
@screenflow.foo.on_touch
def on_foo_touch():
    # TODO : Callback action here.
```

CHAPTER 5

SelectScreen

Description incomming.

XML definition

```
<screen name="foo" type="select">
    <message>displayed message</message>
    <option>first choice</option>
    <option>second choice</option>
    ...
</screen>
```

Callback binding

Given a *screenflow* instance, with registered *foo* select screen, callback binding can be achieved using `on_select` decorator:

```
@screenflow.foo.on_sekect
def on_foo_select(option):
    # TODO : Callback action here.
```

To document

Python Module Index

S

screenflow.screenflow, 3
screenflow.screens.input_screen, 9
screenflow.screens.message_screen, 6
screenflow.screens.screen, 2
screenflow.screens.select_screen, 7

Index

S

screenflow.screenflow (module), 3
screenflow.screens.input_screen (module), 9
screenflow.screens.message_screen (module), 6
screenflow.screens.screen (module), 2
screenflow.screens.select_screen (module), 7