Kivy_Basics

October 18, 2020

1 Grundlagen

Eine `Kivy`-App besteht im Wesentlichen aus drei Elementen: - Unterklassifizierung der Klasse `App` - `build()`-Methode implementieren, sodass eine Widget-Instanz zurückgegeben wird - Instanziierung dieser Klasse und Aufruf der `run()`-Methode

1.1 Python-Module herunterladen

Installation: Windows

[ ]:
```
! pip install --upgrade pip wheel setuptools
! pip install docutils pygments pypiwin32 kivy-deps.sdl2 kivy-deps.glew
! pip install kivy-deps.gstreamer
! pip install kivy-deps.angle
! pip install --upgrade kivy
```

Installation: OS X

[ ]:
```
! python -m pip install kivy
```

1.2 Beispiel

Minimale `Kivy`-Anwendung

[ ]:
```
import kivy
from kivy.app import App
from kivy.uix.label import Label


class MyApp(App):

    def build(self):
        return Label(text='Hello world!')


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

1.3 Erläuterungen

1.3.1 Import / Laden der Module

```
[ ]: from kivy.app import App
```

Die Basisklasse der App muss von der Klasse `App` aus `Kivy` erben.

```
[ ]: from kivy.uix.label import Label
```

Pakete und Klassen sind in `Kivy` in Unterordnern angelegt. So befindeen sich beispielsweise die Elemente der Benutzerschnittstelle (auch User Interaction and Experience aka. UIX) wie **Layouts und Widgets** im Modul `uix`.

1.3.2 Klassen definieren / Widgets hinzufügen

```
[ ]: class MyApp(App):
```

Hier wird die Basisklasse der Kivy-App definiert. > Die Basisklasse muss von der Klasse `App` aus dem Modul `kivy.app` erben. Folglich ist `App` die ***Oberklasse*** und `MyApp` die ***Unterklasse***.

```
[ ]:     def build(self):
             return Label(text='Hello world!')
```

Mithilfe der `build()`-Funktion wird das `Root Widget` initiiert und wiedergegeben. In diesem Beispiel wird auf die Klasse `Label` aus dem `kivy.uix.label` zurückgegriffen und ein Objekt erzeugt. Der Klasse `Label` wird der Parameter `text='Hello world!'` übergeben und mit `return` das Objekt aus der Methode `build()`.

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

Der Code `if __name__ == '__main__':` ist als Kontrollstruktur (eng. ***Execution Control***, siehe www.realpython.com) bekannt. Eine Python-Datei hat in der Regel zwei Zwecke, entweder: - es handelt sich um ein eigenständiges Programm, was ablaufen soll (z.B. wenn in der Konsole `python first_kivy_app.py` getätigt wird) oder - es handelt sich um eine Ansammlung von Funktionen, Klassen oder Konstanten auf die zurückgegriffen werden soll (z.B. wenn mithilfe von `from kivy.app import App` die vordefinierte Klasse `App` importiert werden soll).

Durch die Verwendung von `if __name__ == '__main__':` müssen nicht zwei Dateien angelegt werden, sondern können durch eine einzige Python-Datei bedient werden.

Der Aufruf von `MyApp()` erzeugt das Objekt aus der Unterklasse `MyApp`, welche von der Oberklasse `App` alle Methoden, Klassenvariablen usw. erbt. Mithilfe der Methode `run()`, die eine Methode der Oberklasse `App` darstellt wird die `Kivy`-App gestartet.

1.3.3 Ausführen der `Kivy`-App

```
[ ]: ! python first_kivy_app.py
```

Die `Kivy`-App kann mittels des Befehls im Terminal ausgeführt werden.

---

2 Weitere `Kivy`-Module

[ ]:
```
! python -m pip install kivy-garden
! python -m pip install ffpyplayer
! python -m pip install kivymd
```

[ ]:
```

```

[ ]:
```

```