Introdução ao Desenvolvimento de Interfaces Multitoque com PyMT

Antonio Miguel Batista Dourado

antonio_dourado@dc.ufscar.br

Agenda

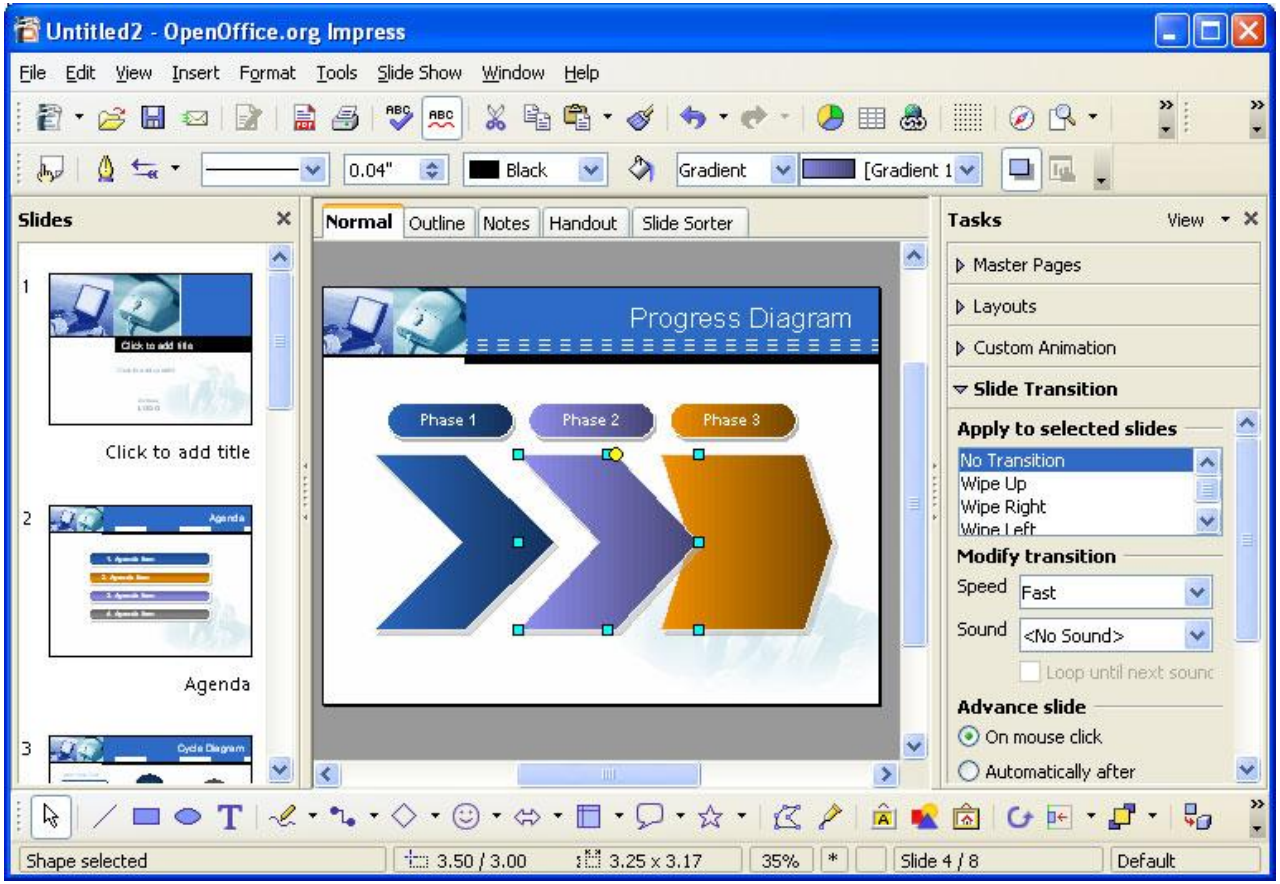
- Apresentação
- O que é o PyMT?
- Instalação
- Hello World
- Widgets
- Events
- Animations
- CSS
- Exercícios
- Showcases

Apresentação

Grande parte das ferramentas de construção de GUIs (Graphical User Interface) são baseadas em WIMP (Windows, Icons, Menus, Pointer).

Hansen, 2009

Apresentação



Interface baseada em WIMP

Apresentação

Interfaces multitoque introduzem novos conceitos de interface GUI e técnicas de interação.

Hansen, 2009

Apresentação

Interface Multitoque

O que é o PyMT?

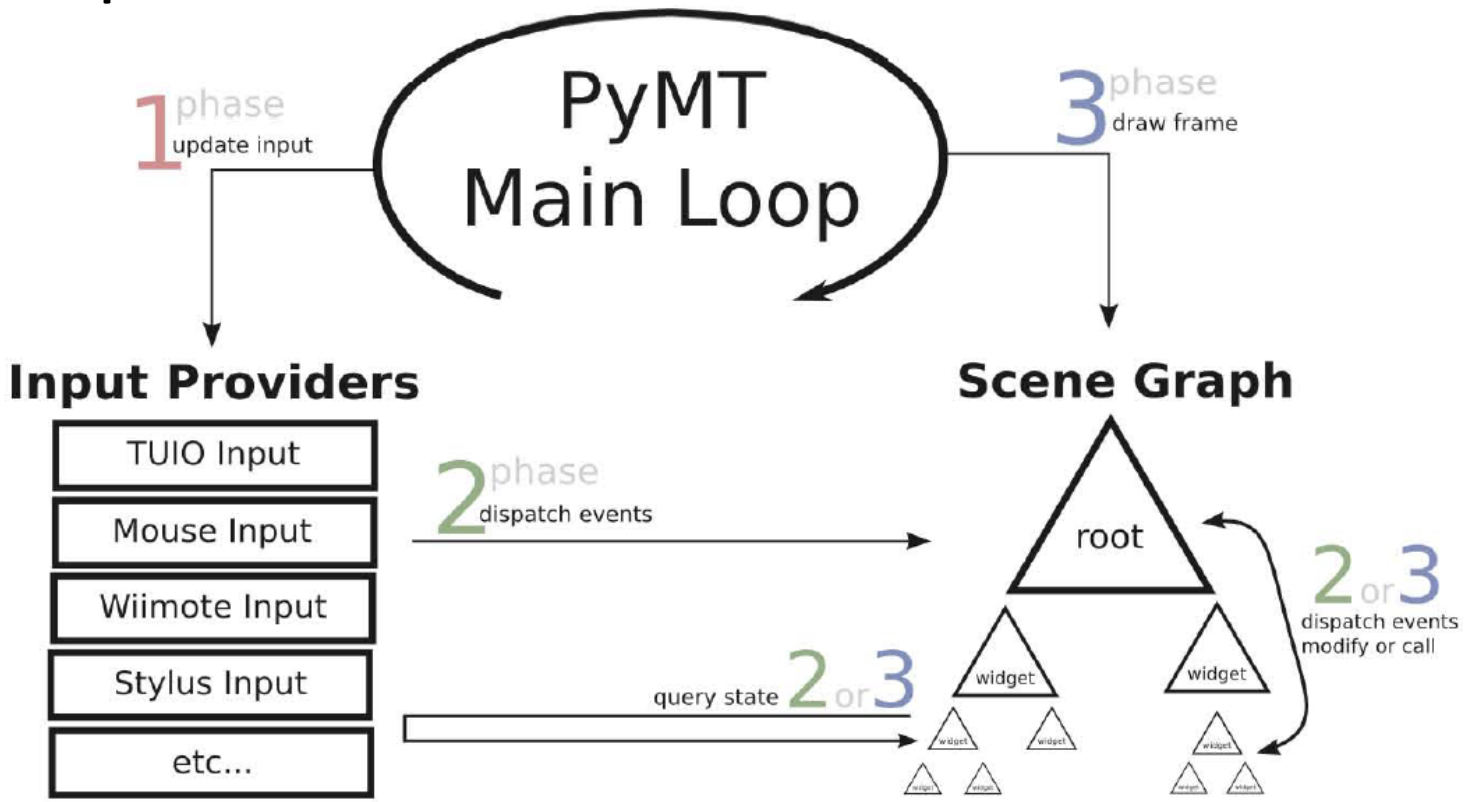
- Framework Open Source baseado em Python para desenvolvimento de aplicações multitoque

- Possui suporte a vários input providers
  - TUIO
  - Mouse
  - Windows 7 multi-touch
  - OSX multi-touch pad e mouse
  - Linux HID multi-touch
  - Outros

- Site oficial: http://pymt.eu

O que é o PyMT?

- Arquitetura



Instalação

- Para Windows 7/Vista/XP
  - Baixe o arquivo em: http://pymt.googlecode.com/files/pymt-0.5.1-win32.zip
  - Extraia o arquivo no diretório C:
  - Adicione o caminho “C:\pymt-0.5-w32” nas variáveis de ambiente do Windows

- **Para o Windows XP, em caso de erro de DLL, usar os mesmo passos acima e os seguintes
  - Baixe a DLL MSVCR71.dll em: http://www.dll-files.com/dllindex/dll-files.shtml?msvcr71
  - Extraia a DLL do zip no diretório “C:\WINDOWS\system32”

Instalação

- Testando a instalação
  - Abra o prompt de comando (Iniciar, Executar, CMD)
  - Ir até o diretório “C:\pymt-0.5-w32\pymt”
  - Digitar o comando “*pymt launcher.py*”

Hello World

```
from pymt import *

if __name__ == "__main__":

    button = MTButton(id="mtbtn1",label="Hello World!")
    window = getWindow()
    window.add_widget(button)
    runTouchApp()
```

Widgets

- Widgets são componentes GUI *(Wikipedia)*
- O PyMT conta com diversos widgets prontos para o uso
- Todos os Widgets do PyMT tem como base a classe MTWidget
- É possível criar Widgets novos, porém estes precisam seguir a linha de derivação da MTWidget

Widgets

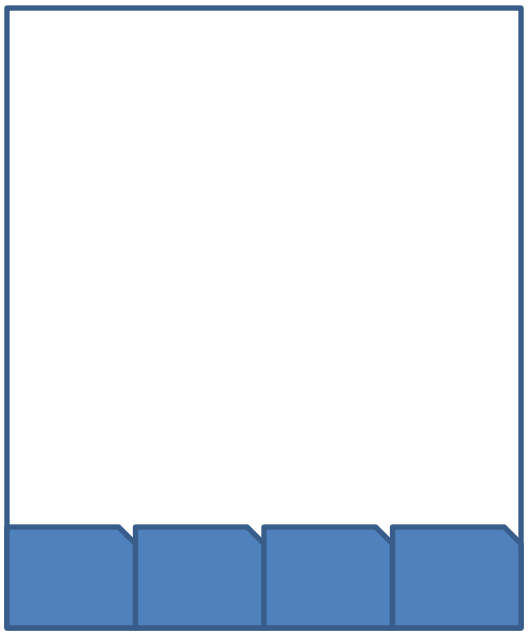
- Alguns Widgets
  - Layout
  - TextInput
  - Popup
  - Sliders
  - Scatters

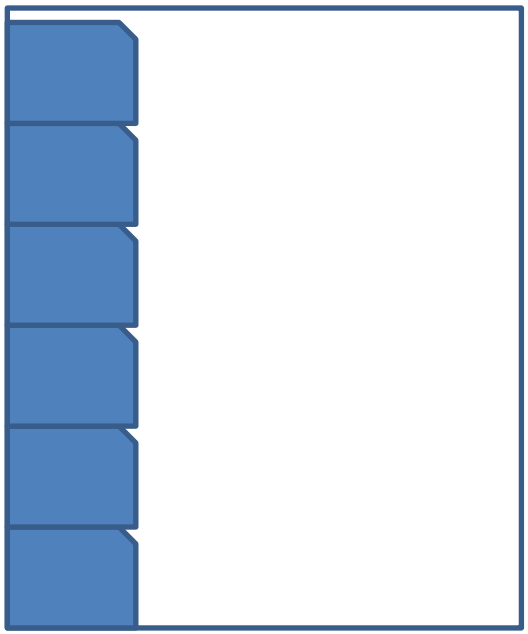
Widgets

- Os Layout Widgets auxiliam o posicionamento dos Widgets.
  - Abstract Layout
  - Anchor Layout
  - Box Layout
  - Grid Layout
  - Screen Layout

Widgets

- O Box Layout arranja os Widgets de forma horizontal ou vertical





Widgets

```
from pymt import *


if __name__ == "__main__":

    layout = MTBoxLayout(orientation='horizontal')

    button1 = MTButton(label="Button 1")
    button2 = MTButton(label="Button 2")
    button3 = MTButton(label="Button 3")

    layout.add_widget(button1)
    layout.add_widget(button2)
    layout.add_widget(button3)

    runTouchApp(layout)
```

Widgets

- TextInput
  - Atributo password (bool) – Define se o campo é destinado a senhas com preenchimento de asteriscos
  - Atributo group (strId) – Define um grupo de TextInputs unidos pela string strId fazendo com que haja a navegação entre os campos através do TAB.
  - Teclado Virtual automático

Widgets

```
from pymt import *


if __name__ == "__main__":

    layout = MTBoxLayout(orientation='vertical',spacing=5)

    txt1 = MTTextInput(group="g1",width=300,height=50)
    txt2 = MTTextInput(group="g1",password=True,width=300,height=50)
    txt3 = MTTextInput(group="g1",width=300,height=50)

    layout.add_widget(txt1)
    layout.add_widget(txt2)
    layout.add_widget(txt3)

    runTouchApp(layout)
```

Widgets

- Pop...

...up!

Widgets

```
from pymt import *


if __name__ == "__main__":

    layout = MTBoxLayout(orientation='vertical',spacing=5)

    popup = MTPopup(width=200,height=200,title="PyMT Popup!")

    layout.add_widget(popup)

    runTouchApp(layout)
```

Widgets

- Sliders
  - Slider
  - Circular Slider
  - XY Slider
  - Multi Slider
  - Boundary Slider

Widgets

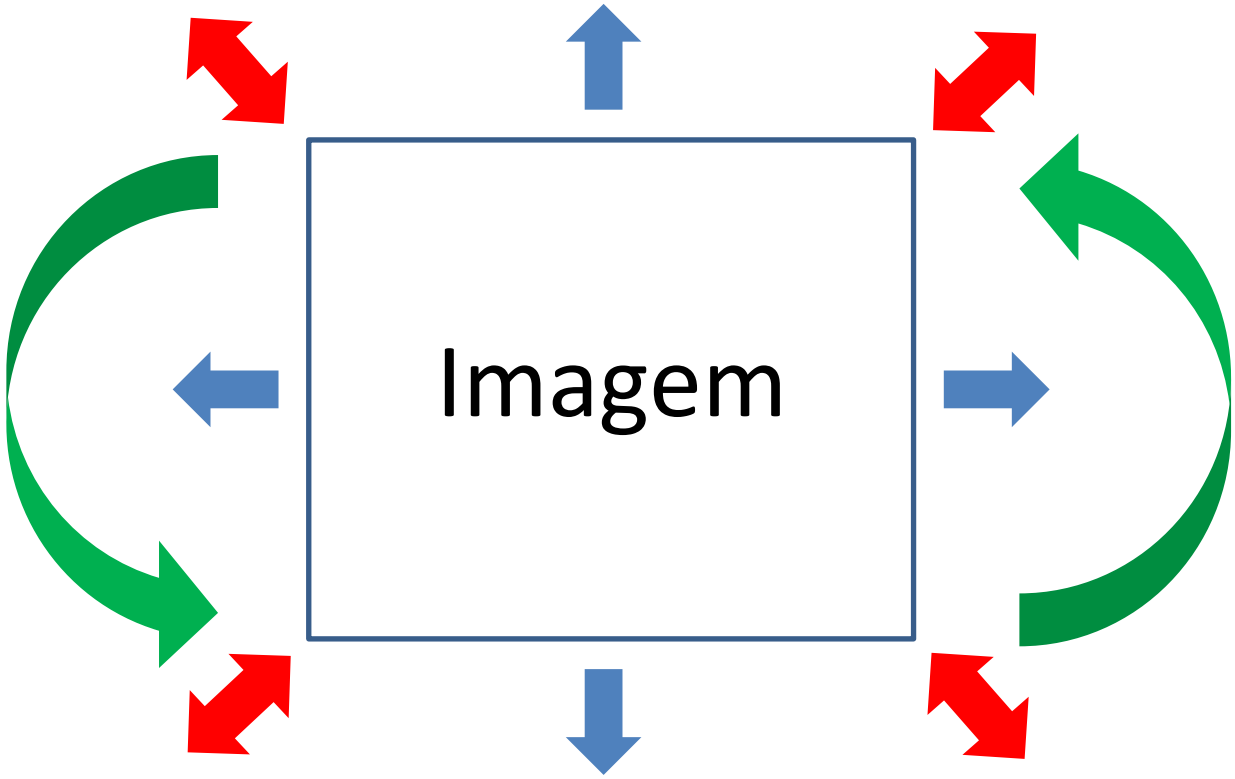
```
from pymt import *


if __name__ == "__main__":
    layout = MTBoxLayout(orientation='vertical',spacing=5)
    sl = MTSlider(value_show=True,value_format='%i')
    layout.add_widget(sl)
    runTouchApp(layout)
```

Widgets

- Scatters Widgets são Widgets que podem ser rotacionados, movidos ou terem seu tamanho alterado utilizando um ou dois dedos.
  - MTScatter - Base
  - MTScatterSvg
  - MTScatterPlane
  - MTScatterImage

Widgets

- MTScatterImage

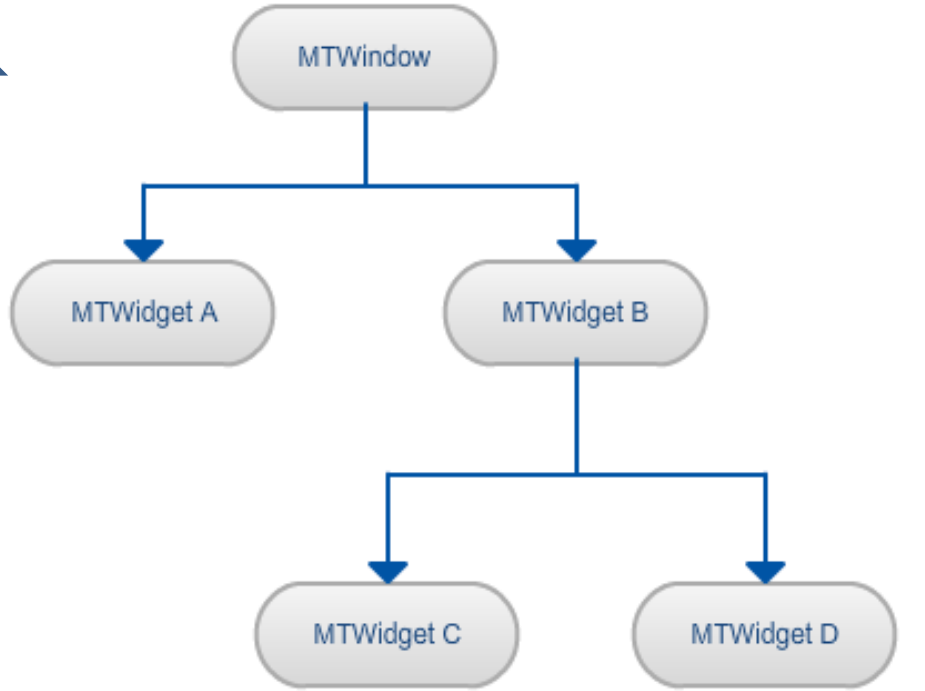


Widgets

```
from pymt import *


if __name__ == "__main__":

    image = MTScatterImage(filename='lena.jpg')

    getWindow().add_widget(image)

    runTouchApp()
```

Events

Os Widgets de uma aplicação PyMT são adicionados de forma BOTTOM-UP.

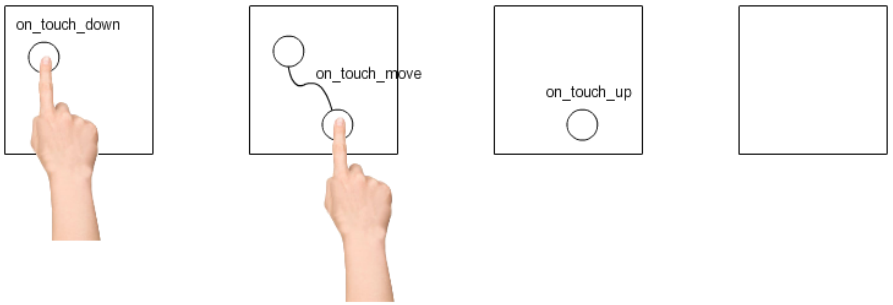


Os eventos dos Widgets no PyMT são disparados de forma TOP-DOWN.

Events

Event name	Arguments	Description	Dispatched in children
on_touch_down()	touch	Fired when a touch is down	Yes
on_touch_move()	touch	Fired when a touch is moving	Yes
on_touch_up()	touch	Fired when a touch is up	Yes
on_update()		Fired to update the widget	Yes
on_draw()		Fired to draw the widget	Yes
on_resize()	w, h	Fired when the size change (except at creation)	No
on_move	x, y	Fired when the position change (except at creation)	No
on_parent_move	x, y	Fired when the parent position change (except at creation)	Yes

MTWidget



Events

- Como maniupular um evento de um Widget

```
@<WidgetInstance>.event
def <event_name>(<event param>):
        ...
        return True //impede propagação de eventos
```

Events

```
from pymt import *

if __name__ == "__main__":

    button = MTButton(label="Event")

    @button.event
    def on_press(touch):
        print "Pressionado"
        return True

    runTouchApp(button)
```

Events

```
from pymt import *

if __name__ == "__main__":

    txt1 = MTTextInput(width=200,height=50)
    txt2 = MTTextInput(width=200,height=50,pos=(250,0))


    @txt1.event
    def on_text_change(text):
        txt2.value = text
        return True

    getWindow().add_widget(txt1)
    getWindow().add_widget(txt2)


    runTouchApp()
```

Animations

- Animations executam transformações nos widgets. Ex: Mover automaticamente um Widget.

Animations

```
from pymt import *

if __name__ == "__main__":

    button = MTButton(label="Disparar")

    animation = Animation(duration=5,pos=(100,100),style={'bg-
    color':(0.0,1.0,0.0,1.0)})

    @button.event
    def on_press(touch):
        #button.do(animation)
        animation.animate(button)
        return True

    @animation.event
    def on_complete(self):
        button.label = 'Finalizado'
        return True

    getWindow().add_widget(button)

    runTouchApp()
```

CSS

- CSS (Cascading Style Sheets)
- Define questões de layout (cores, fontes, etc..) dos widgets do PyMT
- Mesmo CSS utilizado em códigos de webpages

CSS

```
from pymt import *

if __name__ == "__main__":

    css_add_sheet('''
        .cssClass {
            bg-color: #FFFFFF;
            color: #111111;
        }

        #idVar1 {
            slider-color: #0000FF;
            slider-color-down: #FF0000;
        }

    ''')

    button = MTButton(label="CSS",cls="cssClass")

    slider = MTSlider(id="idVar1",x=150)

    getWindow().add_widget(button)
    getWindow().add_widget(slider)

    runTouchApp()
```

Exercício

- Crie uma aplicação PyMT que tenha as seguintes características/funcionalidades
  - Um campo (texto) para nome
  - Um campo (texto) para senha
  - Um campo (texto) para descrição
  - Deve ser possíveis navegar entre os campos de textos através da tecla TAB
  - Um botão REGISTRAR
  - Ao clicar em registrar, fazer a validação dos campos e em caso positivo, exibir um popup de boas vindas

Exercício

- Crie uma aplicação PyMT que tenha as seguintes características/funcionalidades
  - Uma foto ao centro da aplicação
  - Dois SLIDERS abaixo da foto
  - O primeiro slider deverá controlar o tamanho da foto
  - O segundo slider deverá controlar a rotação da foto

Showcases

Lab 64

Fim

Obrigado!

Antonio Dourado
antonio_dourado@dc.ufscar.br
http://www.dc.ufscar.br/windis