clutter mx Documentation

Release 1.0

2012, Jose Maria Garcia Perez

October 26, 2015

Contents

Contents:

CHAPTER 1

Introduction

The purpose is to have a small tutorial about the use of Clutter and Mx under Python3.

1.1 Clutter

Quotting the Clutter web page: “Clutter is an open source (LGPL 2.1) software library for creating fast, compelling, portable, and dynamic graphical user interfaces. It is a core part of MeeGo, and is supported by the open source community. Its development is sponsored by Intel.”

It uses OpenGL.

1.2 Mx

It is a toolkit developed on top of Clutter. Originally developped for Maemo/Moblin then merged into Moblin (now dead and evolved into Tizen), but as for now is an independent project.

1.3 Other technologies

It seems that this can be used with other technologies such as:

- Gtk
- Gstreamer
- Cairo

CHAPTER 2

Basic Usage

2.1 Clutter

In Clutter we work in a Stage where we would put a number of Actors. The typical Hello World example would be:

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter
import sys

if __name__ == '__main__':
    Clutter.init( sys.argv )

    # Create Stage
    _stage = Clutter.Stage()
    _stage.set_title( "Basic Usage" )
    _stage.set_size( 400, 200 )

    # Create Actor
    _red = Clutter.Color().new(255, 0, 0, 255) # R,G,B,alpha
    _actor = Clutter.Text().new_full(
                  "Mono 10",
                  "Hello World!",
                  _red )
    _actor.set_position( 100,100 )
    # Add Actor to the Stage
    _stage.add_actor( _actor )
    _stage.connect("destroy", lambda w: Clutter.main_quit() )
    _stage.show_all()

    Clutter.main()
```

where:

- Clutter is initialized.
- A Stage is created.
- An actor is created (just some formatted text) and its position is set.
- The actor is added to the stage.
- The signal “destroy” in the stage is connected to a function

being the result:

2.2 Mx

Mx is a toolbox based on Clutter. We could say it creates a number of actor that can be used in a Clutter stage:

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter, Mx
import sys

if __name__ == '__main__':
    Clutter.init( sys.argv )

    # Create Stage
    _stage = Clutter.Stage()
    _stage.set_title( "Basic Usage" )
    _stage.set_size( 400, 200 )

    # Create Actor
    _label = Mx.Label()
    _label.set_text("Hello world")

    # Add Actor to the Stage
    _stage.add_actor( _label )
    _stage.connect("destroy", lambda w: Clutter.main_quit() )
    _stage.show_all()

    Clutter.main()
```

where:

- Clutter is initialized.
- A Stage is created.

- An actor (a label from the Mx toolkit) is created.
- The actor is added to the stage.
- The signal “destroy” in the stage is connected to a function

being the result:

CHAPTER 3

Clutter: capturing key events

The following example shows how to capture key presses in Clutter. The example if heavily based on snippet from Nick Veitch:

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter
import sys

# define some colours (takes RGBA values)
red = Clutter.Color().new(255,0,0,255)
green = Clutter.Color().new(0,255,0,255)
blue = Clutter.Color().new(0,0,255,255)
black = Clutter.Color().new(0,0,0,255)
white = Clutter.Color().new(255,255,255,255)

# Define a callback method for key presses
def keyPress(self, event):
    # Parses the keyboard input generated by a callback from the Stage object
    # The event object has the property 'keyval'
    # clutter also defines constants for keyvals
    # so we can use these to compare
    if event.keyval == Clutter.q:
        #if the user pressed "q" quit the test
        Clutter.main_quit()
    elif event.keyval == Clutter.r:
        #if the user pressed "r" make the actor red
        text_actor.set_color(red)
        text_actor.set_text("This is Red")
    elif event.keyval == Clutter.g:
        #if the user pressed "g" make the actor green
        text_actor.set_color(green)
        text_actor.set_text("This is Green")
    elif event.keyval == Clutter.b:
        #if the user pressed "b" make the actor blue
        text_actor.set_color(blue)
        text_actor.set_text("This is Blue")
    elif event.keyval == Clutter.Up:
        # 'Up' is equal to the cursor up key
        stage.set_color(black)
    elif event.keyval == Clutter.Down:
        # 'Down' is equal to the cursor down key
        stage.set_color(white)
```

```
    #event feedback, useful for working out values of tricky keys
    print('event processed',  event.keyval)

if __name__ == '__main__':
   Clutter.init( sys.argv )

   # Stage creation
   stage = Clutter.Stage()
   stage.set_size(450, 180)
   stage.set_color(black)

   # Text Actor
   text_actor=Clutter.Text()
   text_actor.set_color(red)
   text_actor.set_position(20,70)
   text_actor.set_font_name('Sans 24')
   text_actor.set_text('Press a Key:\nUp, Down, r, g, b, q')

   # Add the actor to the stage
   stage.add_actor( text_actor )

   # Show the stage
   stage.show_all()

   # Connect signals
   stage.connect("destroy",lambda w: Clutter.main_quit() )
   stage.connect('key-press-event', keyPress)

   # Pass execution to the main clutter loop
   Clutter.main()
```

CHAPTER 4

Clutter: JSON

The GUI can be defined by mean of a JSON file, which is kind of low fat XML file.

4.1 ui.json

The description of the GUI is done by mean of the following file

```
[
  {
    "id" : "stage",
    "type" : "ClutterStage",
    "width" : 400,
    "height" : 400,
    "color" : "#333355ff",
    "title" : "Scripting example",
    "children" : [ "box" ],
    "signals" : [
      { "name" : "destroy", "handler" : "clutter_main_quit" }
    ]
  },

  {
    "id" : "box",
    "type" : "ClutterBox",
    "width" : 400,
    "height" : 400,

    "layout-manager" : {
      "type" : "ClutterBinLayout",
      "x-align" : "center",
      "y-align" : "center"
    },

    "children" : [
      {
        "id" : "rectangle",
        "type" : "ClutterRectangle",
        "width" : 200,
        "height" : 200,
        "color" : "red"
      }
    ]
```

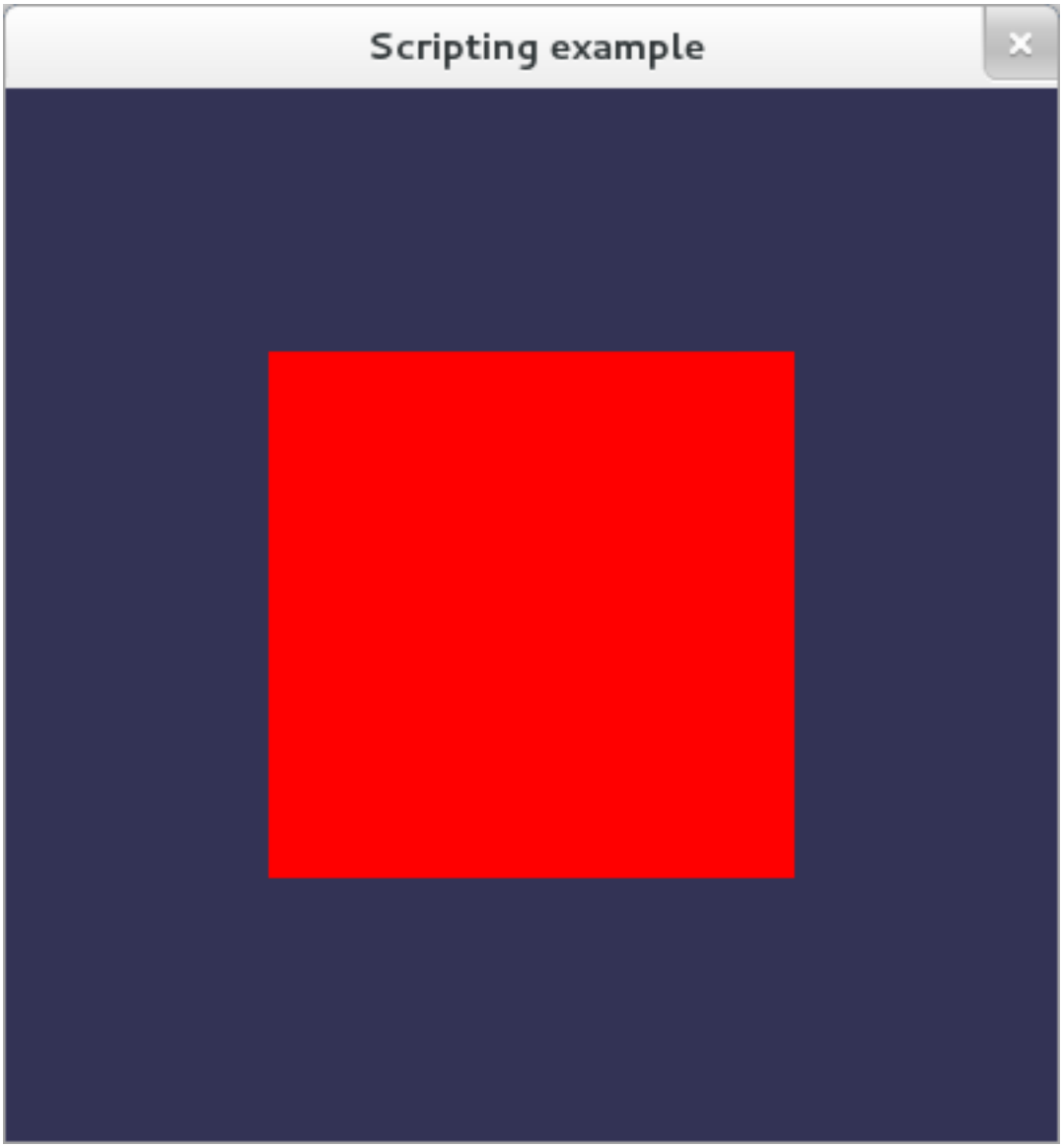
```
  }

]
```

4.2 main.py

The GUI is created from that file by mean of:

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter
import sys

if __name__ == '__main__':
    Clutter.init(sys.argv)

    _script = Clutter.Script()
    _script.load_from_file( "ui.json")
    stage = _script.get_object("stage")
    _script.connect_signals( stage )

    stage.show_all()
    Clutter.main()
```

The result looks like:


Scripting example

CHAPTER 5

Clutter: animations

5.1 Simple implicit animations

It is very easy to get something animated. For an actor we might use something like:

```
_actor_anim = _actor.animatev(
                  Clutter.AnimationMode.EASE_OUT_BOUNCE,
                  1500,
                  ["x"],
                  [20] )
```

where:

- EASE_OUT_BOUNCE: this is the effect of the animation
- 1500 represents how long will be animated in milliseconds.
- ["x"]: this is a list with the elements that will be animated.
- [20]: means that at the end of the animation, "x" will be equal to 20.

5.1.1 Full example

Using this base we could write:

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter
import sys

if __name__ == '__main__':
    Clutter.init( sys.argv )

    # Create Stage
    _stage = Clutter.Stage()
    _stage.set_title( "Basic Usage" )
    _stage.set_size( 400, 200 )

    # Create Actor
    _red = Clutter.Color().new(255, 0, 0, 255) # R,G,B,alpha
    _actor = Clutter.Text().new_full(
                  "Mono 10",
```

```
                    "Hello World!",
                    _red )

    _actor.set_position( 100,100 )
    _actor_anim = _actor.animatev(
                        Clutter.AnimationMode.EASE_OUT_BOUNCE,
                        1500,
                        ["x"],
                        [20] )

    # Add Actor to the Stage
    _stage.add_actor( _actor )
    _stage.connect("destroy", lambda w: Clutter.main_quit() )
    _stage.show_all()

    Clutter.main()
```

more information about Implicit animations.

5.2 Animation based on state

The following code is failing for some reason (any help is welcomed):

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter
import sys

def keyPress(self, event, _transitions):
    Clutter.State.set_state(_transitions, "move-down")

if __name__ == '__main__':
    Clutter.init( sys.argv )

    # Create Stage
    _stage = Clutter.Stage()
    _stage.set_title( "Animation using states" )
    _stage.set_size( 400, 400 )

    # Create Actor
    _red = Clutter.Color().new(255, 0, 0, 255) # R,G,B,alpha
    _actor = Clutter.Text().new_full(
                    "Mono 10",
                    "Press any key...",
                    _red )

    _actor.set_position( 100.0,100.0 )

    # Transition
    # - State creation
    _transitions = Clutter.State()



    # - Defines de behaviour of a number of actors
```

```
    _transitions.set_key( None, "move-down",  # source_state, target_state
          _actor, "x", Clutter.AnimationMode.EASE_OUT_CUBIC, 300,
          pre_delay=0.0, post_delay=0.0)

    # - All state transitions take 250ms
    _transitions.set_duration(None,None,3000)

    # Add Actor to the Stage
    _stage.add_actor( _actor )
    _stage.connect("destroy", lambda w: Clutter.main_quit() )
    _stage.connect('key-press-event', keyPress, _transitions)

    _stage.show_all()

    # Create animation


    Clutter.main()
```

Currently this code fails complaining of:

```
(anim_states.py:12376): Clutter-CRITICAL **: clutter_interval_set_final_value: assertion `G_VALUE_TYP
/usr/lib/python3.2/site-packages/gi/types.py:43: Warning: g_value_set_float: assertion `G_VALUE_HOLDS
  return info.invoke(*args, **kwargs)
/usr/lib/python3.2/site-packages/gi/types.py:43: Warning: g_value_get_float: assertion `G_VALUE_HOLDS
  return info.invoke(*args, **kwargs)
```

5.3 Animator

The reference documentation for the animator.

Not working for me yet:

```
#!/usr/bin/env python
#! -*- coding: utf-8 -*-

from gi.repository import Clutter
import sys

def keyPress(self, event, _anim):
    _anim.start()

if __name__ == '__main__':
    Clutter.init( sys.argv )

    # Create Stage
    _stage = Clutter.Stage()
    _stage.set_title( "Animation using states" )
    _stage.set_size( 400, 400 )
    _stage.set_reactive( True )

    # Create Actor
    _red = Clutter.Color().new(255, 0, 0, 255) # R,G,B,alpha
    _actor = Clutter.Text().new_full(
                  "Mono 10",
                  "Press any key...",
                  _red )
```

```
    _actor.set_position( 100.0,100.0 )

    # Transition
    # - State creation
    _anim = Clutter.Animator()
    _anim.set_duration( 4000 )
    #_anim.set_properties(
       # start of the animation
    #   _actor, "x", Clutter.AnimationMode.LINEAR, 0.0, 0.0,

       # half-way
    #   _actor, "x", Clutter.Animation.LINEAR, 0.5, 100.0,

       # end of animation
    #   _actor, "x", Clutter.Animation.LINEAR, 0.5, 100.0,
    #   )

    _anim.set_key( _actor, "x", Clutter.AnimationMode.LINEAR, 0.0, 50.0)

       # half-way
       #     _actor, "x", Clutter.Animation.LINEAR, 0.5, 100.0,

       # end of animation
       #    _actor, "x", Clutter.Animation.LINEAR, 0.5, 100.0,
       #    )
    # - Defines de behaviour of a number of actors

       #          pre_delay=0.0, post_delay=0.0)

    # - All state transitions take 250ms
    #    _transitions.set_duration(None,None,3000)

    # Add Actor to the Stage
    _stage.add_actor( _actor )
    _stage.connect("destroy", lambda w: Clutter.main_quit() )
    _stage.connect('key-press-event', keyPress, _anim)

    _stage.show_all()

    # Create animation


    Clutter.main()
```

CHAPTER 6

Resources

How everything fits together

6.1 Clutter

- Cookbook
- Reference Manual
- My old web
- Basic old tutorial
- Wiki
- Example
- Old but complete examples
- A little bit of COGL
- Old but well structured tutorial
- Similar source reference
- Rounded circle with COGL
- Container example
- Those who REALLY knows
- Texture

6.2 Mx

- Vala example

6.3 Others

- Gtk3

CHAPTER 7

Indices and tables

- genindex
- modindex
- search