Maemo multimedia applications - Part II

The article Maemo multimedia applications - Part I, describes how to use two important GStreamer tools (`gst-inspect` and `gst-launch`) when developing multimedia applications for the maemo platform. This article describes a simple maemo multimedia application (in Python).

The application is an MP3 player with a GTK+/Hildon interface. The application has 4 buttons (Play, Pause, Stop, and Open).

The constructor

```
#!/usr/bin/python
import pygst
pygst.require("0.10")
import gst
import pygtk
import gtk
import hildon

class Player(hildon.Program):
    def __init__(self):
 hildon.Program.__init__(self)
        self.window = hildon.Window()
        self.window.connect("delete_event", gtk.main_quit)
        self.window.set_title("Player")
        self.add_window(self.window)

 self.create_interface()
 self.create_pipeline()

 self.window.add(self.table)
 self.window.show_all()

    ... [see the next source code] ...

#creates a player
start=Player()
gtk.main()
```

The code above is divided into two main methods: `create_interface()`, which creates the GUI elements of the application such as buttons, and the `create_pipeline()`, which creates the GStreamer elements and pipeline.

Creating GUI elements

The method `create_interface()` creates a layout (gtk.Table) and inserts 4 buttons. Then, a callback function is connected to each inserted button. For example, the callback function `OnPlay()` is connected to the button `buttonPlay`. The callback functions

(C) Copyright Nokia 2014. All rights reserved.

modify the state of pipeline (READY, PLAYING, PAUSED and NULL), according to the action. The callback function onopen also creates a Hildon dialog to choose the file media to be played.

```
    #this method creates the GUI elements (layout and buttons)
    def create_interface(self):
self.table = gtk.Table(2,2,True)
buttonPlay = gtk.Button("Play")
buttonPause = gtk.Button("Pause")
buttonStop = gtk.Button("Stop")
buttonOpen = gtk.Button("Open")

buttonPlay.connect("clicked",self.OnPlay)
buttonPause.connect("clicked",self.OnPause)
buttonStop.connect("clicked",self.OnStop)
buttonOpen.connect("clicked",self.OnOpen)

self.table.attach(buttonPlay,0,1,0,1)
self.table.attach(buttonPause,1,2,0,1)
self.table.attach(buttonStop,0,1,1,2)
self.table.attach(buttonOpen,1,2,1,2)

    #sets GStreamer pipeline state to PLAYING
    def OnPlay(self, widget):
        self.pipeline.set_state(gst.STATE_PLAYING)

    #sets GStreamer pipeline state to PAUSED
    def OnPause(self, widget):
        self.pipeline.set_state(gst.STATE_PAUSED)

    #sets GStreamer pipeline state to NULL
    def OnStop(self, widget):
if self.pipeline.get_state() != gst.STATE_PAUSED:
 self.pipeline.set_state(gst.STATE_PAUSED)
if self.pipeline.get_state() != gst.STATE_NULL:
         self.pipeline.set_state(gst.STATE_NULL)

    #opens a dialog to choose the media file to play
    def OnOpen(self, widget):
dialog = hildon.FileChooserDialog(self.window,gtk.FILE_CHOOSER_ACTION_OPEN)
        response = dialog.run()
        dialog.hide()
        file_name = selector.get_filename()
        if response == gtk.RESPONSE_OK:
                self.audiosrc.set_property("location", file_name)
                self.pipeline.set_state(gst.STATE_READY)
```

Creating GStreamer elements and pipeline

It is necessary to create a GStreamer pipeline to insert the elements that are created. Then, we have to create two elements: one from a `gnomevfssrc`, and the other from `dspmp3sink`. These elements are inserted on the pipeline so they can be "executed". In order to check if the elements exist on your machine, use the `gst-inspect` command. This code cannot be executed on the desktop because the element dspmp3sink exists only for the Internet Tablet OS. Therefore, if you try to execute the code, the system throws an error (`gst.ElementNotFoundError: dspmp3sink`). Finally, link the elements inserted in the pipeline by using the `link()` method.

(C) Copyright Nokia 2014. All rights reserved.

```
#this method creates GStreamer pipeline and elements
    def create_pipeline(self):
        self.pipeline = gst.Pipeline("mypipeline")

        self.audiosrc = gst.element_factory_make("gnomevfssrc","source")
        self.pipeline.add(self.audiosrc)

        self.audiosink = gst.element_factory_make("dspmp3sink", "sink")
        self.pipeline.add(self.audiosink)

        self.audiosrc.link(self.audiosink)
```

See also

- Maemo multimedia applications - Part I
- Developing Python Applications for Maemo

(C) Copyright Nokia 2014. All rights reserved.