2. Playbin

Prev Next

2. Playbin

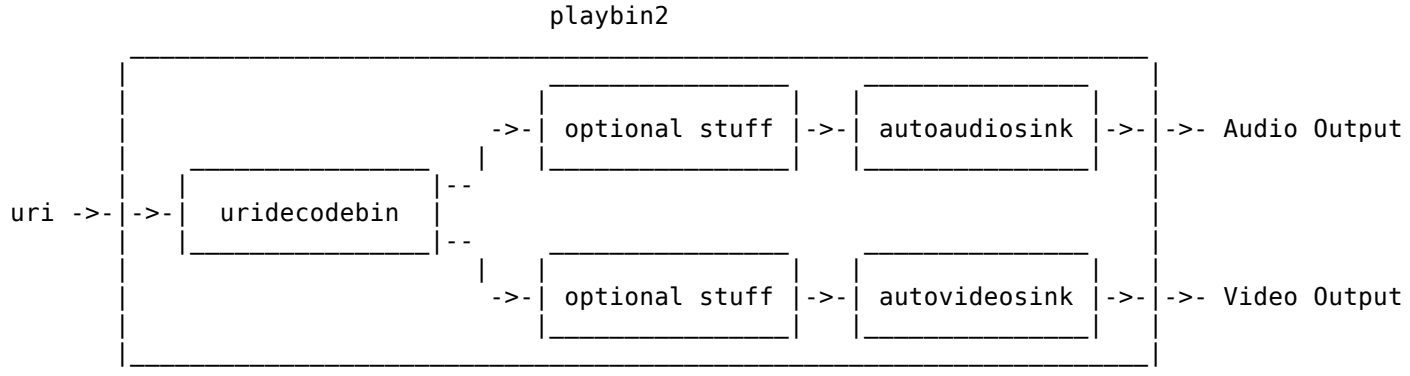
A playbin is a highlevel, automatic, audio and video player. You create a playbin object with:

```
my_playbin = gst.element_factory_make("playbin2", "my-playbin")
```

To get information about a playbin run:

```
$ gst-inspect-0.10 playbin2
```

This figure shows how playbin is built internally. The "optional stuff" are things that could be platform specific or things that you may set with properties.



The "uri" property should take any possible protocol supported by your Gstreamer plugins. One nice feature is that you may switch the sinks out for your own bins as shown below. Playbin always tries to set up the best possible pipeline for your specific environment so if you don't need any special features that are not implemented in playbin, it should in most cases just work "out of the box". Ok, time for a few examples.

This first example is just a simple audio player, insert a file with absolute path and it'll play.

Example 2.1

```
 1 #!/usr/bin/env python
 2
 3 import sys, os
 4 import pygtk, gtk, gobject
 5 import pygst
 6 pygst.require("0.10")
 7 import gst
 8
 9 class GTK_Main:
10
11         def __init__(self):
12                 window = gtk.Window(gtk.WINDOW_TOPLEVEL)
13                 window.set_title("Audio-Player")
14                 window.set_default_size(300, -1)
15                 window.connect("destroy", gtk.main_quit, "WM destroy")
16                 vbox = gtk.VBox()
17                 window.add(vbox)
18                 self.entry = gtk.Entry()
19                 vbox.pack_start(self.entry, False, True)
```

```
20                 self.button = gtk.Button("Start")
21                 self.button.connect("clicked", self.start_stop)
22                 vbox.add(self.button)
23                 window.show_all()
24
25                 self.player = gst.element_factory_make("playbin2", "player")
26                 fakesink = gst.element_factory_make("fakesink", "fakesink")
27                 self.player.set_property("video-sink", fakesink)
28                 bus = self.player.get_bus()
29                 bus.add_signal_watch()
30                 bus.connect("message", self.on_message)
31
32         def start_stop(self, w):
33                 if self.button.get_label() == "Start":
34                         filepath = self.entry.get_text()
35                         if os.path.isfile(filepath):
36                                 self.button.set_label("Stop")
37                                 self.player.set_property("uri", "file://" + filepath)
38                                 self.player.set_state(gst.STATE_PLAYING)
39                 else:
40                         self.player.set_state(gst.STATE_NULL)
41                         self.button.set_label("Start")
42
43         def on_message(self, bus, message):
44                 t = message.type
45                 if t == gst.MESSAGE_EOS:
46                         self.player.set_state(gst.STATE_NULL)
47                         self.button.set_label("Start")
48                 elif t == gst.MESSAGE_ERROR:
49                         self.player.set_state(gst.STATE_NULL)
50                         err, debug = message.parse_error()
51                         print "Error: %s" % err, debug
52                         self.button.set_label("Start")
53
54 GTK_Main()
55 gtk.gdk.threads_init()
56 gtk.main()
```

A playbin plugs both audio and video streams automagically so I've switched the videosink out to a fakesink element which is Gstreamer's answer to /dev/null. If you want to enable video playback just comment out the following lines:

```
fakesink = gst.element_factory_make("fakesink", "my-fakesink")
self.player.set_property("video-sink", fakesink)
```

If you want to show the video output in a specified window you'll have to use the enable_sync_message_emission() method on the bus. Here is an example with the video window embedded in the program.

Example 2.2

```
 1 #!/usr/bin/env python
 2
 3 import sys, os
 4 import pygtk, gtk, gobject
 5 import pygst
 6 pygst.require("0.10")
 7 import gst
 8
 9 class GTK_Main:
10
11         def __init__(self):
12                 window = gtk.Window(gtk.WINDOW_TOPLEVEL)
13                 window.set_title("Video-Player")
14                 window.set_default_size(500, 400)
15                 window.connect("destroy", gtk.main_quit, "WM destroy")
16                 vbox = gtk.VBox()
17                 window.add(vbox)
18                 hbox = gtk.HBox()
19                 vbox.pack_start(hbox, False)
20                 self.entry = gtk.Entry()
21                 hbox.add(self.entry)
22                 self.button = gtk.Button("Start")
```

```
23                 hbox.pack_start(self.button, False)
24                 self.button.connect("clicked", self.start_stop)
25                 self.movie_window = gtk.DrawingArea()
26                 vbox.add(self.movie_window)
27                 window.show_all()
28
29                 self.player = gst.element_factory_make("playbin2", "player")
30                 bus = self.player.get_bus()
31                 bus.add_signal_watch()
32                 bus.enable_sync_message_emission()
33                 bus.connect("message", self.on_message)
34                 bus.connect("sync-message::element", self.on_sync_message)
35
36         def start_stop(self, w):
37                 if self.button.get_label() == "Start":
38                         filepath = self.entry.get_text()
39                         if os.path.isfile(filepath):
40                                 self.button.set_label("Stop")
41                                 self.player.set_property("uri", "file://" + filepath)
42                                 self.player.set_state(gst.STATE_PLAYING)
43                 else:
44                         self.player.set_state(gst.STATE_NULL)
45                         self.button.set_label("Start")
46
47         def on_message(self, bus, message):
48                 t = message.type
49                 if t == gst.MESSAGE_EOS:
50                         self.player.set_state(gst.STATE_NULL)
51                         self.button.set_label("Start")
52                 elif t == gst.MESSAGE_ERROR:
53                         self.player.set_state(gst.STATE_NULL)
54                         err, debug = message.parse_error()
55                         print "Error: %s" % err, debug
56                         self.button.set_label("Start")
57
58         def on_sync_message(self, bus, message):
59                 if message.structure is None:
60                         return
61                 message_name = message.structure.get_name()
62                 if message_name == "prepare-xwindow-id":
63                         imagesink = message.src
64                         imagesink.set_property("force-aspect-ratio", True)
65                         gtk.gdk.threads_enter()
66                         imagesink.set_xwindow_id(self.movie_window.window.xid)
67                         gtk.gdk.threads_leave()
68
69 GTK_Main()
70 gtk.gdk.threads_init()
71 gtk.main()
```

And just to make things a little more complicated you can switch the playbins videosink to a gst.Bin with a gst.GhostPad on it. Here's an example with a timeoverlay.

```
bin = gst.Bin("my-bin")
timeoverlay = gst.element_factory_make("timeoverlay")
bin.add(timeoverlay)
pad = timeoverlay.get_pad("video_sink")
ghostpad = gst.GhostPad("sink", pad)
bin.add_pad(ghostpad)
videosink = gst.element_factory_make("autovideosink")
bin.add(videosink)
gst.element_link_many(timeoverlay, videosink)
self.player.set_property("video-sink", bin)
```

Add that code to the example above and you'll get a timeoverlay too. We'll talk more about ghostpads later.

On peoples requests we add a CLI example which plays music, just run it with:

```
$ python cliplayer.py /path/to/file1.mp3 /path/to/file2.ogg
```

Example 2.3

```
#!/usr/bin/env python

import sys, os, time, thread
import glib, gobject
import pygst
pygst.require("0.10")
import gst

class CLI_Main:

        def __init__(self):
                self.player = gst.element_factory_make("playbin2", "player")
                fakesink = gst.element_factory_make("fakesink", "fakesink")
                self.player.set_property("video-sink", fakesink)
                bus = self.player.get_bus()
                bus.add_signal_watch()
                bus.connect("message", self.on_message)

        def on_message(self, bus, message):
                t = message.type
                if t == gst.MESSAGE_EOS:
                        self.player.set_state(gst.STATE_NULL)
                        self.playmode = False
                elif t == gst.MESSAGE_ERROR:
                        self.player.set_state(gst.STATE_NULL)
                        err, debug = message.parse_error()
                        print "Error: %s" % err, debug
                        self.playmode = False

        def start(self):
                for filepath in sys.argv[1:]:
                        if os.path.isfile(filepath):
                                self.playmode = True
                                self.player.set_property("uri", "file://" + filepath)
                                self.player.set_state(gst.STATE_PLAYING)
                                while self.playmode:
                                        time.sleep(1)

                time.sleep(1)
                loop.quit()

mainclass = CLI_Main()
thread.start_new_thread(mainclass.start, ())
gobject.threads_init()
loop = glib.MainLoop()
loop.run()
```

A playbin implements a gst.Pipeline element and that's what the next chapter is going to tell you more about.

Prev Next
1. Introduction Home 3. Pipeline