6. Capabilities

Capabilities, gst.Caps, is a container where you may store information that you may pass on to a gst.PadTemplate. When you set the pipeline state to either playing or paused the elements pads negotiates what caps to use for the stream. Now the following pipeline works perfectly:

```
$ gst-launch-0.10  videotestsrc ! video/x-raw-yuv, width=320, height=240 ! xvimagesink
```

But if you try to switch out the xvimagesink for a ximagesink you will notice that it wouldn't work. That's because ximagesink can't handle video/x-raw-yuv so you must put in an element BEFORE in the pipeline that does.

```
$ gst-launch-0.10  videotestsrc ! video/x-raw-yuv, width=320, height=240 ! ffmpegcolorspace ! ximagesink
```

And as ximagesink does not support hardware scaling you have to throw in a videoscale element too if you want software scaling.

```
$ gst-launch-0.10  videotestsrc ! video/x-raw-yuv, width=320, height=240 ! videoscale ! ffmpegcolorspace ! ximagesink
```

To put the above examples in code you have to put the caps in a capsfilter element.

Example 6.1

```
 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("Videotestsrc-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.button = gtk.Button("Start")
19                 self.button.connect("clicked", self.start_stop)
20                 vbox.add(self.button)
21                 window.show_all()
22
23                 self.player = gst.Pipeline("player")
24                 source = gst.element_factory_make("videotestsrc", "video-source")
25                 sink = gst.element_factory_make("xvimagesink", "video-output")
26                 caps = gst.Caps("video/x-raw-yuv, width=320, height=230")
27                 filter = gst.element_factory_make("capsfilter", "filter")
28                 filter.set_property("caps", caps)
29
30                 self.player.add(source, filter, sink)
31                 gst.element_link_many(source, filter, sink)
32
33         def start_stop(self, w):
34                 if self.button.get_label() == "Start":
35                         self.button.set_label("Stop")
36                         self.player.set_state(gst.STATE_PLAYING)
37                 else:
38                         self.player.set_state(gst.STATE_NULL)
39                         self.button.set_label("Start")
40
41 GTK_Main()
42 gtk.gdk.threads_init()
43 gtk.main()
```

A frequently asked question is how to find out what resolution a file has and one way to do it is to check the caps on a decodebin element in paused state.

Example 6.2

```
 1 #!/usr/bin/env python
 2
 3 import sys, os, time
 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("Resolutionchecker")
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("Check")
21                 self.button.connect("clicked", self.start_stop)
22                 vbox.add(self.button)
23                 window.show_all()
24
25                 self.player = gst.Pipeline("player")
26                 source = gst.element_factory_make("filesrc", "file-source")
27                 decoder = gst.element_factory_make("decodebin", "decoder")
28                 decoder.connect("new-decoded-pad", self.decoder_callback)
29                 self.fakea = gst.element_factory_make("fakesink", "fakea")
30                 self.fakev = gst.element_factory_make("fakesink", "fakev")
31
32                 self.player.add(source, decoder, self.fakea, self.fakev)
33                 gst.element_link_many(source, decoder)
34
35                 bus = self.player.get_bus()
36                 bus.add_signal_watch()
37                 bus.connect("message", self.on_message)
38
39         def start_stop(self, w):
40                 filepath = self.entry.get_text()
41                 if os.path.isfile(filepath):
42                         self.player.set_state(gst.STATE_NULL)
43                         self.player.get_by_name("file-source").set_property("location", filepath)
44                         self.player.set_state(gst.STATE_PAUSED)
45
46         def on_message(self, bus, message):
47                 t = message.type
48                 if t == gst.MESSAGE_STATE_CHANGED:
49                         if message.parse_state_changed()[1] == gst.STATE_PAUSED:
50                                 for i in self.player.get_by_name("decoder").src_pads():
51                                         structure_name = i.get_caps()[0].get_name()
52                                         if structure_name.startswith("video") and len(str(i.get_caps()[0]["width"])) < 6:
53                                                 print "Width:%s, Height:%s" %(i.get_caps()[0]["width"], i.get_caps()[0]["height"])
54                                                 self.player.set_state(gst.STATE_NULL)
55                                                 break
56                 elif t == gst.MESSAGE_ERROR:
57                         err, debug = message.parse_error()
58                         print "Error: %s" % err, debug
59                         self.player.set_state(gst.STATE_NULL)
60
61         def decoder_callback(self, decoder, pad, data):
62                 structure_name = pad.get_caps()[0].get_name()
63                 if structure_name.startswith("video"):
64                         fv_pad = self.fakev.get_pad("sink")
65                         pad.link(fv_pad)
66                 elif structure_name.startswith("audio"):
67                         fa_pad = self.fakea.get_pad("sink")
68                         pad.link(fa_pad)
69
70 GTK_Main()
71 gtk.gdk.threads_init()
72 gtk.main()
```

Note: The examples here in this tutorial have grown a bit lately and it's not easy to show working gstreamer stuff in not so many lines but I'll try as hard as I can to do just that. Maybe we have to have the code in separate files that you may take a look at if you find anything interesting. The examples share much code too and when the numbers of chapters beefs up so does the readers knowledge and they can make sense out of just a few lines of code instead of runable code examples. Well, time will tell how far we gets.

In this example we will use the playbin from example 2.2 and switch its video-sink out for our own homemade bin, stuffed with some goodies. Now, let's say that you run a tv-station and you want to have your logo in the top right corner of the screen. For that you can use a textoverlay but for the fonts to be the exact same size on the screen no matter what kind of resolution the source has you have to specify a width so everything is scaled according to that.

Example 6.3

```
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("playbin", "player")
30                 self.bin = gst.Bin("my-bin")
31                 videoscale = gst.element_factory_make("videoscale")
32                 videoscale.set_property("method", 1)
33                 pad = videoscale.get_pad("sink")
34                 ghostpad = gst.GhostPad("sink", pad)
```

```
35                 self.bin.add_pad(ghostpad)
36                 caps = gst.Caps("video/x-raw-yuv, width=720")
37                 filter = gst.element_factory_make("capsfilter", "filter")
38                 filter.set_property("caps", caps)
39                 textoverlay = gst.element_factory_make('textoverlay')
40                 textoverlay.set_property("text", "GNUTV")
41                 textoverlay.set_property("font-desc", "normal 14")
42                 textoverlay.set_property("halign", "right")
43                 textoverlay.set_property("valign", "top")
44                 conv = gst.element_factory_make ("ffmpegcolorspace", "conv")
45                 videosink = gst.element_factory_make("autovideosink")
46
47                 self.bin.add(videoscale, filter, textoverlay, conv, videosink)
48                 gst.element_link_many(videoscale, filter, textoverlay, conv, videosink)
49                 self.player.set_property("video-sink", self.bin)
50
51                 bus = self.player.get_bus()
52                 bus.add_signal_watch()
53                 bus.enable_sync_message_emission()
54                 bus.connect("message", self.on_message)
55                 bus.connect("sync-message::element", self.on_sync_message)
56
57         def start_stop(self, w):
58                 if self.button.get_label() == "Start":
59                         filepath = self.entry.get_text()
60                         if os.path.exists(filepath):
61                                 self.button.set_label("Stop")
62                                 self.player.set_property("uri", "file://" + filepath)
63                                 self.player.set_state(gst.STATE_PLAYING)
64                 else:
65                         self.player.set_state(gst.STATE_NULL)
66                         self.button.set_label("Start")
67
68         def on_message(self, bus, message):
69                 t = message.type
70                 if t == gst.MESSAGE_EOS:
71                         self.player.set_state(gst.STATE_NULL)
72                         self.button.set_label("Start")
73                 elif t == gst.MESSAGE_ERROR:
74                         self.player.set_state(gst.STATE_NULL)
75                         self.button.set_label("Start")
76                         err, debug = message.parse_error()
77                         print "Error: %s" % err, debug
78
79         def on_sync_message(self, bus, message):
80                 if message.structure is None:
81                         return
82                 message_name = message.structure.get_name()
83                 if message_name == "prepare-xwindow-id":
84                         imagesink = message.src
85                         imagesink.set_property("force-aspect-ratio", True)
86                         imagesink.set_xwindow_id(self.movie_window.window.xid)
87
88 GTK_Main()
89 gtk.gdk.threads_init()
90 gtk.main()
```

Prev 5. Seeking | Home | Next 7. Videomixer