# How to play video in Qt on MeeGo using GStreamer

This article explains how to embed a video played with GStreamer inside a Qt app. The article applies only in X11 based system such as Maemo and MeeGo.

11 Sep 2011

## Introduction

With this article I want to show you how to create a small and basic video player using Qt and GStreamer. Even if GStreamer can run on Symbian, this article works only in X11 based system such as Maemo and MeeGo. That's why we are going to use gstxoverlay.

## What's GstXOverlay

By default GStreamer reproduces videos inside a new window. By the way this solution is not so much flexible and application developers don't like that, especially if they want to catch mouse events. XOverlay is a Gstreamer interface that allows the developer to embed a video inside an application. This operation is quite simple and to achieve that XOverlay just needs to know the WinID of the widget where the video should be displayed.

## The gstreamer engine class

Even if XOverlay is an important part of the code, to create a basic video player we need something more. We need an engine which manage GStreamer and that we can call from our Qt application. That's engine is here shown. Basically this class create a playbin2 in the constructors along with a videosink which instead of create a new window, overlays thw video on the widget a specific WinID. As it's possible to see from the code, the class provides several getters and setters methods useful to control the playback and query information about the source played.

```
#include "gstengine.h"
#include <QDebug>
#include <gst/interfaces/xoverlay.h>
#include <QWidget>

GstEngine::GstEngine(QObject *parent) :
    QObject(parent)
{
    gst_init(NULL, NULL);

    mPlayer = gst_element_factory_make("playbin2", "play");
    GstElement *videosink = gst_element_factory_make("xvimagesink", "videosink");
    g_object_set(G_OBJECT(mPlayer), "video-sink", videosink, NULL);

    //gst_bus_set_sync_handler(gst_pipeline_get_bus(GST_PIPELINE(m_players[id].player)), bus_cb, &m_players[id]);
    //gst_bus_add_watch (gst_pipeline_get_bus (GST_PIPELINE(m_players[id].player)), message_cb, NULL);
}

GstEngine::~GstEngine()
{
    gst_object_unref(GST_OBJECT (mPlayer));
}

void GstEngine::setUrl(const QUrl &url)
{
    g_object_set(G_OBJECT (mPlayer), "uri", qPrintable(url.toString()), NULL);
    gst_element_set_state (mPlayer, GST_STATE_PAUSED);
}

void GstEngine::play()
{
   gst_element_set_state(mPlayer, GST_STATE_PLAYING);
}

void GstEngine::stop()
{
    gst_element_set_state(mPlayer, GST_STATE_NULL);
}

void GstEngine::pause()
{
    gst_element_set_state (mPlayer, GST_STATE_PAUSED);
}

bool GstEngine::isPlaying() const
{
    GstState state;
    gst_element_get_state (mPlayer, &state, NULL, -1 );
    return state == GST_STATE_PLAYING;
}

bool GstEngine::isPaused() const
{
    GstState state;
    gst_element_get_state (mPlayer, &state, NULL, -1 );
    return state == GST_STATE_PAUSED;
}


bool GstEngine::seek(uint ms)
{
    gst_element_get_state(mPlayer, NULL, NULL, 100*GST_MSECOND);
    if (!gst_element_seek (mPlayer, 1.0, GST_FORMAT_TIME, GST_SEEK_FLAG_FLUSH, GST_SEEK_TYPE_SET, ms*GST_MSECOND, GST_SEEK_TYPE_NONE
    {
      qWarning("Can't seek");
      return false;
    }
    return true;
}
```


(C) Copyright Nokia 2013. All rights reserved.

```
uint GstEngine::position() const {

    GstFormat fmt = GST_FORMAT_TIME;
    gint64 value = 0;
    gst_element_query_position( mPlayer, &fmt, &value );
    return static_cast<uint>( ( value / GST_MSECOND ) );
}
uint GstEngine::duration() const {
    GstFormat fmt = GST_FORMAT_TIME;
    gint64 value = 0;
    gst_element_query_duration( mPlayer, &fmt, &value );
    return static_cast<uint>( ( value / GST_MSECOND ) );
}
void GstEngine::setVolume(int percent)
{
    g_object_set(G_OBJECT(mPlayer), "volume", percent*0.01, NULL );
}

int GstEngine::volume()
{
    double volume;
    g_object_get(G_OBJECT(mPlayer), "volume", &volume, NULL );
    return volume * 100;
}
bool GstEngine::setVideoWidget(QWidget *w)
{
    GstElement *videosink = 0;
    g_object_get(mPlayer, "video-sink", &videosink, NULL);
    if (videosink && GST_IS_X_OVERLAY(videosink) )
    {
        gst_x_overlay_set_xwindow_id(GST_X_OVERLAY(videosink), w->winId() );
        return true;
    }
    return false;
}
```

## Conclusion

This article covers all the basic aspects that a Qt application developer need to know about creating a video player with GStreamer. With just a simple class your Qt application can play files, http streams and more. There is no limit to things that GStreamer or Qt can do.


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.