How to record audio in Qt using GStreamer

This article explains how to record audio captured by microphone to mp3 using GStreamer.

Introduction

Qt Multimedia offers a convenient API to record audio. While this fulfils most use cases, this API does not allow us to record an http stream, or and audio stream coming from video, and it does not allow us to save it in a particular format. GStreamer allows us to achieve these goals without great effort, and works in both Maemo and MeeGo.

Code

The application which I'm going to show you in this article, is a Qt application which contains GStreamer API. Target of this application is record a stream coming from the microphone, convert it using lame to mp3 using a bitrate of 192 and finally saving it in a file. This can be achieved with GStreamer by using a pipeline like the following one:

```
gst-launch autoaudiosrc ! audioconvert ! lame bitrate=192 ! filesink location=./ciao.mp3
```

That pipeline can be seen converted to C in the snippet below. As we can see the filename for the filesink is set later in the code using "g_object_set". To make the application more useful I added a playback pipeline to play the audio file saved. The implementation is pretty simple to understand and does not require further explanation.

```
audioMgr::audioMgr() {
    gst_init (NULL,NULL);
    // Record pipeline
    // autoaudiosrc ! audioconvert ! lame bitrate=192 ! filesink location=./ciao.mp3
    recordBin = gst_pipeline_new("record-pipeline");
    g_assert(recordBin);
    audioSrc = gst_element_factory_make("autoaudiosrc", "audio_in");
    g_assert(audioSrc);
    audioConvert = gst_element_factory_make("audioconvert", "audio_converter");
    g_assert(audioConvert);
    lameEncoder = gst_element_factory_make("lame", "lame_encoder");
    g_assert(audioConvert);
    fileSink = gst_element_factory_make("filesink", "file_sink");
    g_assert(fileSink);
    gst_bin_add_many(GST_BIN(recordBin), audioSrc, audioConvert, lameEncoder, fileSink, NULL);
    gst_element_link_many(audioSrc, audioConvert, lameEncoder, fileSink, NULL);
    // Playback
    playbackBin = gst_element_factory_make("playbin2", "playbin");
    g_assert(playbackBin);
}
audioMgr::~audioMgr() {
    gst_element_set_state(GST_ELEMENT(playbackBin), GST_STATE_NULL);
    gst_object_unref(GST_OBJECT(playbackBin));
    gst_element_set_state(GST_ELEMENT(recordBin), GST_STATE_NULL);
    gst_object_unref(GST_OBJECT(recordBin));
}

void audioMgr::record(const QString& filename, const int bitrate) {
    qDebug() << "RECORD";
    gst_element_set_state(GST_ELEMENT(playbackBin), GST_STATE_READY);
    gst_element_set_state(GST_ELEMENT(recordBin), GST_STATE_NULL);
    g_object_set(G_OBJECT(fileSink), "location", qPrintable(filename), NULL);
    g_object_set(G_OBJECT(lameEncoder), "bitrate", bitrate, NULL);
    gst_element_set_state(GST_ELEMENT(recordBin), GST_STATE_PLAYING);
}

void audioMgr::stop() {
    qDebug() << "STOP";
    gst_element_set_state(GST_ELEMENT(playbackBin), GST_STATE_READY);
    gst_element_set_state(GST_ELEMENT(recordBin), GST_STATE_READY);
}

void audioMgr::play(const QString& uri) {
    qDebug() << "PLAY";
    g_object_set(G_OBJECT(playbackBin), "uri", qPrintable(uri), NULL);
    gst_element_set_state(GST_ELEMENT(recordBin), GST_STATE_READY);
    gst_element_set_state(GST_ELEMENT(playbackBin), GST_STATE_PLAYING);
}
```

The complete code, tested on the Nokia N900 can be download from the link on the right side of this page.

Conclusion

GStreamer API is a very powerful API which permits developer to achieve things which cannot be obtained by just using Qt Mobility Multimedia API. Just changing the audiosink this code can be used to record any other audio source available on the device. To know more about audio sources available you can run gst-inspect, a command line tool belonging to the GStreamer tool set.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.