ADT_GStreamer API
Manipulación de dispositivos de captura de vídeo mediante GStreamer en C++

Mario Chririnos Colunga
Áurea - Desarrollo Tecnológico

30 de noviembre de 2010

Índice

1. Introducción **1**

2. ADT_GStreamer API **2**
2.1. Requisitos 2
2.2. Listado de dispositivos 2
2.3. Conexión y desconexión 4
2.4. Manipulación del buffer de datos 4
2.5. Adjuncion de objetos 4

3. Ejemplo **5**
3.1. Requisitos 5

4. Notas **5**

1. Introducción

Gstreamer es un marco conceptual multimedia multiplataforma escrito en C, que permite construir una gran variedad de dispositivos para manejar audio y vídeo, como reproductores multimedia, editores de audio y vídeo y filtros digitales. En ciertos aspectos GStreamer es a Linux lo que DirectX es a Windows. Para facilitar el manejo de dispositivos de vídeo utilizando GStreamer en Linux creamos la interfaz de aplicación o *API* **ADT_GStreamer** la cual es descrita en este documento. ADT_GStreamer permite 1) Listar todos los dispositivos de captura de vídeo compatibles con *video for Linux 2* (**V4L2**), que estén conectados al sistema; 2) Visualizar en vivo el vídeo proveniente de un dispositivo de captura; 3) Acceder y manipular el buffer de datos del vídeo y 4) Capturar imágenes.

Este documento está estructurado de la siguiente manera: en la sección §2 describe el funcionamiento de nuestra código, la sección §3 describe detalles relacionados al ejemplo incluido con el código fuente y la sección §4 proporciona notas y comentarios finales sobre este documento.

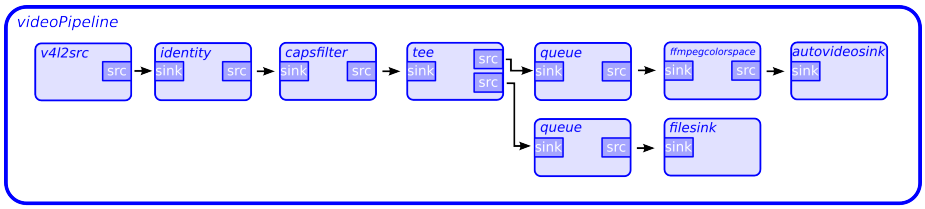

Figura 1: Estructura de la linea de flujo en ADT_GStreamer.

2. ADT_GStreamer API

Gstreamer es un marco conceptual implementado como una línea de flujo o *pipleline*, los datos de audio o vídeo fluyen a través de la tubería pasando por componentes insertables o *plugins* que pueden ser distribuidos a lo largo de la linea de flujo permitiendo crear una gran variedad de aplicaciones. GStreamer provee mas de 150 componentes diferentes de audio y vídeo, como dispositivos de origen, multiplexores y demultiplexores o filtros de datos. Información detallada sobre GStreamer puede ser encontrada en el manual de desarrollo [3].

GStreamer se distribuye bajo licencia LGPL la cual permite crear ejecutables que sean ligados estáticamente a la biblioteca siempre y cuando se distribuya el código binario del programa (en código objeto) para el uso del cliente únicamente, de forma que este si modifica la biblioteca GStreamer pueda crear un nuevo ejecutable del programa. También se debe notificar que el programa usa la biblioteca y que la biblioteca está sujeta a las condiciones de la licencia [1].

La interfaz de aplicación aquí descrita crea una línea de flujo, figura 1, para obtener vídeo de dispositivos compatibles con *video for linux 2* (**V4L2**) y visualizarlo. La línea de flujo también permite acceder al buffer de vídeo y modificarlo.

2.1. Requisitos

Para compilar código de esta API se requiere contar con el paquete **libgstreamer0.10-dev**,**libgstreamer-plugins-base0.10-dev** y agregar al compilador C++ las banderas obtenidas mediante: `pkg-config --cflags gstreamer-0.10 gstreamer-plugins-base-0.10`, `pkg-config --libs gstreamer-0.10 gstreamer-plugins-base-0.10` y `-lgstinterfaces-0.10`.

2.2. Listado de dispositivos

Para enlistar los dispositivos de vídeo conectados al sistema se utiliza la función miembro `int enumCapDev(void)` la cual devuelve el numero de dispositivos encontrados. Una vez enlistados los dispositivos de vídeo se puede acceder a la lista de dispositivos con las siguientes funciones miembro:

- `int getDevListSize(void)` permite obtener el número de dispositivos de video listados.
- `int getResListSize(unsigned int i)` permite obtener el número de resoluciones disponibles para el dispositivo i.

```
#ifndef ADTGSTREAMER_H
#define ADTGSTREAMER_H
//-------------------------------------------------------------------------------------
#include <gst/gst.h>
#include <gtk/gtk.h>
#ifdef GDK_WINDOWING_X11
        #include <gdk/gdkx.h>  // for GDK_WINDOW_XID
#endif
#include <gst/interfaces/xoverlay.h>
#include <string>
#include <vector>
#include "ADT_DataTypes.h"
//-------------------------------------------------------------------------------------
using namespace std;
typedef void(*pt2Function)(unsigned char*, unsigned int, unsigned int, unsigned int, void*);
//-------------------------------------------------------------------------------------
class ADT_GstVideo
{
 private:
        int selectedDev;
        int selectedRes;
        unsigned int width;
        unsigned int height;
        pt2Function fpt;
        vector<string> devList;          //Device list
        vector<string> devNameList;      //Device name list
        vector< vector<ADT_Point2D_ui> > resList;//Resolution list for each device

        //video window
        unsigned long int video_window_xid;
        GtkWidget* videoWindow;
        GtkWidget* drawArea;

        //main pipeline
        GstElement* videoSrc;
        GstElement* filter;
        GstElement* filterCaps;
        GstElement* tee;
        GstElement* videoQueue;
        GstElement* colorSpaceConverter;
        GstElement* videoSink;
        GstElement* videoPipeline;

        int enumFormats();
        int createMainPipeline();
        vector<ADT_Point2D_ui> getResList(GstElement* videoSrcTmp) const;
        vector<ADT_Point2D_ui> getResList(const char* devName) const;

        //callback functions
        static int videoplayer_bus_callback(GstBus* bus,
                                            GstMessage* message, void* data);
        static int filter_handoff_callback(GstElement* filter,
                                           GstBuffer* buffer, void* user_data);
        static void video_widget_realize_cb(GtkWidget* widget, void* user_data);
        static GstBusSyncReply bus_sync_handler_cb(GstBus* bus,
                                                   GstMessage* message, void* user_data);
        static void hideanddisconnect_cb(GtkObject* object, void* user_data);
        static void hide_cb(GtkObject* object, void* user_data);

 public:
        void* userdata;
        string name;

        int enumCapDev();
        int connect(const char* devName, unsigned int _width, unsigned int _height);
        int connect(unsigned int devIndex, unsigned int resIndex);
        void disconnect() const;

        void setFilter(pt2Function  _ftp);
        void unsetFilter();
        int startFilter();
        int stopFilter();

        unsigned int getDevListSize() const;
        unsigned int getResListSize(unsigned int i) const;
        int getSelectedDev() const;
        int getSelectedRes() const;
        const char* getDev(unsigned int i) const;
        const char* getDevName(unsigned int i) const;
        const ADT_Point2D_ui getRes(unsigned int d, unsigned int i) const;
        ADT_GstVideo();
};
#endif
```

Figura 2: ADT_GStreamer.h

- `const char *getDev(unsigned int i)` regresa la dirección del dispositivo i, e.g. */dev/video0*.
- `const char *getDevName(unsigned int i)` regresa el nombre del dispositivo i.
- `const ADT_Point2D_ui getRes(unsigned int d, unsigned int i)` regresa la resolución i del dispositivo d.

2.3. Conexión y desconexión

Dos funciones permiten la conexión de un dispositivo de vídeo. La función miembro `connect(unsigned int devIndex, unsigned int resIndex)` permite conectarse al dispositivo de vídeo $devIndex$ usando la resolución $resIndex$. Los parámetros $devIndex$ y $resIndex$ indican la posición en la lista de dispositivos y resoluciones disponibles respectivamente, los cuales se generan al ejecutar `int enumCapDev(void)`. La función miembro `connect(const char*, unsigned int, unsigned)` permite indicar la dirección del dispositivo de vídeo y la resolución a la cual se desea conectar el dispositivo.

Para desconectar el dispositivo se utiliza la función `void disconnect()`.

2.4. Manipulación del buffer de datos

Nuestra API también permite acceder y modificar el bloque de memoria en donde se encuentra la imagen obtenida por el dispositivo de vídeo. Para poder manipular este bloque de memoria primero se debe de crear una función con la firma `void filtro(unsigned char *data, unsigned int width, unsigned int height, unsigned int nChannels, void* userdata)`, en donde el puntero *data* apunta al bloque de memoria, *width*, *height*, *nChannels* son las dimensiones de la imagen y *userdata* apunta a **this**. Esta función será ejecutada cada vez que el dispositivo de vídeo adquiere una nueva imagen, en ella se puede manipular el bloque de memoria. Las funciones que permiten controlar el acceso a la nueva función son:

- `int setFilter(pt2Function)` modifica el puntero *fpt* con la dirección de la función en el parámetro de entrada.
- `int unsetFilter()` fpt = NULL.
- `int startFilter()` Conecta la señal *"signal-handoffs"* del dispositivo de vídeo a la función especificada por setFilter. Esta señal es emitida cuando el dispositivo de vídeo entrega el buffer de memoria.
- `int stopFilter()` Desconecta la señal *"signal-handoffs"*.

2.5. Adjuncion de objetos

El puntero `void* userdata` permite ajuntar un objeto genérico al objeto `ADT_GstVideo`. Este puntero fue añadido a la clase pues ha resultado útil al trabajar en aplicaciones multicámara.

3. Ejemplo

Junto con el código de esta API se provee de un programa ejemplo para demostrar su funcionamiento. El programa ejemplo consiste en una interfaz gráfica en GTK+ y Glade con la cual se accesa a las funciones del API. Para mayor información acerca de la programación con GTK+ se puede consultar [2].

3.1. Requisitos

Para compilar el programa ejemplo se requiere contar con el paquete **libgtk2.0-dev**. Las banderas de compilación se especifican en el archivo *makefile* de este ejemplo. La interfaz gráfica se puede editar con el entrono de diseño Glade el cual es el paquete **glade-gnome**.

4. Notas

El código fuente de esta API puede ser descargado en nuestro sitio web, en donde también se pueden reportar errores en el código fuente. Para reportar errores en este documento favor de escribir a errata@aurea-dt.com.

En las siguientes versiones de esta API se agregara la captura de imagen y vídeo.

Referencias

[1] Andrew St. Laurent. *Understanding Open Source and Free Software Licensing*. O'Reilly Media, Inc., 16 August 2004.

[2] Tony Gale, Ian Main, and the GTK team. GTK+ 2.0 Tutorial.

[3] Wim Taymans, Steve Baker, Andy Wingo, Ronald S. Bultje, and Stefan Kost. *GStreamer Application Development Manual*. gstreamer.freedesktop.org, (0.10.28.1) edition.