PERCCOM
asmus undus

LULEÅ TEKNISKA UNIVERSITET

M7017E Multimedia Systems

Lab 1 – Capturing and Playing Audio/Video

By Baptiste Louis, Martin Bumba and Fisayo Sangogboye, November 2014.

Table of Contents

INTRODUCTION

This report is about LAB1 of the course M7017E Multimedia Systems, proposed by Karl Anderson for Luleå University of Technology (LUT).

The objective of this Lab is capturing and playing audio/video by using the GStreamer Framework [1]. To achieve this, an audio capture tool has been developed with an embedded audio player.

The development team comprised of three international students. Two team members, Baptiste Louis and Fisayo Sangogboye are from France and Nigeria respectively and are undergoing an Erasmus Mundus PERCCOM program. The third member, Martin Bumba is from Czech Republic and he is undergoing a master program in distributing systems and web development.

All program codes were written in JAVA using NetBeans IDE 8.0 on Windows 8.1 systems.

This project was a great experience.

NOTE

In software development, it is important to document your work. Also, it is a skill to know how to write so that other people can understand and potentially reuse your. This report follow the recommendation describe by Karl Anderson and Christer Wahlberg, LTU – Luleå (Sweden), Nov-14.

PROBLEM SPECIFICATION

Description of the problem and sub-problems

The expectations for this LAB are to implement an application with a graphic user interface that allows a user to record an audio clip, to listen it, to save it and to play it back. GStreamer has to be use.

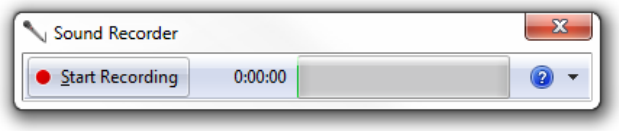


Figure 1 - Sound Recorder application in windows 7.

Coming from these requirements, we had to learn more about GStreamer, understand how it works and how we can use it in order to design an architecture of our application. Also, we had to learn how to well document our code in order to be clearly understandable and reusable.

We had to choose first the coding language to develop the application, an Integrated Development Environment (IDE) and its associated Operating System (OS), according to some compatibility issues of GStreamer Framework.

Figure 2 - Netbeans IDE.

Figure 3 - GStreamer framework.

We had two weeks to realise, document and provide our solution in order to stay on schedule for following labs and the coming seminar work. Additionally, our background in application development and coding was different but we worked in team with the objective to learn more from each other even if this choice can affect our productivity for the project.

We are here to learn and it should be beneficial for our future.

Declaration of assumption

We assumed that an audio recorder is a familiar tool even for a basic user. I remember when I was just a kid to use an audio capture toy which whom I could record my voice and play it back. Therefore, we decided to skip text buttons and to implement icon buttons in our interface. Also, we assumed that mp3 is the main used audio format nowadays so our application save and play mp3 files.

Note: it is possible that ACC audio format would become more and more used because it has better quality and compression capacity than mp3.

Presentation of extra features implemented

Remember, the basics features are to record an audio clip, to listen it, to save it and to play it back.

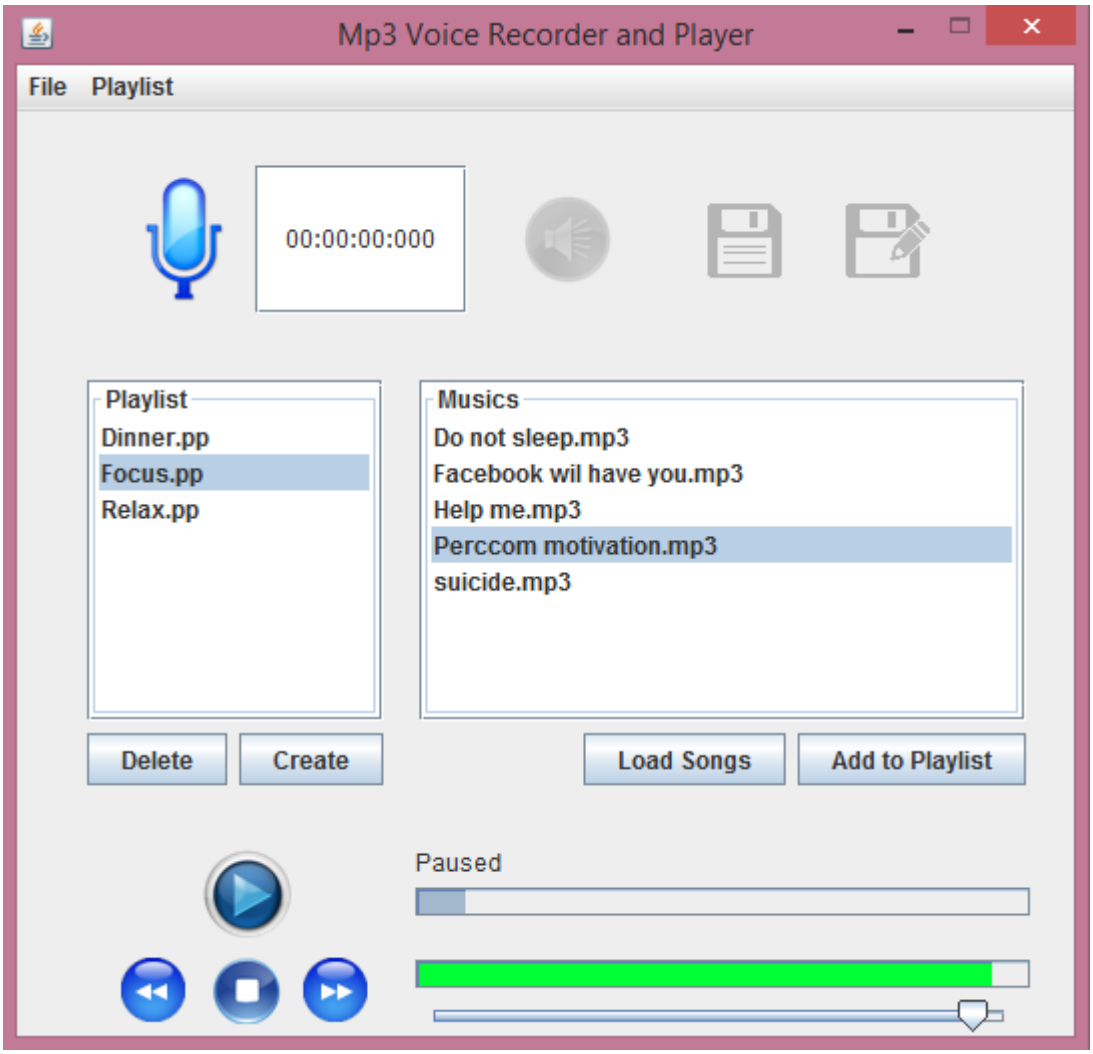


Figure 4 - Ergonomic easy-to-use bug-free GUI.

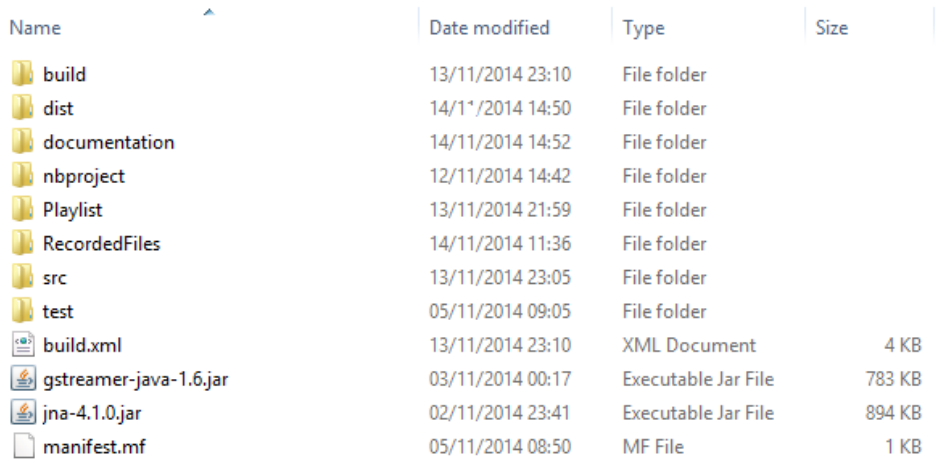
We thought that the following extra features improve the application and the user experience:

- SAVE AS… functionality additionally to the generic SAVE possibility:
  - SAVE will save the recorder film with a generic name *yyyy-mm-dd_hh-mm-ss.mp3* in the default folder *RecordedFiles* created by the application itself.
  - SAVE AS… will open a browser windows enabling the user to choose the place where to save the file and with the desired name.
- Load some songs to listen your last crazy mp3 music files.
- A playlist manager:
  - Create some playlist which will be saved automatically on the default folder *Playlist* created by the application itself.
  - Delete a playlist.
- A volume control slider.
- A playback progression bar.
- A beautiful-easy-to-use-bug-free-friendly interface.

FILES, FOLDERS AND DOCUMENTATION GUIDE

Remember, our application have been developed in JAVA on Windows 8.1 with NetBeans IDE 8.0.

We were advised to develop in C++ instead of JAVA to avoid some potential issue during our coding part. However, our skills in JAVA are much more developed than for C++. We decided to develop our knowledge in Java rather than learning a new programming language leading us to a slower development phase. This section present the folders & files structure of our application.



Name	Date modified	Type	Size
build	13/11/2014 23:10	File folder	
dist	14/1*/2014 14:50	File folder	
documentation	14/11/2014 14:52	File folder	
nbproject	12/11/2014 14:42	File folder	
Playlist	13/11/2014 21:59	File folder	
RecordedFiles	14/11/2014 11:36	File folder	
src	13/11/2014 23:05	File folder	
test	05/11/2014 09:05	File folder	
build.xml	13/11/2014 23:10	XML Document	4 KB
gstreamer-java-1.6.jar	03/11/2014 00:17	Executable Jar File	783 KB
jna-4.1.0.jar	02/11/2014 23:41	Executable Jar File	894 KB
manifest.mf	05/11/2014 08:50	MF File	1 KB

Figure 5 - Structure of the application folder

Netbeans folders:

- *src:* folder where you will find the source code and images used for the GUI;
- *build:* the sources files are compiled into this folder;
- *dist:* the JAR file containing the project built and related libraries are stored in this folder;
- *nbproject:* this folder contains the project's metadata;
- *test:* this folder contains eventually some Unit Tests for the project.

Specific application folders:

- *Playlist:* this folder stores the playlist created by the user;
- *RecordedFile:* this folder is the default folder to save recorded files;
- *Documentation:* this folder contains the documentation for this project.

Documentation files:

- *M7017E_report.pdf:* this document is the report of the whole audio capture Lab1 project.
- *Javadoc_MyAudioCapture:* shortcut to the Javadoc created for MyAudioCapture application.

All the code is available on GitHub https://github.com/Udacity2048/M7017E_LAB1 for open purpose. A .jar file is provided but an error occurs during the execution on windows ("gstreamer not found") despite all libraries are linked with the jar file. Instead, please use Netbeans to open our application for the moment. Do not hesitate to contact us at baplou-4@student.ltu.se .

SYSTEMS DESCRIPTION

Internal design and structure of the application

The application is composed of 4 packages and 7 classes. This architectural choice allows the user to develop his own graphic user interface by using the API documentation provided and the icons package. Moreover, the application' core is separated from the main to protect the operation of the application when the user want to link is new GUI.

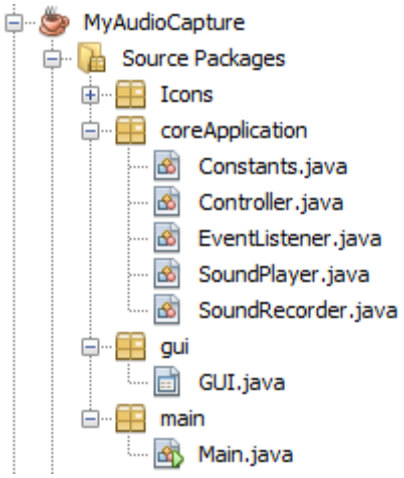


Figure 6 - Classes' architecture in MyAudioCapture project.

Here a short and clear description of each packages:

- *Icons:* gather all the icons layers for a nicer GUI;
- *coreApplication:* gather all classes making our application working, including the middleware;
- *gui:* gather our own ergonomic and easy to use GUI;
- *main:* gather the main class which have to be executed.

Here a short and clear description of each classes:

- *Constants:* gather all the Constants variables used by different part of our application;
- *Controller:* middleware of our application;
- *EventListener:* listener of events from the player, useful for the GUI implementation;
- *SoundPlayer:* GStreamer pipeline implemented a sound player;
- *SoundRecorder:* GStreamer pipeline implemented a sound recorder;
- *GUI:* Graphical User Interface;
- *Main:* main class which launch our application with the GUI.

The details of each classes, their contractors, their methods, are available on the Javadoc provided within the project folder, under *documentation* folder.

Architectural logic

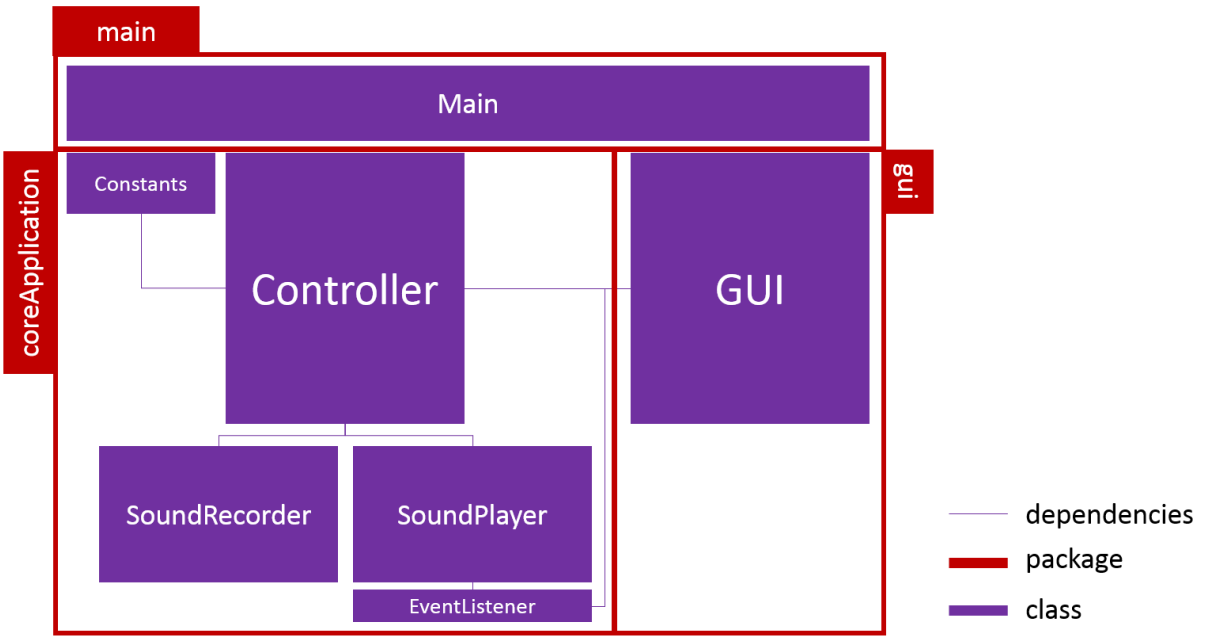


Figure 7 – Classes and packages dependencies.

The classes and packages dependencies have been thought to enable a clear development, to facilitate the reusability of our solution and to bring high scalability.

Controller handle the two pipelines for respectively play and record a sound. Then, Constants gather all the important static variables used in our project. Also, GUI is loosely coupled with the coreApplication so that if an end user wants to develop his own GUI, he just needs to call methods from Controller. Additionally, he can use EventListener to dynamically react to a specific event from the soundPlayer like the end of a song. Finally, the Main launches the Controller and the GUI and the application is running.

ALGORITHM DESCRIPTION

For this lab which is the first lab in a row of 3, the goal was mainly to approach the GStreamer Framework with the pipeline abstraction [2]. Consequently, there is no complex algorithms used which would require pseudo-code explanation. Also, the code is well commented and an API documentation is provided for the whole application [3].

However, let's take a look in our pipeline enabling the audio capture functionality.

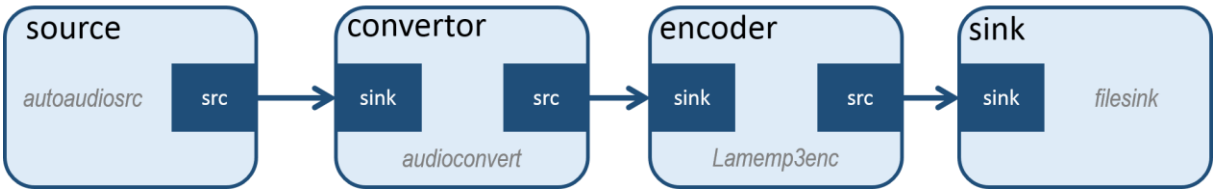


Figure 8 - Audio capture pipeline created with GStreamer framework.

- *autoaudiosrc*: automatically detect the audio source, i.e. the microphone [4].
- *audioconvert*: converts raw audio to supported audio format for encoding [5].
- *Lamemp3enc*: high quality free MP3 encoder [6].
- *filesink*: write the audio stream to a file [7].

Concerning the player part, we did not have time to implement our own pipeline so we used the pre-built pipeline "playbin2" [8].

PROBLEMS AND REFLECTIONS

GStreamer is an interesting framework for developing media-handling components. However, it is too often (and it was the case) that you spend a lot of time to understand and to be able to use a new tool. So, we read information on different web site, including [9] [1] [10]. Through tutorials available at [11], we saw that it is possible to code a simple audio or video player with no more than 20 lines of code, user graphic interface included. These examples helped use to understand the pipeline abstraction and to imagine our audio capture pipeline.

The graphic user interface (**GUI**) is often a gulf in the time spend for a project. Our choice has been to use NetBeans drag-and-drop jFrame to obtain a nicer GUI. However, implement all functionalities behind buttons, text fields and area lists was kind of challenging. Also, it was thinking intensive to provide a bug free interface to the end user, i.e. that the interface will work without freezing when the user click on that button in that situation by mistake.

```
if (!musicsArea.isSelectionEmpty()) {
    if (controller.playerIsPlaying()) {
        controller.playerPause();
        playButton.setPressedIcon(new ImageIcon(getClass().getResource("/Icons/001d.png")));
        playButton.setIcon(new ImageIcon(getClass().getResource("/Icons/001.png")));
        playButton.setRolloverIcon(new ImageIcon(getClass().getResource("/Icons/001g.png")));
        lblDuration.setText("Paused");
    } else if (controller.playerIsPaused()) {
        controller.playerPlay();
        playButton.setPressedIcon(new ImageIcon(getClass().getResource("/Icons/icPaused.png")));
        playButton.setIcon(new ImageIcon(getClass().getResource("/Icons/icPause.png")));
        playButton.setRolloverIcon(new ImageIcon(getClass().getResource("/Icons/icPauseg.png")));
        lblDuration.setText("Playing");
    } else {
        String path = listMView.get(musicsArea.getSelectedValue().toString());
        controller.playerPlayFile(path);
        playButton.setPressedIcon(new ImageIcon(getClass().getResource("/Icons/icPaused.png")));
        playButton.setIcon(new ImageIcon(getClass().getResource("/Icons/icPause.png")));
        playButton.setRolloverIcon(new ImageIcon(getClass().getResource("/Icons/icPauseg.png")));
```

Figure 9 - Example of complex thinking in the management of icon buttons.

GStreamer **configuration** on windows environment for JAVA development was kind of problematic. The basic version of GStreamer does not include plugin like the ugly plugging [12] which contains the MP3 encoder element for our pipeline' architecture. Then, GStreamer v1.0 seems not working (Date 11/11/2014) with JAVA and GStreamer v0.10 has an python issue on Windows. Also, in order to work with JAVA, you have to download and add libraries gstreamer-java-1.6.jar and jna-4.1.0.jar. Lastly, it seems to have some issues with JDK x64 so GStreamer and JDK have to be x86 version.

Go to `C:\Program Files\OSSBuild\GStreamer\v0.10.6\sdk\bindings\python\v2.7\lib`

and copy

`gstreamer-0.10` and `site-packages` to

`C:\Program Files\OSSBuild\GStreamer\v0.10.6\lib`, replacing any existing files.

Then delete the file `C:\Program Files\OSSBuild\GStreamer\v0.10.6\lib\gstreamer-0.10\libgstpython-v2.6.dll` (or change the extension to `.dllx` or something).

Figure 10 - Example of manipulation to skirt the python issue.

CONTRIBUTION

This first LAB1 project has been realized by a team of three international students working as a team on all aspect of the project and each decisions were approved by every members. However, we can identify some specific responsibilities for each person.

Baptiste Louis was mainly responsible for handling the project, defining the architecture, defining extra functionalities and implementing some of them, reviewing the code/comments, and producing the documentations including this report.

Martin Bumba was mainly responsible for developing the audio capture pipeline, the audio player pipeline and the controller which has been evolving with the implementation of extra functionalities.

Fisayo Sangogboye was mainly responsible for developing a beautiful graphic user interface and integrating some of extra functionalities to improve our application in the time we had.

ACKNOWLEDGEMENT

We, Baptiste LOUIS Martin BUMBA and Fisayo SANGOGBOYE, state that our project is compliant with the Academic ethics / Code of honour defined in the Study Guidance of the course M7017E Multimedia Systems, Luleå University of Technology, Semester1 Quarter2 2014.

Figures

References

[1] gstreamer, "Home page," 8 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/.

[2] gstreamer, “gstreamer documentation,” 8 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/section-intro-basics-bins.html.

[3] B. Louis, “API documentation,” Netbeans, Lulea, 2014.

[4] GTK-Doc, "autoaudiosrc," 11 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-good-plugins/html/gst-plugins-good-plugins-autoaudiosrc.html.

[5] GTK-Doc, “audioconvert,” 11 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-base-plugins/html/gst-plugins-base-plugins-audioconvert.html.

[6] GTK-Doc, “Lamemp3enc,” 11 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-ugly-plugins/html/gst-plugins-ugly-plugins-lamemp3enc.html.

[7] GTK-Doc, "filesink," 11 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer-plugins/html/gstreamer-plugins-filesink.html.

[8] GTK-Doc, "playbin2," 11 November 2014. [Online]. Available: http://www.freedesktop.org/software/gstreamer-sdk/data/docs/latest/gst-plugins-base-plugins-0.10/gst-plugins-base-plugins-playbin2.html.

[9] Wikipedia, "GStreamer," 8 November 2014. [Online]. Available: http://en.wikipedia.org/wiki/GStreamer.

[10] Google Project Hosting, "Project Home page," 8 November 2014. [Online]. Available: https://code.google.com/p/gstreamer-java/.

[11] Google Project Hosting, “gstreamer-java tutorials,” 18 April 2011. [Online]. Available: https://code.google.com/p/gstreamer-java/wiki/Tutorials. [Accessed 23 November 2014].

[12] GTK-Doc, "Ugly Plugins," 11 November 2014. [Online]. Available: http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-ugly-plugins/html/.