|  |
| --- |
| **Institute of Technology Blanchardstown** |
| StreaMe |
| Live Streaming Cross-Platform Broadcasting Client |
|  |
| **Benoit Brayer, Nans Plancher, Romaric Delaunoy** |
|  |
| **2012-2013** |

|  |
| --- |
|  |

Submitted in part fulfilment for the degree of

**B.Sc. Computing**

School of Informatics and Engineering,

Institute of Technology Blanchardstown,

Dublin, Ireland

# Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Degree of **B.Sc. in Computer Science** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Signed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Dated: \_\_\_\_/\_\_\_\_\_/\_\_\_\_\_

# Abstract

Regarding to the third year program of the bachelor in computing science in information technology, we had to handle an entire project by ourselves during the year. Most of them were proposed by the institute of technology of Blanchardstown, but it was also possible to choose our own project.

In our case, we already had an idea of what we wanted, and we knew that this idea was able to be realised in the due time: one year. We also knew that our idea was useful and able to live by itself in the real world.

After stating the main project target and objectives talking to each other, we planned to look for a teacher who was able to trust and support our ambition. After a while, Dr Luke Raeside believed in our motivation, and we have been happy to count him as our project supervisor. Thanks to him, the StreaMe project has been born.

# Contents

[Declaration 1](#_Toc355090580)

[Abstract 2](#_Toc355090581)

[Contents 3](#_Toc355090582)

[Illustration Table 6](#_Toc355090583)

[I. Chapter 1: Introduction 7](#_Toc355090584)

[I.1 Project Introduction 7](#_Toc355090585)

[I.2 Objectives 7](#_Toc355090586)

[I.3 Project Design 7](#_Toc355090587)

[I.4 Technologies 7](#_Toc355090588)

[II. Chapter 2: Literature Review 9](#_Toc355090589)

[III. Chapter 3: System Analysis 10](#_Toc355090590)

[III.1 Overview 10](#_Toc355090591)

[III.1.1 Uses case 11](#_Toc355090592)

[III.2 Functional requirements 12](#_Toc355090593)

[III.2.1 Store the project 12](#_Toc355090594)

[III.2.2 Configure the streaming platform and parameters 12](#_Toc355090595)

[III.2.3 Add media sources into the project 12](#_Toc355090596)

[III.2.4 Send the stream to a server 12](#_Toc355090597)

[III.2.5 Display stream in the software 12](#_Toc355090598)

[III.2.6 Display feedback of the streaming 12](#_Toc355090599)

[III.3 Use Cases 13](#_Toc355090600)

[III.3.1 Create a new project 13](#_Toc355090601)

[III.3.2 Load the source available 13](#_Toc355090602)

[III.3.3 Add a source into the project 14](#_Toc355090603)

[III.3.4 Remove a source from the project 14](#_Toc355090604)

[III.3.5 Choose a broadcasting platform 15](#_Toc355090605)

[III.3.6 Configure the streaming parameters 16](#_Toc355090606)

[III.3.7 Save the project 17](#_Toc355090607)

[III.3.8 Save the project as 18](#_Toc355090608)

[III.3.9 Open a project 18](#_Toc355090609)

[III.3.10 Rename a project 19](#_Toc355090610)

[III.3.11 Start streaming 20](#_Toc355090611)

[III.3.12 Stop streaming 20](#_Toc355090612)

[III.3.13 Streaming display 21](#_Toc355090613)

[III.3.14 Streaming feedback 21](#_Toc355090614)

[IV. Chapter 4: System Design 22](#_Toc355090615)

[IV.1 User Interface Design 22](#_Toc355090616)

[IV.1.1 Main Window 22](#_Toc355090617)

[IV.1.2 Starting Assistant 23](#_Toc355090618)

[IV.1.3 New Project 24](#_Toc355090619)

[IV.1.4 Streaming Parameters 24](#_Toc355090620)

[IV.1.5 Advanced Parameters 25](#_Toc355090621)

[IV.1.6 Rename 25](#_Toc355090622)

[IV.2 Functional Design 26](#_Toc355090623)

[IV.2.1 Starting assistant 26](#_Toc355090624)

[IV.2.2 New Project Assistant 26](#_Toc355090625)

[IV.2.3 Streaming Parameters 27](#_Toc355090626)

[IV.2.4 Advanced Parameters 27](#_Toc355090627)

[IV.2.5 Rename 27](#_Toc355090628)

[IV.3 Classes Design 28](#_Toc355090629)

[IV.3.1 Design Pattern MVC 28](#_Toc355090630)

[IV.3.2 Overall Class Diagram 29](#_Toc355090631)

[IV.3.3 Class Diagram 30](#_Toc355090632)

[V. Chapter 5: Implementation 31](#_Toc355090633)

[V.1 Languages, Libraries and Tools 31](#_Toc355090634)

[V.1.1 C++ 31](#_Toc355090635)

[V.1.2 QT 32](#_Toc355090636)

[V.1.3 Qt Creator 32](#_Toc355090637)

[V.1.4 Phonon 32](#_Toc355090638)

[V.1.5 FFMPEG 33](#_Toc355090639)

[V.1.6 Broadcasting Platforms 33](#_Toc355090640)

[V.2 Prototype Implementation 34](#_Toc355090641)

[V.2.1 User Interface 34](#_Toc355090642)

[V.2.2 Project File 34](#_Toc355090643)

[V.2.3 Video Capture 36](#_Toc355090644)

[V.2.4 Video Broadcast 38](#_Toc355090645)

[V.2.5 Video Display 38](#_Toc355090646)

[V.3 Installer on Windows and Linux 39](#_Toc355090647)

[V.3.1 Windows 39](#_Toc355090648)

[V.3.2 Linux 39](#_Toc355090649)

[V.4 Others ??? 39](#_Toc355090650)

[VI. Chapter 6: Testing and Evaluation 40](#_Toc355090651)

[VII. Chapter 7: Conclusion and Further Work 41](#_Toc355090652)

[VII.1 Further Work 41](#_Toc355090653)

[VII.2 Conclusion 41](#_Toc355090654)

# Illustration Table

[Figure III‑1 Use Case Diagram 11](#_Toc355090721)

[Figure 2 Sequence Diagram Create a new project 13](#_Toc355090722)

[Figure 3 Sequence Diagram Create a new project 13](#_Toc355090723)

[Figure 4 Sequence Diagram Add a media source 14](#_Toc355090724)

[Figure 5 Sequence Remove a source 14](#_Toc355090725)

[Figure 6 Sequence Diagram Choose a broadcasting platform 15](#_Toc355090726)

[Figure 7 Sequence Diagram Configure the streaming parameters 16](#_Toc355090727)

[Figure 8 Sequence Diagram Save the project 17](#_Toc355090728)

[Figure 9 Sequence Diagram Save the project as 18](#_Toc355090729)

[Figure 10 Sequence Diagram Open a project 18](#_Toc355090730)

[Figure 11 Sequence Diagram Rename a project 19](#_Toc355090731)

[Figure 12 Sequence Diagram Start the streaming 20](#_Toc355090732)

[Figure 13 Sequence Diagram Stop the streaming 20](#_Toc355090733)

[Figure 14 Sequence Diagram Streaming display 21](#_Toc355090734)

[Figure 15 Sequence Diagram Streaming feedback 21](#_Toc355090735)

[Figure 16 Main widow draft 22](#_Toc355090736)

[Figure 17 Starting assistant window draft 23](#_Toc355090737)

[Figure 18 New project assistant window draft 24](#_Toc355090738)

[Figure 19 Streaming parameters window draft 24](#_Toc355090739)

[Figure 20 Advanced parameters window draft 25](#_Toc355090740)

[Figure 21 Rename window draft 25](#_Toc355090741)

[Figure 22 Interaction Diagram Starting Assistant 26](#_Toc355090742)

[Figure 23 Interaction Diagram New Project Assistant 26](#_Toc355090743)

[Figure 24 Interaction Diagram Streaming Parameters 27](#_Toc355090744)

[Figure 25 Interaction Diagram Advanced Parameters 27](#_Toc355090745)

[Figure 26 Interaction Diagram Rename 27](#_Toc355090746)

[Figure 27 StreaMe Class Diagram 30](#_Toc355090747)

[Figure 28 Classification of computing language usage 31](#_Toc355090748)

[Figure 29 QT logo 32](#_Toc355090749)

[Figure 30 UStream.tv logo 33](#_Toc355090750)

[Figure 31 Logo Justin.tv Figure 32 Logo Twitch.tv 33](#_Toc355090751)

[Figure 33 Example of project file 35](#_Toc355090752)

# Chapter 1: Introduction

This grouped project will be introduced in four parts. First of all, it will be explained in a general way: what is it, for whom it is for, how it works. Next, we’ll explain the objectives of this software, and how it’s designed. By the end, we'll explain which technologies we choose.

## Project Introduction

The idea we had was to permit people to be shown and broadcast to the world without paying a high price material and services. Concretely, we wanted to allow people to capture themselves playing music, sharing experiences, teaching lessons, playing video games, commenting news, etc. and broadcast it to a large public that doesn't have to pay for watching them.

Starting from this main objective, we analysed that the device that was the friendliest for this feature was the personal computer. Indeed, the power of these machines is enough to manage a software that handles this functionality. The user needs also an internet connection to be connected to a free third party service that will broadcast the audio video stream captured by his computer to the entire world.

Finally, we wanted to build this project free and open. First of all, we decided to realize it with the GPL Licence and consequently make it become an open source software. Second of all, we decided to do not restrict this software to a single operating system. Therefore, this software, that we called StreaMe, is cross-platform, and is able to be executed on Microsoft Windows 7 and Ubuntu 12.10 (Linux).

## Objectives

The main objective of StreaMe is to be open and easy. It doesn't need a huge configuration, just to select which sources the user wants to use, to select the quality of the streaming, and to select the hosting platform. By clicking on the start button, the user is displayed into multiple screens all over the world using the internet. The second objective of StreaMe is to be accessible on Windows and Ubuntu with regarding installers (.exe installer on Windows, .deb package on Ubuntu).

## Project Design

StreaMe is a windowed software and most of its functionalities are grouped in a single main window to keep features easy to access. Therefore, the user is able to add and remove an input source to the project directly from the main window. He's also able to start and stop the streaming from this main window. The video streamed is displayed in a caption into this main window, and the user can choose to listen or not the streamed sound.

These functions are the most important, but some other features are accessible from a top menu bar that contains more specific features. From this menu, the user can configure the streaming parameters, create/rename/save/save in a specific place a project. All these configuration options are displayed into “popup” windows.

## Technologies

StreaMe is a software that uses a command line application already created: FFmpeg. FFmpeg is “a solution to record, convert and stream audio and video” in command line. Using FFmpeg, we don't have to reinvent the wheel. Therefore, StreaMe might be described as a simpler interface between the user and FFmpeg which is hard to use.

To make StreaMe cross-platform, we previously thought about java or python as the programming language. However, for performance and compatibility reasons with FFmpeg, C++ was the best solution. Therefore, we designed our code with the MVC design pattern to separate the model, the view and the controller, in the objective to make the software easy to manage on several operating systems. Moreover, we choose Qt as the Graphical User Interface, which is fortunately cross-platform.

Finally, we had to define how to manage the broadcasting platform: the one that receive the stream from the user and share it to anyone. After a research, we selected two free and open platforms: Ustream and Justin.tv. The user only needs to register at least onto one of them to use StreaMe. Therefore, if a third party watcher wants to see the user streaming itself, he just has to connect to the regarding web page: an integrated player displays the broadcaster.

# Chapter 2: Literature Review

# Chapter 3: System Analysis

## Overview

The system must be capable of capturing video from camera such as a webcam and audio from a microphone of a computer and send a video and audio stream to a broadcasting platform to play it in live.

In order to address these issues six key elements were defined:

1. Create a project

The application must be capable to assist the user for the creation of a new project. Also the application must give the possibility to save and load configurations to the user by using project files.

1. Configure parameters

The possibility of fully configure the audio and video quality, and the size of frames before sending the stream must be given to the user but an simplified configuration must be available for users based on the user internet broadband speed.

1. Add media sources into the project

Adding video and audio source must be easy and intuitive.

1. Send the stream to a server

Sending audio and video to the broadcasting platform must be automatically available when the platform parameters are filled in.

1. Display stream in the software

A player must be present into the interface to show the content which is currently streaming.

1. Display feedback of the streaming

Feedback on the streaming must be given to the user into the interface.

### Uses case

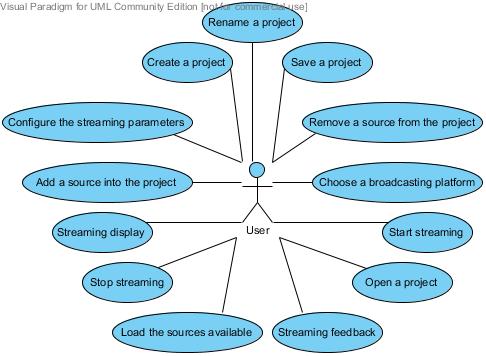


Figure III‑1 Use Case Diagram

## Functional requirements

### Store the project

1. Ensure the application can access the repertory where the user wants to save the project.
2. Ensure the project file is not corrupted when the application have to load it.

### Configure the streaming platform and parameters

1. Ensure to allow only coherent values using combo-box widgets and set a range of values for the upload broadband speed selection.
2. Ensure to limit the allowed platform to the platform where the application is able to stream to.

### Add media sources into the project

1. Ensure to not add two times the same source to the used source
2. Ensure that the is only one source for each type of source (video and audio)

### Send the stream to a server

1. Ensure that image and video codecs are the codecs allowed by the broadcasting platforms
2. Ensure to ask the user if something is wrong with the broadcasting platform like a bad streaming key, the broadcasting platform server down, no source defined as used.

### Display stream in the software

1. Use a Queue store each video and audio to play in the player and automatically reload this queue to keep playing the streaming since the streaming was not stopped

### Display feedback of the streaming

1. Ensure to display every useful feedback into the interface and to filter every useless feedback.

## Use Cases

For each use case: description, diagrams (main use case first)

### Create a new project

The user wants to create a new project. The user clicks on the “File” menu and click on the “create new project “button in the interface. A new window appears and asks the user to give a name to the project. After clicking on the create button, the new project is created and the streaming configuration windows appears.

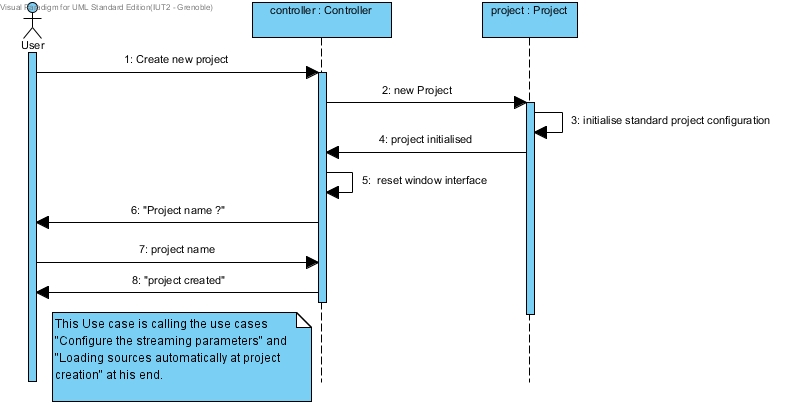


Figure 2 Sequence Diagram Create a new project

### Load the source available

When starting a new project the sources available are automatically load and appear in the sources list of the software.

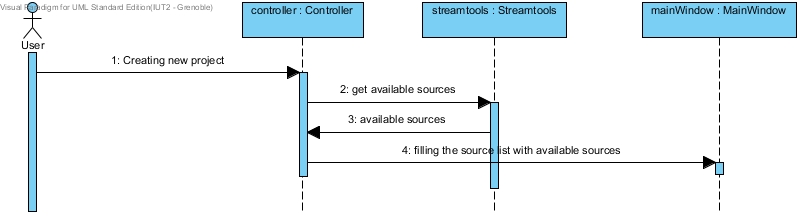


Figure 3 Sequence Diagram Create a new project

### Add a source into the project

When the user wants to add a source into his project he simply click on the source wanted in the source list and click on a button with an arrow to the right to add this source into the project, he can now see his selected source into the project source list call “Show”.

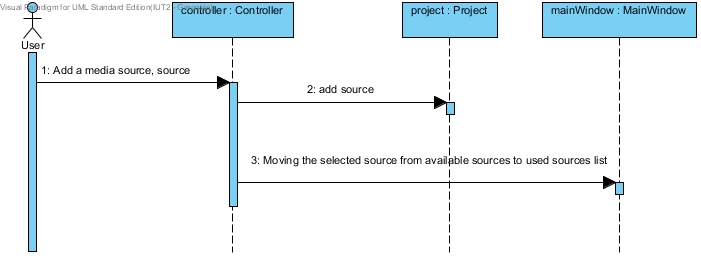


Figure 4 Sequence Diagram Add a media source

### Remove a source from the project

When the user wants to remove a source from the project he click on the source wanted into the source list of the project named “show” and click on the left arrow, he can see that the source is no more into the project list but in the source list.

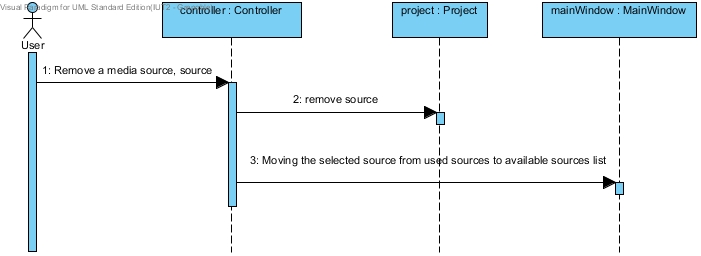


Figure 5 Sequence Remove a source

### Choose a broadcasting platform

The user wants to choose a broadcasting platform. He clicks on the menu “Config”, then “Choose platform“, a window appear with the platforms he can choose, he can select a platform then enter a streaming key and click on the « ok » button. He can access to the streaming parameters too.

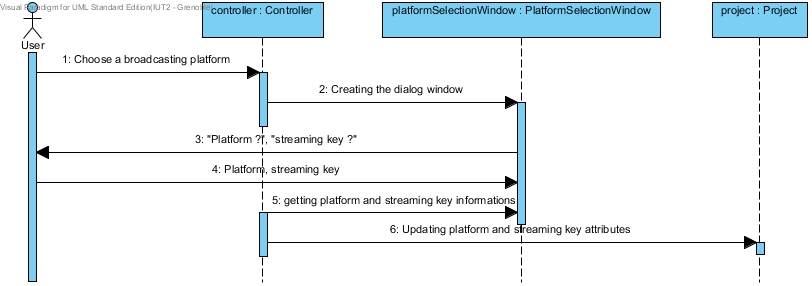


Figure 6 Sequence Diagram Choose a broadcasting platform

### Configure the streaming parameters

The user wants to configure the streaming parameters. He clicks on the “Config” menu and click on the “Configure parameters” button in the interface. A new window appear witch allow the user to configure the broadcasting platform parameters (size, format, bitrate, speed). The user validate the changes by clicking on the ok button.

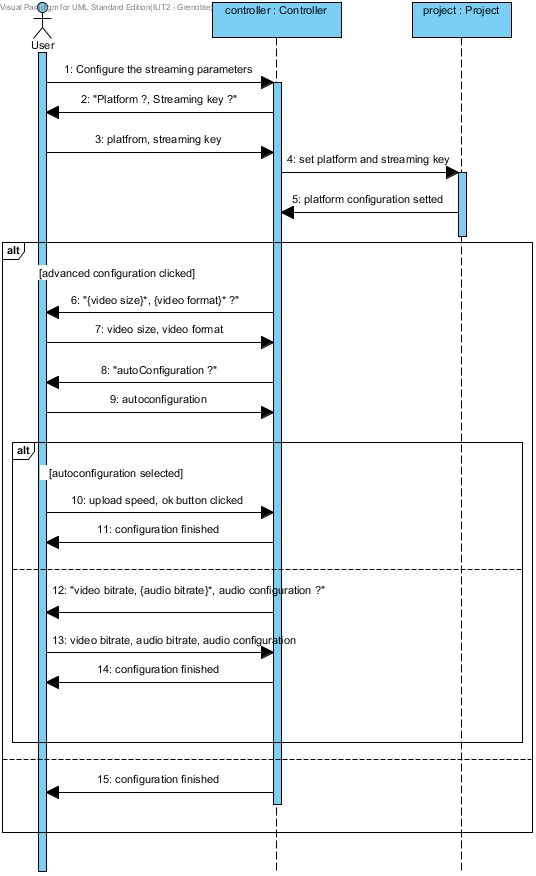


Figure 7 Sequence Diagram Configure the streaming parameters

### Save the project

After configuring the project, the user can save it into a file, which can be reuse later. All parameters defined can be saved in the file, like the sources selected, the configuration, the platform chosen. To save the project, the user has to go on the “file” menu, by clicking on “save” the save function is called if a project file is already created, else the function “save as” is called.

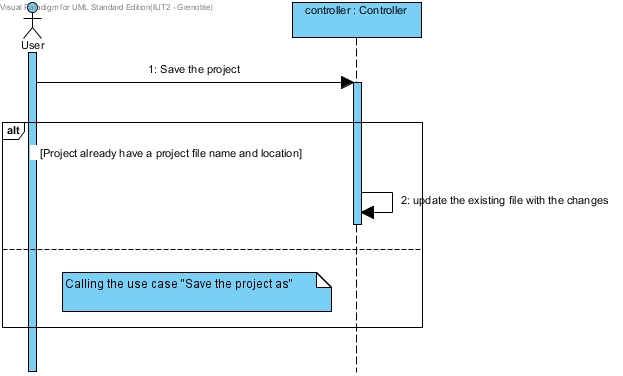


Figure 8 Sequence Diagram Save the project

### Save the project as

When the user want to save his project into a new project file, he go to the “file” menu, click on the “save as” button, he has to select a new path and a name for the project file, the parameters saved are the same than in the “save the project” use case.

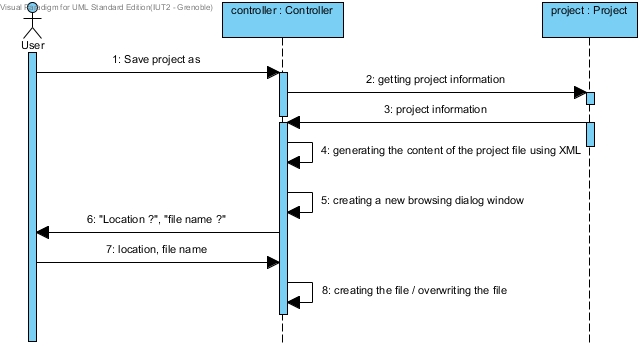


Figure 9 Sequence Diagram Save the project as

### Open a project

The User want to open an existing project, he go on the “file” menu and select “open a project “, he has to select the file of the existing project and click on ok.

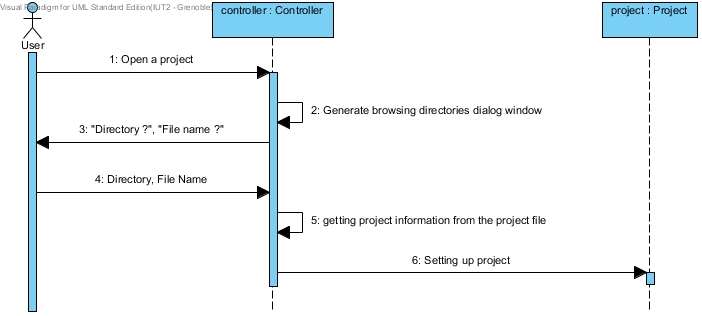


Figure 10 Sequence Diagram Open a project

### Rename a project

The user can rename a project when he want to, he just has to go in the “edit” menu and click “rename” a new window appears where he can enter the new name of the project and click on “ok” to validate it.

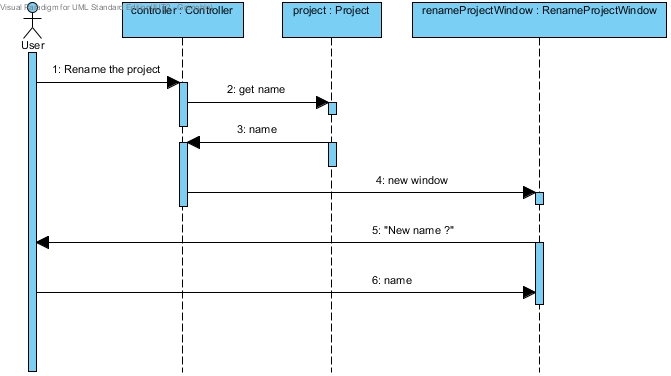


Figure 11 Sequence Diagram Rename a project

### Start streaming

The user wants to start streaming. If he has selected some sources for his project and configure the streaming parameters he can click on the play button in the main window or in the “Show” menu, then the streaming start and after a few seconds the stream is displayed into the main window.

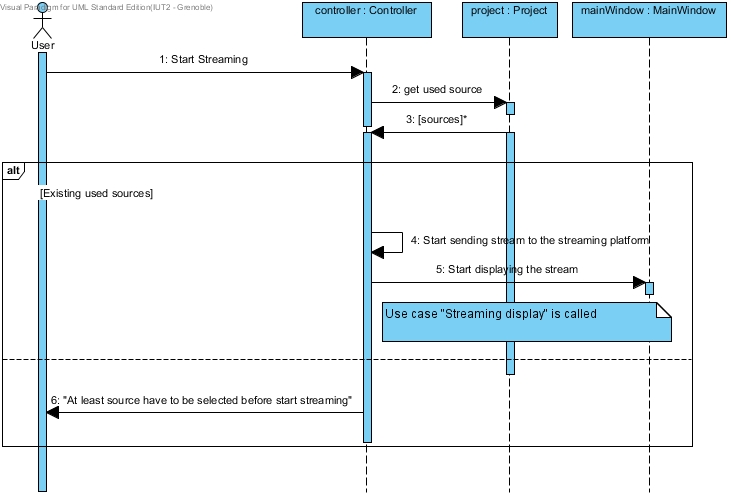


Figure 12 Sequence Diagram Start the streaming

### Stop streaming

The user can at every-time stop the streaming, by clicking on the button stop in the main window or in the “show” menu. Clicking on stop will stop the display of the stream and the broadcast on the streaming platform. When the user stops the streaming, the project is still available. If he wants to stream again he just has to click on the play button again.

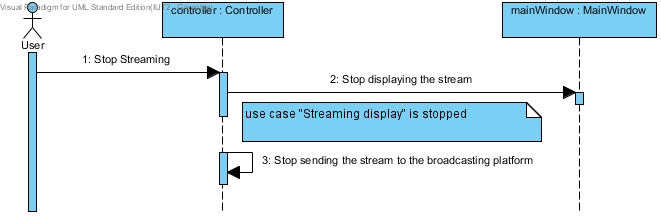


Figure 13 Sequence Diagram Stop the streaming

### Streaming display

When the user wants to stream the result of what he sent is displayed into the software with just a little delay, the user can control the sound of the output.

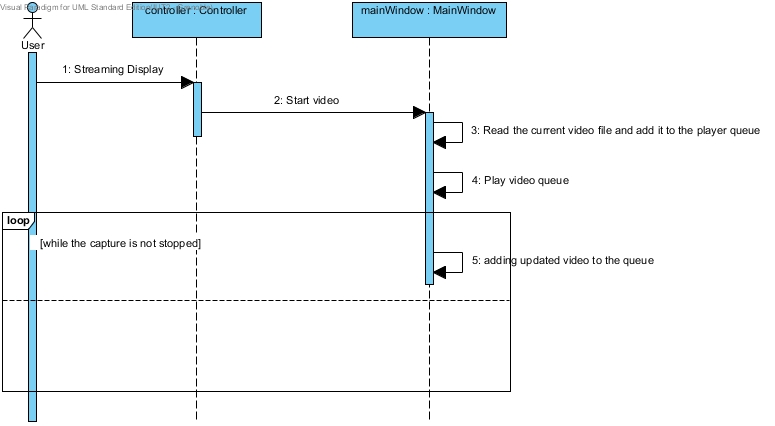


Figure 14 Sequence Diagram Streaming display

### Streaming feedback

When the user is sending a stream he also has a text feedback into the software that will tell the state of the streaming and of the source capture.

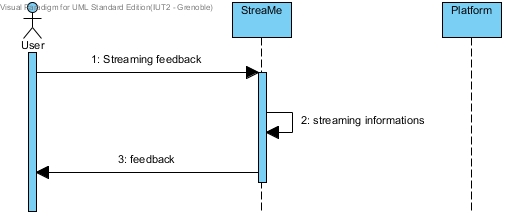


Figure 15 Sequence Diagram Streaming feedback

# Chapter 4: System Design

This chapter will outline how the software works in term of display and implementation; first we will take a look to the interface design.

## User Interface Design

The user interface must allow the user to create a project, select some sources, configure the streaming and start it.

To do that we have a few different window. Each window has its own functionality, but all the software is depending on one window, the main one, which we will describe first.

### Main Window

This window is the main window of the software, this is the most important window of the project, all the functionalities are depending on this window.

We have a menu where we can access to the project functionalities, the streaming parameters, the sources selection and where we can start and stop the streaming.

A list of the sources available is displayed in this window and we can add or delete a source to the show by clicking on the arrows between the two lists.

A player is also available where we have the streaming display, we can click on play and stop to play and stop the streaming, we also have a volume slider to control the sound in output.

The stream time is also displayed, this is the time since the stream is started.

And finally we have two tabs, one for the software feedback and one for the ffmpeg feedback (the streaming feedback), and we also have a status bar that display the streaming status (started, stopped).

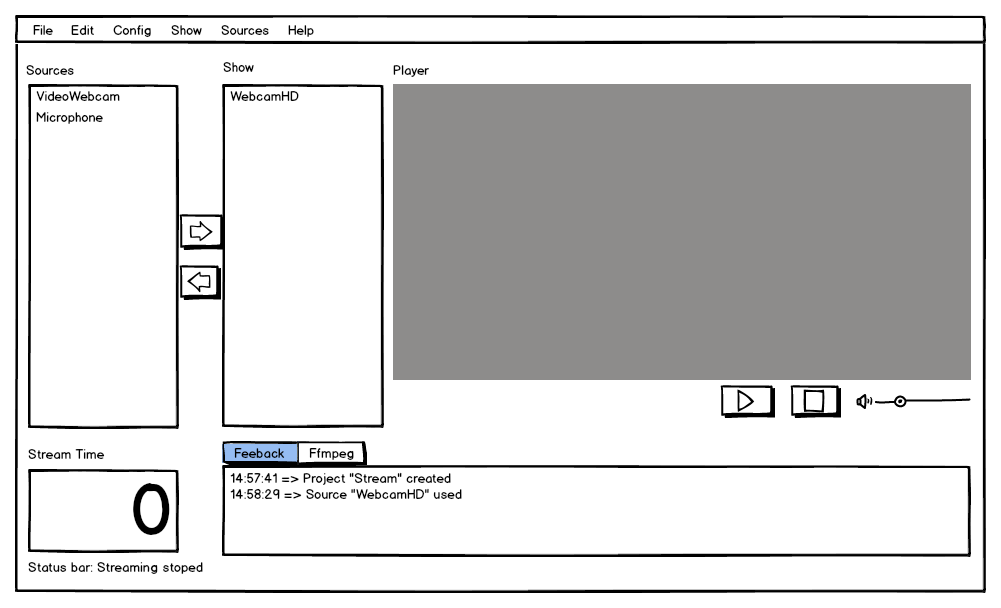


Figure 16 Main widow draft

### Starting Assistant

The starting assistant window is a window that is automatically displayed when we start the project.

It simply propose to the user to create a project or to open one, the user can skip this window but everything in the main window is disable until a project is created or opened.

If create project is selected the new project window is displayed, if open project is selected the open project window is displayed.

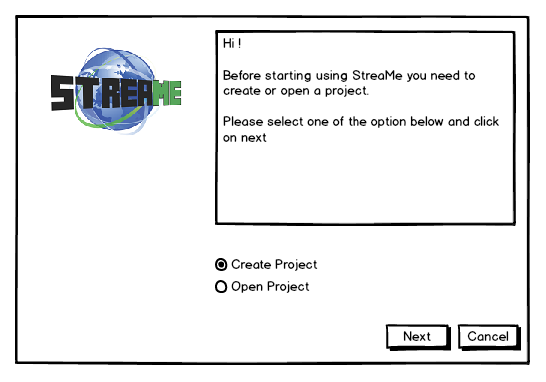


Figure 17 Starting assistant window draft

### New Project

The new project window is the window displayed when the user choose to create a project in the menu or with the starting assistant.

The user just has to enter a project name and click to next, which will display the streaming parameters window.

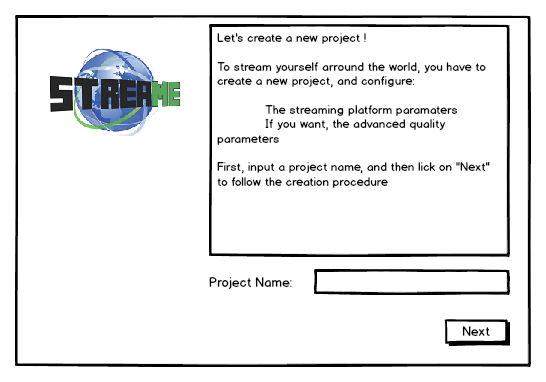


Figure 18 New project assistant window draft

### Streaming Parameters

The streaming parameters window make the user choose a streaming platform and enter the streaming key to send a stream on this platform, the streaming key is something a platform gave to a user when he create an account on it, it is obligatory to send a stream on a platform.

After selected a platform and enter the streaming key the user can validate by clicking on ok or configure the advanced parameters by clicking on advanced

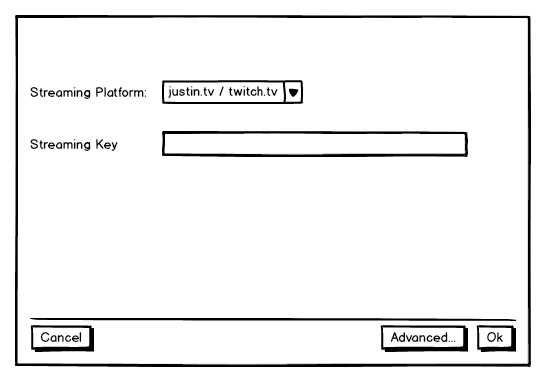


Figure 19 Streaming parameters window draft

### Advanced Parameters

The advanced parameters window allows the user to configure the parameters he want for his streaming.

He can select the size and the format of the video.

He can enable or not the auto configuration, if it is enable he just can change the upload speed, if the auto configuration is disable he can change more parameters like the video bitrate, the audio bitrate and format (mono, stereo).

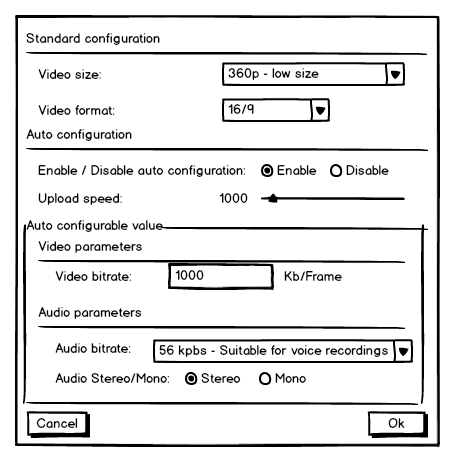


Figure 20 Advanced parameters window draft

### Rename

The rename window simply allows the user to change the name of the project, he can enter the new name and click on ok to validate, or click on cancel to cancel.

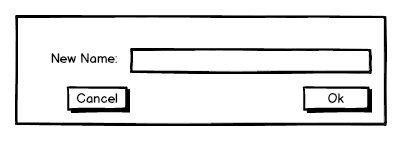


Figure 21 Rename window draft

## Functional Design

Most of the interactions are done within the main window in this software, except a few that will call the other windows described above. We will take a look to how the interaction with these windows are done in the software.

### Starting assistant

When we start the software the first interaction is the starting assistant, we select a starting option, open or create a new project.

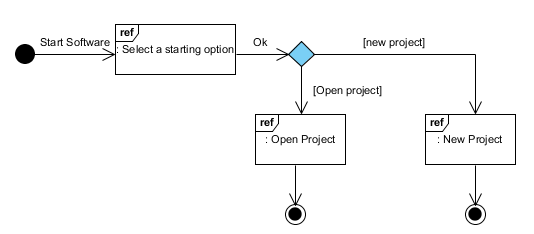


Figure 22 Interaction Diagram Starting Assistant

Then depending on the option selected another window is opened, a simple file browser to open a project or the new project assistant to create a new project.

### New Project Assistant

The new project assistant proposes to enter a project name, and then it brings the user to the streaming parameters window.

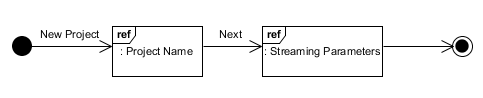


Figure 23 Interaction Diagram New Project Assistant

### Streaming Parameters

The streaming parameters window propose to the user to select a streaming platform in a list, the user has to enter his streaming key for this platform, then he can choose to end the configuration or to change the advanced parameters.

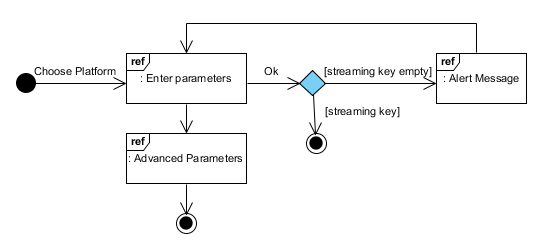


Figure 24 Interaction Diagram Streaming Parameters

### Advanced Parameters

The advanced parameters window propose to the user to change a lot of streaming parameters as we can see in the user interface part. The user can change these parameters and then save or cancel. The parameters in the auto configuration part can be changed only if the user select “disable” in the Enable/Disable auto configuration option.

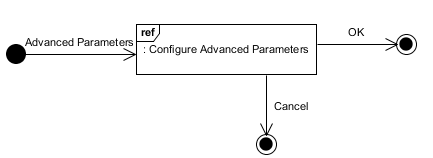


Figure 25 Interaction Diagram Advanced Parameters

### Rename

The user can choose to rename his project after creating it, a simple window to enter the new name is displayed, the user can save the new name or cancel the change.

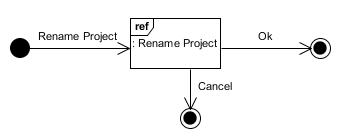


Figure 26 Interaction Diagram Rename

## Classes Design

This software requires an important number of classes to work, due to its complexity. To handle that correctly we implemented a design pattern, the model view controller pattern (MVC).

### Design Pattern MVC

The MVC pattern is very useful to structure the architecture of the software. Indeed this pattern separates the classes into three parts.

**The Model:** It contains the data classes of the software, example in our software the *camera* class.

**The View:** It contains the visual part of the software, the user interface, example in our software the *mainwindow* class.

**The controller:** This is what control everything, when an action is asked by a user, the view send the information to the controller who will apply it by using the model if necessary. In our software this is the *controller* class.

**Example of the use of the MVC in our software:**

When the user want to rename a project. The user enter the new project name in the view and tell the view “I want to save this new name”, the view call the controller and tell him “the user want to change the name of the project with this one”, the controller will change the name by calling the project model and tell him “your name is now this one”.

The point for this software to use this pattern was to structure the classes and make it easier to implement.

### Overall Class Diagram

Here is an overall description of each class, with the name and the position in the MVC pattern.

| Class | MVC | Description |
| --- | --- | --- |
| Source | Model | Super class of the different sources describe the name and the type of a source. |
| Camera | Model | Model of a camera, inherited from the source. |
| Microphone | Model | Model of a microphone, inherited from the source. |
| Project | Model | Model of a project, describe the name, the configuration of a project. |
| Controller | Controller | The main controller of the software. |
| StreamThread | Controller |  |
| StreamTools | Controller |  |
| AvTools | Controller |  |
| *ChooseCreateOpenProject* | View | View of the starting assistant. |
| *MainWindow* | View | View of the main window. |
| *NewProjectAssistant* | View | View of the new project assistant. |
| *PlatformSelectionWindow* | View | View of the streaming parameters. |
| *RenameProjectWindow* | View | View of the rename window |
| *StreamingParametersConfigurationWindow* | View | View of the advanced parameters |

### Class Diagram

Here is the class diagram of the software.

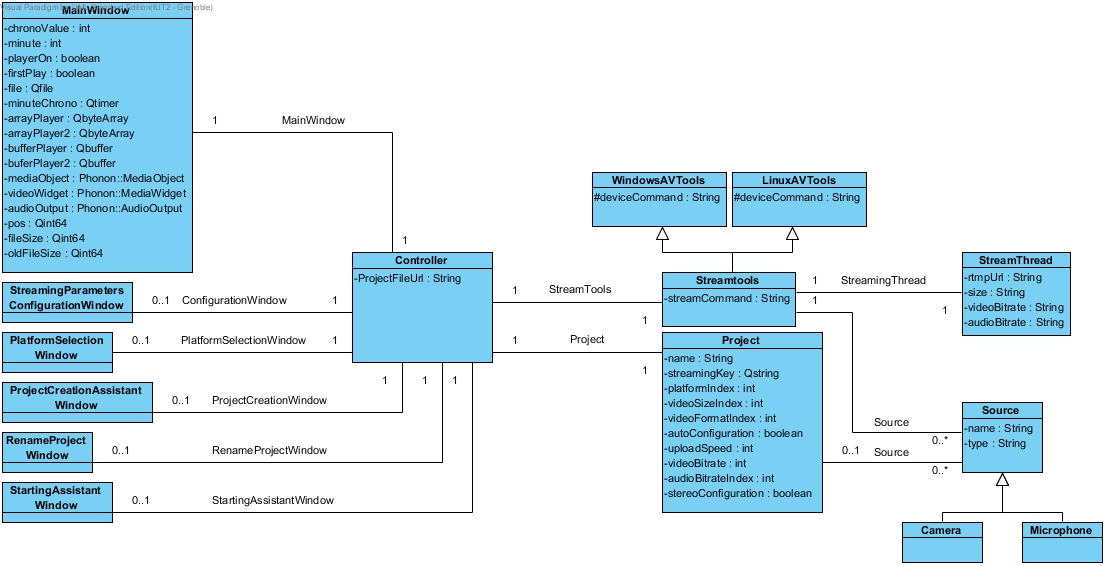


Figure 27 StreaMe Class Diagram

# Chapter 5: Implementation

## Languages, Libraries and Tools

### C++

C++ is a low level language well know and powerful.

Adaptable on every platform by simply compile the software on the desire platform, this language is good to be used for cross-platform development.

Because C++ is a low level language it greater performances than an higher level language like java or python.

C++ is also the third most used language according to the TIOBE website (website making a classification of computer languages usage based on the hits into the different search engines).

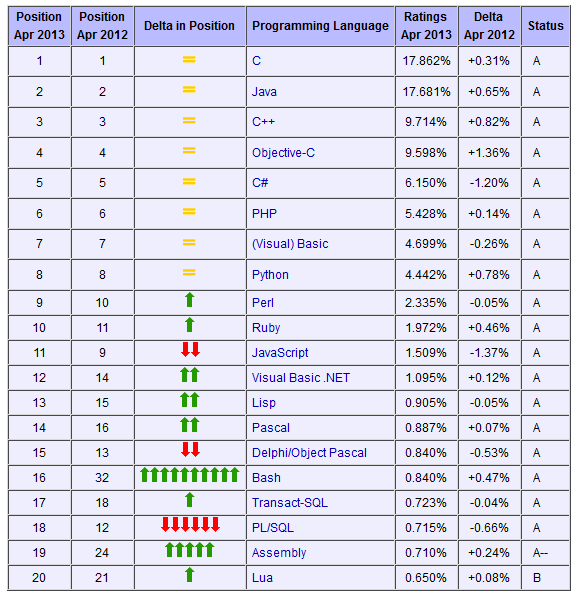


Figure 28 Classification of computing language usage[[1]](#footnote-1)

The percentage of use for this language show us that it is a powerful language but also gives a guaranty to find documentation and solve problem easily.

### QT



Figure 29 QT logo

QT is a cross platform framework for C++ developed initially by Trolltech and now maintaining by Nokia.

QT is a framework giving a lot of tools to add features like build easily GUI interfaces, Dialog with SQL databases, generate and read XML standard code, thread management with the same code on all platforms, to C++.

QT contains more than 800 different classes and more than 16000 different methods according to the QT documentation[[2]](#footnote-2).

As C++ QT is used by a large community, used for the well know KDE interface for Linux operating systems, and used for a lot of well-known applications like Amarok, Kdevlop.

### Qt Creator

QT Creator is an cross-platform IDE (integrated development environment) software useful for QT development including a windows builder to simplify the conception of GUI.

QT Creator is also very powerful because it includes auto-completion for QT library by default and C++ functions by default into it.

QT Creator has also a simplified interface for debugging because a different window template is available for debugging.

### Phonon

Phonon is a cross-platform multimedia framework that enables the use of audio and video content in Qt applications.

With Phonon you can easily implement a video and audio player into your application, by using classes like mediaObject for the video source, audioOutput for the audio and videoWidget for the display of the mediaObject.

To use phonon into a Qt project we just have to add the following line in the qmake project file:

QT += phonon

And include the classes needed like this:

#include <phonon/MediaSource>

### FFMPEG

“FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video. It includes libavcodec, the leading audio/video codec library.”[[3]](#footnote-3) In other words FFmpeg is a command line software that includes libav, a cross-platform and open source audio and video processing tools[[4]](#footnote-4). FFmpeg is able to detect the audio and video sources connected to the computer, capture an audio-video stream, convert them into a mpeg file and stream them all to a broadcasting platform using rtmp[[5]](#footnote-5). Rtmp is a protocol that has been developed by Macromedia for streaming audio, video and data over the Internet, between a Flash player and a server.

FFmpeg is therefore able to handle all features of StreaMe, but the best advantage is that it is highly configurable. Indeed, with FFmpeg StreaMe is able to specify the resolution desired, the quality of the audio and video render, the number of frames per seconds, the audio and video encoder…

### Broadcasting Platforms



Figure 30 UStream.tv logo

UStream.tv is the well know live broadcasting platform all around the globe with more than 50 million of unique monthly viewers.

This platform got a good reputation by having a lot a different content like sport; music and gamming live web-TV shared every day by his community.

UStream is also working as well as this because it proposes to the streamers to remunerate them by sending the money directly by paypal.



Figure 31 Logo Justin.tv Figure 32 Logo Twitch.tv

Justin.tv and Twitch.tv are two separate website for the same company.Justin.tv was the first website proposing live broadcasting like UStream.tv but with more themes available to define the different web-TV viewable on the website.

After a while the company decided to create a specific platform for game streaming where a large community of gamers stream every day. This website is a success because twitch makes his own knows by sponsoring gamming competitions.

Twitch.tv is considered now like the best platform for gamming streaming with a powerful tchat client included on the watching webpage of every website, options to set a delay, and a good flexibility on duration of adds for the streamer.

## Prototype Implementation

In this part we will see how each of the main functionalities are implemented in the prototype but also how the user interface has been created

### User Interface

### Project File

For the storage of the project data we decided to use the XML file format to define our own file storage format policy. The goal of using this format is to make it readable also for humans using a simple text editor to simplify the debugging. Indeed, using this standard an experimented user can easily understand what configuration was selected by reading the project file. Also, the C++ framework we use (QT) include methods to create and read XML formatted files. These included methods accentuated our decision to use it because it also simplified the algorithm we had to perform to open a project from an existing project file.

Conform to the standard way of saving files into software, we decided to allow the user to save a project using two different way (save and save as). Also, to simplify the finding of the StreaMe project file, a StreaMe file type was defined (.sm). This file type helps the user when this one wants to open a project file because it allows filtering the files by this extension easily.

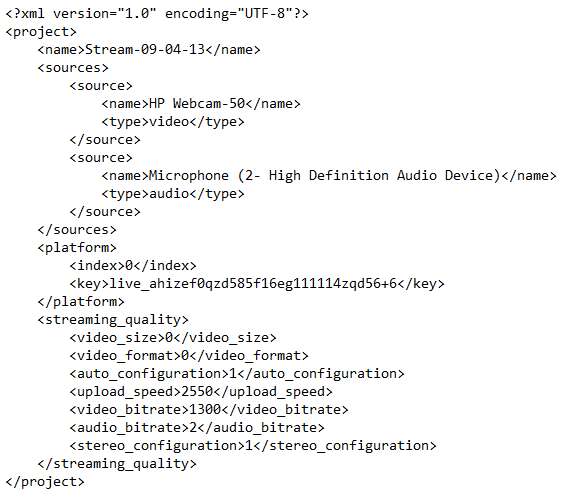


Figure 33 Example of project file

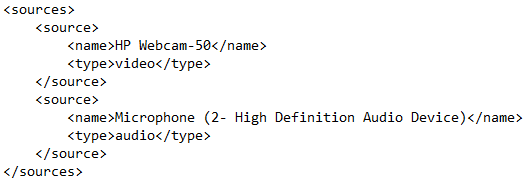
Into this storage file we choose to store each part of what defined the project itself.



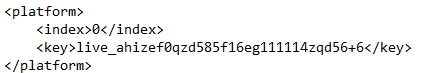
This line is just the xml header using for when loading the project file because this line define the encoding of the file and the xml version used. Without this line the Qt xml parser is not able to get the information form the file.



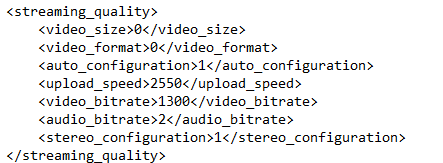
This line is just basicly the name of the project defined into the software. This name can be diferent that the name of the project file.



This part of the project file define each used source with his name as it apear into the software and the type of the source.



These information are the authentication information for the broadcasting platform.



Finally this part is the most important because it save every configuration for the audio and the video quality the user want to have when streaming the project.

### Streaming Settings

#### Auto configuration

#### Manual configuration

### Video Capture

To capture the video, we decided to use FFmpeg because it is a powerful and highly configurable software. To implement the video capturing, we had to reply to two main questions: how to do it with FFmpeg, and how to use FFmpeg inside StreaMe?

#### The FFmpeg command

After a long research, we’ve seen that FFmpeg uses different codecs on Windows and Linux and that it was impossible to use the same codecs on both platforms. Therefore, we knew that the command line had to be different regarding the operating system.

To detect available devices, FFmpeg is able to use dshow, a codec part of DirectX, hence the command on Windows would be: “FFmpeg –list\_devices true -f dshow –I dummy”. To lunch this command from StreaMe, we used a powerful feature of Qt: QProcess. QProcess is a cross platform class that is made for run external tools to the program. A StreaMe class has also been implemented to manage the audio and video detecting: avtools. In avtools, a function called detectSources() contains a QProcess that lunch the FFmpeg command, and analyses the output of this command. The following is a simplified way to run the dshow detection of devices on Windows.

QProcess \*process = new QProcess();

//Declares the QProcess

QString path("ffmpeg\\bin\\ffmpeg.exe");

//Indicates the path of FFmpeg

QStringList arguments;

arguments << "-list\_devices" << "true" << "-f" << "dshow" <<

"-i" << "dummy"

//Constructs the arguments to give to FFmpeg

process->start(path,arguments,QIODevice::ReadWrite);

//And start the process with the given arguments

On Linux, the command is different. Indeed, dshow doesn’t exist and we only found a separate way to detect sources. For video sources, video for linux (v4l2) is able to detect the cameras and for audio sources, a configuration file lists all microphones plugged to the sound card. Therefore, the detection function executes two QProcesses that lunch first of all the video detection, and then the reading of this file with the linux command “cat”. Respectively, the FFmpeg commands are: “vl4l2-ctl –list-devices” and “cat /proc/asound/cards”.

For both platforms, we implemented the function for read each output, construct a regarding source object (Camera or Microphone) and push each source detected into the class vector of sources. Each source contains a user name, a system name and a type. The type is “audio” or “video” and the user name is a human readable source name. For the system name, Dshow uses the same name as the user name, therefore on Windows they are equals. However, we’ll see that on Linux the command that we choose for capture the audio and video stream uses a different system name.

To introduce the main command of FFmpeg, the one that captures, configures and sends the stream to a broadcasting platform, we can separate the FFmpeg command in three parts. First, the input part which gives to FFmpeg the input sources. Secondly, the parameters part that contains the user specified parameters. Finally, the output part which is what FFmpeg renders with the input and the parameters.

The input part is given by the class avTools. Indeed, avTools contains a string attribute which is the audio and video input sources part of the final FFmpeg command. It contains also a getter and setters. The getter just return this string, but the setters is a bit more complex. Indeed, the getter has to handle four cases:

1. A Windows streaming with only a video source
2. A Windows streaming with a video and audio source
3. A Linux streaming with only a video source
4. A Linux streaming with a video and audio source

To handle the streaming with one and two sources, we’ve made the choice to surcharge a protected function that we called “setDevicesCommand()”. Therefore, setDevicesCommand() is represented twice in the class avTools : once with only one string parameter to pass the video name and once with two string parameters to pass the video and audio name. In each case, the function constructs the input part of the final FFmpeg command.

To handle the fact that the command is different regarding the operating system, we had to use pre-defined compiler macros to make the compiler compile each function for the relating platform. Hence, we use the macro “#ifdef \_WIN32 ” to give to the compiler the windows code, and “#elif \_\_linux\_\_” for the Linux code. The following is the complete code of the command that creates de “devices command” (the first part on the final command) for one video and one audio input source.

void AvTools::setDevicesCommand(string videoDevice, string audioDevice){

#ifdef \_WIN32 // if the OS where we compile is Windows

this->devicesCommand = "video=" + videoDevice + ":audio="+ audioDevice;

#elif \_\_linux\_\_ // else if it's Linux

this->devicesCommand = videoDevice + " -f alsa -i " + audioDevice ;

#else //else, error

#error

#endif

}

This code just constructs a part of the entire devices part. At the end, on Windows, this part will be “-f dshow -i video=Video WebCam:audio=Microphone (High Definition Audio Device)” for two sources, and “-f dshow -i video=Video WebCam” for a single source.

On Linux, we’ll use video4linux2 and the video system name for the camera, and alsa with the audio system name for the microphone: “-f video4linux2 -i /dev/video0” for one source, “-f video4linux2 -i /dev/video0 -f alsa -i hw:0” for two.

### Video Broadcast

### Video Display

The video display is implemented in the main window. It simply display the video captured with a delay of only five seconds.

To display the video, a video file is created by ffmpeg when capturing and streaming, it means that what is send to the streaming platform is also stored in a video file.

To read and display this video file into our software we use the QT framework named “Phonon” as described above.

Creating the basic player with phonon was really simple with the following lines:

mediaObject = new Phonon::MediaObject(this);

videoWidget = new Phonon::VideoWidget(ui->videoPlayer);

Phonon::createPath(mediaObject, videoWidget);

audioOutput = new Phonon::AudioOutput(Phonon::VideoCategory, this);

Phonon::createPath(mediaObject, audioOutput);

The media object represents the video source, the video widget represents the player in the software, and the audio output represents the audio in output.

Now that we have our player defined we need to add a media to it to play it. To add the video that we capture with our software into the player we use, as we say above, a video file.

It will be very easy to play a completed and finished video file in our player, but the problem is that, because we are capturing the video and we want to display it with a small delay we need to play the file before it’s finished.

Play a not finished file in our player is not as simple as a finished file. To do that we need to read the current file, add it to a buffer, start playing the buffer, and before the video end read the current file again (which will not be the same as before, new frames will be in the video) and add it to the buffer, we do that will the capture is working.

Here is how we implement that the first time in C++:

file->setFileName("why.mpeg");

file->*open*(QIODevice::ReadOnly);

fileSize=file->*size*();

\*array1 += file->read(file->*size*());

bu->setBuffer(array1);

mediaObject->setCurrentSource(bu);

mediaObject->play();

We can see in detail that we open the file, read in fully, add it to a bytes array and then add that array to a buffer, we set the buffer as a source for the media object and then play it.

For the rest of the video a signal will be send when the video is almost finished, we will then in the same way add the new part of the video to the buffer.

*(schema pour resumer le fonctionnement)*

### Feedback tabs

## Installer on Windows and Linux

### Windows

### Linux

# Chapter 6: Testing and Evaluation

# Chapter 7: Conclusion and Further Work

## Further Work

## Conclusion

1. Source : http://www.tiobe.com [↑](#footnote-ref-1)
2. Source : http://doc.qt.digia.com [↑](#footnote-ref-2)
3. Ffmpeg.org [↑](#footnote-ref-3)
4. Libav.org [↑](#footnote-ref-4)
5. adobe.com/devnet/rtmp.html [↑](#footnote-ref-5)