Amalia.js : an Open-Source Metadata Driven HTML5 Multimedia Player

Nicolas Hervé, Pierre Letessier, Mathieu Derval, Hakim Nabi
Institut National de l'Audiovisuel - INA
Bry-sur-Marne, France
nherve@ina.fr

ABSTRACT

Amalia.js is a new extensible and versatile HTML5 multimedia player that allows you to view any type of metadata synchronized with your video or audio streams. It manages metadata that are localized both temporally and spatially. They can also be hierarchical. Several visualization plugins have already been developped, enabling amalia.js to be deployed in a huge variety of web applications. We believe it can be used in various research areas to quickly visualize analysis results and to share them with the community. Amalia.js is an open-source software under a GPL license. It is available for download at `http://ina-foss.github.io/amalia.js`.

Categories and Subject Descriptors

H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia; I.3.8 [**Computer Graphics**]: Applications

General Terms

Algorithms, Design, Documentation

Keywords

multimedia, player, metadata, plugin, html5

1. INTRODUCTION

The INA Research Department is working on several multimedia document analysis topics : audio segmentation, speaker diarization and recognition, visual segmentation, visual object detection, text/audio synchronization, ... As part of our Research Area [1], several of our prototypes have to be developped as web applications so as to be easily accessible. Among these prototypes, most of them are used to analyze video streams and enhance their metadata. However, the analysis being at different levels (keyframes, video segments, audio channels, associated synopses, one video alone or relative to a corpus, ...) results are obtained in very different natures, often needing quite specific approaches to be presented to the end user. It is also common that a user interaction is required as part of research on the navigation aid or data mining in large multimedia data sets. Finally, it is sometimes useful to engage the user by asking him to confirm the results of automatic algorithms or to annotate the media and thus create groundtruth. Despite the disparity in research fields and resulting interfaces, working on a common object, a video, allows us to set a generic framework in which we hope to insert all of our work. The objective is to develop a universal media player, extensible and accessible through a web browser allowing on one hand to present all the results of video and audio analysis algorithms and also to be a solid foundation for applications of manual or semi-automatic annotation. The main characteristics expected of such a player are: ergonomics, ease of technical integration, genericity of the manipulated metadata and the use cases addressed, scalability, related tools and technical robustness.

After studying the various existing solutions, we concluded that it was better to develop our own software to meet all of our needs. Furthermore, we believe that this kind of tool can be useful to others. This is the case in all the areas of image analysis: biology, medicine, satellite, cultural heritage, sport ... Whatever the field of analysis, we always need to be able to quickly and easily view the results of the methods we are developing. This problem is particularly true as we often manipulate large amount of videos. Developing visualization and annotation tools is often a tedious task that deserves to be pooled. Thus, we choose to share it as an open-source project under a GPL license.

Amalia.js is composed of three main parts that will be described in the following sections : the unified metadata format, the core player and the visualization plugins.

2. PREVIOUS WORK

2.1 Metadata Format

We need a unified metadata model that will be used to convey all the necessary informations for the visualization plugins of our player. We already mentioned the huge diversity of analysis result types that can be produced for a media stream. Thus, the main difficulty in choosing or defining a metadata model is to ensure it will be generic enough and stable. Many formats already exist to represent technical

[1] `http://research.ina.fr`

MM'15, October 26–30, 2015, Brisbane, Australia.

ACM 978-1-4503-3459-4/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2733373.2807406 .

or indexing metadata on video and audio streams. Amoung the most common one can find Dublin Core [2], MPEG-7 [3], Cinelab data model (Advene project) [1] or the Open Annotation Data Model [3]. Work on creating such a model is conducted for several years in our Research Department. One can cite the work done around Feria [2] that allowed such thinking and formalized the representation of metadata temporally positioned on a stream. This paper also showed why MPEG-7 is unable to fulfil all our requirements.

2.2 Web Players

The list of available HTML5 video players is growing every day. Among the well known softwares in this category, we can cite JWPlayer, video.js or Popcorn.js [4]. A good overview is available with a comparison matrix [5]. However, there are only a few players able to synchronize metadata with the playback of the stream. Generally they are limited to the display of subtitles or clickable overlays for ads insertion in the video. Some tools meet quite the needs we have presented. Most of them are developed in research labs. This is the case of the IRI player [6]. There are also some tools for video annotation, like Vatic [7] and the Open Video Annotation Project [7]. In [5] an HTML5 player is presented with a special attention on non-linear navigation and user annotations. VideoJot [6] allows the user to annotate both spatially and temporally. These two software seems promising but, unfortunately, they are not publicly available.

3. AMALIA SOFTWARE ARCHITECTURE

3.1 Metadata Model

The existing formats either do not allow for an accurate representation of temporal metadata or they are too complex to implement. Indeed, we need a representation format of the information for displaying them in the player. This means it must be easy to parse and to be in a standard language for HTML5 developments, typically JSON. Thus, algorithms for analyzing multimedia streams retain, if they need it, their own data model, which can be richer and more complex than the one used for the player.

The main principle of amalia.js is to have a unified metadata model. Ensuring that all the metadata types are consistent and use the same standards will facilitate their use and enable us to design generic visualization plugins. A typical usage of amalia.js is to have a video file with a few metadata blocks. When instantiating the player, a binding has to be done between the metadata blocks and the visualization plugins so as to decide which metadata is displayed and in which way. It is possible to have a single metadata block bound with several visualization plugins, each one responsible for displaying a specific facet of the data.

A generic metadata model has thus been designed, it is intended to represent all the metadata of a media stream. It is able to describe both the audio and video of a content, technical informations, any documentary notes and all the results of automatic metadata extraction algorithms. Currently, this model is only able to manage metadata of a single stream. For analysis results of a corpus, involving links between streams (clustering, mining, similarity detection ...), a specific data model extension has to be created and is currently investigated. A full XML schema (XSD) has been written to represent this metadata model. This schema has much more elements than what is strictly needed to display simple metadata with amalia.js. The current version is 0.2.5. We use this schema to automatically generate Java classes using Jaxb and Jackson to serialize the metadata stream in JSON. The amalia-model project [8] is the reference implementation. It comes with several helper methods through a general factory and is also available on the Central Maven Repository. This JSON format is currently the only supported metadata format.

The model is generic and extensible, it is able to represent any metadata that is :

- localized temporally in the stream : at a given timecode (*tc*) or during a specific time segment (*tcin*, *tcout*)
- localized spatially in a video frame : currently ellipses and rectangles bounding boxes are available
- hierarchical : each metadata is associated to a specific level (*tclevel*)
- labeled : each metadata may have a label
- scored : each metadata may have a confidence scored
- extended : each metadata can have its own additionnal data that is not represented in this model

Each metadata block has a unique *id* that is used for the binding with the visualization plugins. A metadata block should only represent a coherent and homogeneous bunch of data. If you need to display distinct types of data, or even several blocks of data of the same type (e.g. the output of different versions of the same algorithm), you should provide them in separate blocks of metadata. One can find example of JSON files on the web site.

3.2 The core player

There are two main difficulties identified for such a project. First, we have to cope with the HTML5 restrictions. Although major progress has been made with the latest versions of HTML, there are still some difficulties compared to the development of desktop applications. This is particularly true for the multimedia streams management (codecs and synchronization). Unfortunately, we also still have to deal with the web browsers diversity. This is especially true with the emergence of mobile devices. Thus, we have to ensure our player is working fine on a huge variety of web browsers, including tablets and smartphones. It has to follow the responsive design guidelines. Secondly, we must ensure that the software architecture enables the player to support the various plugins that will be developed and that it does not put too many constraints on future developments. Furthermore, we must guarantee the good overall performances of the whole system.

For our player development, we could have started with an existing software and adapt it. However, the main development effort is on the visualization plugins management infrastructure and the metadata synchronization. It quickly

[2] `http://dublincore.org`
[3] `http://www.openannotation.org/spec/core`
[4] `http://popcornjs.org`
[5] `http://html5video.org/wiki/HTML5_Video_Player_Comparison`
[6] `http://www.iri.centrepompidou.fr/outils/metadata-player-2`
[7] `http://www.openvideoannotation.org`
[8] amalia-model project : `https://github.com/ina-foss/amalia-model`

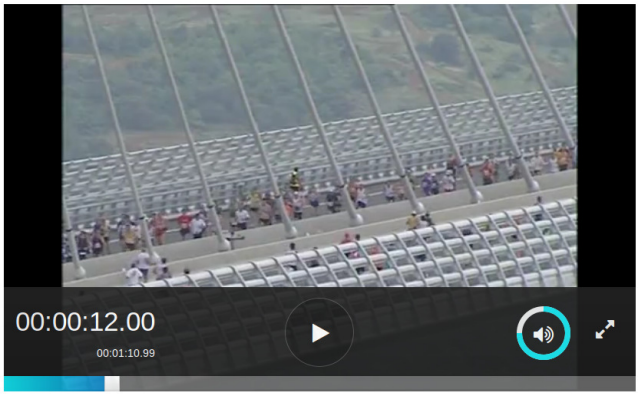

Figure 1: Amalia.js player interface

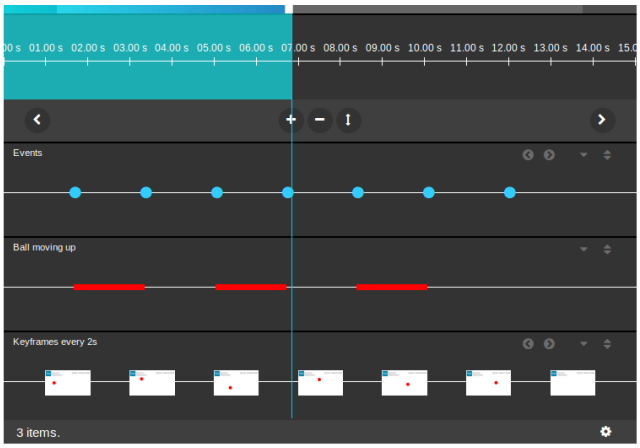

Figure 2: The timeline plugin

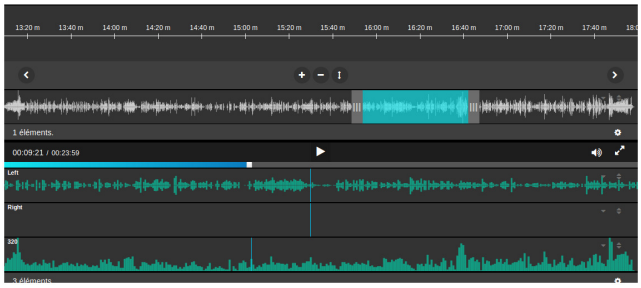

Figure 3: The histogram visualization for the timeline plugin

appears more complex and time-consuming to adapt existing software rather than developing a new one from start. Indeed the basic functionalities of a media player are now managed by web browsers, the additional cost to manage these features (mainly the player control bar) is minimal compared to adapting an existing software. Amalia.js core is mainly an event-driven framework based on a plugin architecture. Both the internal technical components and the visualization widgets are plugins. This way, we can easily extend and modify the behaviour of the player or adapt it to an evolving technical environment. The core player is responsible for the overall configuration, the media asset being played, the plugins management (instantiation, configuration), the metadata management (loading, parsing, keeping up-to-date), the binding between metadata blocks and visualization plugins and the event loop (user interaction, synchronization between the plugins and the stream playback, metadata update notification). It provides an API for the plugins to access the metadata and to control the stream playback. The loading and parsing of the metadata blocks are two technical plugins. They are responsible for, respectively, the transmission protocol and the data format. We have implemented two protocols. The first, and simplest, is loading a full list of metadata blocks from an URL. On the server side, this URL can be a simple JSON file (such as the demonstrations on the amalia.js web site) or a dynamic web application using, for example, the amalia-model implementation (such as DigInPix [4]). The second protocol uses web sockets. We can thus manage metadata streaming to the player. This feature is particularly interesting when one need real-time synchronization of the metadata. It is the case, for example, with collaborative annotation of a media stream or if we need to display the results of an automatic analysis algorithm as they are produced. As already mentioned, we currently only support the JSON format of our metadata model. But, as the parsing of these data is also a plugin, it can easily be replaced so as to manage another data representation. The binding between a metadata block and a visualization plugin can be achieved in two ways. One can either bind them statically when configuring the player (using the metadata block *id*) or use the dynamic binding that will associate metadata blocks and plugins based on the data types.

The figure 1 shows the amalia.js player default layout. Indeed, the control bar is a plugin which contains many widgets. The global layout and the visual aspect of these widgets can easily be modified so as to adapt the player to your application visual design.

3.3 Visualization Plugins

3.3.1 Timeline

The purpose of this plugin is to represent all the timecoded metadata on a timeline. This plugin is composed of two parts. At the top, the time axis where you can navigate and zoom in the timeline. Below the time axis, all the timelines are displayed. For a given media stream, one can display as many timelines as desired, each of which may be bound to a different metadata block and displayed with a specific type of line. For the moment, we have four types of lines :

- cuepoint: simple temporal points (tc field)
- segment: temporal segments (tcin and tcout fields)
- image: temporal points with images, typically keyframes (tc and thumb fields)
- histogram: any information that is temporally distributed

The three first types of lines are used in the figure 2. The histogram visualization is presented in figure 3 where a wave form is displayed. One can notice that an audio stream is played as no video playback window is displayed. Furthermore, the timeline plugin is instanciated twice and bound each time on the same metadata block. Once with the zoom enabled (at the top), and once with the zoom disabled.

The timeline is synchronized with the player, a progress bar indicates the current position in the media and a click on any object represented in the timeline will seek in the media at the correct timecode. Optionnaly, the hierarchical structure of the metadata can be exploited. One can associated a level in the data hierarchy with a zoom level of the player. This option is particularly usefull when too many data are available and when we choose to display only the most important ones at a first glance. The other data will only be displayed if the user chooses to zoom in and explore a specific part of the metadata

3.3.2 *Caption and synchonized text*

Textual metadata can be displayed by two specific plugins: caption and synchonized text. Depending on the textual metadata you have, it may be usefull to exploit its hierarchical structure. Typically, one may choose to have the words, sentences and paragraphs of a text temporally localized. In such a case, one would choose a specific *tclevel* for each level of text. When binding a text metadata block with one of the text visualization plugin, one has to choose which level will be displayed. The caption plugin is relatively conventional and simply used to represent the text over the video. It is used to display subtitles. The synchronized text plugin allows you to browse all the text in a specific window and synchronize the playback with the video.

3.3.3 *Overlay*

This plugin is used to visualize objects detected in the video with bounding boxes. Moreover, it is able to track them. A track is represented by a series of spatial locations. The exact position of the bounding box is then extrapolated by the plugin during the playback of the video.

3.3.4 *Edition*

Apart from the new technical or visualization plugins, we are currently working on the editing version of amalia.js. This will allow us to enter or correct metadata directly through the player. We plan to use this version in groundtruth management applications so as to enable the creation of huge research datasets used to train supervised algorithms or to assess analysis methods performances. Entering information in the player is also a feature that could be useful for any business applications dealing with temporal annotations of multimedia streams. The current version is shown in figure 4.

4. CONCLUSION AND FUTURE WORK

We have developped a HTML5 media player with its synchronized metadata visualization plugins. This software is used in some of our research prototypes : DigInPix [4] and SyncNotes [9]. It is also used at INA for professional applications. We provide it as an open-source software in the hope it will be usefull, especially in the multimedia analysis community. The work is not yet finished. We are currently developing the edition version of the plugins and improving the server side libraries so as to ease applications development. Furthermore, new visualization plugins will be available shortly. We encourage anyone to participate by providing their own plugins or help us with existing ones.

[9] `http://syncnotes.ina.fr/en`

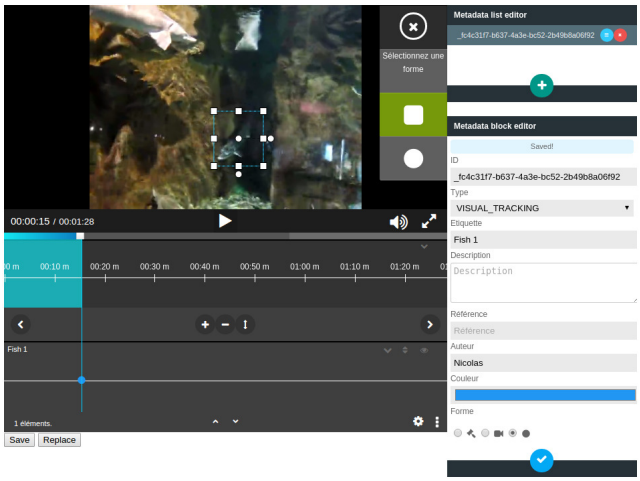

Figure 4: The edition application

Acknowledgments

We wish to thank the main developer of this software that, unfortunately, can not be in the authors list for legal reasons.

5. REFERENCES

[1] O. Aubert and Y. Prié. Advene: An open-source framework for integrating and visualising audiovisual metadata. In *Proceedings of the 15th International Conference on Multimedia*, MULTIMEDIA '07, pages 1005–1008, New York, NY, USA, 2007. ACM.

[2] V. Brunie, J. Carrive, and L. Vinet. Ingénierie des documents audiovisuels : le projet feria. une approche centrée sur la description des contenus. *Technique et Science de l'Information*, 2006.

[3] S.-F. Chang, A. Puri, T. Sikora, and H. Zhang. Introduction to the special issue on mpeg-7. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):685–687, 2001.

[4] P. Letessier, N. Hervé, A. Joly, H. Nabi, M. Derval, and O. Buisson. Diginpix: visual named-entities identification in images and videos. In *ACM International Conference on Multimedia Retrieval*, 2015.

[5] B. Meixner, B. Siegel, P. Schultes, F. Lehner, and H. Kosch. An html5 player for interactive non-linear video with time-based collaborative annotations. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, MoMM '13, pages 490:490–490:499, New York, NY, USA, 2013. ACM.

[6] M. Riegler, M. Lux, V. Charvillat, A. Carlier, R. Vliegendhart, and M. Larson. Videojot: A multifunctional video annotation tool. In *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, pages 534:534–534:537, New York, NY, USA, 2014. ACM.

[7] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, pages 1–21. 10.1007/s11263-012-0564-1.