Synchronized Delivery of 3D Scenes with Audio and Video

Cyril Concolato* Jean Le Feuvre† Emmanouil Potetsianakis‡

Institut Mines-Telecom; Telecom ParisTech; CNRS LTCI
46, rue Barrault
75013, PARIS

Abstract

Nowadays, 3D graphics have established their presence on the web - alongside audio and video. In fact, 3D scenes are often used in conjunction with audio and video, to create virtual worlds. However, the diverse nature of these various media components raises synchronization and packaging challenges. In order to address these challenges, we propose packaging 3D scenes, with audio and video, inside MP4 containers. This way, the 3D and other media are delivered as a whole, and on the receiving end, we are able to extract and synchronize the content, from within the browser. In this paper, we explain our methodology, present an end-to-end example scenario, and its associated implementation, using open-source tools.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—[Standards, Languages] H.5.1. [HCI]: Multimedia Information Systems—;

Keywords: X3D, HTML5, MP4, Streaming, Synchronization

1 Introduction

Recent innovations in browser technology, such as the introduction of HTML5 `canvas`, `audio` and `video` elements, and the associated Javascript APIs such as `MediaSourceExtension` (MSE) and `Webaudio`, simplified the media content integration in web pages. In conjunction with the WebGL API, which allows hardware accelerated rendering of graphics without the need of plug-ins, interactive multimedia from the browser is more accessible than ever. The aforementioned evolutions not only facilitated the use of audiovisual (AV) content, but popularized the use of 3D scenes for the web. Especially in the case of virtual worlds, AV content is often hosted inside 3D scenes to create immersive user experiences. There are numerous examples, like the "Flight of The Navigator"[1] by Mozilla, which demonstrates a 3D environment incorporating video texturing for virtual billboards, or the "No Fun Cube"[2] by The Presets, which couples auditory experiments with the exploration of a video-textured 3D cube.

Despite the progress in the topic of multimedia content representation, existing solutions fail to establish an efficient transportation technology. The predominant approach is to fetch the 3D documents from a location, typically using AJAX, while the AV content is streamed from a different location, even if they are rendered in the same context and share a common distributor. As a result, the content is often unsuitable for offline consumption, without using caching tools, and even in online use cases, delivery issues between the AV and 3D content might occur (different latency between sources, unavailability of a server, etc).

In this paper, we propose and describe how to package both AV and 3D content in a single MP4 container, to benefit from the features that the MP4 file format offers in terms of synchronization, delivery and interoperability. We also demonstrate an end-to-end solution using this packaging, based on open-source tools, for the consumption of synchronized video-textured 3D content from within the browser.

In the following Section 2, we review the current trends in the field of 3D scenes with AV content, in order to illustrate the mechanisms of our proposal in Section 3. Then, in Section 4 we demonstrate a sample application we developed, implementing our solution. Finally, in Section 5 we summarize and present our future work.

2 State of The Art

In a survey by Evans et al. [Evans et al. 2014] various 3D technologies for the web were studied. Even though the publication is not focused on transportation, packaging and synchronization aspects of 3D graphics, it indicates that a widespread distribution mechanism for 3D scenes does not exist. We believe that all the major web-based technologies for 3D graphics mentioned in the study, are compliant with our solution.

Limper et al. [Limper et al. 2013], published a paper focusing on the delivery of 3D content for the web. Their work is thorough in the domain of compression and encoding, but it does not mention any specific streaming and packaging technology. It is relevant nonetheless with our work, since MP4 files can carry compressed 3D scenes (e.g. BIFS, MPEG-3DGC).

Kapetanakis et al. demonstrated a basic virtual reality world built with X3DOM, that integrates video streaming over MPEG-DASH (with Dash.js) [Kapetanakis et al. 2014]. MPEG-DASH is a standardized mean of transmitting multimedia content and it is utilized to do adaptive streaming of videos. The authors chose to statically load the 3D model with the webpage. Dash.js could be modified to stream the 3D content, but without solving the distributed resources issue, or having consideration for offline scenarios. This approach can benefit from our solution, since one or more X3D scenes can be inserted in the same MP4 file used for the video (and then delivered over DASH if desired).

3 Packaging 3D scenes in MP4 files

As mentioned in Section 1, the main idea of our proposal is to package the 3D content in the same MP4 container as the AV content, to benefit from the MP4 properties. MP4 files allow storing media data of a particular type in a structure called track. A track is used

*e-mail:cyril.concolato@telecom-paristech.fr
†e-mail:jean.lefeuvre@telecom-paristech.fr
‡e-mail:emmanouil.potetsianakis@telecom-paristech.fr

[1] https://videos.cdn.mozilla.net/uploads/mozhacks/flight-of-the-navigator/

[2] http://nofun.thepresets.com/

to carry samples, with associated timestamps. These timestamps indicate, in the MP4 timeline, at which moment the media data is to be consumed by the application. Multiple tracks can be stored in the same file, either providing different types of data (e.g. audio and video streams), or the same type of data for alternative or complementary streams (e.g. audio in different languages). Regarding delivery, MP4 files can be delivered as a whole over HTTP, using progressive download, or split as mutliple segments and streamed over HTTP, using MPEG-DASH.

We apply these concepts for the carriage of 3D data in MP4, by placing 3D scenes in tracks. Tracks for this type of data are indicated by the type 'meta'. For XML-based 3D scenes (e.g. X3D) we indicate the sub-type of the track to be 'metx'. For other formats (e.g. Three.js scenes), the appropriate sub-type is 'mett', originally reserved for text. Other sub-types are available for binary (possibly compressed) data.

Each sample of a track, carries one 3D scene. In other words, for XML-based scenes, one sample carries one full XML document. Fragmentation of XML per sample [Concolato and Potetsianakis 2015] would also be possible if progressive rendering of XML documents is available.

The effects of this approach are:

- The delivery of the 3D and AV content altogether facilitates the synchronization of these media components.
- Alternative 3D representations (e.g. providing different levels of detail) can be realized by using several tracks.
- Streaming of 3D content can be achieved by providing multiple samples (i.e. 3D documents) per track.
- The file can be delivered using DASH.
- The AV content of the file is still playable by clients without 3D support.

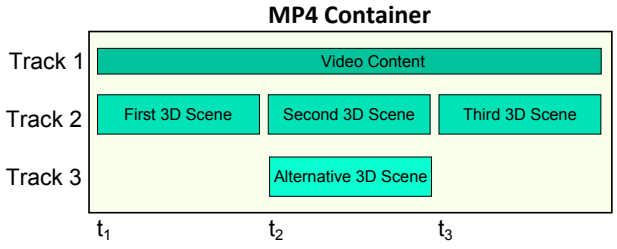


Figure 1: *Outline of a valid MP4 container authored for 3D document streaming*

An illustration of an example MP4 file is shown in Figure 1. Track 1 contains the video stream, Track 2 a stream of three 3D scenes, and Track 3 one 3D scene. At t_1 the 3D scene inside the sample of Track 2 is rendered, while at t_2 the client is able to chose between the 3D scene in Track 2, or the alternative scene in Track 3 and so forth.

4 Application Example

In order to demonstrate our proposed solution, we realized a web application which dynamically loads video-textured X3D scenes during video playback. MP4Box is used for the packaging, while the javascript library MP4Box.js handles the extraction and synchronization of the content (both tools are part of the GPAC [Le Feuvre et al. 2007] open-source multimedia framework[3]). Finally, X3DOM [Behr et al. 2009] is used to render the X3D scenes (via WebGL).

To create valid MP4 files, each stream (audio, video and 3D) is stored in a separate MP4 track. MP4Box natively supports audio and video import, while for X3D (or any otherwise non-supported stream) requires NHML descriptor files as shown in Listing 1. For each MP4 track, one NHML file is used, with parameters specified as attributes of one `NHNTStream` element. Since we are using X3D documents (which is an XML-based 3D format), we set 'meta' as `mediaType` and 'metx' as `mediaSubType`, followed by the XML namespace. Other optional arguments that can be set are `trackID`, `text_encoding` and `xml_schema_location`. Every 3D scene is imported into a separate sample by providing its filename in `mediaFile` and the timeslot in `DTS` (according to the defined `timeScale`). We can specify a `duration`, otherwise the samples are considered to be consecutive (without gap). Each sample has a self-contained 3D document, therefore it is always a Random Access Point (set at `isRap`). This property allows seeking inside the MP4 file.

```
<NHNTStream timeScale="90000" mediaType="
    meta" mediaSubType="metx" xml_namespace=
    "http://www.web3d.org/specifications/x3d
    -namespace" >
  <NHNTSample DTS="0" isRAP="yes"
      mediaFile="first.x3d"/>
  <NHNTSample DTS="900000" isRAP="yes"
      mediaFile="second.x3d" duration="
      9900000"/>
</NHNTStream>
```

Listing 1: *Sample NHML descriptor file*

On the client side, the application uses MP4Box.js to analyze the tracks of the received MP4 file, and then prepares the web page for the streams, according to the type of the available tracks. After the rendering context is initialized for every stream, it parses the samples of each track. Audio or video streams are decoded through MSE, and the video samples are rendered in a `canvas` HTML element. In case a sample is parsed containing an X3D document with `video` (or `canvas`) texture node(s) referring to the same MP4 file, the reference(s) is (are) replaced with a reference to the already active `canvas` element used for the video track. If multiple video tracks are present, fragment identifiers are used to point to a video track in particular (e.g. `URL="file.mp4#trackID=2"`). This is the only modification of the X3D document that might occur, throughout the process. Then, the updated X3D document is parsed to X3DOM, with a `reload` content instruction. Finally, after X3DOM sets the DOM context, it uses WebGL to render the 3D scene. An overview of the web application architecture is shown in Figure 2.

We created a sample MP4 file with two tracks, containing a video stream in the first one, a 3D scene with a video-textured cylinder and a 3D scene with a video-textured box in the second. Both of the 3D scenes reference the same video, and we used the same NHML descriptor for the import as in Listing 1, placing them in consecutive samples. We also kept the original video output visible, for the frame-accurate synchronization to be evident. Figure 4a shows the output at 6" of playback, while Figure 4b at 47", at which point we have already switched to the second 3D scene.

We tested the scenario, using the `requestAnimationFrame` to refresh the `canvas` content, in Google Chrome, running on a machine with an Intel Xeon CPU of 4 Cores at 3.6GHz, 16GB of

[3] https://github.com/gpac/

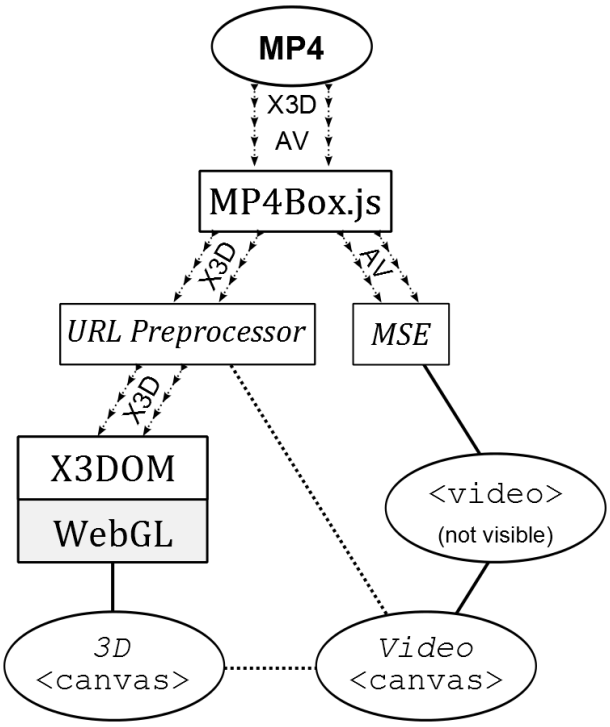


Figure 2: *Structure of web application*

RAM and a Nvidia Quadro 600 graphics card. In a set of 20 runs, the mean delay measured from the cue of the first X3D sample, until the rendering of the first textured frame was $21ms$. For the second X3D scene, the mean delay measured from the cue, until rendering was $119ms$.

The aforementioned delays include the latency caused by 3D scene texture URL processing, setting up and switching to the new 3D rendering context and the X3DOM reloading. However, the MP4 file structure is such that it allows MP4Box.js to analyze it and expose the timing information in advance, as shown in Figure 3. Since we receive the 3D scene early (at t_1), we start the video playback when the 3D scene is set up (at t_3), thus keeping the video and 3D synchronized from the beginning. In a similar manner, the second scene is received early (at t_4), and it can be constructed inside a new -initially invisible- `canvas` element (at t_5), then at the right time (at t_6), replace the previous (which is destroyed in the background), ergo circumventing the gap between 3D scenes.

5 Conclusion and Future Work

Our work proposes a novel approach to 3D and AV distribution for the web, by packaging the contents in a single MP4 container. This way, we are able to consider streaming and offline scenarios and resolve synchronization issues. We are able to insert 3D scenes in dedicated tracks, thus constructing valid MP4 files, that allow the AV content to be consumed even by clients that lack 3D content support. Also, this approach is content-agnostic, since we do not modify the original content prior to importing, thus offering support for most 3D formats, given a compliant client.

On the receiving end, the client is able to extract the 3D scenes from the MP4 container, and once they are rendered, to provide frame-accurate synchronization with the AV content. We demonstrated our solution by implementing it in a web application for X3D scenes, where the 3D content is dynamically loaded and tightly timed to the video. All the tools used to build the application are freely available and were used without any modification of the source code. Since MP4Box.js only extracts the samples from the tracks, any format can be used (e.g. XML3D) in a similar manner.

Due to the aforementioned reasons, our mechanism can be used to improve the content distribution of existing solutions [Jankowski and Decker 2012] [Kapetanakis et al. 2014], or extent applications that require tight synchronization between the AV and 3D content [Potetsianakis et al. 2014].

Our future work will target scenarios with 3D formats that support stream-based animations (e.g. BIFS). A possible implementation would be to carry the base 3D scene in a track, while the timed animations are placed in another track. The client will be able to render the scene and update it with an animation, whenever it occurs.

Finally, we are also working on tackling latency issues that occur when rendering complex 3D scenes. More specifically, we plan on implementing a progressive loading mechanism for large 3D documents. With this mechanism, the backbone of the scene is rendered as fast as possible, and it is updated as the rest of the elements are parsed. Alternatively, since the scene information is known in advance, a render time estimate will be calculated, thus allowing preparation of the scene in the background, for seamless playback.

References

BEHR, J., ESCHLER, P., JUNG, Y., AND ZÖLLNER, M. 2009. X3DOM: A DOM-based HTML5/X3D Integration Model. In *Proceedings of the 14th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '09, 127–135.

BERJON, R. 2006. Remote Events for XML (REX) 1.0. *World Wide Web Consortium, Working Draft WD-rex-20061013*.

CONCOLATO, C., AND POTETSIANAKIS, E. 2015. In-Browser XML Document Streaming. In *XML Prague 2015*, 197–205.

EVANS, A., ROMEO, M., BAHREHMAND, A., AGENJO, J., AND BLAT, J. 2014. 3D Graphics on The Web: A Survey. *Computers & Graphics 41*, 43–61.

JANKOWSKI, J., AND DECKER, S. 2012. A dual-mode user interface for accessing 3d content on the world wide web. In *Proceedings of the 21st international conference on World Wide Web*, ACM, 1047–1056.

KAPETANAKIS, K., PANAGIOTAKIS, S., MALAMOS, A., AND ZAMPOGLOU, M. 2014. Adaptive video streaming on top of Web3D: A bridging technology between X3DOM and MPEG-DASH. In *Telecommunications and Multimedia (TEMU), 2014 International Conference on*, 226–231.

LE FEUVRE, J., CONCOLATO, C., AND MOISSINAC, J.-C. 2007. GPAC: Open Source Multimedia Framework. In *Proceedings of the 15th International Conference on Multimedia*, ACM, New York, NY, USA, MULTIMEDIA '07, 1009–1012.

LE FEUVRE, J., CONCOLATO, C., DUFOURD, J.-C., BOUQUEAU, R., AND MOISSINAC, J.-C. 2011. Experimenting with Multimedia Advances Using GPAC. In *Proceedings of the 19th ACM International Conference on Multimedia*, ACM, New York, NY, USA, MM '11, 715–718.

LIMPER, M., WAGNER, S., STEIN, C., JUNG, Y., AND STORK, A. 2013. Fast Delivery of 3D Web Content: A Case Study. In *Proceedings of the 18th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '13, 11–17.

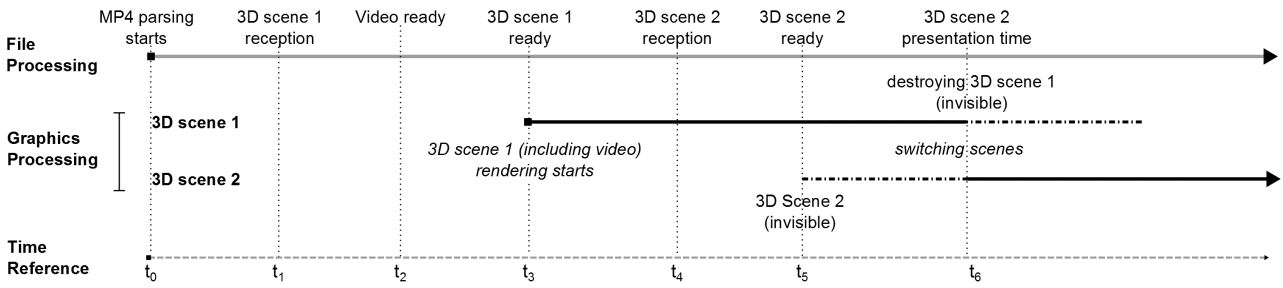
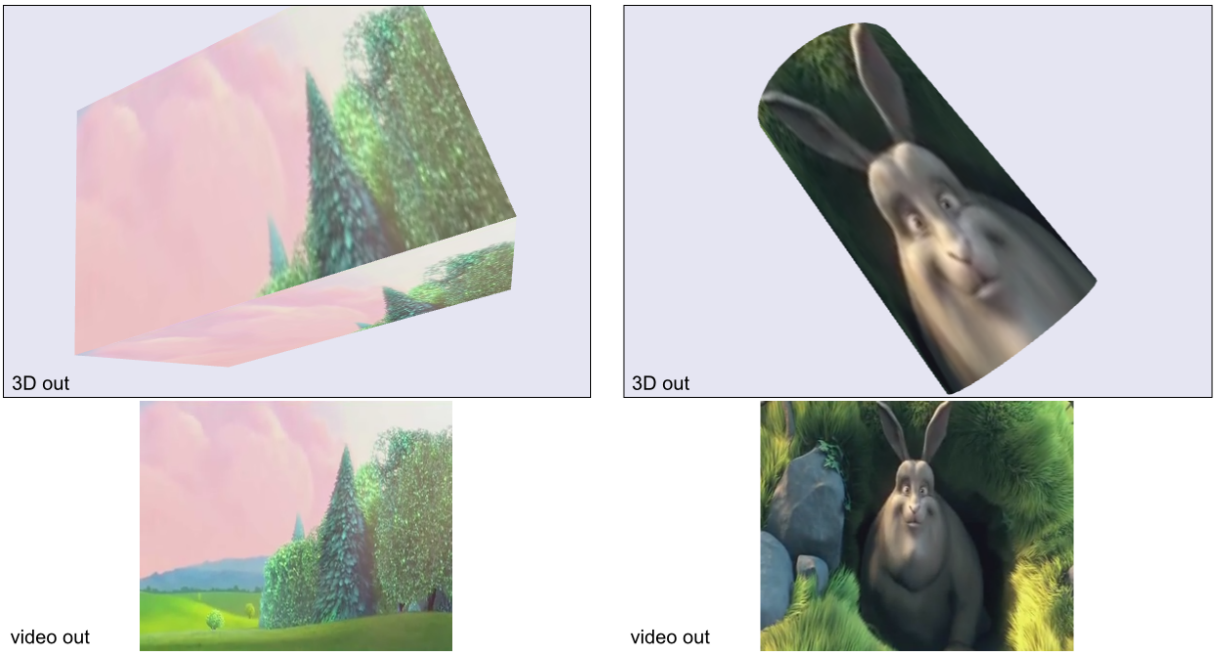


Figure 3: *Seamless 3D rendering timeline*



(a) *During rendering of the first 3D scene ($t = 6s$)*

(b) *During rendering of the second 3D scene ($t = 47s$)*

Figure 4: *Web application screenshots*

NIEDERMEIER, U., HEUER, J., HUTTER, A., STECHELE, W., AND KAUP, A. 2002. An MPEG-7 Tool For Compression And Streaming of XML Data. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, vol. 1, IEEE, 521–524.

POTETSIANAKIS, E., KSYLAKIS, E., AND TRIANTAFYLLIDIS, G. 2014. A Kinect-based Framework For Better User Experience In Real-Time Audiovisual Content Manipulation. In *Telecommunications and Multimedia (TEMU), 2014 International Conference on*, IEEE, 238–242.

SINGER, D. 2012. ISO/IEC 14496-12: 2012 Part 12: ISO Base Media File Format. *International Organization for Standardization*.