Grant Agreement No.: 687645
Research and Innovation action
Call Topic: H2020 ICT-19-2015

ORPHEUS
Object-Based Audio Experience

Object-based broadcasting – for European leadership in next generation audio experiences

D5.2: Implementation and documentation of intermediate version of object-based renderers and user interfaces

Version: v1.0

Deliverable type	D (Demonstrator)
Dissemination level	PU (Public)
Due date	28/02/2017
Submission date	3 March 2017
Lead editor	Benjamin Duval (Trinnov), Niels Bogaards (ElephantCandy)
Authors	Matthew Paradis (BBC), Niels Bogaards (ElephantCandy), Benjamin Duval (Trinnov), Martin Ragot (b<>com), Nikolas Färber (FhG), Michael Weitnauer (IRT)
Reviewers	Werner Bleisteiner (BR), Nicolas Epain (b<>com)
Work package, Task	WP5
Keywords	Renderers, User interfaces

Abstract

This document aims to present an overview of work that has been carried by the various partners in the ORPHEUS consortium related to the renderers and user interfaces for object-based broadcasting clients on the reception (consumer) side.

Demos and documentation is provided for development done related to WebAudio implementations, MPEG-H integration efforts and prototype client designs.

[End of abstract]

Document revision history

Version	Date	Description of change	List of contributor(s)
V0.1	15/02/2017	Initial structure	Benjamin Duval (Trinnov)
V0.2	16/02/2017	Description of the integration of MPEG-H library in AVR receiver	Benjamin Duval (Trinnov)
V0.3	21/02/2017	Description of web browser rendering examples	Matthew Paradis (BBC)
V0.4	22/02/2017	Description of iOS rendering implementation	Niels Bogaards (ECANDY)
V0.5	22/02/2017	UI design	Niels Bogaards (ECANDY), Martin Ragot (b<>com)
V0.6	23/02/2017	Chromium browser	Nikolaus Färber (FhG IIS)
V0.7	23/02/2017	Abstract, Executive Summary and Introduction drafts	Niels Bogaards (ECANDY)
V0.7.1	23/02/2017	Corrections and updates	Benjamin Duval (Trinnov)
V0.7.5	23/02/2017	Corrections and updates	Niels Bogaards (ECANDY), Martin Ragot (b<>com)
V0.8	24/02/2017	Conclusion, abbreviations and corrections	Benjamin Duval (Trinnov)
V0.8.5	24/02/2017	Suggestions to the conclusions	Niels Bogaards (ECANDY)
V0.9	24/02/2017	Cleanup, ready for review	Benjamin Duval (Trinnov)
V1.0	28/02/2017	Reviews integration	Werner Bleisteiner (BR), Nicolas Epain (b<>com), Benjamin Duval (Trinnov)

Disclaimer

Copyright notice

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive Summary

This document aims to present an overview of work that has been carried by the various partners in the ORPHEUS consortium related to the renderers and user interfaces for object-based broadcasting clients on the reception (consumer) side.

Most of the work so far has been to develop, implement and test individual components that will be integrated into the various clients used during Pilot 1. On the rendering side, a significant portion of the efforts in the last months has revolved around developing and integrating MPEG-H decoding libraries (as MPEG-H has been selected as the primary file format for object-based audio distribution in ORPHEUS) and the development of tools and components based on the WebAudio API, for rendering inside common web browsers.

User Interface design, implementation and validation was mainly done for the iOS mobile client app, but many of the concepts, visual designs etc. will be portable to other platforms.

Most of the work presented in this document refers to actual working prototypes of components of the object-based broadcasting chain. Where possible, movies, images or detailed documents are provided to convey the state of the development and provide context to what role the components will fulfil.

Table of Contents

List of supplied additional documents and demos

2.1_BBC_DASH_client.mp4

2.2_BBC_Adaptive_mixing.mov

4_MPEG-H_in_AVR.mp4

6_EC-app-demo-v1.mov

6_EC_UIDesign.pdf

Abbreviations

AAC	Advanced Audio Coding
ADM	Audio definition model
DASH	Dynamic Adaptive Streaming over HTTP
JSON	JavaScript Object Notation
MHAS	MPEG-H Audio Stream
MPD	Media Presentation Description
MPEG	Moving Picture Experts Group
MSE	Media Source Extension
OSS	Open Source Software
W3C	World Wide Web Consortium

1. Introduction

This document is dedicated to present an overview of work that has been carried by the various partners in the ORPHEUS consortium related to the renderers and user interfaces for object-based broadcasting clients on the reception (consumer) side.

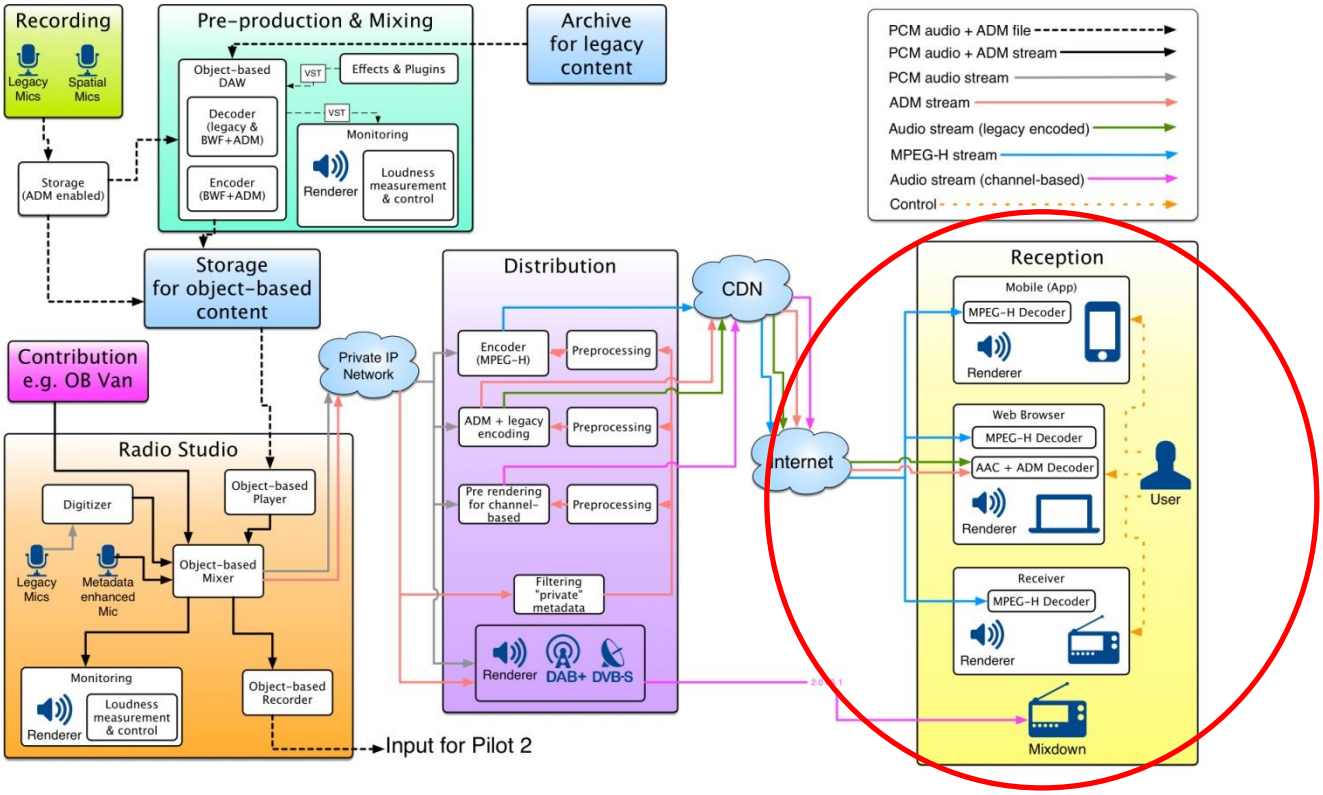


Figure 1: *reception macro-block in the ORPHEUS architecture*

The ORPHEUS object-based broadcasting system is an extensive system, where innovative solutions are required throughout. This work-package, WP5, focuses on the reception side: the systems that consumers will see and use to experience object-based broadcasts. The challenges are therefore not only technical (how to get the objects transported to the end device and render them into sound), but also ergonomic and conceptual: how can the plethora of options inherent to object-based audio's flexibility be presented in such a way that consumers will be able to fully experience and appreciate the improvements that object-based broadcasting has to offer.

As individual components need to be shown to work before they can be integrated successfully with others, most of the work so far has been to develop, implement and test individual components that will be integrated into the various clients that will be used during Pilot 1.

Two distribution formats were chosen for use during Pilot 1: MPEG-H (for use in the mobile app client and hardware receiver) and AAC+ADM (for common web browsers). A special case is the use of MPEG-H inside a custom version of the Chromium browser: this implementation serves mainly as a proof of concept for future MPEG-H support in commercial browsers.

For the integration of MPEG-H into the mobile app and the AV Receiver, FhG IIS has provided custom MPEG-H decoding libraries that can be integrated by the other partners. This integration work is ongoing, but simple test demonstrators have already been developed, paving the way for full MPEG-H functionality in Pilot 1.

Current versions of common browsers do not support MPEG-H decoding yet and within ORPHEUS, a combination of AAC encoded audio streams and ADM metadata streams will be used as an alternative. The emerging WebAudio API standard is used as a client-side rendering platform. Test

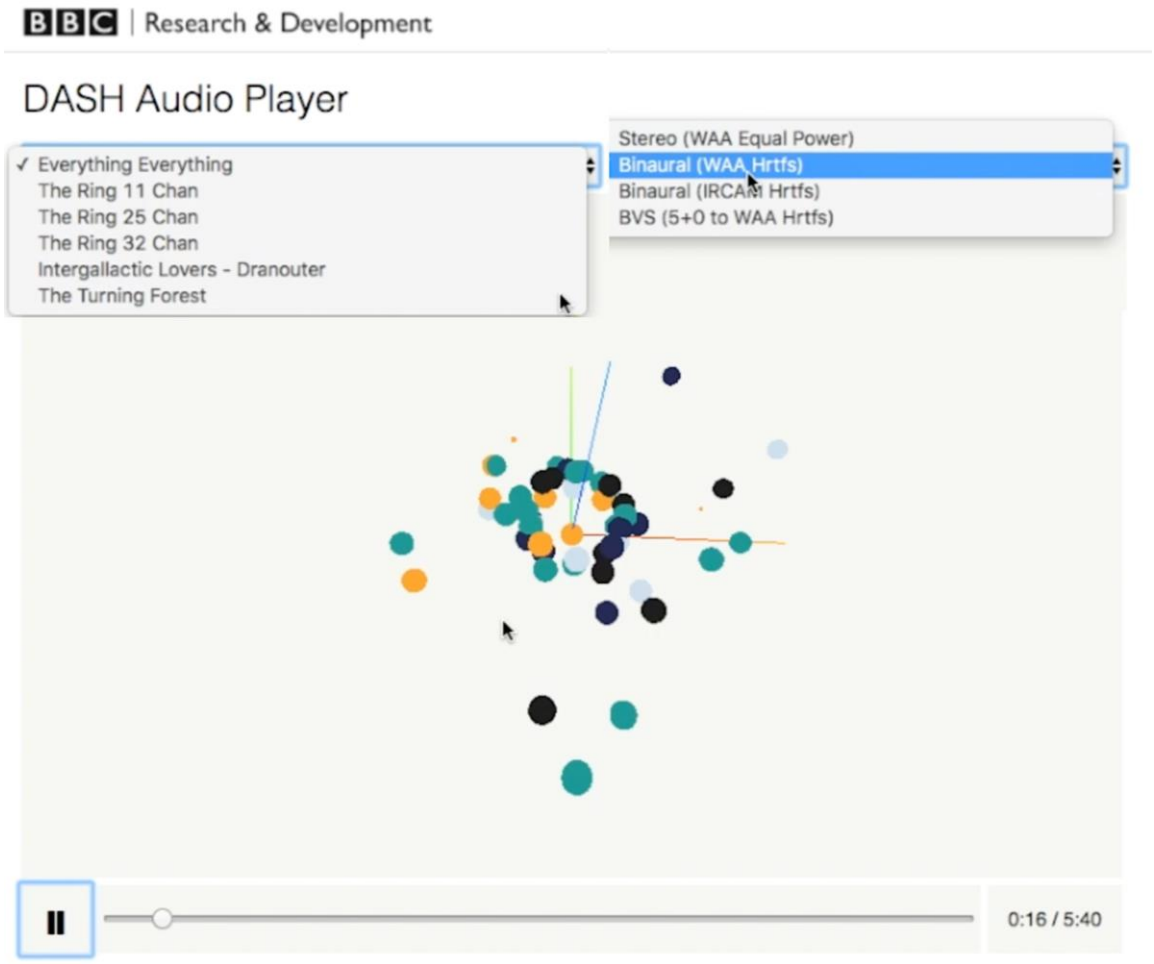
components have been developed for critical parts of the rendering that will be needed in Pilot 1.

User Interface design, implementation and validation was mainly done for the iOS mobile client app. Extensive experiments were carried out to test and validate user interface concepts and designs. At a later stage, concepts, visual designs and other results of the evaluation of the mobile app prototype can be re-used on other platforms.

Most of the work presented in this document refers to actual working prototypes of components of the object-based broadcasting chain. Where possible, movies, images or detailed documents are provided to convey the state of the development and provide context to what role the components will fulfil.

2. Rendering in the web browser

2.1. Dash client



>Demo file: 2.1_BBC_DASH_client.mp4

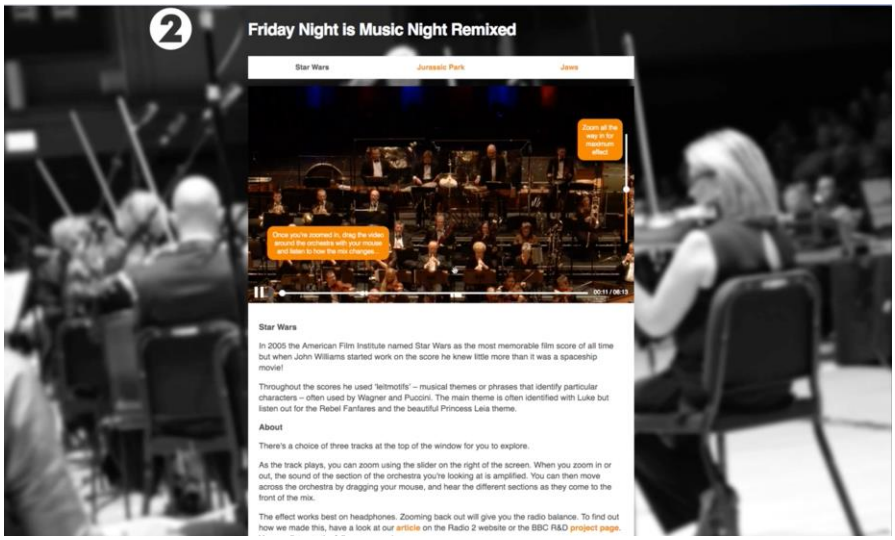
A client side JavaScript library has been developed at the BBC R&D, which allows the streaming of multiple audio objects (channels) to the browser. Streams are synchronized at playback along with segmented JSON formatted metadata. This solution allows the development of live and on-demand prototypes which can incorporate personalization and user interaction.

Multiple renderers have been developed that can be selected in real time to enable content to be presented according to the capabilities of the users' device or preferences.

This library allows to rapidly prototype new audio experiences, for example user interaction to select foreground/background audio or to deliver alternative audio streams. Multi object playback is now possible directly from the IPStudio platform, enabling the output from the ORPHEUS radio studio to be rendered in the browser, simulating a live or on demand broadcast.

In this example, the object metadata is being visualized in the browser to show object locations and movement.

2.2. Adaptive mixing



> **Demo file: 2.2_BBC_Adaptative_mixing.mov**

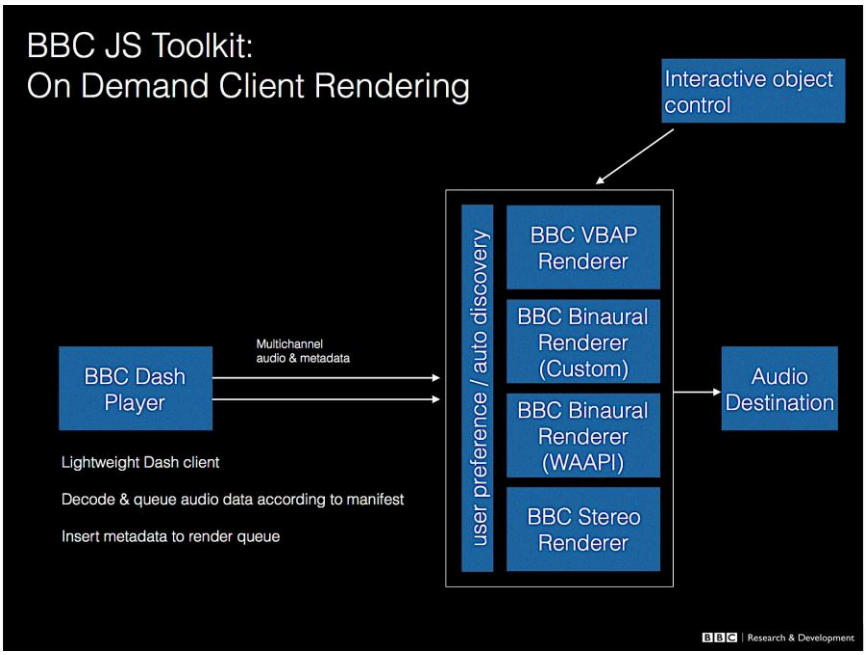


Figure 2: BBC JS Toolkit

To produce the audio remixing effect, multiple audio channels from the BBC outside broadcast truck are captured. Rather than streaming a complete stereo broadcast mix to the browser, the audio is splitted into its component parts (stems) which, when combined (or summed), produce the stereo mix. This means that at any time up to 24 channels of audio are streamed to the browser. The streams are generally separated by instrument group, for example, Piano, First Violins, Harp, Woodwinds...

When zooming in or panning the video, the zoom-level and pan-position in the video scene are tracked and modify the level and spatial position of the audio sources or objects in response. Static sources, such as the overall ambience sound of the venue, are louder when zoomed out and quieter when zoomed in. Conversely, dynamic sources such as individual instrument groups are louder when zoomed in and quieter when zoomed out, and also louder when closer to the pan-position of the user. The level of the dynamic sources in the mix is adjusted to reflect the distance of the sound source from the user's pan position in the scene.

Any audio processing is carried out completely in the browser using the Web Audio API.

3. Integration of MPEG-H Decoder in *Chromium* Browser

3.1. Background

Though the actual integration of MPEG-H into commercial browser deployments will have to be done by the browser providers (Microsoft, Google, Mozilla, Apple), it is important to first understand the technical feasibility and constraints of different integration options. For that purpose ORPHEUS developed a prototype client to verify our assumptions and the technical feasibility.

3.2. Development

The implemented approach for integrating MPEG-H into browsers follows the high-level architecture illustrated in Fig. 3. The overall streaming client consists of a web-application defined in HTML/CSS and JavaScript (JS). For DASH streaming it relies on the dash.js project [2] and uses the Media Source Extension (MSE) API [3] for media decoding and play-out. Because the MSE is a one-way API, it is necessary to handle the User Interaction (UI) outside the MSE within a UI-Manager (uimanager.js) to support user interactivity with MPEG-H. This UI-Manager has to parse and modify the incoming MP4 fragments, the included MHAS bitstream and the included Audio Scene Information (ASI) on the fly. A pre-requirement for this architecture is that MPEG-H is supported in the Media Engine of the browser and MPEG-H can be decoded using the MSE API.

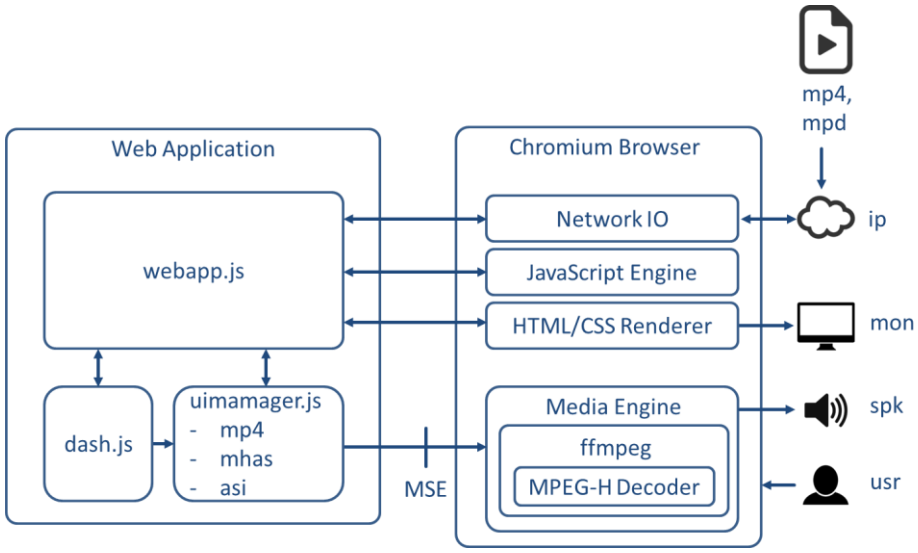


Figure 3: *High-level architecture for integrating MPEG-H into the Chromium browser*

The Chromium browser [1] is a natural choice to develop such a prototype because it is an Open Source Software (OSS) project and the basis for Google's popular Chrome browser. The Media Engine of Chromium is based on ffmpeg [4] which needed to be extended with the MPEG-H decoder from Fraunhofer IIS. The challenge of this project was the management of quite complex C/C++ projects (Chromium, ffmpeg) and the interaction with JavaScript applications and browser-APIs (dash.js, MSE). In addition the implementation of the UI-Manager in JavaScript did require optimization to achieve real-time operation. The task was completed successfully, i.e. MPEG-H encoded MP4 file segments were downloaded via DASH and played out on a Chromium browser. In addition, basic user interaction could be demonstrated by pressing buttons. The example application also provides extensive debugging information for the modified bitstream elements as illustrated in the screenshot in Fig. 4.

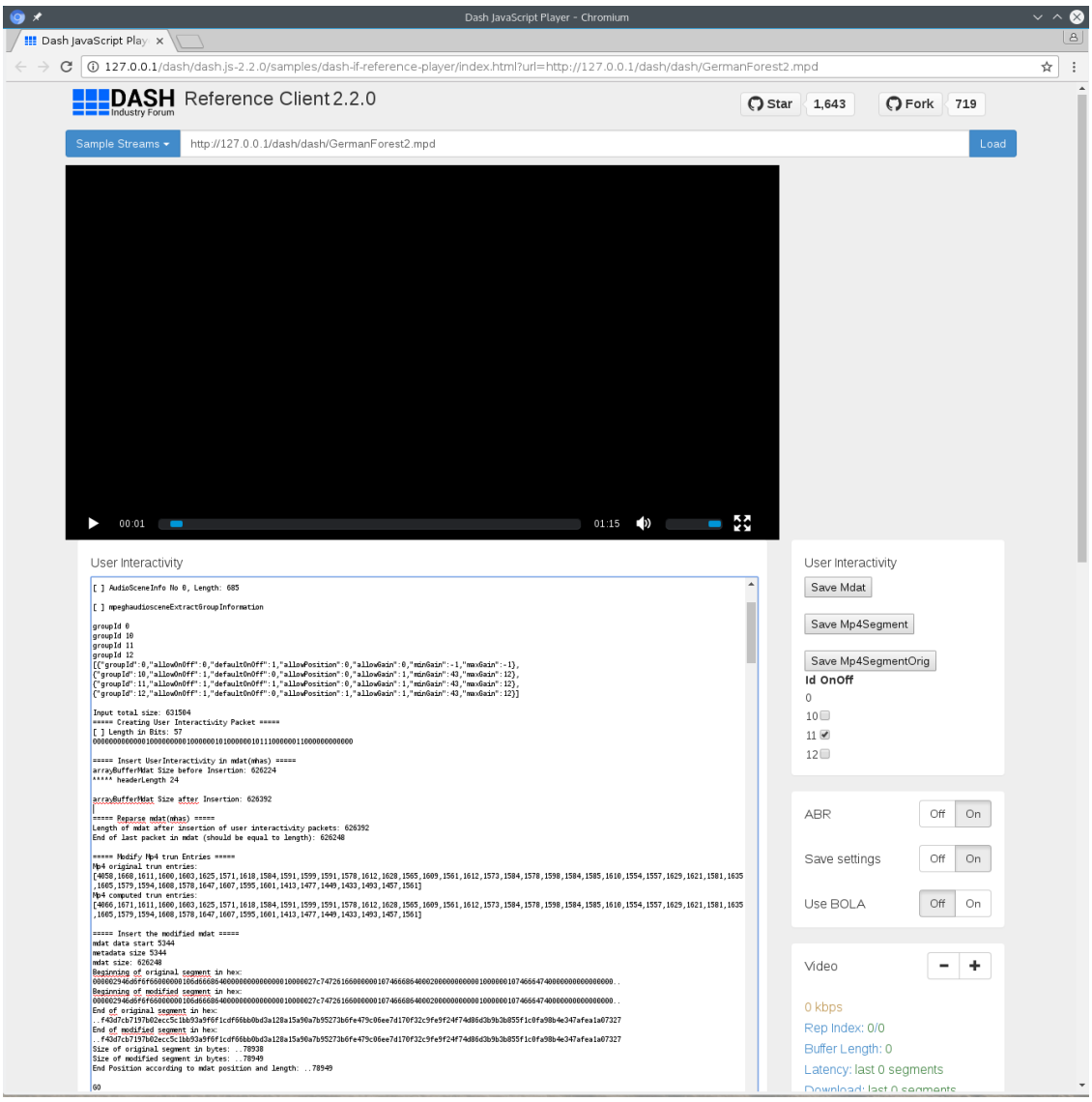


Figure 4: *Screenshot of Interactive Example Application playing MPEG-H in Chromium Browser*

3.3. Conclusion

Though the basic approach of implementing MPEG-H in browsers could indeed be verified, there are two main reasons why developments on the modified Chromium browser are not continued within ORPHEUS.

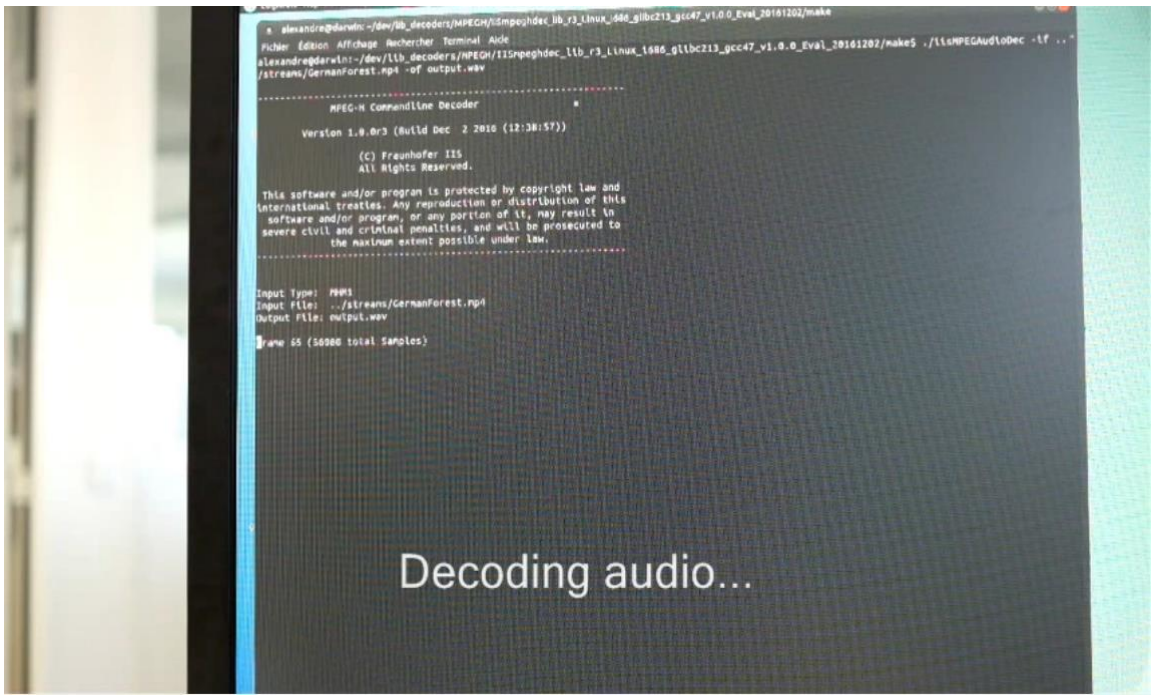
First, it became apparent that the delay introduced through the MSE makes a responsive user interaction difficult, i.e. the delay of >1 second between pushing a button and hearing the effect in the audio output is too high. Though some use-case may tolerate such a delay (e.g. switching language) and ideas to further reduce the delay are available, it was decided that it is a safer approach to extend the MSE towards bi-directional communication. However, as this requires long-term standardization activities at W3C and WAVE, this effort is followed up by Fraunhofer IIS outside ORPHEUS.

The second reason for discarding the Chromium browser as a client platform for MPEG-H playback is related to legal reasons. Chromium is a huge OSS project, which combines a multitude of other OSS projects. The analysis of the relevant OSS licenses revealed, that it will be almost impossible to achieve legal approval for re-distribution because of a licensing conflict with the included commercial MPEG-H decoder. Hence, even if the Chromium browser would finally work perfectly on a technical level, ORPHEUS would not be able to distribute this special build to the public as e.g. required for user trials.

3.4. References

[1] The Chromium Project, https://www.chromium.org/
[2] DASH.JS player, https://github.com/Dash-Industry-Forum/dash.js/wiki
[3] Media Source Extension API, https://www.w3.org/TR/media-source/
[4] ffmpeg, https://www.ffmpeg.org/

4. Intermediate implementation of Fraunhofer IIS MPEG-H decoder and renderer in the high-end AV receiver



>Demo file: 4_TRI_MPEG-H_in_AVR.mp4

The award winning Trinnov *Altitude32* AV receiver is an AV preamplifier with decoding capabilities for several 3D audio formats, object or (non-object) channel based. It can drive up to 32 loudspeakers, making it the most powerful audio processor on the market. It is used in luxury home-theatres to reproduce contents from any physical or network source. The goal is to integrate the MPEG-H decoder and renderer to add the support of object-audio contents.

Figure 5:*Trinnov Altitude32 processor*

The integration of the MPEG-H decoder and renderer library from Fraunhofer IIS into the Trinnov AVR receiver has begun by a software architecture improvement to gain computing efficiency, which allows the dedication of some CPU usage to the demanding decoding of MPEG-H. Then, the library has been compiled for receiver's platform, a small embedded PC running Linux. The decoder and renderer are now running offline on the target architecture.

The second part of the integration will be to put the decoder and renderer into the main audio processing chain to have the contents decoded in real time. For this, a test environment is needed, with a local DASH streaming module. This work is ongoing.

On the first example of MPEG-H audio file provided by FhG IIS, the decoder library allows to switch the language between 2 dialogue objects (one in German, the other in English), to change the volume and/or the prominence of the audio objects (the dialogue one and another object, a bee, that moves around the scene), and to change the position of the last audio object (the bee) within a range authorized by the sound designer.

Since the interface to control these parameters is not ready yet, the object properties can't yet be edited in real time: the source code of the example program has to be compiled again at every change, so they have for the moment to be set at the compilation time. This will be enabled after the second part of the integration.

5. Rendering implementation in the iOS mobile application

The mobile iOS application that will be developed by Elephantcandy for Pilot 1 and 2 receives object-based audio streams encoded in MPEG-H transmitted over MPEG-DASH. To this end a custom MPEG-DASH stream receiver has been developed, which is currently able to receive and play mp4/AAC audio streams. The system has been tested with streams and Media Presentation Descriptions (MPD files) from both Fraunhofer IIS and the BBC.

The mp4 container format will also be used for MPEG-H encoded streams. Internal test apps have been developed using the MPEG-H decoder library provided by Fraunhofer IIS. Work is well under way to integrate the MPEG-DASH streaming with the MPEG-H decoder.

To interactively manipulate the MPEG-H rendering of the object-based streams, the prototype iOS app described in section 6 will be interfaced with the MPEG-H UI Manager component.

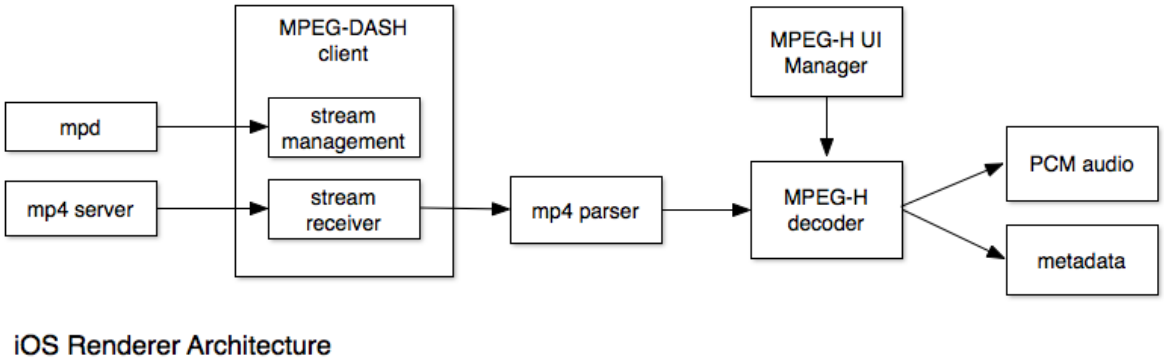


Figure 6: *iOS Rendering Architecture schematic*

6. User interface and personalisation

>Demo file: 6_EC_app-demo-v1.mov

>Additional file: 6_EC_UIDesign.pdf

To fully experience and profit from object-based broadcasting, consumers need to be provided with tools that strike a good balance between feature-richness, understandability and ease of use. In object-based audio, the objects are rendered to a final sound signal at the receiving end, allowing a plethora of options to be made available. Many of them involve concepts that may be new and challenging to all but the most technically experienced listeners. Apart from technical complexity, broadcast content may also be adapted to user's preferences and listening conditions.

To come up with user interfaces that are both convincing, engaging and ergonomically sound, an iterative workflow was planned involving b<>com and ECANDY, where a series of experiments has been devised for every step of the design. To date three tests have been performed at b<>com's Ergonomics Lab, leading to a first prototype iOS app for Pilot 1.

Firstly, a Benchmark Study was carried out on a wide selection of user interfaces that could serve as points of departure for aspects of the UI design. Evaluated products included podcast players, 3D sound processing plugins and radio websites. A second study focussed on the logic of the first drawn prototype app from ECANDY. For the third test, two iOS apps were developed to perform a specific A/B test on how rendering profiles and situations should best be linked.

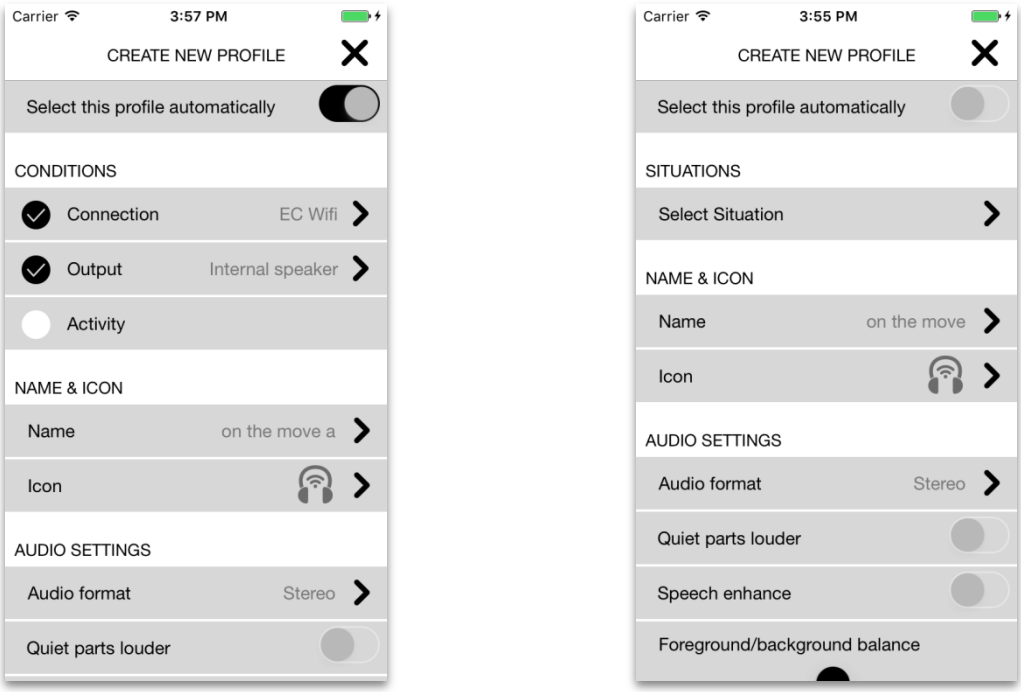


Figure 7:*Test scenario's A and B*

The user interface design has so far mainly been carried out by ECANDY and is focused on designing an iOS app for Pilots 1 and 2. The effort so far has resulted in the UI design and implementation of a prototype app for Pilot 1. Depending on the final content of Pilot 1, specific interaction elements may still need to be designed, but the general workflow and presented concepts has been defined.

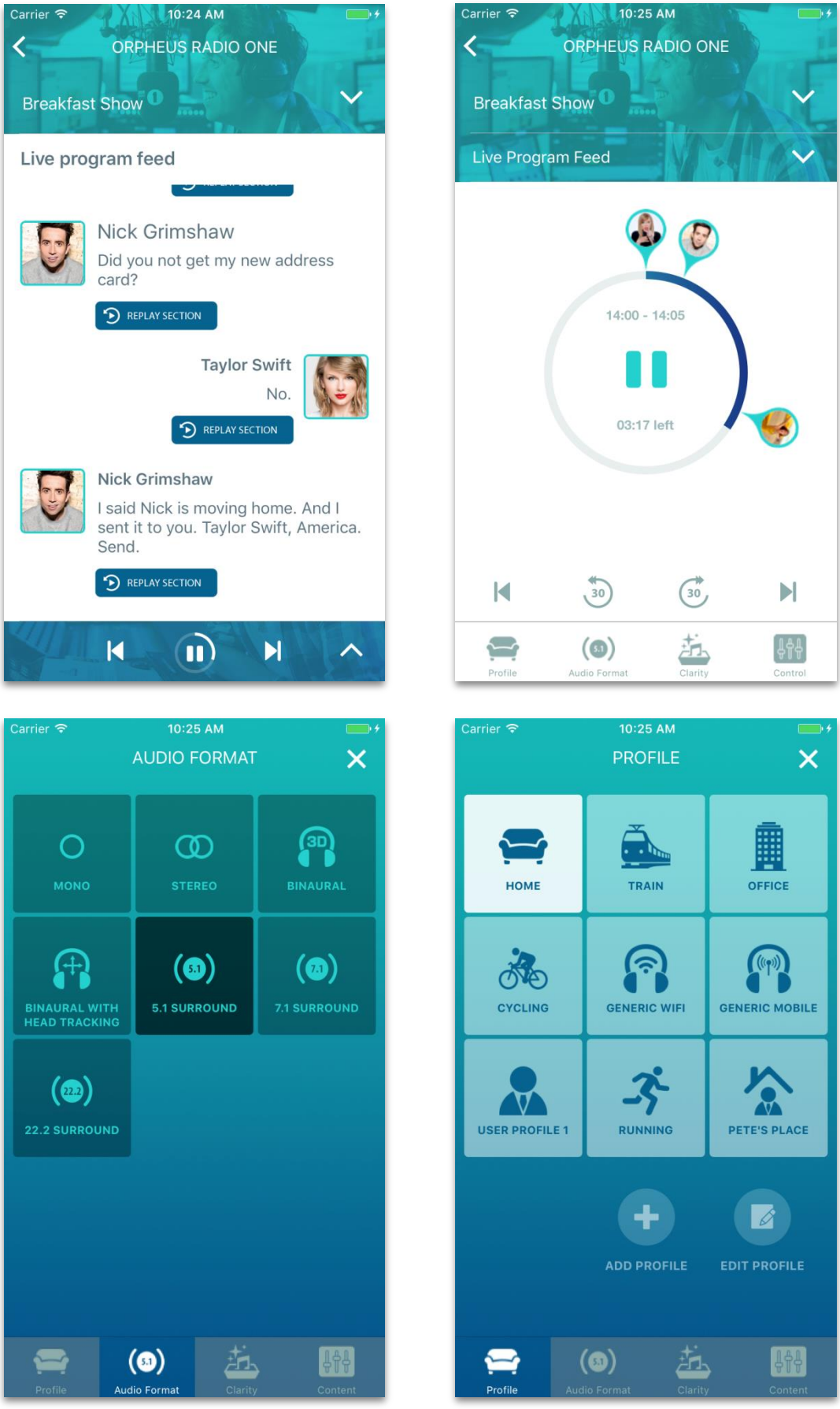


Figure 8 *Screenshots of the Pilot 1 Prototype iOS app*

7. Conclusions

The demonstrations presented in this document give an overview of the work currently carried by the partners of the ORPHEUS consortium in accordance with deliverable D5.1 and towards Pilot 1. The focus has been on the implementation and design of individual components, which will be integrated and connected over the course of the coming months. As such, not all work lends itself to compelling demonstrations and many developments are still ongoing. The documentation presented should however give a good impression of the progress that has been made, which is in line with the proposed planning.

The implementation and integration of the decoders and renderers is ongoing, especially MPEG-H, which has been selected as the primary file format for object-based audio distribution in ORPHEUS. Its integration is being carried out in the different devices. A prototype User Interface for Pilot 1 for a mobile client app has been designed, and is currently under implementation and validation.

The developments on renderers and user interfaces are progressing well towards a system that is usable for Pilot 1, and in the coming months the individual components will be connected. Findings from the present developments, such as concepts arising from the user interface design, can be applied to other platforms, and it is expected that MPEG-H over MPEG-DASH will prove itself as a powerful and flexible way to deliver object-based audio broadcasting to consumers.

[end of document]