Deliverable

Project Acronym:	**IMAC**
Grant Agreement number:	761974
Project Title:	*Immersive Accessibility*

D4.2-Audio Production Tools

Revision: 2.0

Authors: Theresa Liebl (IRT), Peter tho Pesch (IRT), Ronald Mies (IRT), Kimiasadat Mirehbar (Anglatecnic)

Delivery date: M24 (30-09-19)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 761974

Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Abstract:

This document provides an overview of developed ImAc Audio Production Tools. It accompanies the software products and describe their main features and their availability.

REVISION HISTORY

Revision	Date	Author	Organisation	Description
0.1	31-07-2018	Kimiasadat Mirehbar	ANGLA	Anglatecnic input
0.2	22-08-2018	Theresa Liebl	IRT	IRT input
0.3	22-08-2018	Theresa Liebl	IRT	Merges IRT and Anglatecnic input
0.4	28-08-2018	Peter tho Pesch	IRT	Comments added, minor text changes.
0.5	04-09-2018	Theresa Liebl	IRT	Text and formatting updates
0.9	06-09-2018	Peter tho Pesch	IRT	Text review of all sections, major formatting update Version finalized for review
1.0	06-09-2018	Peter tho Pesch	IRT	First version for publication
1.1	21-08-2019	Ronald Mies	IRT	Template for final version
1.2	13-09-2019	Ronald Mies	IRT	IRT and Anglatecnic updates included
1.3	26-09-2019	Ronald Mies	IRT	Version for internal review
2.0	30-09-2019	Ronald Mies	IRT	Final version for submission

EXECUTIVE SUMMARY

This deliverable documents the audio production tools developed in T4.2 and the accompanying work. The software tools that have been developed, are the actual deliverables – this document describes their concepts and functionalities. The tools that have been investigated and developed in the project (Web AD editor, cloud renderer, object-based audio editor) are described put into the context of the overall ImAc audio production workflow.

In T4.2 of the ImAc project, the consortium is looking for an environment in which professional users would be able to create and edit audio description files for 360° videos in an interactive manner.

The requirements for this environment were extracted from the work done in T2.3 and have been gradually completed during the project (see also ImAc deliverable D2.3 [5]).

CONTRIBUTORS

First Name	Last Name	Company	e-Mail
Enric	Torres Feixas	ANGLA	enric@anglatecnic.com
Kimiasadat	Mirehbar	ANGLA	kimia@anglatecnic.com
Àlex	Soler de Ferrater	ANGLA	alex@anglatecnic.com
Theresa	Liebl	IRT	theresa.liebl@irt.de
Peter	tho Pesch	IRT	peter.thopesch@irt.de
Ronald	Mies	IRT	ronald.mies@irt.de
Matthias	Urban	IRT	matthias.urban@irt.de
Robert	Schwering	IRT	robert.schwering@irt.de

CONTENTS

Revision History 1
Executive Summary 2
Contributors 3
Tables of Figures and tables 6
List of acronyms 7
1. Introduction 8
1.1. Purpose of this document 8
1.2. Scope of this document 8
1.3. Status of this document 8
1.4. Relation with other ImAc activities 8
2. Audio Description workflow 10
3. Web AD Editor 12
3.1. Introduction 12
3.2. Structure and concept 12
3.3. First Iteration vs. second iteration 13
4. Cloud Renderer 15
4.1. Introduction 15
4.2. Concept 15
4.3. Web API 17
5. Object-based Audio Editor 24
5.1. Introduction 24
5.2. Concept 25
5.3. Functional description 27
References 31
Annex I Installation of the Web AD editor code 32
Annex II Web AD Editor Quick User manual 36
1. Introduction 38
2. What is new 38
3. Before starting 38
4. How to start 39
5. How to produce Audio Description 41
6. More options 48

Annex III Object-Based Audio Editor Quick User manual 56
General functionalities 56
Audio object editing 57

TABLES OF FIGURES AND TABLES

List of figures

Figure 1 - Dependencies between tasks and deliverables in ImAc 9
Figure 2 - Picture of Audio (Description) Workflow 10
Figure 3 - AD Editor access via ACM 13
Figure 4 – Main ImAc ACM login page 13
Figure 5 – Eddie interface synchronized with a DAW 24
Figure 6 – Simplified structure of ADM metadata which describe an audio mix with an optional AD mix 26
Figure 7 – Eddie menu option “Scene” 27
Figure 8 – Eddie timeline with keyframes 28
Figure 9 – Eddie item position space with three items of different height and position in the scene 30
Figure 10 – Eddie menu option “File” 56
Figure 11 – Eddie menu option “Edit” 56
Figure 12 – Eddie menu option “View” 57
Figure 13 – Eddie menu option “Windows” 57
Figure 14 – Eddie Transport Widget 57
Figure 15 – Eddie menu option “Scene” 58
Figure 16 – Eddie timeline with keyframes 59
Figure 17 – Eddie item position space with three items of different height and position in the scene 61

List of tables

Table 1 – Comparison of the web AD Editor iterations 14
Table 2 - List of audio output streams generated by the cloud renderer 17
Table 3 – Available properties in the Item Editor 28
Table 4 – Available properties in the Keyframe Editor 29
Table 5 – Available properties in the Item Editor 58
Table 6 – Available properties in the Keyframe Editor 60

LIST OF ACRONYMS

Acronym	Description
ACM	Accessibility Content Manager
AD	Audio Description
ADM	Audio Definition Model
API	Application Programming Interface
AST	Audio Subtitles
DAW	Digital Audio Workflow
ED	Editor Interface
FOA	First Order Ambisonics
MMC	Midi Machine Control
MTC	Midi Time Code
SL	Sign Language
ST	Subtitle
TC	Time Code
WP	Work Package

1. INTRODUCTION

1.1. Purpose of this document

This document provides an overview of the tools developed in ImAc for audio production for immersive content, specifically for providing Audio Description and Audio Subtitles with it. These tools focus on professional users such as broadcasters and access service providers. The document accompanies the software developments and describes their main features and availability.

1.2. Scope of this document

This document aims to be a guide through the progress made in development of the audio production tools of the ImAc platform. Whereas the (software) tools are the actual deliverables, this document describes their concepts and functionalities and puts them in the context of the overall ImAc audio workflow.

1.3. Status of this document

This deliverable has been developed as an iterative document reflecting the status of the audio production tools in ImAc. The developments from the first iteration were reflected in the first version of this deliverable (version 1.0, delivered in September 2018, M12).

This version is the final version of this deliverable and addresses the improvements and status of the audio tools in second iteration.

1.4. Relation with other ImAc activities

Figure 1 shows the relation between Task T4.2, in which this deliverable has been produced, and other ImAc activities:

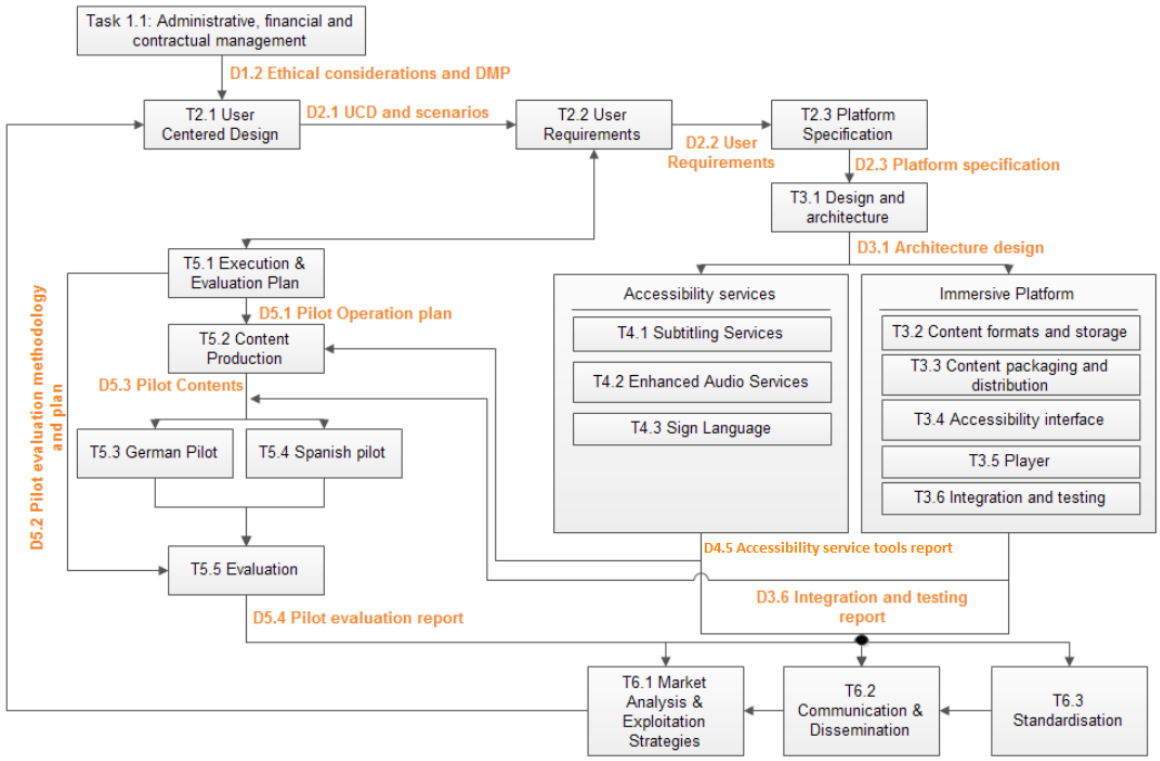


Figure 1 - Dependencies between tasks and deliverables in ImAc

Related activities include especially:

- D4.5: In deliverable 4.5 [1] the status of all production tools is documented.
- Integration in Workflow: Audio production tools are part of the overall ImAc workflow.

2. AUDIO DESCRIPTION WORKFLOW

This section provides an overview of the integration of the audio production workflow within the ImAc platform. Different views on the technical components in ImAc, including AD production tools, can also be found in ImAc Deliverable D3.1 [2].

For audio production in ImAc two strategies were pursued:

- Establishment of a functional workflow to create AD to perform user tests in the ImAc project, based on state-of-the-art audio formats;
- Use of a standardised production format for object-based audio production and creation of a tool which enables a fully object-based audio production workflow.

There is a large difference in the workflow depending on the audio format of the main mix. When main audio has been produced as a channel-based production (e.g. stereo or 5.1 surround sound) or a First Order Ambisonics (FOA) production, the Accessibility Content Manager (ACM, see D3.2 [3]) orchestrates the audio pre-processing and packaging tools. This workflow has been installed and runs fully automated after the human editor has completed the AD production and triggered the pre-processing. It is also used for all ImAc productions. When the main audio has been produced as object-based audio, the audio pre-processing requires additional steps which do not run automated. The two workflows are described separately in the following.

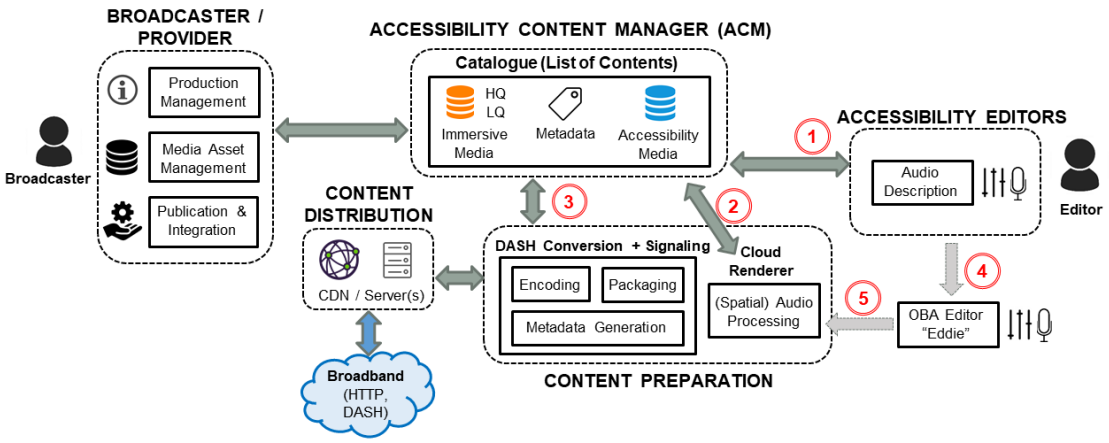


Figure 2 - Picture of Audio (Description) Workflow

Workflow including channel-based main audio

The only interface of the audio production workflow to the ImAc platform is the ACM, see Figure 2. The ACM provides access to 360° content ("immersive media") and manages the produced AD audio files and their metadata (no. 1 in Figure 2, for further details of the ACM see D3.2 [2]). It further triggers the cloud renderer for audio pre-processing and monitors the pre-processed files (no. 2 in Figure 2). Finally, the ACM provides the packager with all assets and metadata when publishing content to the ImAc portal (no. 3 in Figure 2).

The authoring and (if applicable) the recording of the AD assets is done using the Web AD editor (refer to section 3). Here, AD is produced as an object-based asset. Basically, the Web AD editor can be understood as a very specific object-based audio editor that is tailored to the specific needs of AD objects. The production can be done using a low-res video asset with the main audio

mix in stereo. The result of this processing step is an object-based audio package containing the AD objects.

Once the AD object-based audio package has been produced, it must be pre-processed and rendered into the FOA format for distribution. The cloud renderer (refer to section 4) merges the AD objects and the main audio (preferably delivered as 5.1 surround sound) into a virtual 3D audio scene and renders a set of FOA streams that are ready for playout. The parameters for these streams are mainly controlled by the AD package, enabling the different AD variations (AD gain and AD mode, refer to section 4.2 for further information). The rendering result (FOA streams plus metadata) is sent back to the ACM. As written above, the ACM may then use these streams when publishing content to the ImAc portal.

Workflow of a fully object-based production

Merging object-based audio with channel-based main audio, as described in the workflow above, is a special feature that was added to the cloud renderer to support the ImAc use cases, because regular audio productions at the moment typically use stereo, 5.1 surround sound or FOA audio. The long-term objective, however, is a fully object-based audio production (OBA, see section 5.2). When using OBA, the cloud renderer would not have to merge different audio formats at its input, but rather only take care of the rendering part of a complete OBA scene. This is the case, when audio production is done fully object-based, i.e. not only the AD production, but also the main audio of the program.

In a fully object-based audio production, the OBA editor Eddie (refer to section 5) would be used to author all relevant audio assets into a 3D audio scene. This editor does not include special attributes or features that are dedicated to AD, but it can include the AD objects provided by the Web AD editor, just as any other audio objects (no. 4 in Figure 2). However, the import of these objects requires manual user actions. The complete scene can again be rendered by the cloud renderer (no. 5 in Figure 2), but in comparison to the channel-based production workflow without automatically feeding the rendered streams back into the ACM.

The next sections describe the audio production tools developed in ImAc:

- Web AD Editor
- Cloud renderer
- Object-based audio editor

Larger user manuals are included as annex at the end of the document.

3. WEB AD EDITOR

3.1. Introduction

The ImAc Web AD Editor is an online editor for AD production with full support for 360º videos. In addition to this deliverable, various documentation is available:

- Quick user manual aimed at professional users who wish to start working with the tool
- Factsheet and posters for dissemination

The editor allows users to create and author audio description files and add all metadata that is required for the AD service in ImAc. The audio description files can be exported as object-based audio assets/objects in an ImAc-specific format.

The chapter describes the main functionality of the Web AD Editor, overviews the new characteristics of the tool in comparison with the first iteration release and gives the complete user manual for professional users who wish to use the tool is included in Annex II Web AD Editor Quick User manual.

3.2. Structure and concept

The structure of and relation between the elements are quite intuitive and clear.

The previously assigned AD tasks to the human editor via the Accessibility Content Manager (ACM) appear in the Editor Interface (ED), which redirects the user to this editor. When the editor's production is finished the task is saved in ACM. The full user manual of this editor is presented as an annex of this deliverable.

The main concept and characteristics for the Web AD editor are as follows:

- Supporting 360º videos
- Cloud-based
- Based on working with LQ videos transferred from ACM
- Easy to work time-cueing
- Soundwave of the video
- Audio segmentation
- Different preview modes
- Integrated to different modes of 360º AD types: dynamic, static and classic
- Personalised user experience due to extremely user-based SETTINGS
- Possibility to split and join audio segments recorded by the audio describer
- Possibility to determine the information required by Audio Renderer (another unit in the workflow to be explained further): AD level, gain
- Responsive environment

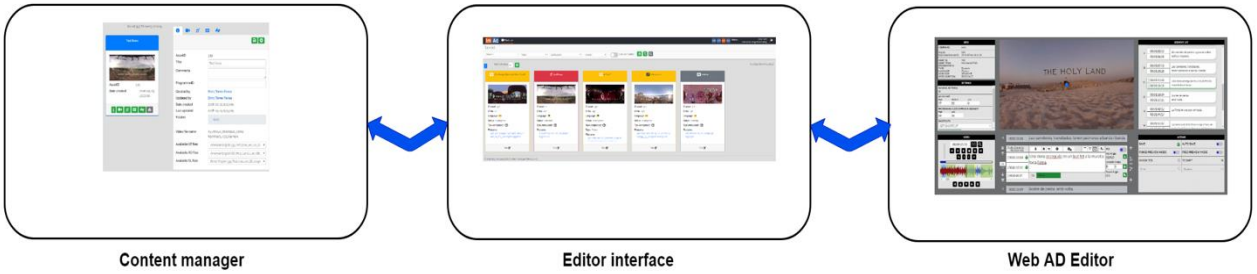


Figure 3 - AD Editor access via ACM

ONCE THE USER HAS EVERYTHING INSTALLED (SEE ANNEX I INSTALLATION OF THE WEB AD EDITOR CODEREFERENCES

[1] ImAc deliverable D5.4 - Pilot evaluation report, Revision 0.4, November 2018, http://www.imac-project.eu/documentation/deliverables/

[2] ImAc deliverable D3.1 – Architecture Design, Revision 2.2, January 2019, http://www.imac-project.eu/documentation/deliverables/

[3] ImAc deliverable D3.2 - Accessibility Content Manager, Revision 2.4, July 2019, http://www.imac-project.eu/documentation/deliverables/

[4] ImAc deliverable D2.2 - User Requirements, Revision 1.0, November 2018, http://www.imac-project.eu/documentation/deliverables/

[5] ImAc deliverable D2.3 – Platform Specification, Revision 1.13, July 2019, http://www.imac-project.eu/documentation/deliverables/

Annex I Installation of the Web AD editor code) and set, it is possible to access the Web Interface (note: installing code is not compulsory for access, username and password are enough). To access the Web AD editor, the user has to login via the ACM, as follows:

1. Open your preferred browser (Chrome and Firefox work better with the interface).
2. Browse to the path where the editor has been installed, e.g.: http://yourServerAddress/editor/
3. A login page will appear:

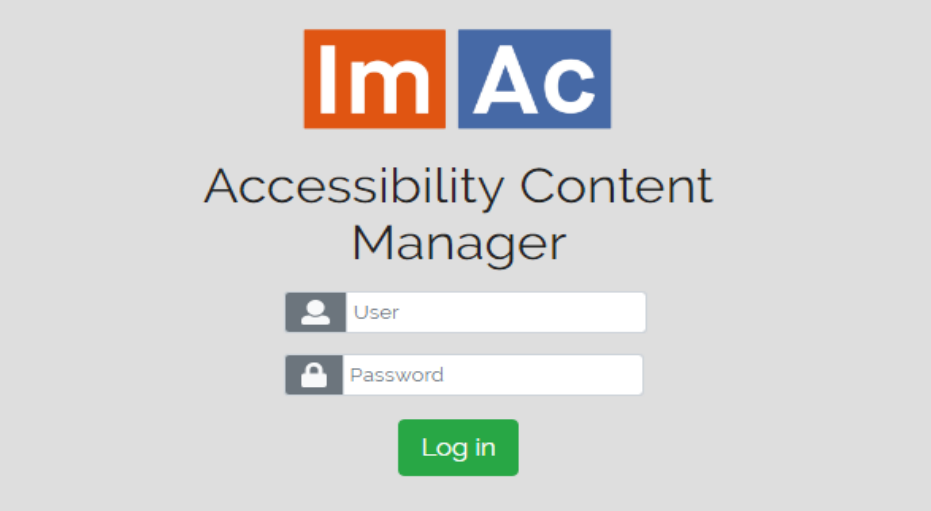
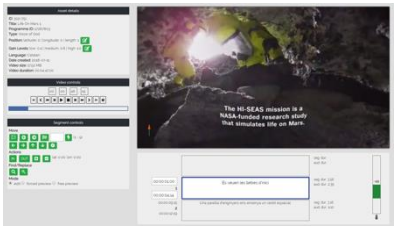


Figure 4 – Main ImAc ACM login page

3.3. First Iteration vs. second iteration

During the ImAc project, the tools have been developed in two iteration cycles. During the second iteration, various improvements have been made in all the tools, which is true for the Web AD Editor as well.

Table 1 shows a comparison between the first and second iteration. This table only reflects the improvements made along second iteration and not all the editor characteristics.

Web AD Editor: first vs. Second iteration		
Feature title		
User friendly environment	No	Yes
Prototype vs. exploitation	Prototype	Exploitation

Shortcut buttons	No	Yes
Time cue checking	No	Yes
Video soundwave	No	Yes
User personalized settings	No	Yes
Copy and paste angles	No	Yes
Join and split segment text	No	Yes
Location angle index	No	Yes

Table 1 – Comparison of the web AD Editor iterations

The web AD editor realises the objectives and professional user requirements (as addressed in WP2 of the project). The full list of user requirements can be found in D2.2 [9].

A detailed overview of the Web AD editor's functionality can be found in the user manual, which is included in this document as Annex II Web AD Editor Quick User manual.

4. CLOUD RENDERER

4.1. Introduction

This chapter describes the concept, structure and the user interface of the cloud renderer. The software is hosted on AWS (Amazon Web Services) in a serverless architecture. This means, that rendering requests are started in a volatile environment, data being released when the rendering event is terminated. The render results are, however, stored on S3 (Simple Storage Service) on AWS, enabling consistent data availability.

The cloud renderer merges audio productions – no matter if channel-based or object-based – with AD objects, based on the required settings of all the AD segments, including positions (in the 3D space) and gain levels of the AD. The interface between the Web AD editor and the cloud renderer is accessible via an HTTP PUT request, using a JSON body containing all relevant information the renderer request handler needs to execute a job. The processing results are communicated to the requestor, i.e. the AD Editor, by a HTTP POST Request callback.

The cloud renderer serves as a proof of concept. In the ImAc end-to-end chain, the renderer serves as processing engine, accessible as cloud service on-demand.

4.2. Concept

The main goals of the cloud renderer engine are:

1. To keep the requirements on the client device (player on the end-user side) low – the cloud renderer produces all requested audio-layouts as specified in the Web AD Editor software. The rendering results are encoded in a predefined audio format, which allows DASH-encoding in a next step (no feature of the renderer).
2. One rendering job produces the results of all output formats as defined in the request call. Currently the renderer is capable of rendering to stereo, binaural, FOA (First Order Ambisonics), or any multichannel loudspeaker setup.
3. Depending on the complexity of the render request the serverless architecture allows scaling of the rendering process to keep the processing time low. Scaling is enabled by default. However, hard limits are introduced to keep scaling costs at a defined maximum.

The generated AD tracks are described in an XML-structured descriptive document (index.xml), which defines general position definitions as well as required gain steps valid for all AD tracks (in the HEAD segment of the XML). Additionally, each AD “segment” is described in this XML document to enable features like “ducking” (i.e. decreasing the overall gain of the audio scene during the lifetime of an AD object). Dynamic positioning is being described within the segment, as well. This is described in detail in section 4.3.

Segmentation of AD tracks:

- AD tracks are separated in segments. This way they are already produced sectional and the segments can be described and addressed separately. The segmentation of audio in chunks and additional descriptive data allows the “transformation” into audio objects as required by the rendering process. It’s also the easiest solution for a dynamic positioning of the AD, as each AD segment can be placed individually in the 3D space.
- The AD tracks are handled like other individual audio sources but with some additional characteristics (will be only rendered for AD streams, ImAc extensions, etc.)

The cloud renderer addresses three interactive parameters which are relevant for an immersive AD experience:

- The "*position*" parameter defines where within the 3D space the different AD tracks will be placed. In ImAc three different positions are defined, two of which are fixed whereas one position is meant to be set "dynamically" by the AD compositor. The fix positions foresee equal placement for all AD segments. The positions are set in the Web AD Editor software and might be adjusted by the person in charge. The default values describe one position being placed over the listener's head (sometimes referred to as "Voice of God", labelled as "classic" in the ImAc context), the other, being placed right next to the listener like a "friend on sofa", labelled "static" in this context. The "dynamic" position allows the editor to place the AD segment without any default suggestion. Intentionally, the dynamic position shall be placed where the "action" is happening.
- The *"gain"* parameter defines the audio level of the AD object. The value ranges from 0 to 1, where 0 means silence and 1 means full gain. The available gain values (e.g. high, medium and low) must be set by the professional user by means of the Web AD Editor.
- The Ducking (defined as "*Dipping*" in the Web AD Editor) parameter defines the attenuation of all active audio objects but the one owning this parameter. The HEAD section in the XML-structured descriptive document includes all possible values of ducking (DippingLevels), which are defined by the Web AD Editor. Furthermore, a boolean parameter "*KeepDipping*" allows a follow-up ducking of all active objects even after termination of the owning one. This feature is helpful if distances between AD segments are too short and gain level regulation would be too fast and distracting.

The Renderer supports the following audio formats:

- PCM/WAV - uncompressed audio
- AAC-LC – low complexity AAC, containerized in MP4
- AC3 – Dolby Encoding

The encapsulation in DASH foresees AAC in the ImAc Project, which makes this codec the preferred one. In case of First Order Ambisonics (FOA), the audio output is additionally also multiplexed with the 360° video source, enabling a FOA-Video result file, mainly for verification purposes.

By combining all relevant output formats with each possible variant of parameters we get a list of the following generated audio output streams as shown in Table 2.

No	Format	Audio mix	AD Description
1	Stereo	Main mix	-
2	Stereo	Main mix + AD	Pos: center, volume: low
3	Stereo	Main mix + AD	Pos: center, volume: medium
4	Stereo	Main mix + AD	Pos: center, volume: high
5	FOA	Main mix	-
6	FOA	Main mix + AD	Pos: classic, volume: low
7	FOA	Main mix + AD	Pos: classic, volume: medium

8	FOA	Main mix + AD	Pos: classic, volume: high
9	FOA	Main mix + AD	Pos: Static, volume: low
10	FOA	Main mix + AD	Pos: Static, volume: medium
11	FOA	Main mix + AD	Pos: Static, volume: high
12	FOA	Main mix + AD	Pos: dynamic, volume: low
13	FOA	Main mix + AD	Pos: dynamic, volume: medium
14	FOA	Main mix + AD	Pos: dynamic, volume: high
15	Binaural (0°)	Main mix	-
16	Binaural (0°)	Main mix + AD	Pos: classic, volume: low
17	Binaural (0°)	Main mix + AD	Pos: classic, volume: medium
18	Binaural (0°)	Main mix + AD	Pos: classic, volume: high
19	Binaural (0°)	Main mix + AD	Pos: Static, volume: low
20	Binaural (0°)	Main mix + AD	Pos: Static, volume: medium
21	Binaural (0°)	Main mix + AD	Pos: Static, volume: high
22	Binaural (0°)	Main mix + AD	Pos: dynamic, volume: low
23	Binaural (0°)	Main mix + AD	Pos: dynamic, volume: medium
24	Binaural (0°)	Main mix + AD	Pos: dynamic, volume: high

Table 2 - List of audio output streams generated by the cloud renderer

4.3. Web API

As mentioned above, the cloud-based renderer is triggered via its web API, listening for an HTTP PUT request, containing a JSON body with the required invocation data. This is described in the following. For processing the AD-data created using the Web AD Editor (see section 3), a metadata format was defined to describe AD audio objects. This format basically adds AD relevant parameters to basic audio object parameters as specified for example by the ADM format. In ImAc this data is transported in an XML structure, stored in a separate file in a data package containing this descriptive file and the audio assets.

An API invocation performing a HTTP PUT requests, follows the following structure:

```
{
"audioFileTemplate": "{outputFormat}/{audioFileFormat}/{uid}-
{mode}.{extension}",
"videoURL": "https://xxx/video_hq.mp4",
"assetId": "350",
"audioFileFormats": [ "aac" ],
 "outputFormats": [ "stereo", "binaural", "FOA" ],
"audioDescriptionFile": "https://xxx/172225.ad",
"audioDescriptionLanguage": "en_GB",
"callbackUrl": https://xxx/audioRenderingReport.php
}
```

In detail, this structure contains following information:

- The aforementioned package containing all audio assets and audio metadata is found under the URL in “audioDescriptionFile”.
- The “assetID” and “audioDescriptionLanguage” are parameters, helping to identify the render output after job completion.
- The “audioFileTemplate” describes the naming convention of finished results.
- The “videoURL” gives access to the high-quality source media file (containing both video and audio), which is needed to extract the origin audio channels for rendering.
- “audioFileFormats” and “outputFormats” list the required formats which shall be produced.
- The “callbackUrl” is the address for status responses and completion messages.

The index.xml file, shipped within the “audioDescriptionFile” contains the metadata required to generate object-based audio sequences for the rendering process. All relevant parameters in the XML structure are described below.

Basic structure

- The AD object description is divided into a “Head” and a “Segments” section.
- The “Head” section contains parameters that apply to all AD segments like the id of the program the AD relates to, or the language. Additionally, it contains the description of all AD variations to be rendered.
- The “Segments” section contains the description of all single AD segments. The AD is produced and provided to the renderer in segments. Each segment is typically a few seconds long and covers e.g. the AD that is placed in a gap in a dialog.

```
<Audiodescription360 lang="ca_ES">
    <Head>
        ...
    </Head>
    <Segments>
        <Segment id="s1">
            ...
        </Segment>
        ......
    </Segments>
</Audiodescription360>
```

Head section – basic elements

- “ID”, “Title” and “ProgrammeID” are not processed by the cloud renderer but may help identifying content and processes in case of manual error tracing.
- The “language” element sets the language for this AD track. It is also used to label the rendered streams correctly.

```
<Head>
    <Id>75</Id>
    <Title>The programs title</Title>
    <ProgrammeID>17267803</ProgrammeID>
    <language>ca_ES</language>

</Head>
```

Head section – AD position

- The "AdPositions" element contains a list of AD object positions in 3D space. For each position a separate output stream will be rendered.
- Each "position" element in the "AdPositions" list contains a name that is used for labelling the corresponding output stream(s) and a position. The position defines where in the 3D audio scene the AD speaker (audio object) shall be placed. The parameters "longitude" (direction in the horizontal plane), "latitude" (the tilt angle) and "length" (the distance to the listener) are used.
- The position named "Dynamic" does not contain position information. It signals a mode, where each AD segment can be placed at a different position in the Head section of the XML; the position information for that mode is included in the "segment" element (see below).

```
<Head>
    ...
    <AdPositions>
        <position>
            <name>VOG</name>
            <staticCoordinates>
                <longitude>0</longitude>
                <latitude>0</latitude>
                <length>1</length>
            </staticCoordinates>
        </position>
        <position>
            <name>Friend</name>
            <staticCoordinates>
                <longitude>90</longitude>
                <latitude>0</latitude>
                <length>1</length>
            </staticCoordinates>
        </position>
        <position>
            <name>Dynamic</name>
        </position>
    </AdPositions>
</Head>
```

Head section – AD gain

The gain "level" consists of a describing parameter ("name") and an adjustable "value" between 0 and 1, where 0 means silence and 1 full gain. In the Head section three gain levels are defined (set by the Web AD editor).

```
<Head>
    ...
    <AdGainLevels>
        <level>
            <name>low</name>
            <value>0.4</value>
        </level>
        <level>
            <name>medium</name>
            <value>0.6</value>
        </level>
        <level>
            <name>high</name>
            <value>1.0</value>
        </level>
    </AdGainLevels>
    ...
</Head>
```

Head section – Dipping Levels

Dipping – also known as Ducking – describes the gain decrease of all running audio objects to enhance the perceivability of the audio object in focus.

The parameter "DippingLevels" describes the list of possible dipping multipliers. All active objects get 'dipped' by multiplying the dipping value to the active gain value during existence of the dipping object. Here a certain value gets connected to a key, which will be used in the "Segments" section to describe the dipping intensity.

```
<Head>
 . . .
 <DippingLevels>
    <level>
        <name>none</name>
        <value>1</value>
    </level>
    <level>
        <name>low</name>
        <value>0.6</value>
    </level>
    <level>
        <name>high</name>
        <value>0.3</value>
    </level>
 </DippingLevels>
 . . .
</Head>
```

Segment description

The following example demonstrates a full description of an AD segment. Containing its ID “Segment id”, a descriptive text (“Text”), the time range with its starting and end point (“TCIN” resp. “TCOUT”) and the position attribute of the dynamic placement ("DynamicPosition”). The global parameters set in the head section are valid for each segment element.

```
<Segments>
 . . .
   <Segment id="s1">
       <Text>Títol, lletres blanques sobre fons negre: Jaunt
             Ryot.</Text>
       <Comments/>
       <Audio>audio_segment_189_1.mp4</Audio>
       <TCIN>00:00:00.000</TCIN>
       <TCOUT>00:00:03.560</TCOUT>
       <Dipping>high</Dipping>
       <KeepDipping>0</KeepDipping>
       <Duration>3.54</Duration>
       <DynamicPosition>
           <Longitude>0</Longitude>
           <Latitude>0</Latitude>
           <Length>1</Length>
       </DynamicPosition>
   </Segment>
 . . .
</Segments>
```

Callbacks

The cloud renderer sends notifications back to the Web AD Editor to a given callback-URL. The callback is sent via HTTP POST and includes error messages in case of break-ups during the rendering job. After successful rendering information about the result files is sent as well.

The following is an example of an erroneous termination:

```
{ "Records":

[ { ..., "MessageId": "b4841654-2214-562d-9d05-43ef9ba7d409",
"Message": "ValueError: could not fetch configurations from
https://xxx/xxx.mp4", "MessageAttributes": {}, "Type":
"Notification", ..., "Subject": null } }

]

}
```

After each finished file a callback is triggered, containing information about the location and metadata of completed files as well as the ones being still in progress.

```
{
  "Message":
[
{
"status": "processing",
"outputFormat": "0+2+0",
"sourceURL": "https://xxx/163355.ad",
"uid": "7356d6578b5edf36d460e28116c83238",
"parameters": { "s9": { "Pos": "TRADITIONAL", "Gain": "low" } },
"assetID": "583",
"contains_video_channels": true,
"output": [
"https://xxx/stereo/aac/583VIDEO_GainlowPosTRADITIONAL-static-
0.aac" ],
"audioDescriptionLanguage": "pl_PL"
},
…
]
}
```


Using the “assetID” and “audioDescriptionLanguage” a unique identification of completed files is possible. Additional metadata contains information about the audio file format and the output format. A list of the download URLs to the rendered files is described in “output”. The parameter “uid” is set and used internally by the renderer and only needed by the requestor for communication with IRT in case of error tracking.

Processing

The renderer itself is built in a functional architecture (FAAS, Function as a Service). Isolated tasks are processed in their own functional environments. Internal communication uses S3 buckets for shared data (assets) and shared database tables for metadata exchange. Therefore, parallel execution is made easy, not only for execution of multiple jobs, but also for executing tasks within a job in parallel, which fastens up the rendering process. The job is divided in the parts aggregation, preparation and the actual rendering, each part depending on the results of the former one. The aggregation part is performing the collection of all assets as described in the job invocation. If data is invalid or incomplete, an exit callback is triggered, informing about the error by sending a HTTP POST request to the given callback URL. The preparation part converts the given metadata, which is a proprietary format, based on the requirements for AD processing as defined in this project, into a format fitting the renderer interface. An object-based scene is generated for all options specified by the Web AD editor. The audio channels from the original video are used as “background” audio sources, a so-called “audio bed” onto which the AD assets are placed. In case of a surround audio bed, the audio channels from the video are placed in the standard 5.1 layout, i.e. horizontal layout of 0-30-110-270-330 (center, left-front,

left-surround, right-surround, right-front). The LFE channel is being dropped. Each combination of position, gain and dipping (ducking) variants is stored in a separate scene file. The rendering part executes renderer tasks for each scene and for each given output format (the list might contain “stereo”, “binaural” and “FOA” (First Order Ambisonics)). Depending on the editor’s choice, a multitude of rendering tasks is being executed in parallel. Even though the number of result files might grow to the dozens, due to the parallel design the job is performed in just a few minutes – notably depending on the content length.

The result files are made available for download: a hard link to the storage location on S3 is prepared after a successful rendering. After each completion a list of available result files is sent to the callback URL as HTTP POST request. After the last completion a final callback message is sent, signalling a successful job termination.

5. OBJECT-BASED AUDIO EDITOR

5.1. Introduction

This chapter describes the user interface and the functionality of the object-based audio editor called *Eddie*.

The object-based editor with graphical user interface was developed by IRT. It allows users to create and author object-based audio scenes, control the object positions as well as other metadata, such as gain. It also controls the signal processing/rendering by sending metadata and control commands via TCP/IP to a renderer instance. The editor can be synchronized to any Digital Audio Workflow (DAW) via MTC (Midi Time Code) and MMC (Midi Machine Control).

The graphical user interface consists of two main parts: the item position space which shows position, height and movement of the objects placed in the scene (right-hand side in Figure 5) and the timeline which shows automations and the timing of each object in the scene (left-hand side in Figure 5).

The software was developed as a lab tool for internal usage.

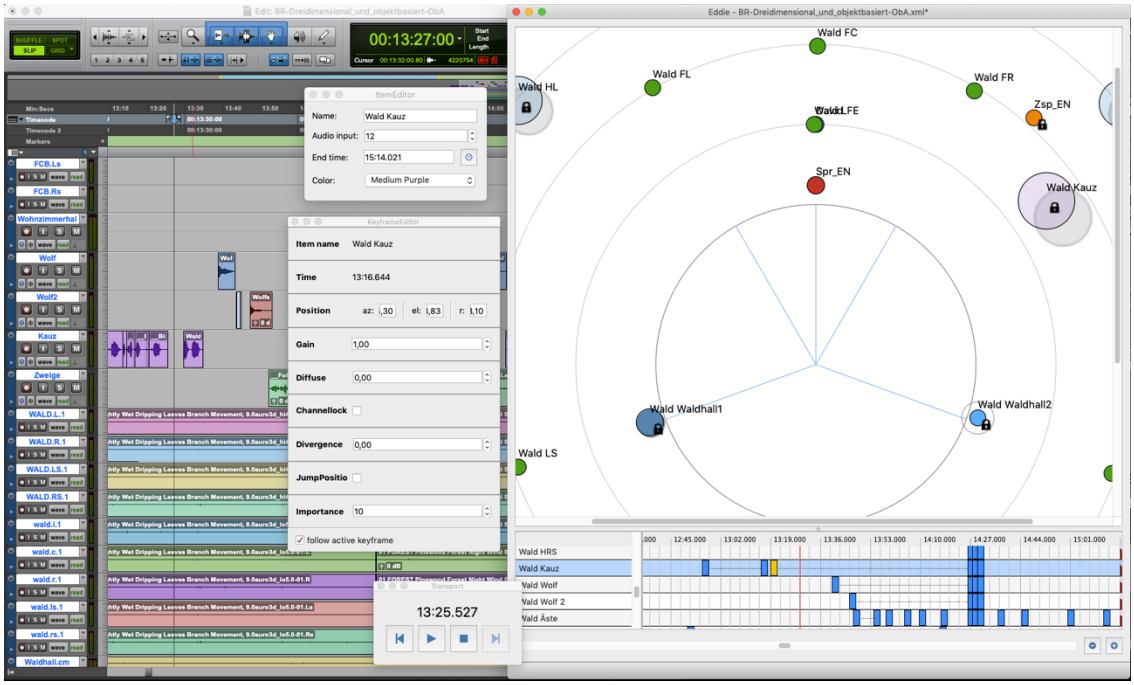


Figure 5 – Eddie interface synchronized with a DAW

5.2. Object-based audio

The term 'object-based media' has become common to describe the representation of media content by a set of individual assets, together with metadata describing their relationships and associations. At the point of consumption these objects can be assembled to create an overall user experience. The precise combination of objects can be flexible and responsive to user, environmental and platform specific factors.

Essentially, the goal is to capture the creative intent of the producer and carry as much information as possible, required or desired, from the production side to the end-user, to ensure the best recreation possible on the consumer side. To achieve this, the final product of an audio production process will be an audio scene that is in turn composed of several audio objects. The metadata associated with each object includes, but is not limited to, the target position of the audio signal, its target loudness and a description of its actual content.

For playback, the object-based content needs to be 'rendered' to the reproduction layout, such as a multi-channel loudspeaker set-up. The term 'rendering' describes the process of generating actual loudspeaker signals from the object-based audio scene. This processing takes into account the target positions of the audio objects, as well as the positions of the speakers in the reproduction room. It may further take into account user interaction such as a change of position or level.

An object-based approach can serve end-users more effectively, by optimizing the experience to best suit their access requirements, the characteristics of their playback platform and the playback environment or personal preferences of the listener. Moreover, it will be highly beneficial for content producers, as workflows can be streamlined and only a single production needs to be created, archived and transmitted in order to support and serve a multitude of potential target devices and environments. This is enabled by the simple fact that the metadata of individual objects can be modified and adjusted, either by the end-user or along the production and transmission chain, without the need to change the audio material itself.

Accessibility features are designed into the format, such as the ability for users to control the relative playback volume of voice commentaries over the rest of the programme audio. Object-based audio (OBA) provides the opportunity to bring immersive audio to the listener and can provide personalized experiences. For the first time, listeners can interact with the audio and adapt it to their preferences, e.g. for accessibility reasons. OBA is more bit rate efficient than previous technologies, so these new features don't come at the cost of higher bit rates or additional audio streams.

5.3. Concept

The initial intention for audio production in ImAc was to realize a whole production including AD with object-based audio (OBA). OBA offers a lot of benefits compared to traditional channel-based audio productions (see section 5.2). Especially for productions with additional content (like AD) and for providing interactive options to the end user (e.g. level control of the AD) OBA is specifically suitable. Only one production needs to be created, including the audio tracks and corresponding metadata to describe all the additional options. To create and use OBA productions, a complete object-based workflow with a metadata model like the Audio Definition Model (ADM) is necessary.

The ADM is a standardised metadata model for describing both the technical properties of an audio production as well as its content on a semantic level. ADM metadata can be attached to audio files to ensure the audio is correctly handled. ADM data is typically stored as XML which is widely used and human readable. An ADM element is represented by an XML element, with its parameters specified either using attributes or sub-elements. The ADM aims to be future proof by providing enough flexibility in its design to not limit the size and scope of definitions. It also aims to be easily extendable in the future where new parameters can be added without breaking the structure or backward compatibility.

Figure 6 shows a simplified example of an audio mix with an optional AD mix described with ADM. It contains two *AudioProgrammes*, which can be interpreted as potentially user-selectable presets. *AudioProgramme 1* is the main mix, *AudioProgramme 2* is a modified version of the main mix with audio description. Both programmes share and reuse the same audio tracks. The *AudioObjects* in each programme allow for a fine-grained description of the technical properties of each object. For example, the object containing the audio description is in fact made up using three audio description segments.

The *AudioContent* elements assign semantic meaning to the content and group the objects accordingly. The audio description object, for example, is explicitly defined as being a "dialogue/audio description" content.

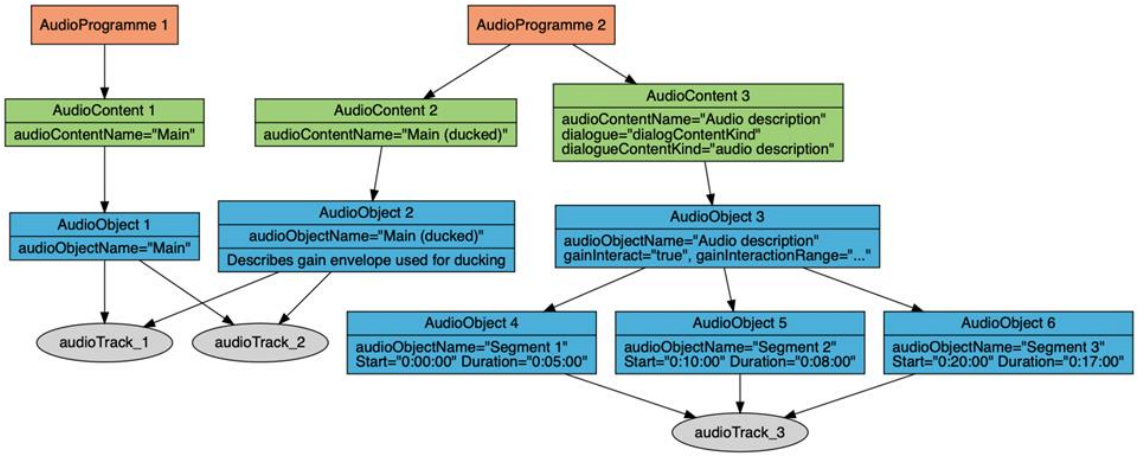


Figure 6 – Simplified structure of ADM metadata which describe an audio mix with an optional AD mix

To realize a complete object-based workflow with ADM, an OBA production tool supporting ADM was needed. Since no such tool existed, ADM should be integrated in Eddie, the OBA editor with graphical user interface developed by IRT. The main goal for the ImAc project was to have an audio production tool which can generate object-based ADM audio scenes. To achieve this, multiple tasks had to be carried out.

First and most importantly, internal data structures in Eddie had to be completely refactored and adapted to match the possibilities and constraints defined by the ADM. This includes, but is not limited to, adding support for polar coordinate systems and lifetime management of individual audio element as, in the ADM, each audio element may have a start time and duration. Previously all metadata in Eddie was dynamic and tied to keyframes. That means it could be changed freely over the course of an audio production. As this is not supported by the ADM for all metadata entities, the notion of *static metadata* had to be introduced. This *static metadata* can be changed by the producer, but its values do not change over time. An example is the mapping of audio elements to audio assets or audio tracks. While the ADM allows multiple audio elements and metadata sets to share a single audio asset, it imposes the constraint that the mapping of an audio asset to a given audio element cannot change within an audio scene.

To make this data structure available to the producer, new user interface elements had to be created and parts of the existing UI had to be adapted. Backward compatibility to support productions already created with the old format was achieved by offering an import functionality.

Finally, the export of ADM from the editor was added to support writing of standardized ADM files in XML format.

With the integration of the basic functions of ADM in the Eddie a big step was made for realizing a complete OBA workflow. During the developments on the Eddie new possibilities to create object-based ADM audio scenes arose. It became clear during the project that a separate software for creating OBA scenes is not the best way to generate a good audio production workflow. Rather, it is much more efficient to integrate the creation of the OBA scene directly in the DAW used for the audio production.

5.4. Functional description

In this chapter, a few of the new functionalities of the object-based audio editor are introduced. The full User Manual for Eddie can be found in Annex III Object-Based Audio Editor Quick User manual.

- Items (objects) can be added, named, duplicated or removed from the scene. They can be placed in any spot in the item position space and moved around the scene.

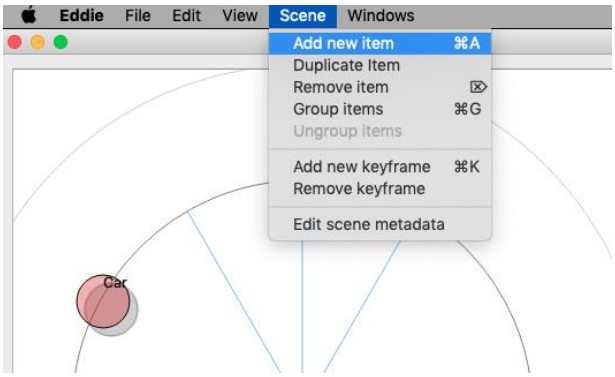
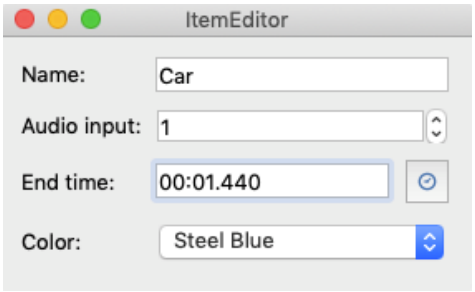


Figure 7 – Eddie menu option "Scene"

- In the Item Editor static (time independent) object metadata can be edited. The available properties are described in Table 3.

Item Editor	
	
Name	A descriptive name of the object.

Audio input	This field allows to assign an audio track to this object. The number refers to an input channel of the renderer, which receives audio from the DAW via some kind of audio interface.
End time	If set, this time specifies when the object stops to exist. The button next to the input field allows to quickly set the current time as the end time.
Color	Display color of the object within the editor.

Table 3 – Available properties in the Item Editor

- Keyframes for automation (time-dependent metadata changes) can be added and removed for each item in the scene in the timeline.
  To make automations for item positions, just jump to a time position of your choice, add a keyframe and place the desired position of the item. You can proceed in the same manner for more position changes.

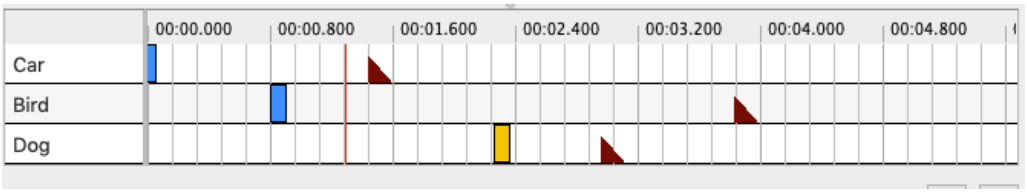


Figure 8 – Eddie timeline with keyframes

- In the Keyframe Editor the dynamic (time dependent) object metadata can be edited. The available properties are described in Table 4.

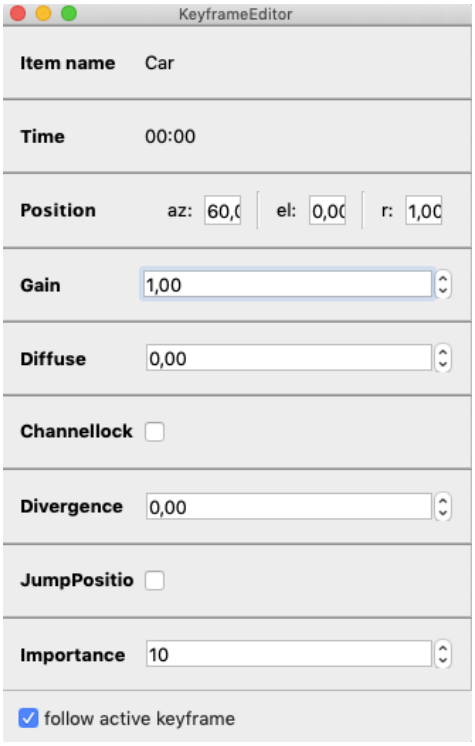
Keyframe Editor	
	
Position	The position of the audio object is represented by polar coordinates. The listener will be located at the origin of the scene. Azimuth (“az”) describes the angle between the forward listening position and the object in the horizontal plane. Elevation (“el”) describes the angle between listener and object in the vertical plane. “r” is the radius and describes the distance between the listener and the object.
Gain	The volume of the audio object within the range of [0, 1], 1 being maximum gain.
Diffuse	Describes the intended diffuseness of an audio object int range of [0, 1], where 0 means not diffuse.
Channellock	Is used to set the ADM channel lock property. It instructs the renderer to lock an audio object to the nearest channel or speaker instead of normal rendering.
Divergence	Adjusts the balance between the object’s specified position and two other virtual objects placed symmetrically around the original object in the range [0, 1]. This allows to fade smoothly from rendering a single object to creating a sound source using stereophonic phantom imaging. A value of 0 means no divergence.
JumpPosition	Is used for smooth transitions from the position of this keyframe to the position of the next keyframe.
Importance	The importance of an audio object in the range [0, 10]. 10 is the most important, 0 the least. Allows a renderer, for example, to discard objects below a certain threshold of importance.

Table 4 – Available properties in the Keyframe Editor

The AD variations which were used in the ImAc user tests allow the user to select between different positions for the AD speaker and also allow to change its gain compared to the rest of the scene. In Eddie, these parameters are set by the position and the gain property of the AD audio source.

- In the item position space, the interface shows a 2D view for object position with 3D visuals for the height of objects. Large and light circles (Bird in Figure 9) show a higher position than small and darker ones (Car). Circles with an additional circle around them (Dog) are items positioned under the middle-layer.

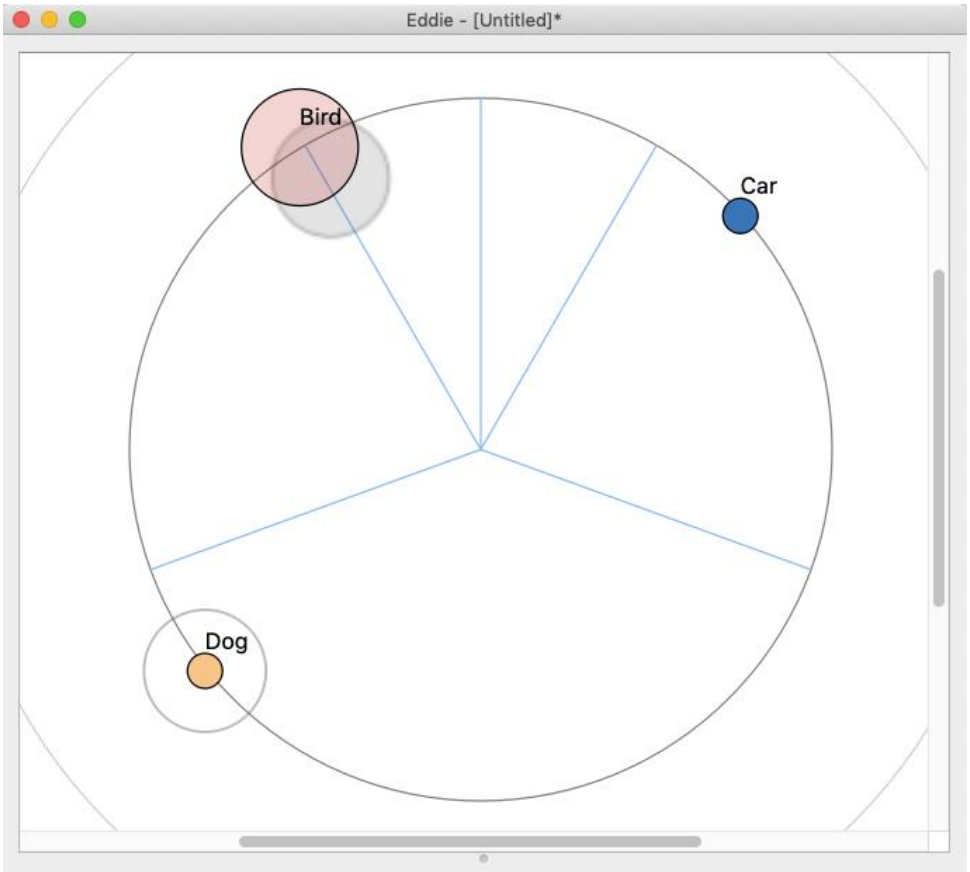


Figure 9 – Eddie item position space with three items of different height and position in the scene

REFERENCES

[6] ImAc deliverable D5.4 - Pilot evaluation report, Revision 0.4, November 2018, http://www.imac-project.eu/documentation/deliverables/

[7] ImAc deliverable D3.1 – Architecture Design, Revision 2.2, January 2019, http://www.imac-project.eu/documentation/deliverables/

[8] ImAc deliverable D3.2 - Accessibility Content Manager, Revision 2.4, July 2019, http://www.imac-project.eu/documentation/deliverables/

[9] ImAc deliverable D2.2 - User Requirements, Revision 1.0, November 2018, http://www.imac-project.eu/documentation/deliverables/

[10] ImAc deliverable D2.3 – Platform Specification, Revision 1.13, July 2019, http://www.imac-project.eu/documentation/deliverables/

ANNEX I INSTALLATION OF THE WEB AD EDITOR CODE

Before describing the code installation procedure, it is essential to mention that the code is only available to project consortium and EC reviewers and it is not publicly available due to rights belonging to developers.

Installation

A Linux operating system, Debian 9 distribution has been installed. The process of installation is completed using ”root” user.

Packages

The user requires to install several packages into the server in order to have all the tools and programs for running:

```
apt-get update

apt-get install vim screen rsync ntp less man net-tools apache2 php php7.0-mysql php7.0-curl php7.0-gettext php7.0-mbstring php7.0-xml openssl mysql-server

php7.0-odbc curl apt-transport-https php7.0-sybase freetds-common libsybdb5 exim4
```

This will install the main tools for the system:

- Apache Web Server
- MySQL Database (MariaDB)
- PHP 7.0 engine
- System Tools

Each package must be configured.

Note: “Vim” editor is utilised to edit each file. After finishing, In order to save and exit: “ESC → :wq”

Apache

Next step is Configuring the project, its locations and parameters using following commands:

```
vim /etc/apache2/sites-available/acm_deliverable.conf
```

```
Alias /acm_deliverable/ /var/www/content_manager_deliverable/html/

<Location /acm_deliverable/>
  order deny,allow
  deny from all
  allow from 172.19.192.0/255.255.255.0
  allow from 127.0.0.1/255.255.255.0
  allow from all
  Options Indexes FollowSymLinks MultiViews
```

```
  php_flag magic_quotes_gpc Off
  php_flag short_open_tag Off
  php_flag register_globals Off
  php_value upload_max_filesize 5G
  php_value post_max_size 5G
  php_value memory_limit 2G
  php_value max_execution_time 600
  php_value max_input_time 60
  php_value max_file_uploads 50
  AddDefaultCharset UTF-8
  php_value session.gc_maxlifetime 14400
  ErrorDocument 404 /acm_deliverable/404_not_found.php
</Location>
```

```
vim /etc/apache2/sites-available/editor_deliverable.conf
```

```
Alias /editor_deliverable/ /var/www/content_manager_deliverable/html/ed/

<Location /editor_deliverable/>
  order deny,allow
  deny from all
  allow from 172.19.192.0/255.255.255.0
  allow from 127.0.0.1/255.255.255.0
  allow from all
  Options Indexes FollowSymLinks MultiViews
  php_flag magic_quotes_gpc Off
  php_flag short_open_tag Off
  php_flag register_globals Off
  php_value upload_max_filesize 5G
  php_value post_max_size 5G
  php_value memory_limit 2G
  php_value max_execution_time 600
  php_value max_input_time 60
  php_value max_file_uploads 50
  AddDefaultCharset UTF-8
  php_value session.gc_maxlifetime 14400
  ErrorDocument 404 /editor_deliverable/404_not_found.php
</Location>
```

Next, it is required to reload Apache Web Server in order to apply this configuration.

```
a2ensite acm
a2ensite editor
service apache2 reload
```

PHP

In order to disable some logs from the Apache server log trace, this command using Vim is used.

```
vim /etc/php/7.0/apache2/php.ini
```

```
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT & ~E_NOTICE
```

CODE

A zipped file will be provided for the installation:

```
content_manager_r20.tgz
```

The user navigates to the web server code folder via these commands:

```
cd /var/www/
sudo mkdir content_manager_deliverable
sudo chown www-data:www-data content_manager_deliverable
sudo chmod 775 content_manager_deliverable
cd content_manager_deliverable
```

It is required to unzip the code file:

```
tar xvfz content_manager_rXX.tgz
```

Configure the path in the following file

```
vim html/includes.inc.php
```

```
<?php
require_once("/var/www/content_manager_deliverable/includes/connection.inc.ph
p");?>
```

Configured paths and database parameters are found inside the following file:

```
vim includes/config-local.inc.php
```

```
//PATHS
define("PATH_ROOT","/var/www/content_manager_deliverable");
//ROOT
```

```
define("ROOT_PAGES","/acm_deliverable");
define("ROOT_PAGES_ED","/editor_deliverable");
//BBDD
$bbdd_usuari='imac';
$bbdd_pwd='****';
$bbdd_servidor='localhost';
$bbdd_bbdd='content_manager_deliverable';
$bbdd_driver='mysqli';
```

MYSQL

The SQL is a MariaDB 10.1.26. The user needs to add the user "imac" and import the database:

```
mysql -u root -p
```

```
GRANT ALL PRIVILEGES ON *.* To 'imac'@'%' IDENTIFIED BY '****';
FLUSH PRIVILEGES;
exit;
```

```
cd /var/www/content_manager/bbdd

mysql -u imac -p content_manager < content_manager_r20.sql
```

Typing the password is obligatory in order to import the database.

Crontab

In this file the user configures the Linux Task Manager to execute periodically a number of scripts:

```
vim /etc/crontab
```

```
#CONTENT_MANAGER
* *  * * *   root /var/www/content_manager/scripts/generate_transcoding.php
0 0    * * *   root /var/www/content_manager/scripts/clean_transcodings.php
```

ANNEX II WEB AD EDITOR QUICK USER MANUAL

Quick User Manual

Project Acronym:	**IMAC**
Grant Agreement number:	761974
Project Title:	*Immersive Accessibility*

Web AD Editor Quick User Manual

Revision: 1.1

Authors: Kimiasadat Mirehbar & Enric Torres (Anglatecnic)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 761974

Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Abstract:

This document is meant to be a concise quick user manual for professional users who wish to use ImAc Web Audio Description Editor.

Contents

Contents

1. Introduction

2. What is new

3. Before starting

4. How to start

5. How to produce Audio Description

6. More options

Annex A: AD types

Annex B: Errors

Annex C: Default shortcuts

1. INTRODUCTION

This document aims to be a quick user manual for professional users who wish to use the ImAc Web AD Editor for the first time.

ImAc project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 761974 is looking for development of solutions for accessibility services in immersive environment.

One of the main objectives – among many – of ImAc is development of online editors aimed to **professional users** who wish to produce access services for 360º environments. This objective is achieved in the forth work package of ImAc. Stakeholders of this environment are broadcasters and access service providers who wish to use the editors to produce access services.

This document is addressed to stakeholders who wish to produce audio descriptions (AD) using the ImAc Web AD Editor and is meant to be a quick user guide of this tool for learning purposes and pilots.

2. WHAT IS NEW

In general the most brand-new feature of this editor is the possibility to work in 360º environment and the angles. The important matter is why do we work with **angles**?

Since we are audio describing a spherical video, events happening are not always positioned in a static Field of View (FoV), because they can move in the 360º space. So when we are describing videos, sometimes the emphasis of the current action will be in our FoV, but if all of a sudden the point of interest moves, the description will be in the new position and the user needs to move around to find it. To locate the positions the editor uses the angle concept, so when the producer sets the angle in a way is giving the coordinates in the 360º sphere of the action or point of interest that is described. The editor takes into account the type of AD as well which definitions are presented as an annex at the end of this document. Basically this is the most significant brand-new feature presented by this online editor.

The users who have worked with the Web AD Editor before will experience changes such as the possibility to **customise shortcut buttons, edit recorded segments, improved front-end** for users convenience alongside with **sound-wave of the video in time.**

3. BEFORE STARTING

Before starting it is important to be sure that the requirements are met:

- Hardware: PC with at least i5 processor, 8 GB RAM with a good graphics card.
- Voice recording setup: good quality microphone and headphones. Additionally the room where the user works needs to be acoustically silenced to achieve a better result.
- Web browser: Last version of Chrome or Firefox (at least Chrome version 74 or Firefox version 65).
- Screen resolution: at least 1920x1080 pixels and in the case of using this resolution make sure that "Size of text, apps and other items" is not scaled to 125% or more. In any case, the user can always click Ctrl+"mouse wheel down" to zoom in so as to get the right layout where all buttons in the edition area are shown (see Illustration i where the

edition area is marked in red).

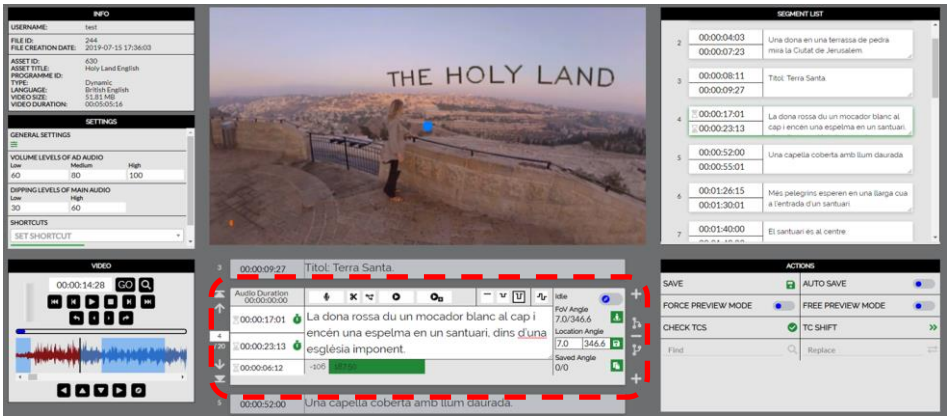


Illustration i: Web AD Editor layout

- Good internet connection because the editors and the videos used during the audio description production are online.
- Although the video is provided to the audio describer producers it is important to notice that the video provided must be HTML5 compatible Low Quality video to assure that the 360º web player runs smoothly.

Also it is important to be aware of the following:

- The web editors are online tools, so after executing them some features may take some time before they are available such as the waveform and some data in the info box.

4. HOW TO START

4.1. Login

User accesses the Editor Interface of ACM via the web browser (illustration ii) and enters username and password previously provided by administrator.

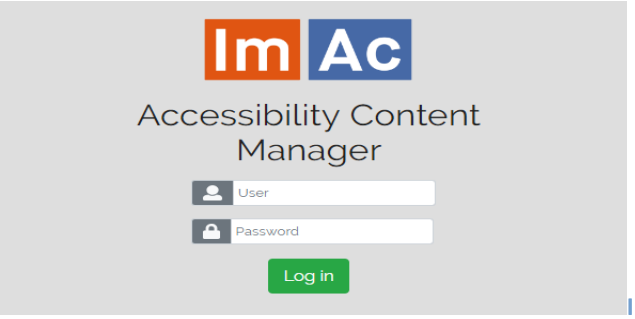


Illustration ii: ACM login page

4.2. Navigation on main page

When entered, a window with the list of assigned production tasks (audio description tasks for the purpose of this document) to the user with their corresponding videos appears (Table i). Then the user can make use of the following tools:

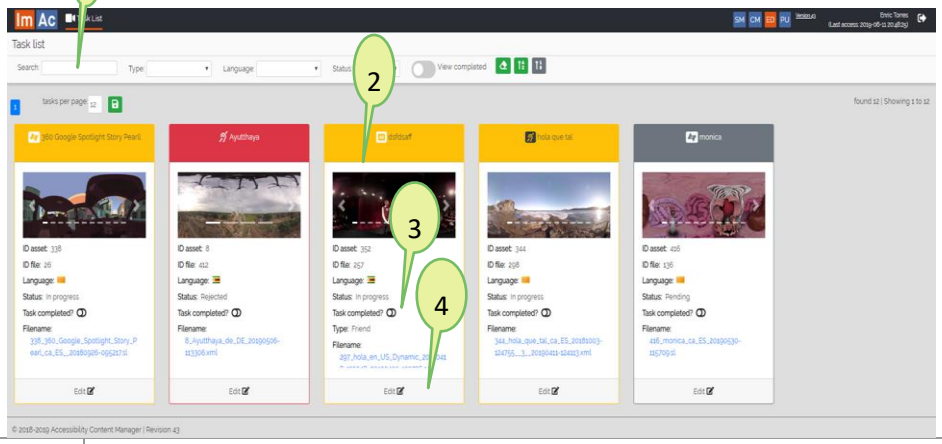
Navigation elements on the Editing interface	
	
Search bar (1)	With this tool, the user can search through their assigned production tasks and view them alphabetically, including completed tasks or not, etc.
Task icon (2)	These icons show the type of production task we are dealing with: it can be either a subtitle, a sign language or an audio description task. For the purpose of this document we will only focus on the audio description tasks. The colour of the row shows the status of the task (“Pending”: in grey, “In progress”: in yellow, “Completed”: in blue, “Rejected”: in red, and “verified”: in green).
Task status (3)	The user can view the current status of the work and only can changed it to “Completed” when the task is done.
Edit icon (4)	By clicking on this button on an audio description task (for the purpose of this document) the Web AD Editor will be executed with the video file and the audio description file for the AD production. The same applies with the other access service tasks but with the corresponding editor.

Table i: Navigation elements on the Editing interface

User selects the audio description task, presses the “Edit” button and the Web AD Editor will appear.

4.3. Web AD Editor

Illustration iii displays the two main areas of the editor. The upper area is only designed for viewing, setting and verification purposes. The down area is purely for edition.

The editor is responsive, so you may wish to set the browser zoom adequately (Ctrl+mouse wheel up or Ctrl+mouse wheel down) to fit all the boxes adequately in the screen.

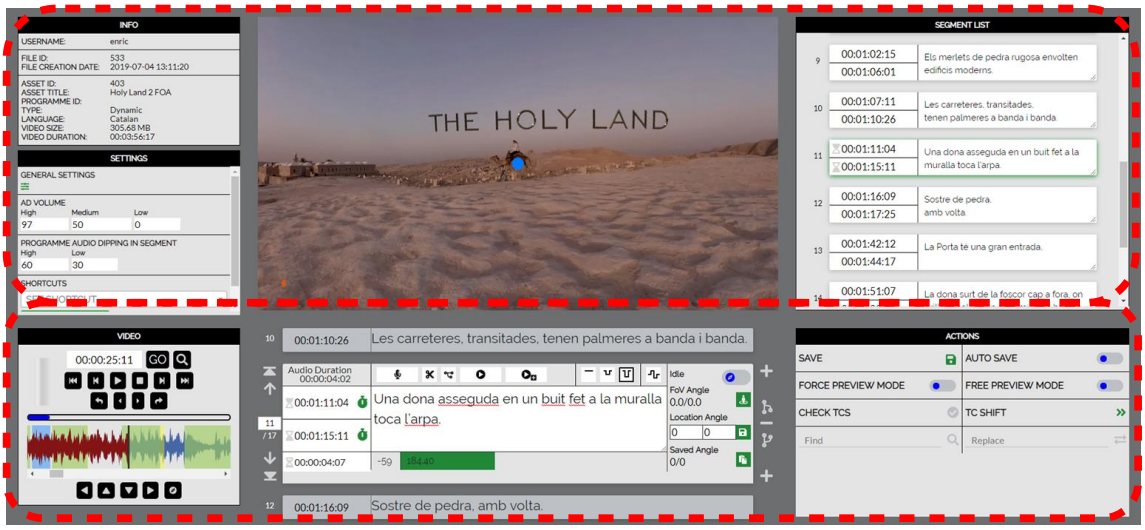


Illustration iii: Web AD Editor main areas

5. How to produce Audio Description

It is assumed that now you are inside the Web AD Editor. Let's take a look at how we use the editor to create an audio description segment from scratch.

Your first tools are the video controls. Table ii demonstrates all the buttons with their functionality. With these buttons you navigate through the video, move the video FoV and jump to the video frame you wish.

Video controls

The sound-wave displays the video sound, the black vertical line shows the current moment (01:37:01 here). Display of segments are reflected as transparent colours along the sound-wave in time as below: **Blue:** corresponds to the TCs window (from TCin to TCout), that is the time that is assigned to this segment during the script production. **Yellow:** corresponds to the recorded audio window (from TCin to TCin + audio duration). **Green:** corresponds to the area where blue and yellow overlaps. The ideal case is that only this colour is displayed meaning that the duration of the recorded audio is the same as the time assigned to this segment during the script production. **Blue** displayed after green means that the TCs window is longer than the recorded audio window. **Yellow** displayed after green means that the recorded audio window is longer than the TCs window.

Video controls	
Frame backward 	This button makes the video go backwards frame by frame (Alt+shift+left).
Fast backward 	This button makes the video go backwards with a fast speed (F5).
Slow backward 	This button makes the video go backwards with a slow speed (F6).
Toggle play/pause 	This button plays and pauses the video (F2). Note: The F2 shortcut is used to Play and to Pause the video alternatively, nonetheless the F3 shortcut has also been added to Pause the video only.
Stop 	This button makes the video stop and go to the beginning (F9).
Slow forward 	This button makes the video go forward with a slow speed (F7).
Fast forward 	This button makes the video go forward with a fast speed (F8).
Frame forward 	This button makes the video go forward frame by frame (Alt+shift+right).
Find segment by TC 	With this button, you can find the segment that contains the TC (Ctrl+Shift+F).
Jump backward 	This button helps the user to jump some frames backward. The number of the frames to be jumped is configurable in General Settings (see table vi) (F1).
Jump forward 	This button helps the user to jump some frames forward. The number of the frames to be jumped is configurable in in General Settings (see table vi) (F4).
Move FoV left 	With this button you move the Filed of View (FoV) to the left in the spherical video (Alt+left). You can also use the mouse and left button over the video, and move to the left to do the same.
Move FoV up 	With this button you move the FoV up in the spherical video (Alt+up). You can also use the mouse and left button over the video, and move up to do the same.
Move FoV down 	With this button you move the FoV to the down in the spherical video (Alt+down). You can also use the mouse and left button over the video, and move to the bottom to do the same.
Move FoV right 	With this button you move the FoV to the right in the spherical video (Alt+right). You can also use the mouse and left button over the video, and move to the right to do the same.

Video controls	
00:00:05:22 GO	Enter a specific time of the video, press GO and you are taken to that video frame.
Move FoV to "Speaker's location"	By pressing this button the FoV moves to the angle where the current segment is set (Alt+F).

Table ii: Video controls

After being in the appropriate moment of the video, you need to enter the script of the segments with their correct time-codes, illustration iv displays all the information you need in this sense. You can enter the script text and record the audio one by one or enter all the script and then start recording all the AD segments. The sequence presented in this document is based on creating a single AD segment from scratch completely, that is to enter the script with TCs and record the audio one by one.

For each segment you have to enter the segment text in the text field and after finding the appropriate video frames the time codes (TCs) must be entered: TCin by clicking on the TCin clock icon (Shift+Page up) and TCout by clicking on the TCout clock icon (Shift+Page down). The third row below them shows the segment duration.

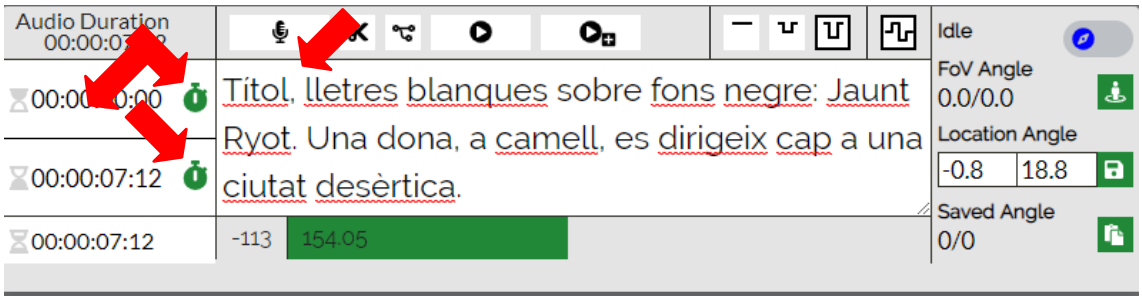


Illustration iv: Script editing

The below reading speed thermometer reflects the difficulty of the segment for the given duration. At first it is green, then if we are excessing the ideal reading difficulty it turns into red. The number (113 here) shows the remaining number of allowed characters.

When you are sure about the text itself and its time-codes next step begins. However this editor works in 360º media and matter of angles is important. So next step is setting proper angles for the segments which is done in the same area demonstrated in illustration iv. Table iii gives you the appropriate information. This step is only done when the AD type (see Annex A) is dynamic.

AD segment edition – Angles for dynamic ADs
This box only appears when the AD type (see Annex I) is dynamic. By default, at first the video has the current angle as longitude: 0.00° and latitude: 0.00°

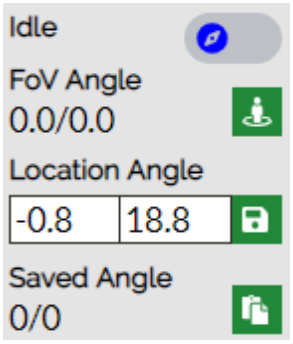
AD segment edition – Angles for dynamic ADs	
Also the **Idle** option can be marked when the segment does not bear a specific spherical position.	
	**Idle**: Only when the segment does not bear a specific spherical position (Alt+O).
	FoV angle: This is the current field of view (FoV) angle and corresponds to the video direction that we see (you can change the FoV angle using the navigation buttons in the video control box or moving the mouse with left button over the video). The green button next to it (Alt+Enter) sets the FoV angle to the "Location angle" of the segment (see next row).
	Location angle: This is set by the audio describer. It corresponds to the angle in the 360º sphere where the segment is located. It is illustrated as a blue dot over the video. It is important to know how to bind an angle to the segment. This is done solely by finding the desired angle by moving the FoV and setting it to the current segment (to know how to set an FoV angle to "Location angle" see the above row). By pressing the green save button next to it, the "Location angle" value is transferred to the "Saved angle" (Alt + C) so it can be used later in other segments (see next row).
	Saved angle: This angle is kept in this register (see previous row) so it can be used in other segments. The button next to it pastes the "Saved angle" to the "Location angle" of the segment (Alt+V). The aim of this button is to use the angle from another segment that was copied previously.

Table iii: AD segment edition – angle setting for dynamic AD

After finishing setting the proper angle for the segment, you may start the procedure of AD recording. Table iv explains the audio controls.

AD segment audio controls

Above the text area, a number of buttons are viewed which are related to the audio of the AD segment and its presentation.

AD segment audio controls	
Record button 	The user is able to record the audio of the AD segment by pressing the “Record” button (Shift + F2). When pressed, a **yellow** bar under the text area appears which goes on for about 2 seconds (this time is customizable by user in the General Settings, see table vi) and it gives time to the user to prepare themselves. When the yellow bar turns **red** it means that the recording has started so the user has to speak now. Finally when the user surpasses the segment duration the red bar starts to blink, this is not potentially an error but it is something that the user needs to be aware of, because it surpasses the time assigned to this segment.
Short & Long test 	After the recording the user can check the result using one of this two buttons: • Short test: it runs a test of the result from 2 seconds before TCin until 2 seconds after TCout of the segment (Shift + F3). • Long test: it runs a test of the result from 4 seconds before TCin until 4 seconds after TCout of the segment (Shift + F4). The durations above are customisable by the user in the General Settings (see table vi).
Set dipping level to main audio 	By pressing one of this buttons the user can change the dipping level that the main audio will perform during the AD segment. For instance the high dipping should be used when there is a lot of noise in the programme audio.
Dipping from previous segment 	Additionally, there is an option named “Dipping from previous segment”. When being on, the dipping starts at the TCout frame of the previous segment and ends at the TCout frame of the current segment, in other words, the dipping includes the time between the previous and the current segment which is useful when the segments are very close.
Cut audio 	By pressing this button a dialogue appears from where the user can cut the end parts of the recorded audio by dragging the points indicated in the following image (Shift + F5).

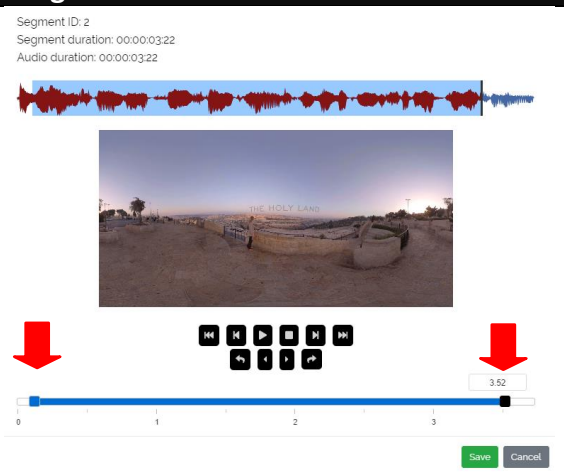
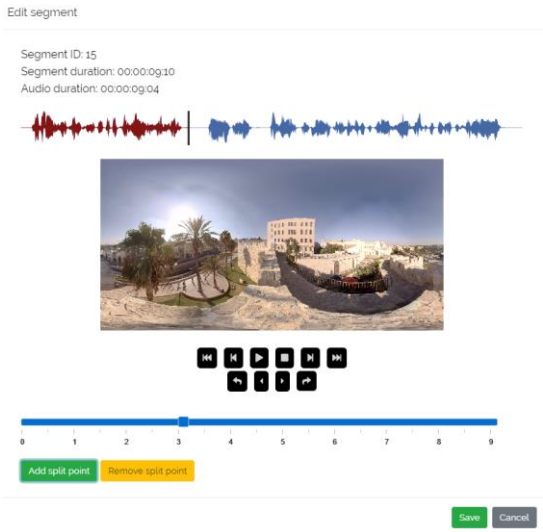
AD segment audio controls	
	
Split audio 	By pressing this button a dialogue appears from where the user can split the recorded audio into several audios. For that the user runs the media until they find the right frame and then presses the "Add split point" button. The user repeats this procedure for additional splits (Shift + F6).  A new segment is created after the current one for each new audio.

Table iv: AD segment audio controls

After finishing you may need some buttons in order to organise/edit/improve the sequence of the AD segments. Table v shows you the buttons (and shortcuts) available for this purpose.

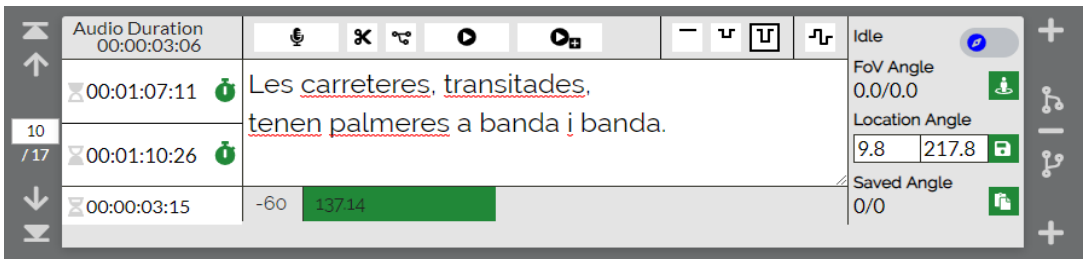
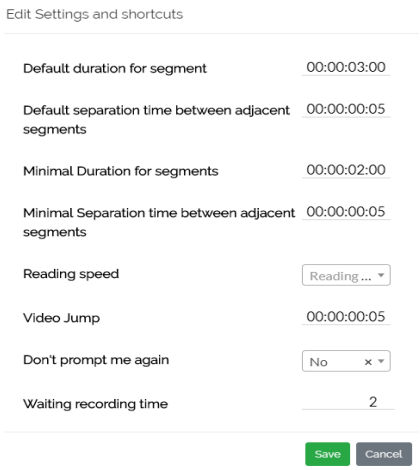
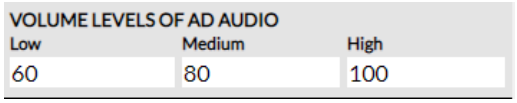
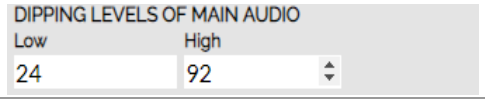
AD segments edition – buttons	
	
Jump to first segment 	This button takes you to the first segment (Alt + Page Up).
Previous segment 	This button takes you to the previous segment in relation to your current position (Page Up).
Next segment 	This button takes you to the next segment in relation to your current position (Page Down).
Last segment 	This button takes you to the last segment (Alt + Page Down).
Jump to segment value 	This part takes you to a specific segment. But you need to enter the number of the segment. It also informs you of the number of segments.
Insert segment before 	This button is useful for the insertion of a new segment before the existing segment (Ctrl + U).
Split text on cursor position 	This button splits the current segment text into two separate segments (Ctrl + Insert).
Remove segment 	This button removes the selected segment (Ctrl + D).
Join this and follow texts 	This button joins the current segment with the next one and creates a single segment containing both texts of the previous ones. The reading speed thermometer is modified accordingly but timings remain the same as the first previous segment (Ctrl + Delete).
Insert after 	This button is useful for insertion of a new segment after the existing segment (Ctrl + Shift + U).

Table v: AD segment edition and navigation buttons

6. MORE OPTIONS

The procedure in which we produce a segment is over, but we still have more options to work with. Remember the dippings and buttons that were introduced earlier, all these are customisable using “SETTINGS”. Table vi shows the setting for the file and editor.

AD settings	
General settings	 By pressing this icon the following dialogue appears for the general setting of the editor that are saved for the user (audio describer).  If any of the minimal numbers above are not met the user gets an error when verifying the AD and needs to fix it before continuing.
Volume levels of AD audio	 These settings are the volumes that the media consumer of the 360º programme will be able to choose for the AD audio. These settings are saved in the AD file.
AD location	 This setting is only available when dealing with a Classic and Static AD types, as in these cases it is necessary to indicate where the AD will be anchored in the 360º sphere. A table with all AD types is available as an annex at the end of this document. These setting are save in the AD file.
Dipping levels of main audio	

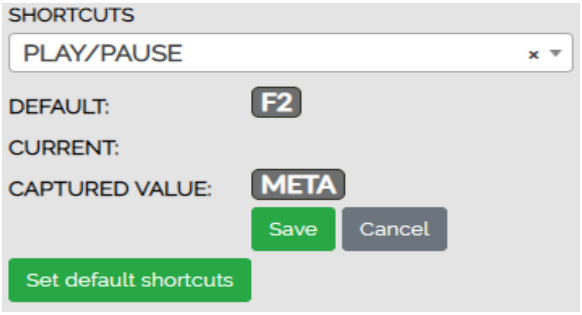
AD settings	
	These settings are the dipping levels that can be used in the segments. When a segment with a dipping level is run, the main audio volume is lowered during the AD segment. These settings are saved in the AD file.
Shortcuts	During the production of AD segments, it is convenient to work only with keyboard instead of changing between keyboard and mouse buttons constantly. As a result, most of the buttons of the editor have default shortcuts. These shortcuts are configurable and can be changed by the user if desired. A full list of the default shortcut buttons of the editor is presented as an annex at the end of this document. The default shortcuts are also shown when hovering over the buttons in the Web AD Editor. Also a button is designed to reset all the user-configured buttons back to default ones.  **WARNING: Some shortcuts are not advisable (for example the ones that are used for editing the segment script and some that are used by the browser). There is a blacklist of key combinations that cannot be used as shortcuts, however the user must be aware when customizing specific shortcuts (for instance, not to use the same key for two different shortcuts).**

Table vi: AD settings

The last step after finishing edition is verification and for that we use the preview modes. The general actions on the AD file are explained in Table vii, so here we can also find a description of the preview modes.

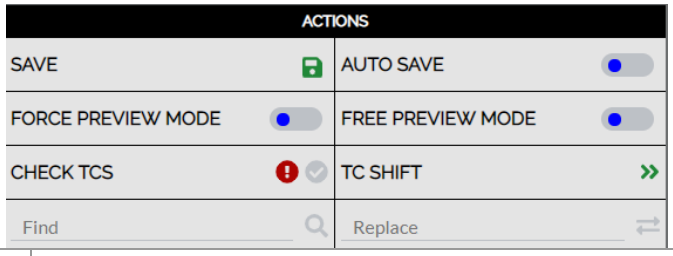
AD actions	
	
Save	This button saves the work.
Auto save	Saves the contents automatically when activated.
Check TCs	By pressing this icon the timings are checked. As mentioned in table vi, if the minimal criteria set in the settings are not met in some segments, an error will appear for those segments. If everything is OK the icon turns green (Ctrl + Q). Hovering over the segment with the error will show the description and solution. A table containing the possible errors is presented as an annex at the end of this document.
Forced preview	This mode is used for verification. This verification mode makes it easier for the describer as the video will change angle when needed during the playback of the video. Segments and angle are bound with the video. You cannot freely change angle at your wish, as the video itself takes you to the “Location angle”. When this mode is being on, the user cannot do editing any more and needs to turn it off (F11). **Note**: when clicking this button, “CHECK TC” (see previous row) is executed first and if there is any error the preview is not executed until the errors are fixed.
Free preview	This mode is used for verification. This verification mode is more real for the describer as if playing back the video with segments using HMD. Segments are bound to the video time code, but angle is not. It means that you can move angle (it is not fixed to the segment location angle) during the playback of the video. When this mode is being on, the user cannot do editing any more and needs to turn it off (F12). **Note**: when clicking this button, “CHECK TC” (see the row before the previous one) is executed first and if there is any error the preview is not executed until the errors are fixed.
TC shift	This shifts the time codes of a group of segments.
Find/Replace	This helps the user find specific words and replace them if needed.

Table vii: Actions

Also at the upper area you have informative sections (Illustration v):

- On your left, general information of the production task.
- On your right, the script: texts with their numbers and time-codes.
- Blue dot, current “Location angle”.

Illustration v: Informative sections

Annex A to the Web AD Editor Quick User Manual: AD Types

Term	Description
Classic	AD centred in the scene. The AD segments don’t have specific angles.
Static	AD anchored to the scene. The AD segments don’t have specific angles.
Dynamic	AD comes from where the described point of interest (AD anchored to the point of interest). It means that the user should assign an angle to each AD segment.

Annex B to the Web AD Editor Quick User Manual: Errors

Error message	Cause
Minimum duration fail	This error appears in the segments that don't meet with the minimum duration (TCout – TCin < minimum duration). The minimum duration can be changed in General Settings.
Minimum separation fail	This error appears in the segments that don't meet with the minimum separation between adjacent segments (TCin current segment – TCout previous segment < minimum separation). The minimum separation can be changed in General Settings.
Segment overlapping	This error appears when two segments are overlapped. Either the script TCs overlap (TCin current < TCout previous) or the audios overlap (TCin current < TCin previous + audio duration).
Unordered TC values	This error appears in the segments that are not in order (TCin current < TCin previous).

Annex C to the Web AD Editor Quick User Manual: Default shortcuts

Functionality	Shortcut button
Toggle play/pause	F2
Pause	F3
Jump backward	F1
Jump forward	F4
Fast backward	F5
Slow backward	F6
Slow forward	F7
Fast forward	F8
Stop (Jump video to first frame)	F9
Frame backward	Alt + Shift + Left
Frame forward	Alt + Shift + Right
Move FoV left	Alt + Left
Move FoV right	Alt + Right
Move FoV up	Alt + Up
Move FoV down	Alt + Down
Move FoV to “Location angle” of the segment	Alt + F
Previous segment	Page up
Next segment	Page down
First segment	Alt + Page up
Last segment	Alt + Page down
Find segment with video TC	Ctrl + Shift + F
Set TCin	Shift + Page up
Set TCout	Shift + Page down

Functionality	Shortcut button
Jump video to TCin frame	Ctrl + Alt + Page up
Jump video to TCout frame	Ctrl + Alt + Page down
Dynamic AD: Set the FoV angle to “Location angle” of the segment Classic/Static AD: Set the FoV angle to “AD location”	Alt + Enter
Copy “Location angle” of the segment to “Saved angle” (only for dynamic AD)	Alt + C
Paste “Saved angle” to “Location angle” of the segment (only for dynamic AD)	Alt + V
Idle on/off (only for dynamic AD)	Alt + O
Record audio segment	Shift + F2
Short test	Shift + F3
Long test	Shift + F4
Cut audio	Shift + F5
Split audio	Shift + F6
Split segment on cursor point	Ctrl + Insert
Join segment with next	Ctrl + Delete
Delete segment	Ctrl + D
Insert segment before	Ctrl + U
Insert segment after	Ctrl + Shift + U
Check TCs	Ctrl + Q
Forced preview	F11
Free preview	F12

ANNEX III OBJECT-BASED AUDIO EDITOR QUICK USER MANUAL

In this annex, we introduce a detailed user manual of the object-based audio editor Eddie.

General functionalities

- The user starts the *Eddie* program and can open an existing scene or create a new one. The new scene can be saved and named.

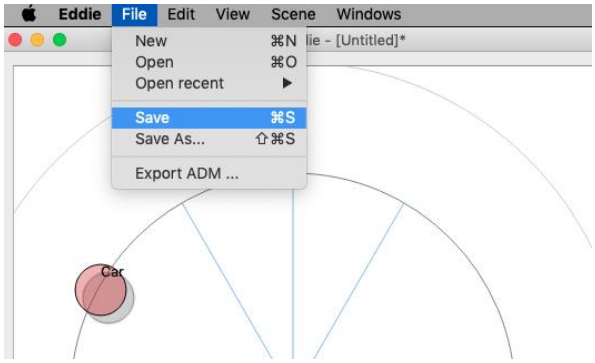


Figure 10 – Eddie menu option "File"

- While editing or creating a scene the user has the option do undo and redo actions.

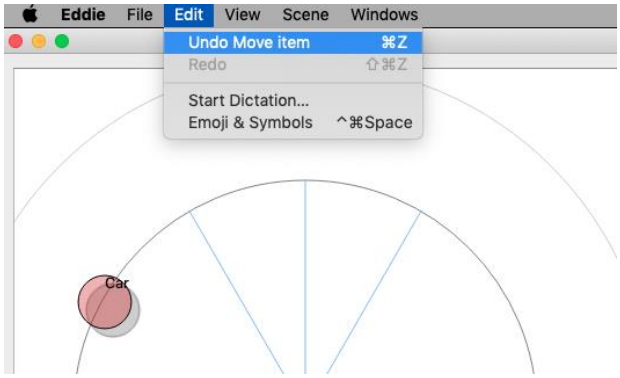


Figure 11 – Eddie menu option "Edit"

- The user has several options to change the settings of the graphical interface with zoom and fit options.

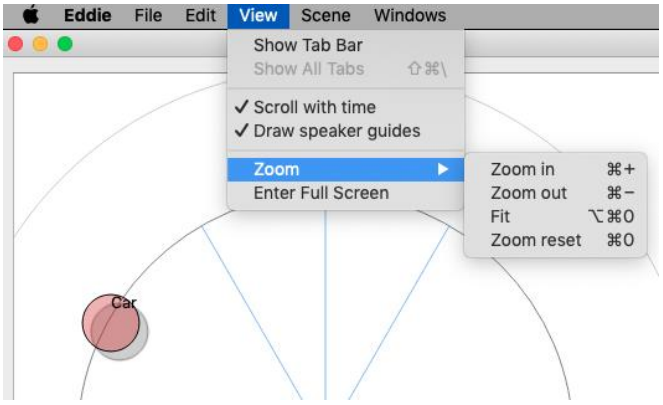


Figure 12 – Eddie menu option "View"

- The user can open two additional windows to the main window. The *Transport Widget*, *Keyframe Editor* and the *Item Editor*.

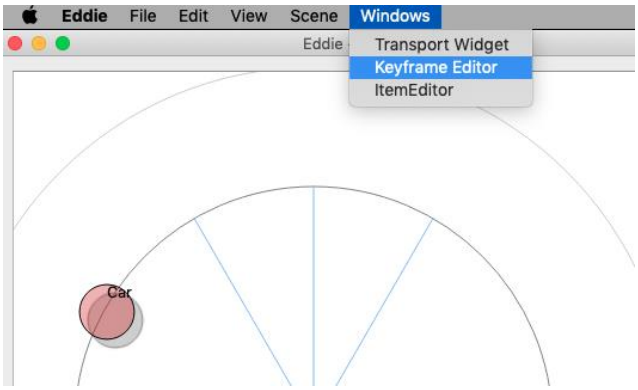


Figure 13 – Eddie menu option "Windows"

- The *Transport Widget* allows to start and stop audio playback. This includes a remote control and sync functionality for a DAW via MIDI (bi-directional) to ensure audio and metadata are always in sync and to provide a smooth user experience.

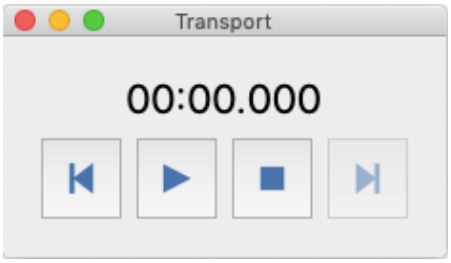


Figure 14 – Eddie Transport Widget

The *Keyframe and Item Editor*s are explained in the next section.

Audio object editing

- Items (objects) can be added, named, duplicated or removed from the scene. They can be placed in any spot in the item position space and moved around the scene.

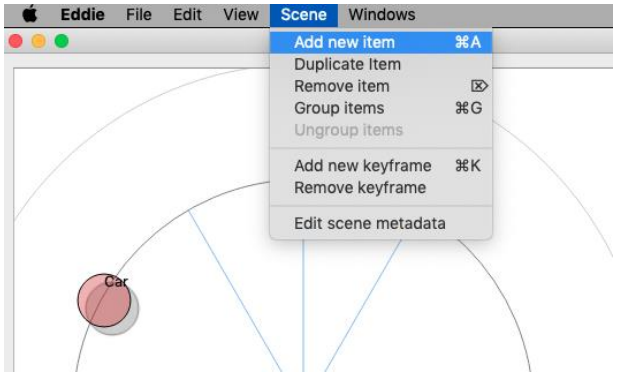


Figure 15 – Eddie menu option "Scene"

- In the Item Editor static (time independent) object metadata can be edited. The available properties are described in Table 5.

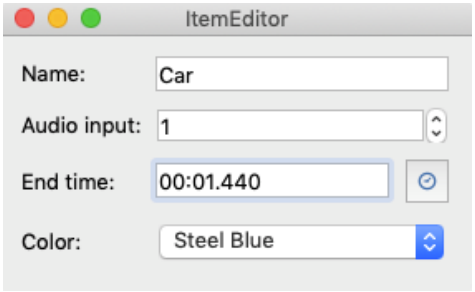
Item Editor	
	
Name	A descriptive name of the object.
Audio input	This field allows to assign an audio track to this object. The number refers to an input channel of the renderer, which receives audio from the DAW via some kind of audio interface.
End time	If set, this time specifies when the object stops to exist. The button next to the input field allows to quickly set the current time as the end time.
Color	Display color of the object within the editor.

Table 5 – Available properties in the Item Editor

- Keyframes for automation (time-dependent metadata changes) can be added and removed for each item in the scene in the timeline.
  To make automations for item positions, just jump to a time position of your choice, add a keyframe and place the desired position of the item. You can proceed in the same manner for more position changes.

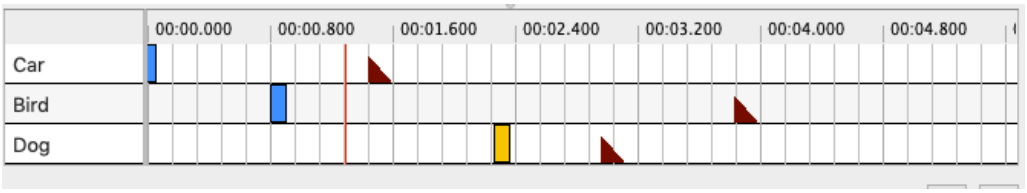
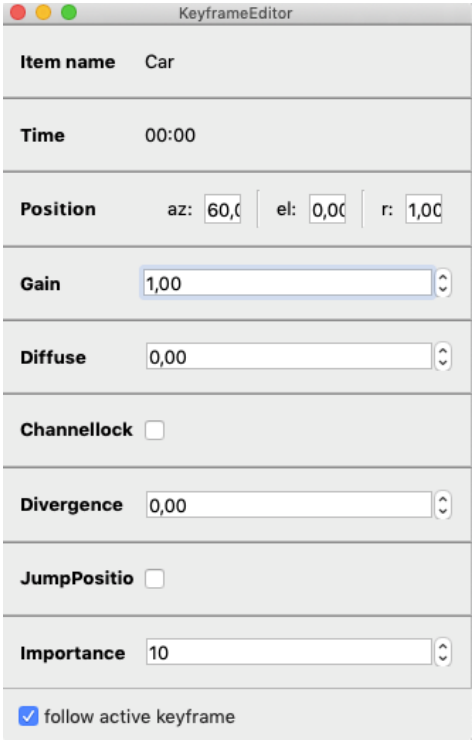


Figure 16 – Eddie timeline with keyframes

- In the Keyframe Editor the dynamic (time dependent) object metadata can be edited. The available properties are described in Table 6.

Keyframe Editor	
	
Position	The position of the audio object is represented by polar coordinates. The listener will be located at the origin of the scene. Azimuth (“az”) describes the angle between the forward listening position and the object in the horizontal plane. Elevation (“el”) describes the angle between listener and object in the vertical plane. “r” is the radius and describes the distance between the listener and the object.
Gain	The volume of the audio object within the range of [0, 1], 1 being maximum gain.
Diffuse	Describes the intended diffuseness of an audio object int range of [0, 1], where 0 means not diffuse.

Channellock	Is used to set the ADM channel lock property. It instructs the renderer to lock an audio object to the nearest channel or speaker instead of normal rendering.
Divergence	Adjusts the balance between the object's specified position and two other virtual objects placed symmetrically around the original object in the range [0, 1]. This allows to fade smoothly from rendering a single object to creating a sound source using stereophonic phantom imaging. A value of 0 means no divergence.
JumpPosition	Is used for smooth transitions from the position of this keyframe to the position of the next keyframe.
Importance	The importance of an audio object in the range [0, 10]. 10 is the most important, 0 the least. Allows a renderer, for example, to discard objects below a certain threshold of importance.

Table 6 – Available properties in the Keyframe Editor

The AD variations which were used un the ImAc user tests allowed the user to select between different positions for the AD speaker and also allow to change its gain compared to the rest of the scene. In Eddie, these parameters are set by the position and the gain property of the AD audio source.

- In the item position space, the interface shows a 2D view for object position with 3D visuals for the height of objects. Large and light circles (Bird in Figure 17) show a higher position than small and darker ones (Car). Circles with an additional circle around them (Dog) are items positioned under the middle-layer.

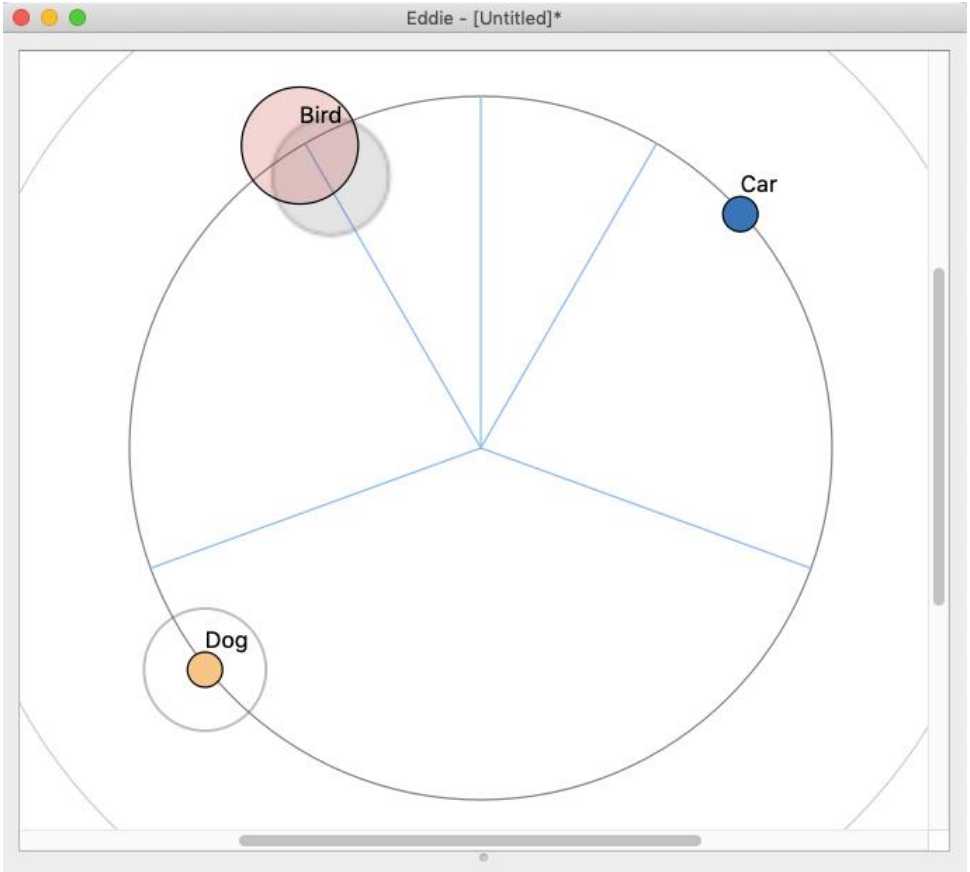


Figure 17 – Eddie item position space with three items of different height and position in the scene

<END OF DOCUMENT>