**INTERNATIONAL ORGANISATION FOR STANDARDISATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11 MPEG2020/M53547**

**April 2020, Alpbach, AT**

|  |  |
| --- | --- |
| **Source** | **Philips** |
| **Status** | **Input** |
| **Title** | **AEP development and open issues** |
| **Author** | Sam Jelfs ([sam.jelfs@philips.com](mailto:sam.jelfs@philips.com)), Jeroen Koppens ([jeroen.koppens@philips.com](mailto:jeroen.koppens@philips.com)) |

# Introduction

This document outlines the work done in the recent period to support the loudness equalization of renderers within the AEP (workplan item 83), modifications to the test configuration file, and updates to the AB testing methodology, as well as documenting the open issues and findings from our informal testing. Philips has also worked to support Fraunhofer IIS in their efforts to find and fix a lot of the issues reported at the previous MPEG meeting.

# Loudness alignment

## Current procedure

### Overview

In order to align loudness of multiple conditions, it must first be measured how these conditions differ in loudness without any level adjustments.

The level alignment process consists of four steps.

1. Generate a ‘consistent’ signal with known conditions.
2. Measure the loudness of the generated signal.
3. Calculate level alignment gains.
4. Apply the level alignment gains.

The following sections will discuss these steps in more detail.

### Generating a ‘consistent’ signal

The AEP is designed to allow plugins generate a signal dependent on the user’s position and interaction in the scene. For assessing loudness differences accurately across multiple renderers, all renderers must be measured with the same conditions (i.e. the same user input). The test scene provider must specify the location(s) and orientation at which the virtual user is to be positioned during the measurement procedure[[1]](#footnote-1). If a single measurement position is sufficient this can be the taken as the “scene start location” that is used in Unity as the teleport target location when the user moves from the pre-scene room to the main scene. For multiple measurement locations either an addition to the EIF or an additional descriptive file would be required.

In order to carry out the measurements an extra patch has been created that allows to set the user pose to fixed values without the need to use Unity or the HMD.

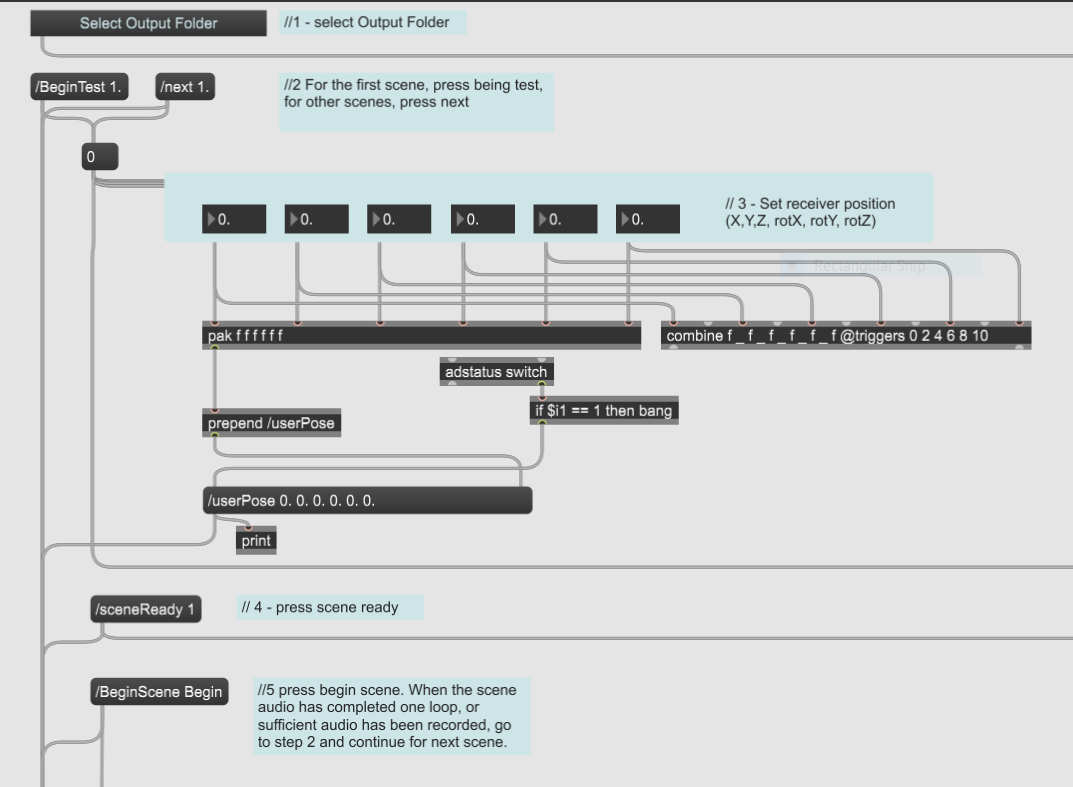


Figure 1 - Measuring renderer output signals

Figure 1 shows the interface for generating the output signals. The procedure is that the user loads a test configuration file containing the scenes and renderers required to be tested.

1. An output folder is first selected where the audio files are to be saved.
2. For the first scene, the user presses the /BeginTest button
3. Enter the coordinates of the virtual measurement position within the scene.
4. Press the /sceneReady button.
5. Press the /BeginScene button

* At this point the audio will start to be played by Max, and the renderers should output audio. One of the renderer’s output audio will be routed through to the headphones, but all outputs are stored to files.

1. After sufficient audio has been recorded, typically for one loop of the scene duration, press the /next button and return to step 3 and repeat for the remaining scenes.

This will generate a set of audio PCM files in the output folder, one binaural recording per renderer per scene, the loudness of which can then be measured.

### Measuring the loudness

Once an External’s output signals are logged for all scenes, the loudness of these signals must be measured using a BS.1770-4 loudness analysis tool, in the level-gated mode.

Plenty of tools conforming to the BS.1770-4 standard are available. For example recent versions of Adobe Audition CC include a BS.1770-3 measurement in the Amplitude Statistics window (up until 5.1 channels, this is equivalent to BS.1770-4).

The tools provide the loudness as dBFS, LKFS or LUFS, which are equivalent. These values should be entered in the TCF tool provided with the AEP. For the CfP, an administrator may be assigned to which all proponents send the measured loudness values for their External. The administrator will then enter the loudness values for all scenes and all conditions into the TCF tool.

### Calculating level alignment gains

With all loudness values entered in the TCF tool (Figure 1), the tool will automatically calculate level alignment gains per condition per scene and include them in the TCF.

Currently, the tool only performs alignment per scene. All conditions are attenuated to be as loud as the condition with the lowest loudness value in the scene. By attenuating only, there is no risk of introducing additional clipping by this feature.

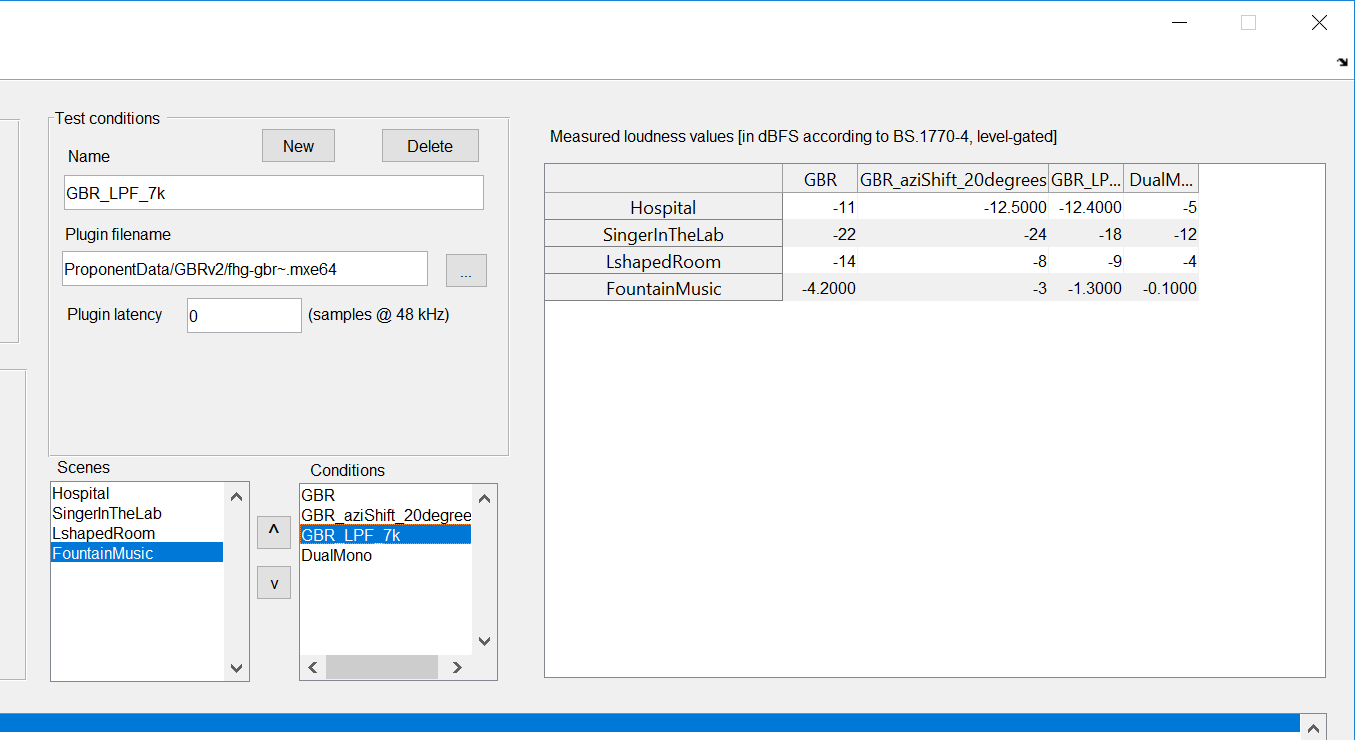


Figure 2 – Part of the TCF tool showing the loudness matrix in which the loudness values are entered.

### Applying the level alignment gain

When the test is performed, the TCF is loaded into the Max environment. The level alignment gains in the file are applied to the outputs of the renderers, before the output selection matrix, and reconfigured at every scene change.

## Improvements

### DualMono loudness

The DualMono condition is a built-in condition and has no Max External associated with it. This means it uses no metadata and just mixes down all signals into a dual mono signal.

This poses some disadvantages:

* No pre-gain is applied to any of the signals. If there is a wide variety of pre-gain values within a scene, the balance between the signals is off and an overall level alignment cannot fix this.
* All signals are downmixed, so also all signals of an HOA source and all signals of multiple HOAs that should be interpolated.
* The DualMono condition does not model distance attenuation or occlusion. So all sound sources will be equally loud, no matter where the user is in the scene. This is very different from proponent plugins, and will cause loudness differences with other conditions in the test when the user is at different locations than the used measurement point. E.g. in a different room.
* For scenes where audio signal gains are controlled by user interactions, such as in the Hospital scene where the radio is playing but muted at the beginning of the scene, the number of signals rendered by the dualMono condition will be different to that which is rendered by the other conditions.

### Dynamic updates

Scenes without continuous signals and only event sounds, such as the Basketball scene, will result in a silent signal for all conditions and therefore not get appropriate loudness alignment gains (all alignment gains will be 0 dB).

Likewise, scenes which are heavily reliant on level 2 updates (User / Unity triggered updates), may also have insufficient audio output from the renderers for an accurate measurement and alignment.

### Inter-scene alignment

Currently, there is no alignment of loudness between scenes, neither at the scene design and specification stage nor at the rendering/testing stage. For user comfort, and to ensure that no one scene is overly loud during testing, some form of level alignment between scenes may be required.

Table 1 shows measured loudness values for a number of scenes at the output of four different conditions. It shows quite some variance between conditions, which will be eliminated by the current loudness alignment. Each condition will be attenuated to be as loud as the blue highlighted loudness. The loudness variation between scenes is still very large.

Table 1 – Measured loudness values for a selection of scenes and conditions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **BS:1770 Loudness [dBFS]** | | | |
| **Scene** | **GBR** | **GBR\_20deg** | **GBR\_LPF** | **DualMono** |
| ***Hospital*** | -2.9317 | -3.4404 | -2.8584 | -8.0745 |
| ***Office*** | -37.6009 | -37.6742 | -37.6879 | -6.0083 |
| ***SITL*** | -20.916 | -20.966 | -21.0273 | -17.073 |

The current level alignment setup, requires all scenes to have a certain decent loudness. E.g. higher than -31 dBFS / LKFS. This is the responsibility of the scene designer.

An alternative would be to do automatic alignment and align all scenes to be as loud as the quietest scene. This does currently not work when scenes are included that have no continuous sound sources (loudness = -Inf dBFS). Moreover, it makes scenes that are intended to be relatively quiet (e.g. Office) equally loud as scenes that are intended to be louder, such as the music from SingerInTheLab.

In order to have appropriate alignment between scenes, it would be best to ask content creators to level their scene relative to a reference. A commonly used reference is speech. For example, if a speech signal of -20 dBFS were included in the scene, how loud should the other sound sources be to have a proper relative loudness to the speech.

### Measurements at multiple points

If measurements at a single point prove to be insufficient to get a good level alignment between conditions at all points, it may be necessary to measure loudness at multiple points and find an alignment gain that is optimal over all points (e.g. by least squares error).

Currently it is not expected that this will be required based on the current experience and scenes tested, but further user experiences are required to be able to make this judgement. Specifically between two different proponent renderers, with different occlusion or distance attenuation behavior, the required loudness alignment may vary over space.

## Proposed further updates

For the measurement of scenes that make use of dynamic updates beyond level 1 (timed) updates, it would be beneficial to have a system whereby a number of updates can be automatically triggered during the measurement procedure. This would likely need to take the form of a configuration file and automatic script such that every time the measurement procedure is run the same updates are triggered in a repeatable manner.

Switching to an automated measurement procedure will also make it easier to measure at multiple positions per scene, if that is required, and also to remove any potential for human error when entering the virtual user coordinates for example.

The quality of the DualMono reference condition is such that the level will always be problematic for between-condition normalization. We propose that in its current form it will not be used beyond the pilot testing phase, and as such the ability to accurately measure and account for the loudness is not required.

# Changes to the Test Configuration File

## Scene randomization

During the previous round of pilot testing a number of sites reported their findings. One issue raised was that the scene presentation order was, when desired, randomized on the loading of the test configuration file, rather than on the start of the testing. That meant that if multiple listening tests were run consecutively, without reloading the TCF between participants, each participant received the same scene presentation order. To fix this, the scenes are no longer randomized when loaded from the TCF, instead a new scene database is created in a javascript object, and the presentation order is controlled via that script, allowing for randomization on every new test.

This new js object, and the messaging to support it, are contained within a SceneManager patch in the main Max patch.

## Loudness Alignment

As detailed in 2.1.4 the TCF generation tool now includes the functionality to calculate and store the required loudness alignment gains into the TCF. These gains are now loaded into the sceneConfig patch on a per-scene basis, and are used to configure live.gain~ objects on the output of each renderer, triggered on scene change.

# Messaging between Max and the Renderers

Two changes have been made to the messaging between Max and the renderers in this round of development.

1. The loadHRIRSet now takes an absolute path, rather than a filename.
2. The loadScene message no longer passes a filename with .bin extension.

Previously, loadHRIRSet passed only the name of the .sofa file to be loaded, with the assumption that the file was either contained within the same directory as the renderer, or that the renderer knew where to find the file. This is now changed such that the message passes the absolute path of the .sofa file.

The LoadScene message was configured such that it always passed the scene identifier as <sceneName>\_<TestID>.bin. This has been changed so that no file extension is added to the message, given that it is not specified what file format the proponent must use, or even that the file must use that naming convention, only that when given the combination of scene name and test ID that the renderer knows which bitstream needs to be parsed.

These changes were proposed on the reflector and raised during one of the ad-hoc calls without any objections.

# AB Testing

The A-B testing methodology, as introduced in [1] and piloted in [2] had been updated to work with the current platform iteration. This was announced via the reflector and made available on request. Due to the fact that the work is based upon the version of the platform intended to be located on the MPEG gitlab servers, and was released only as a zip file to the wider group, the updated A-B testing could not be made available as a branch as would normally be the case.

The usage and functionality has not changed since the initial release, only the underlying mechanics have been updated to work in the new AEP.

# Conclusions

A number of tasks have been carried out during the previous period, as documented above. Notably there are a number of outstanding issues to be addressed, particularly in relation to the loudness alignment procedures, but the current work status gives a good base for further investigation and development.

# References

1. M46086 - MPEG-I evaluation framework software update for A-B testing
2. M47444 - A-B Testing validation pilot

1. These user poses should be documented somewhere in the CfP documentation, or a fully automated procedure should get them from a file. [↑](#footnote-ref-1)