AUDIO RENDERING SYSTEM DESIGN FOR AN OBJECT ORIENTED AUDIO VISUAL HUMAN PERCEPTION ASSESSMENT TOOL

Ulrich Reiter

Institute of Media Technology
Technische Universität Ilmenau, Germany
ulrich.reiter@tu-ilmenau.de

ABSTRACT

The cognitive processes behind human bimodal (audio visual) perception are not well understood. This contribution presents an approach to reach a deeper understanding by means of subjective assessments of (interactive) audio visual applications. A tool developed for performing these assessments is introduced, and the audio rendering system design of that tool is explained: its modular character, the signal processing flow as well as the possible reproduction setups are discussed. Finally, an example for the assessment of geometrically based room simulation and preliminary test results are given.

1. INTRODUCTION

Opposite to subjective assessments of audio or visual only presentations, subjective assessments of audio visual presentations still remain a field of scientific research full of question marks. To give an example: in its 1999 dated “Question ITU-R 102/6” [1] the study group 6 of the International Telecommunication Union (ITU) has declared the existing methods for the subjective assessment of audio systems with accompanied visual presentation to be insufficient. Yet there have not been significant contributions nor public discussions since then, and the same statement unfortunately is valid for the subjective assessment of systems where the auditory and visual domain are considered equally important.

One of the reasons for this is that there exists an already large variety of audio visual systems with very different areas of application. It is common ground to assume that the perceived quality of a system is depending on the expectations of the user, which themselves depend on the area of application, on the presentation environment, etc. The distinction between systems can be based upon a number of different criteria. It is difficult to compare these systems, and even more difficult to derive knowledge about the cognitive processes behind human bimodal perception from assessments performed on these systems.

Yet without a better understanding of these processes, further enhancements of these applications in terms of “experienced realism for the user” will be hard to achieve. In this paper a tool is presented which is capable of performing basic audio visual subjective assessments for the evaluation of the human bimodal cognitive processes. Due to its modular concept it can be extended to allow more specific tests with focus on auditory, visual as well as audiovisual phenomena.

For the tool, an object oriented approach was chosen. Therefore, the tool itself as well as the audio visual content to be assessed are object oriented. On the one hand this means that the part of the software responsible for the audio rendering is based on a modular design, where modules can be easily exchanged with other modules. This concept will be presented in section 2 of this article. On the other hand this means that the audio visual content itself (the “scene”) is organized in a hierarchical way, so that elements of the scene can be transformed, scaled, exchanged with other elements and re-used in other scenes. This concept was widely introduced with the VRML 2.0 standard [2], a concept which ran into the current ISO/IEC 14496 standard MPEG-4, where a scene description called BIFS (*BInary Format for Scene description*) based on VRML is used [3]. For the auditory part of the scene description, A(*udio*)BIFS and A(*dvanced*)A(*udio*)BIFS is used.

Therefore the tool presented here is actually an MPEG-4 player with an advanced audio render engine called TANGA which allows to use arbitrary algorithms for the rendering of audio. As an example, different methods for the real time calculus of early reflections in an interactive environment could be compared and assessed subjectively by only exchanging modules or “components” in the player, but leaving the scene description itself unaltered. The player has been developed over the last three years in the IAVAS project [4] at the Institute of Media Technology, Technische Universität Ilmenau, Germany.

The visual display part of the test setup consists of an acoustically transparent screen of $2.80m$ of width and a video projector, the auditory part is described in subsection 2.5. The test lab is in accordance with ITU-R BS.1116 [5].

2. MODULAR AUDIO ENGINE

In the Tanga System, the processing of audio signals is done through Tanga Components. Each Tanga Component constitutes a signal processing unit with a given number of output and input channels. The audio signal arriving at the input channels is then transformed by the signal processing logic implemented in the Component. Therefore the Tanga System can be expanded to virtually any audio functionality by means of writing a new plugin.

The system uses the ‘PortAudio’ API for audio input and output to any multichannel sound card and supports many different drivers on different platforms, e.g. ASIO and DirectSound on Windows and ALSA on Linux [6]. Latency of the Tanga System depends on the Tanga Components themselves, but is dominated by the time necessary to calculate the early reflections and late reverberation parts of a scene’s room impulse response (RIR).

The Tanga System consist of three main parts: the Tanga Engine which provides the link between the Tanga System and the underlying Audio API, the Tanga Components which provide the signal processing units and some helper classes.

2.1. The Tanga Engine

The `TangaEngine` class is defined as an abstract interface and hides the details of the underlying audio API to the rest of the system. This interface is currently implemented through the `PaTangaEngine` class, which provides a 'PortAudio' based render engine. It could be replaced with any audio API that provides some means of real time audio output.

One of the most important requirements for the Tanga Engine is that it should provide a DAC output timestamp. This should be the time when the samples being buffered will be played at the audio output, which is essential for synchronization purposes. 'PortAudio' was chosen because it provides such a timestamp and has in general very good real time support. Whereas 'PortAudio' also provides audio streams in blocking read/write mode, this feature is not useful for the Tanga Engine and we rely on the non-blocking audio streams which use a callback function for filling the output buffers.

This callback function invoked by 'PortAudio' is used to control the Tanga Engine, since 'PortAudio' ensures that this function is always called in time such that the output buffers are filled as needed by the audio hardware.

2.2. The Tanga Components

The signal processing units of the Tanga System are implemented through classes derived from the `TangaComponent` class. This base class defines the basic functionality and layout of any component of the system:

- Each component has a number of output channels and of input channels.
- It provides methods for attaching buffers to those input and output channels.
- The principal method is the perform() method, which will read the samples from the input buffer, perform some signal processing action on them and then write the result to the output buffer.

The buffers are implemented in the `TangaBuffer` class, which provides some basic buffer handling methods, as well as a method to fill the entire buffer with silence.

An example for a `TangaComponent` is the `TangaMix` component, which is used to mix n input signals into m output signals by using an $n \times m$ matrix. The matrix is set using a method that requires an array of $n \cdot m$ float values as its parameter and an instance of such a `TangaMix` component should have n input and m output buffers attached. This component can be used to implement, for example, an MPEG-4 AudioMix node.

2.3. Helper classes

These are classes that are used by the different components or by the engine itself, but have no specific, common layout or functionality. They mainly provide mathematical operations, as for example in the `TangaVbap` class. This is a class used to compute the mixing matrix for the `TangaMix` component, which will distribute the audio signals from the sound sources to the attached loudspeakers according to the VBAP algorithm. For more details on this see subsection 2.5 on loudspeaker setups.

A simplified UML (Unified Modeling Language) diagram for the Tanga Engine, some Tanga components and a helper class can be found in fig. 1.

2.4. Signal processing

As the Tanga Engine is part of the IAVAS MPEG-4 Player, the audio input will be provided by the corresponding nodes of an MPEG-4 scene being played. Usually this will be an AudioSource node attached to a Sound or a DirectiveSound node. In order to explain how the system works, for now we will only consider the case of Sound nodes which have an AudioSource node attached to them.

A `TangaSource` component will then be used to grab the audio samples from the MPEG-4 audio stream and write them into a `TangaBuffer`. Such a component has only one output channel and no input channel. The perform() method of this component will grab the correct audio frame with the Composition Time Stamp (CTS) corresponding to the provided DAC output timestamp and eventually multiply the samples by an intensity factor as defined in the Sound node.

At the end of the signal processing line we have a `TangaMix` component with a number of input channels corresponding to the number of Sound nodes present in the scene and a number of output channels corresponding to the number of loudspeakers used. The mixing matrix of this component is set through the TangaVbap helper class which is fed by the current locations of the Sound nodes.

To ensure that the components are performed in the correct order, the engine interprets the MPEG-4 scene graph. The callback() method invoked by 'PortAudio' will first go through all the Sound nodes defined in the scene calling the perform() method of the associated `TangaSource` components, and then, at the end, the `TangaMix` component's perform() method will be executed, which fills the final output buffers. Fig. 2 illustrates this. When DirectiveSound nodes are used in the scene and a physical approach based on early reflections and late reverberation is used for aural-

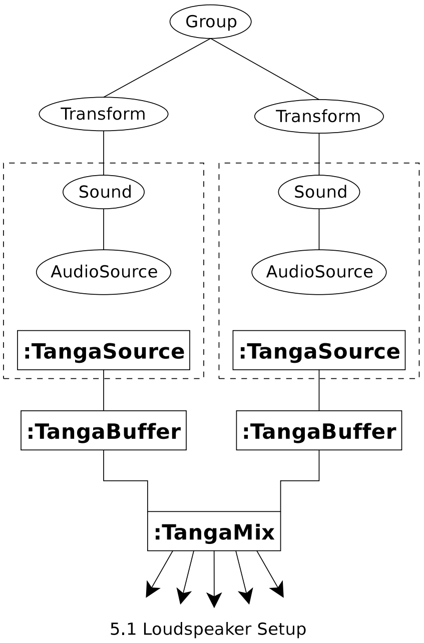

Figure 2: *The order in which the Tanga Components are performed is derived from the MPEG-4 scene graph.*

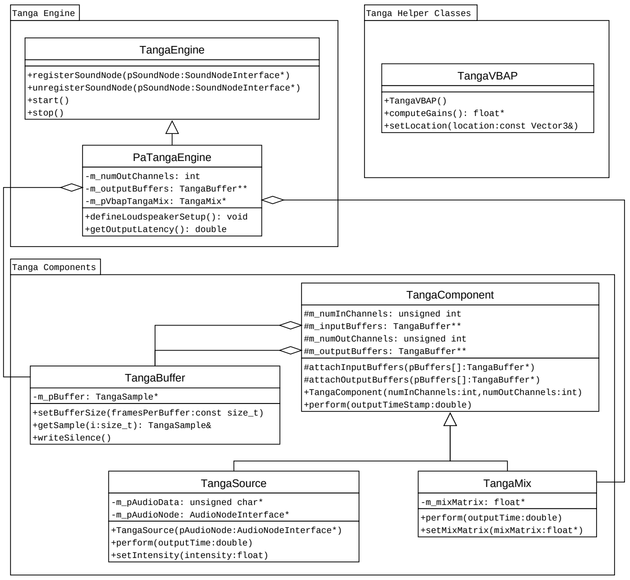
Tanga Engine
TangaEngine
+registerSoundNode(pSoundNode:SoundNodeInterface*)
+unregisterSoundNode(pSoundNode:SoundNodeInterface*)
+start()
+stop()
PaTangaEngine
-m_numOutChannels: int
-m_outputBuffers: TangaBuffer**
-m_pVbapTangaMix: TangaMix*
+defineLoudspeakerSetup(): void
+getOutputLatency(): double
Tanga Helper Classes
TangaVBAP
+TangaVBAP()
+computeGains(): float*
+setLocation(location:const Vector3&)
Tanga Components
TangaComponent
#m_numInChannels: unsigned int
#m_inputBuffers: TangaBuffer**
#m_numOutChannels: unsigned int
#m_outputBuffers: TangaBuffer**
#attachInputBuffers(pBuffers[]:TangaBuffer*)
#attachOutputBuffers(pBuffers[]:TangaBuffer*)
+TangaComponent(numInChannels:int,numOutChannels:int)
+perform(outputTimeStamp:double)
TangaBuffer
-m_pBuffer: TangaSample*
+setBufferSize(framesPerBuffer:const size_t)
+getSample(i:size_t): TangaSample&
+writeSilence()
TangaSource
-m_pAudioData: unsigned char*
-m_pAudioNode: AudioNodeInterface*
+TangaSource(pAudioNode:AudioNodeInterface*)
+perform(outputTime:double)
+setIntensity(intensity:float)
TangaMix
-m_mixMatrix: float*
+perform(outputTime:double)
+setMixMatrix(mixMatrix:float*)

Figure 1: *Simplified UML diagram for the Tanga Engine, Tanga components and helper class.*

ization, the signal processing evidently becomes more complex.

2.5. Loudspeaker setups

As the panning of sound sources in the horizontal plane is based on Ville Pulkki's *Vector Base Amplitude Panning* (VBAP), an arbitrary loudspeaker setup can be used for the reproduction of audio [7]. It is easily possible to change the position or number of loudspeakers used. A higher number of loudspeakers in the setup adds to the accuracy of localization, because phantom sources are of higher precision. On the other hand, more loudspeakers also demand higher computing power. It is therefore necessary to find a balance between number of loudspeakers and perceived precision of phantom sources. Two experimental setups with 8 resp. 12 loudspeakers located in a circular array with equal angles between the speakers are currently being tested. They will be compared to a 'classic' 5.0 or 5.1 surround loudspeaker setup according to ITU-R BS 775.1 [8].

The algorithm implemented is also suitable for VBAP in a future 3D loudspeaker setup. Whether it is desirable to have an elevation component and what its effects on human audio visual perception are in this context still needs to be evaluated.

3. EXAMPLE: GEOMETRICALLY BASED ROOM SIMULATION

A number of different methods for the numerical calculus of early reflections are widely known, among them the mirror source method, raytracing method, beamtracing method and some others. Common among them is that they are all based on the geometrical description of the room. (In the MPEG-4 context this is called the 'physical approach'.) As yet, a room acoustic calculus method based on the mirror source method has been implemented in the tool. Starting from the position of sound source and receiver and the acoustically relevant objects of the scene, a mirror source 'tree' is generated. MPEG-4 allows the author of a scene to define which objects of a scene are to be considered in the acoustic calculus. Each node of the tree represents a mirror source, and its position in the tree indicates its order. It is therefore possible to change the order of mirror sources to be computed dynamically. The 'tree' is only traversed up to a level where the maximum order is reached.

To further control the mirror source calculus, a maximum total number of mirror sources to be computed can be set. Mirror sources can be sorted according to their contribution to the overall simulation (e.g. the sound level coming from these mirror sources), and thus the simulation can be further fine-tuned. In case more than one sound sources are located in a scene, a sound 'forest' made up from a number of 'trees' is generated.

In an assessment performed to evaluate the suitability of the audio engine for interactive real time applications, the following test setup was chosen: A room acoustic simulation based on the mirror source method was performed, while test subjects were asked to play a game of *catch the ball* within the virtual room (see fig. 3), using the mouse as a navigation device like known from ego-shooter games. Each contact with the ball increased the test subject's score, and subjects were asked to reach a score as high as possible. The score reached by each test subject was expected to represent the subject's involvement with the scene. Placed at three possible locations was an omni-directional sound source reproducing three different sound snippets (spoken words, acoustic guitar music, and a funky drum / bass / guitar tune). After each round (approx. 20 sec long), test subjects were asked to rate the amount of agreement between visual and auditory impression of the scene during that round on a continuous scale of 0 - 100. To enter the rating a virtual slider on the screen was used, whose position could be modified using the computer mouse. After entering a rating, a new combination of sound snippet, location of sound source and order of mirror sources was accidentally determined. The whole test session took about 20 - 25 minutes for each test subject. Eight laymen and three expert listeners participated. In a second assessment, the

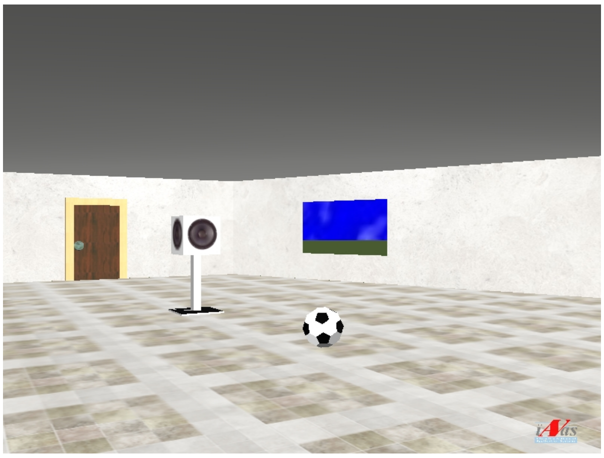

Figure 3: *Test subjects were asked to play a game of* catch the ball *within the virtual room. An omni-directional sound source was spatialized using the mirror source method.*

same test subjects were presented with a video/audio recording of the game (*passive* assessment). The order of presentation of the scenes was the same as in the preceding *active* assessment of that subject. Again, after each round test subjects were asked to rate the amount of agreement between visual and auditory impression of the scene.

Unfortunately, the test results did not render significant data regarding audio-visual perception. Yet, some tendencies can be clearly seen. As the laymen very often did not use the provided scale from 0 - 100 to its full extent but only a small part of it, their ratings varied to a great degree. Because a normalization of the data causes a number of problems detailed for example in [9], we decided not to take this data into account and rely only on the experts' ratings. Of course, with a number of three remaining test subjects the results are not representative any more.

The tendencies which can be derived from this data can be summarized as follows: for interactive applications (e.g. where the user can move freely inside the scene) the number of mirror sources necessary for a satisfying auditory impression is lower than for applications where the user can only participate in a passive way (by watching and listening). The two parameters *maximum order of mirror sources* and *maximum total number of mirror sources* allow a very good control of scalability of the system. This will be further studied in the near future when the system is completely operative.

With regard to the test proceedings, extra care has to be taken to clearly indicate to the test subjects what the attributes are that should be rated. This topic and a number of related problems are discussed in detail in another publication [10].

4. OUTLOOK

Another very interesting approach to the problem of room acoustic simulation in real time applications is the MPEG-4 'perceptual approach' introduced by Jot [11]. As the name suggests, it is based on perceptual parameters which have been derived from psychoacoustic experiments. The simulation process is not based on the geometry of the room anymore, but on parameters which describe subjective acoustic properties of that room. An enhanced version of this approach based on Miller Puckette's 'Pd' [12] has already been evaluated in the IAVAS context [13] and will be carried forward into the Tanga Engine. This is especially interesting because it will allow a direct comparison (regarding subjective quality and measured computing power necessary) between the two approaches.

5. ACKNOWLEDGEMENTS

Thanks go to Mathias Schwark, who did the design and most of the implementation of the Tanga Engine, and who has helped with section 2 on the modular character of the audio engine.

6. REFERENCES

[1] Question ITU-R 102/6, *Methodologies for subjective assessment of audio and video quality*, International Telecommunication Union, Geneva 1999.

[2] ISO/IEC 14772-1, *The Virtual Reality Modeling Language (VRML)*, 1997

[3] ISO/IEC 14496-1, *Coding Of Audio-Visual Objects: Systems*, Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2501, October 1998.

[4] Kühhirt, Uwe; Drumm, Helge; Reiter, Ulrich; and Rittermann, Marco: "Application Systems for MPEG-4", in *Proceedings of the 2002 IEEE 6th International Symposium on Consumer Electronics (ISCE 2002)*, Erfurt, Germany, September 2002.

[5] Recommendation ITU-R BS.1116-1, *Methods for the subjective assessment of small impairments in audio systems including multichannel sound systems*, International Telecommunication Union, Geneva 1997.

[6] PortAudio, an Open-Source Cross-Platform Audio API, http://www.portaudio.com/

[7] Pulkki, Ville: "Virtual Sound Source Positioning Using Vector Base Amplitude Panning", in *J. Audio Eng. Soc.*, Vol. 45, No. 6, 1997 June, pp 456-466.

[8] ITU-R BS. 775, *Multichannel Stereophonic Sound System With and Without Accompanying Picture*, Geneva 1994.

[9] Mason, Russell: "Elicitation and measurement of auditory spatial attributes in reproduced sound", PhD Thesis, University of Surrey, 2002.

[10] Reiter, Ulrich; Köhler, Tom: "Criteria for the Subjective Assessment of Bimodal Perception in Interactive AV Application Systems", in *Proceedings of the 2005 IEEE 9th International Symposium on Consumer Electronics (ISCE 2005)*, Macau, SAR, June 14-16, 2005, pp 186-192.

[11] Jot, Jean-Marc: "Real-Time Spatial Processing of Sounds for Music, Multimedia and Interactive Human-Computer Interfaces", in *ACM Multimedia Systems Journal*, vol. 7, no. 1, January 1999, pp 55-69.

[12] Pure Data (Pd), a graphical Computer Music System, http://pd.iem.at/

[13] Dantele, Andreas; Reiter, Ulrich: "Description of Audiovisual Virtual 3D Scenes: MPEG-4 Perceptual Parameters in the Auditory Domain", in *Proc. IEEE Int. Symposium on Consumer Electronics 2004*, Reading, UK, September 2004, pp 87-90.