An 'Unreal' Framework for Creating and Controlling Audio-Visual Scenes for the rtSOFE

Felix Enghofer, Ľuboš Hládek, Bernhard U Seeber

Audio Information Processing, Technical University of Munich, 80333 Munich, E-Mail: Felix.Enghofer@gmail.com

Motivation

Audio-visual scenes have been used in the gaming industry since the advent of computer games, but they have focused more on the visual than the auditory component. Recent advances in audio virtual reality, stipulated mainly by auditory research, have opened the potential of highly precise audio for many other areas including architecture, art, or more general perceptual research. Here we present a concept for integration of the "real time Simulated Open Field Environment" (rtSOFE) [1], an advanced real-time room acoustic simulator for rooms of arbitrary shapes, with a popular game engine [2], to create a pipeline for complete production of highly realistic audio-visual scenes to be used in hearing research.

Performance of the framework is demonstrated with an audio-visual interactive simulation of a highly realistic classroom with changeable acoustics and a movable sound source. This was achieved by integrating four room acoustic models from [3] into a model of a realistic classroom (Figure 1). The user can instantly change the classroom acoustics by pressing a button on a hand-held controller. The virtual teacher can be moved around the classroom such that the acoustical and visual simulations are aligned. The purpose of this project was to apply and verify the newly developed framework in an interactive setting, permitting the illustration of perceptual consequences of different configurations of sound absorbing materials in a realistic classroom situation.

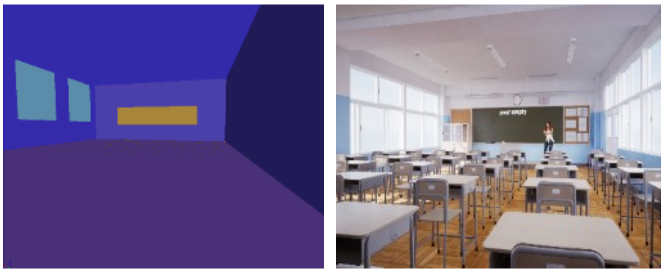

Figure 1: An exemplar simplified acoustical model of a classroom in low-polygon 'obj' file. (left) and the visual UE4 representation (right) from a classroom demo project.

Aims

Because of rtSOFE's highly flexible design, setting up audio-visual scenes was quite cumbersome. Several executables need to be launched across several computers which is time consuming and prone to errors. Additionally, after the discontinuation of the Blender Game Engine (BGE) [4], a new VR-ready, high performance, real-time integration platform was sought.

First, a comfortable and yet flexible way to define, control and play audio-visual scenes in the rtSOFE is necessary. A user-friendly frontend with a graphical user interface would be ideal, as expert programming knowledge should not be required to create and conduct experiments. Furthermore, a way of incorporating such a frontend with the already existing rtSOFE components would be desirable. These components ought to be launched and configured in an automatic way and communication between them, as well as audio-video output, needs to be synchronized. Hereby it is important to note that it must still be possible to consider the auditory and visual components of the scene separately, one of the reasons being different geometry complexity constraints for the room acoustic simulation. Also, backwards compatibility and flexibility of the current systems should be preserved, and future changes of the system such as multiple sound source support and internal audio synchronization shall be considered.

Related Work

In the history of hearing research, experiments were usually done with static sounds, using abstract stimuli such as noises played over headphones, and no visual stimuli, which has been shown to not represent real life situations effectively [5]. Adding visual cues can, for example, influence gaze and head movements [6]. Creation of an immersive, interactive experience can give a subject the impression of being part of a simulated environment [7] that can lead to multi-modal interactions.

Immersive playback of recordings or interactive simulations of audio-visual content can be realized through Head Mounted Displays (HMD) [8, 9] or projection screens in a Cave Automated Virtual Environment (CAVE) [1, 8, 10–12] in conjunction with headphones or loudspeaker arrays.

There are strengths and weaknesses in each approach: In some situations, particularly for audiological research, a CAVE is favorable over HMDs, since an array of loudspeakers will reproduce a physical sound-field of the environment, enabling the use of natural sound localization cues for normal hearing listeners and users of hearing devices alike. On the other hand, combining an HMD with binaural synthesis over headphones allows for a portable system and the ear-individual manipulation of binaural cues. However, the weight and dimensions of the HMD and headphones can influence head movements and sound localization. There are additional safety concerns if the participants are not able to see the surroundings, or if they cannot easily exit the control room.

The scene model for such systems could deal with the audio and visual components separately, using different computer programs. rtSOFE's auditory part can be defined using a room model in .obj format that is integrated with a spatial movement model (and visual model) in the Blender game engine. Programs such as TASCAR [13] or the SoundScape Renderer[14] integrate an acoustic scene description. Some of those also include the (room) acoustic simulation with the audio playback, others are reliant on additional programs.

Visual scenes can be designed using Blender, 3ds MAX, game engine editors etc. and played back either via their included renderers like the Blender Game Engine (BGE) or yet other (custom) renderers based on graphics APIs such as OpenGL/CL, as done in previous SOFE incarnations [10, 11]

rtSOFE Overview

The rtSOFE consists of an interactive room acoustic simulation and real-time convolution software, 60 loudspeakers as well as a 4-sided video projection in a CAVE-like setting with 12 optical motion tracking cameras within an anechoic chamber. The participant is wearing light stereoscopic glasses with tracking markers, while receiving natural sound cues through the loudspeaker array [15].

The custom room acoustic simulation software of the rtSOFE [16] was designed as a flexible cluster application from the outset. Multiple instances, running on different computers, can be used to simulate different sources or acoustical conditions, and they can also be configured to dynamically update different parts of the impulse response.

The system can also be controlled from a motion tracker and position data are checked for collisions in the BGE running on the Video-PC (Figure 2). The real-time convolution engine "Convolver" [17] (Audio-PC) receives room impulse responses from the room acoustic simulation and processes it with input audio and the loudspeaker equalization filters, which is then played back over the loudspeaker array. The BGE can also display visual scenes, created in Blender, via four 3D projectors. It can be controlled via Open Sound Control (OSC) messages (UDP) [15]. The software for rtSOFE is designed for the Windows operating system.

Solution Concept

Synchronized playback and control of several soft- and hardware components suggest that it is sensible to develop a single application which fulfils the previously mentioned aims by providing graphical audio-visual scene definition and playback, using the existing audio framework, controlled from a centralized perspective. This app will be referred to as "rtSOFE Controller". Implementing everything from the ground up would have been infeasible for this project and unnecessary, since there already are many tools which can be used as a starting point.

As such, several real-time game engines were considered for this project. The lightweight Godot engine stands out with its human readable scene definition format outside the editor. However, similar to CopperCube, these engines lack in certain features compared to some competitors and are not able to boast years of development nor documentation, therefore not meeting the requirements for this project.

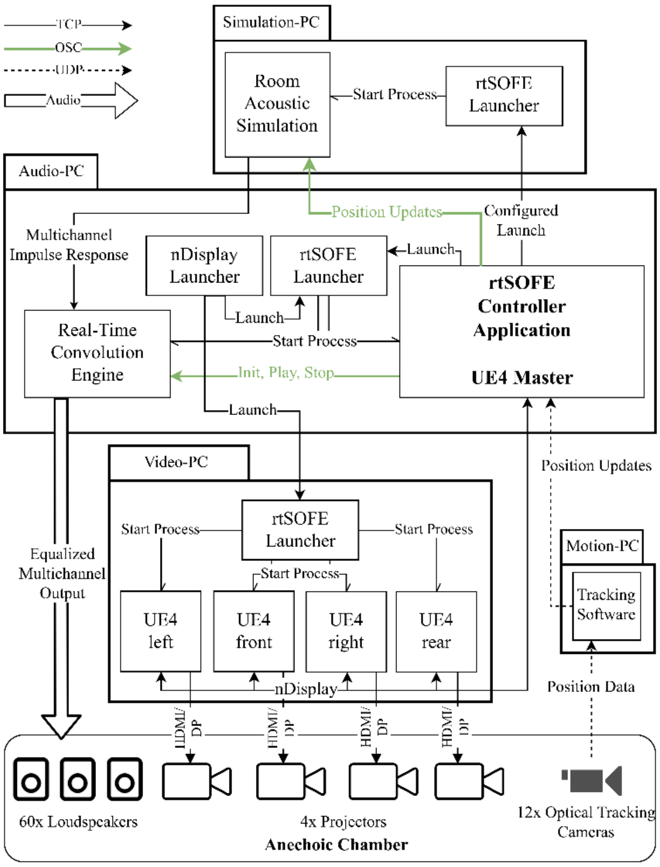

Figure 2: Concept scheme for the rtSOFE Controller and Launcher. The Controller master receives position data, launches and controls various applications.

The most established engines are the Unreal Engine 4 (UE4) [2], Unity [18] and Lumberyard [19] (based on CryEngine). All of them convince with a powerful editor for scene definition, high performance real-time rendering and various scene import capabilities. However, Lumberyard does not support cluster rendering natively, and licensing issues presented themselves with Unity. UE4 on the other hand, fulfills all criteria and stands out with its Blueprint visual scripting language and its ability to create complete scenes including game logic without any programming knowledge requirements for the user. Furthermore, UE4 provides many useful, free plugins and assets for process communication, content creation and playback, such as the cluster rendering plugin nDisplay. Also, the engine is extendable and completely written in C++. Besides the obvious advantage of memory management and therefore real time capabilities, any third-party C++ library can be integrated in the engine, project and/or plugin code. Moreover, any C++ function can be exposed to Blueprints in order to simplify possibly complex features for the end user. Thus, the decision was made to use UE4.

rtSOFE Controller

The rtSOFE Controller was implemented as a UE4 plugin that integrates rtSOFE's hard- and software framework. All functionality is exposed to Blueprints for simple control and scene design. Once a scene is arranged in the Unreal Editor, objects such as a virtual loudspeaker or the "player" can be defined and configured as a possibly moving or tracked rtSOFE sound source or receiver. Subsequently, UE4's "Play In Editor" (PIE) feature allows the user to test the scene immediately. When this scene is played back, the rtSOFE Controller launches, updates and controls the respective rtSOFE audio applications as configured.

Control and Communication

An overview of the system communication can be seen in Figure. In order to launch the room acoustic simulation and convolution software instance, an additional, simple C# console application, called "rtSOFE Listener" was created based on the already existing UE4 cluster app "nDisplay Listener". It automatically starts on each relevant computer and listens for appropriately formatted TCP messages from the rtSOFE Controller in order to launch executables or execute Windows commands across the network.

The Controller can transform and resend tracking data, coming from a motion tracking system via UDP. Collision and custom calculations can be done within the engine. These data are handled in the same way as the positions of (predefined) moving objects and are sent to the room acoustic simulation instances across the network using the existing OSC over UDP messaging protocol. The generated multichannel impulse response(s) by the rtSOFE room acoustic simulation instance(s) are sent to the convolution software (Convolver), which then streams the convolved multichannel audio to the sound card, as it was done previously. The rtSOFE Controller implements the Convolver's OSC protocol, allowing the Controller to initialize, play and stop sound.

For the playback of a scene's visual component, the cluster rendering plugin "nDisplay" was used to simplify real-time object synchronization and virtual CAVE projection screen setup as well as improve the performance for multiple GPUs and scalability. The cluster master hereby is executed on the Audio-PC and represents the rtSOFE Controller. Four additional instances are launched on the Video-PC with the sole purpose of rendering 4 stereoscopic viewports to the projectors in the anechoic chamber: left, front, right and rear. Cluster nodes, input devices and the physical dimensions of the CAVE projection screens are defined in a configuration file. The "nDisplay Launcher" is used in conjunction with the rtSOFE Listener to appropriately launch the cluster nodes via TCP messages.

Discussion

The presented integration of rtSOFE into UE4 not only allows for easier and faster design and conduct of experiments, but also opens the door to many features, which have been developed for the gaming industry. The system has already been used to recreate the project of [3], which benefits from the high render performance and quality of UE4. Also, a way of incorporating real-time facial tracking with Motion Builder and UE4 is in progress.

While the current system is functioning and stable, there are some things to address. Although nDisplay is a powerful plugin for cluster rendering, it comes with certain drawbacks regarding object synchronization. For one, it uses thread locks, which impacts the real-time component of the system. Additionally, all cluster instances are locked to the same framerate. This means, that the game loop of the master instance, which computes tracking data, predefined trajectories etc. can only operate on the framerate of the weakest link in the cluster, which depends on the playback quality of the visual scene. Possible solutions to this problem are currently being investigated and might include altering the nDisplay source code and/or exploiting UE4's multiplayer capabilities.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Projektnummer 352015383 – SFB 1330, Project C5. rtSOFE development is supported by the Bernstein Center for Computational Neuroscience, BMBF 01 GQ 1004B.

References

[1] Seeber, B. U. ; Clapp, S. W.: Interactive simulation and free-field auralization of acoustic space with the rtSOFE. In: The Journal of the Acoustical Society of America 141 (2017), Issue 5, 3974.

[2] Epic Games: Unreal Engine 4 URL: www.unrealengine.com.

[3] Pulella, P.: Auralization of acoustic design in primary school classrooms. Master's Thesis. Department of Electrical Engineering, Technical University of Munich. 2019.

[4] Removing Blender Game Engine from Blender 2.8 URL: https://developer.blender.org/rB159806140fd33e6ddab951c0f6f180cfbf927d38.

[5] Bentler, R. A.: Effectiveness of Directional Microphones and Noise Reduction Schemes in Hearing Aids: A Systematic Review of the Evidence. In: Journal of the American Academy of Audiology 16 (2005), Issue 7, 473–484.

[6] Hendrikse, M. M.E., et al.: Influence of visual cues on head and eye movements during listening tasks in multi-talker audiovisual environments with animated characters. In: Speech Communication 101 (2018), 70–84.

[7] Cruz-Neira, C., et al.: The CAVE: audio visual experience automatic virtual environment. In: Communications of the ACM 35 (1992), Issue 6, 64–72.

[8] Llorach, G., et al.: Towards Realistic Immersive Audiovisual Simulations for Hearing Research. In: Hilton, A., et al. (Hrsg.): Proceedings of the 2018 Workshop on Audio-Visual Scene Understanding for Immersive Multimedia - AVSU'18. New York, New York, USA: ACM Press, 2018, 33–40.

[9] Vorländer, M.: Acoustic Virtual Reality URL: www.akustik.rwth-aachen.de/cms/Technische-Akustik/Forschung/Forschungsgebiete.

[10] Hafter, E. R. ; Seeber, B. U.: The Simulated Open Field Environment for auditory localization research (2004).

[11] Seeber, B. U. ; Kerber, S. ; Hafter, E. R.: A system to simulate and reproduce audio–visual environments for spatial hearing research. In: Hearing Research 260 (2010), 1-2, 1–10.

[12] Vorlaender, M., et al.: Audiotechnik des aixCAVE Virtual Reality-Systems. In: 41. Jahrestagung für Akustik (DAGA2015).

[13] Grimm, G., et al.: Toolbox for acoustic scene creation and rendering (TASCAR): Render methods and research applications. In: Musikinformatik & Medientechnik : Bericht (2015).

[14] Geier, M. ; Spors, S.: Spatial Audio with the SoundScape Renderer. In: 27th Tonmeistertagung – VDT International Convention (2012).

[15] Seeber, B. U. ; Kolotzek, N. ; Clapp, S. W.: Creating interactive audio-visual space for hearing research with the real-time Simulated Open Field Environment (rtSOFE).

[16] Hornung, M. G.: Implementation and Optimization of a Software Framework for interactive Room Acoustics Simulation. Master's Thesis. Department of Computer Science, Technical University of Munich. 2015.

[17] Pods, S.: Echtzeit-Faltung von mehrkanaligen Impulsantworten. Bachelor's Thesis. Department of Electrical Engineering, Technical University of Munich. 2016.

[18] Unity URL: https://unity.com/.

[19] Amazon: Lumberyard URL: https://aws.amazon.com/lumberyard/.