GAME AUDIO LAB - AN ARCHITECTURAL FRAMEWORK FOR NONLINEAR AUDIO IN GAMES.

SANDER HUIBERTS, RICHARD VAN TOL, KEES WENT

Music Design Research Group, Utrecht School of the Arts, Netherlands.
adaptms[at]kmt.hku.nl

Nonlinear and adaptive systems for sound and music in games are gaining popularity due to their potential to enhance the game experience. This article is about the Game Audio Lab: a framework for academic purposes which enables research and rapid prototyping of nonlinear sound for games. It enables researchers and designers to map composite variables and adapt sound and music design in real-time during active game play.

INTRODUCTION

The Music Design Group of the Utrecht School of the Arts [1] in the Netherlands has developed a lab for academic purposes which enables research and rapid prototyping of nonlinear audio. The system offers designers the ability to construct composite variables based on variables that are available in the original game system. In this paper, the architecture of the hardware and software as well as the motivation for this setup will be discussed.

1 BACKGROUND

The use of nonlinear and adaptive systems for sound and music is becoming more and more common in the game industry as these systems can be quite effective for enhancing the game experience. In order to innovate in this area, research is indispensable, but R&D can be time-consuming and costly for commercial developers. Academics have the ability to contribute to the development of new techniques and can afford to take larger risks concerning innovative design, as they are not bound to commercial constraints.

Unfortunately, most of the current game systems are not freely accessible for researchers, and publication of the research outcomes in collaborations is often prohibited due to intellectual property restrictions. In addition to the commercial restraints for academic usage, there is another major motivational factor for developing a more flexible game sound design platform: when using third-party games, the design researcher is often unable to transfer the desired variables of the game to his system simply because the original game designer or programmer has not included these in the game code. Many games do not offer the variables that researchers want to use as input for their nonlinear or adaptive music systems.

2 RELATED TECHNOLOGIES

Audio systems for games make use of audio integration tools that enable sound designers to link audio to game objects, scripted events and locations in the game world [2]. Such tools, which are often third party software (or: middleware), are very time-efficient to developers as they provide audio integration functionality as well as a sound playback engine (often with real-time DSP-effects). Examples of software include GameCODA, ISACT, Wwise, XACT, FMOD, Miles Sound System and Unreal 3 Sound System [3].

For academic purposes, these versatile middleware solutions have two shortcomings. Firstly, they do not allow control over game variables once they arrive from the game engine. Researchers are unable to converge the standard variables into more meaningful variables with which they wish to control their nonlinear or adaptive music systems. Secondly, the tools follow industry standards and are therefore (mostly) sample-based, thus not tailored towards experimentation with techniques such as procedural generation of sound.

3 DESCRIPTION OF GAME AUDIO LAB

3.1 Principle

The framework is to fill the need for a flexible system that allows for rapid-prototyping of and experimentation with adaptive or nonlinear audio in games. It should allow the designers to modify parameters, processing and engine architecture in real-time.

A unique feature of the framework is that it allows researchers to converge plain variables of the game to composite variables - variables that express meaningful information about the game that is not available otherwise. Composite variables are constructed from

multiple game variables to form a single variable which can then be used to directly control the audio engine. An example of a composite variable that was used in the Game Audio Lab is "Level of Threat", which expresses the amount of threat the player's avatar is in during the game by a ranged value. This information is not readily available in most games, but can be constructed by converging variables that are present in the game: the number of enemies within a certain radius that are attacking the avatar, the distances to these attacking enemies, the amount of health of the avatar, the location of the avatar in the game world, etc.

3.2 Hardware Setup

A Dell computer (Xenon Quad processor) with Microsoft Windows Vista is used as game platform. This computer is connected via an UDP network to an Apple Mac Pro computer which functions as audio workstation. The audio workstation runs the GSToolkit (the completely modifiable audio engine used in the lab) [4] and is connected to a MOTU multichannel audio interface that distributes the sound over a 5.1 surround powered speaker system. This configuration keeps the game and the GSToolkit separated on two computers, making it possible for two programmers to work simultaneously both on the game or a game mod and on the GSToolkit.

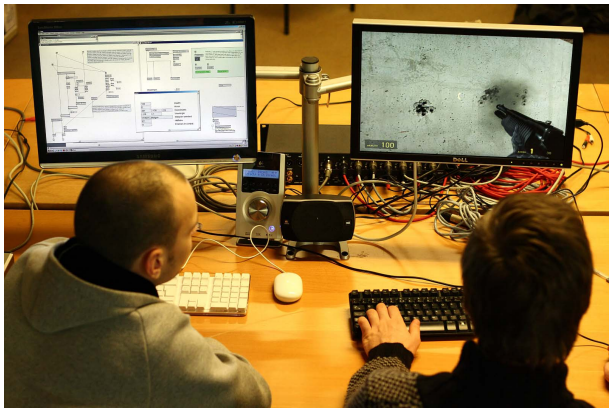

Figure 1: A photo of the Game Audio Lab showing the audio workstation and the game platform.

3.3 Software framework

A first implementation was based on the game Half Life 2 (Valve, 2004) [5]. Half Life 2 is a First-Person-Shooter and the open source SDK of this game offers various tools for 'modding' (making a customized version of the game for a special purpose). A level editor named Valve Hammer Editor is included in the SDK, which provides the customisation and creation of levels [6].

The Game Audio Lab uses a modified version of this game, with implemented Open Sound Control (OSC) communication [7].
OSC is a protocol for communication among computers, sound synthesisers, and other multimedia devices that is optimised for modern network technology. There are several implementations of OSC available as C++ code. For the GSToolkit the OSCpack of Ross Bencina was used [8]. OSC makes use of the UDP network for sending messages from the game engine to the GSToolkit.

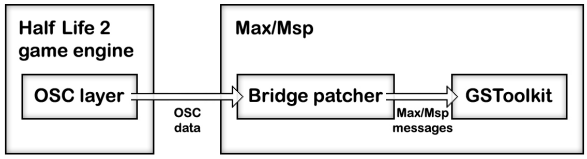

Figure 2: An overview of the software framework.

Plain variables from the game platform are transferred to the so-called Bridge software (or: Bridge) of the audio workstation. The Bridge offers the designer the ability to converge plain variables to composite variables - variables that express meaningful information about the game that is not available otherwise.

The GSToolkit in MAX/MSP is used as a dynamic sample player with real-time DSP, which is designed to react to the composite variables. The designer is able to change the mapping of variables to composite variables as well as the dynamic response of sound while actually playing the game. During these adaptations, it is not necessary to stop or pause the game as the Bridge is running on the audio workstation, separated from the game platform.

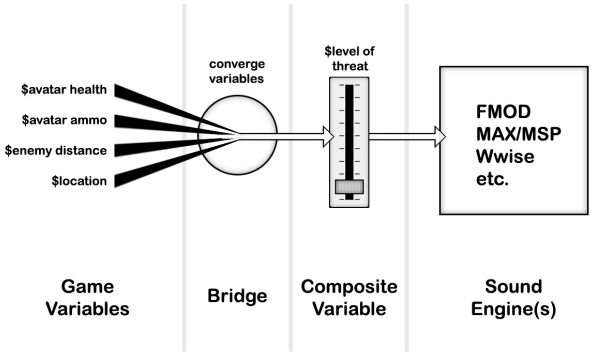

Figure 3: An overview of the routing of variables through the system.

In order to study the design of nonlinear or adaptive sounds, it is valuable to be able to change only specific sound instances, while leaving the other sound objects

of the game intact. Game Audio Lab provides this by enabling the designer to choose which sound triggering information is communicated to the Bridge of the audio workstation. Sound instances that are not affected by this step are played back normally by the sound engine of the game platform (which in Half-Life 2 is FMOD). The intercepted sound triggering information is transferred to the audio workstation, where the designer can implement the sound playback of this instance in the GSToolkit. To give an example, in the Half-Life 2 application, only the weapon sounds and ambience layers of the game are replaced with alternatives on the audio workstation, while the other sounds, such as the zombie and interface sounds are unaffected.

Besides intercepting and creating new playback calls for the custom sound engine, meaningful gameplay data is needed to dynamically control the playback of sounds. Some variables can be transmitted and used directly to control a sound engine parameter, others can be combined in the Bridge module into composite variables. Sound designers can decide which gameplay variables are used and how they are combined in the Bridge, without the intervention of game programmers. Although the current implementations of Game Audio Lab use samples, the system also allows the use of Max/MSP patches that use generative (procedural) audio.

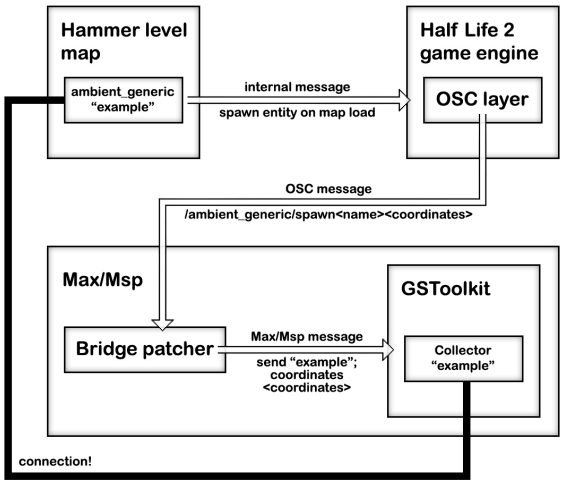

Figure 4: An illustration of the connection of the linked software of the Game Audio Lab.

4 PRACTICAL APPLICATIONS

In 2008, the system was put into practice to research the possibilities of adding a narrative disposition to gun sounds and ambient sound in First-Person Shooter games. As the system enables using composite variables, the designers were able to successfully add dynamic behaviour to the gun and ambient sounds, conforming to the (presupposed) experience of the player. The effect of the system on players has not been evaluated yet, as it was the purpose of the setup to validate the practical use of the framework.

An advantage of the framework is that it is relatively uncomplicated to research, evaluate and design new audio techniques in current games. The Bridge software enables the researcher to select only certain sound instances and helps with the mapping of variables. A restriction, however, is that the intended game has to be modifiable up to some extent, so that the game parameters can be communicated to the Bridge via OSC.

The dynamic sound design was researched in testing environments of Half-Life 2 (Valve, 2004). The designers have chosen to build Half-Life 2 testing environments with the help of the Valve Hammer Editor of the Source SDK. These test-levels helped to evaluate the nonlinear techniques by changing various settings in the environment. Based on the composite variables *Level of Threat* and *Level of Success* (henceforth: LoT and LoS), the weapon sounds were processed with real-time DSP. LoT and LoS influenced the dynamic behaviour and aesthetic qualities of the gun sounds. To exemplify, having a higher LoS makes the gun sound more appealing, while having a lower LoS reduces the sonic impact of the sound.

Figure 5: Video still of a testing environment for the nonlinear response of the weapon sounds.

The ambiences were generated following a different framework [9]. Sounds of the Setting of the soundscape of the game - the non-directly reactive sound objects [10] - were adaptive to LoT and LoS. Real-time processing as volume, filtering and reverberation also responded to these composite variables for an intensification of the effect of the ambiences.

The application was successfully implemented and the Game Audio Lab was demonstrated at the Utrecht

School of the Arts. Some demonstration videos of the test environment can be found at the Adaptive Music Systems research page of the Music Design Research Group [11].

5 DISCUSSION

The Game Audio Lab has been developed and evaluated in 2007 and 2008. The framework was tested with Half-Life 2, a commercial first-person shooter game, because the source of this game is freely accessible. From the Source SDK, multiple variables were transferred to the Bridge that calculated two composite variables: Level of Success and Level of Threat. These composite variables proved very useful to research nonlinear sound design and composition. They define a grid of game states that can be linked to the dynamic behaviour of sound objects.

Game Audio Lab is not the first practical application of a combination of Half-Life 2, OSC and a modular audio editing environment, such as Max/MSP, PureData or CPS [12]. For instance, Leonard J. Paul has been using Half-Life 2 and PD for audio prototyping for many years [13]. Unique about this framework is the modular architecture in combination with the Bridge and GSToolkit. This setup allows a useful way of working with composed variables that is valuable for future applications, as the game platform can be exchanged for any OSC-compatible platform.

For the projects of designers and researchers at the Utrecht School of the Arts, the framework showed to be very flexible and versatile for exploring and investigating the concepts of nonlinear audio. As many academics work with MAX/MSP or PureData and they can easily use these environments to develop innovative concepts for representative video games. A big advantage is that the academics are not restricted to the dedicated functions of audio middleware, tools and engines that are publicly available. Especially for generative composition and procedural processes, the system shows its strength. The system is capable of following a narrative in real-time, conforming to the (presupposed) experience of the player.

A point of critique may be that the Game Audio Lab designs do not conform to the reputable middleware systems and audio engines used in game development. The Game Audio Lab was never intended to offer the same quality, durability and features of current middleware. The system is purely intended for rapid prototyping, which is what systems such as MAX/MSP and PureData excel at. Furthermore, academics are taught to reflect upon current systems, in order to be able to expand the possibilities and innovate.

From here, the system will be evaluated and further developed. Other project groups will use and explore the use of this system for design and research of adaptive sound and music in games. The available open source games will also be replaced with innovative game concepts developed at the Utrecht School of Art & Technology. Furthermore, we assume that further research into the composite variables might prove useful for the cooperation of sound designers and composers as they can both use the same extended set of variables to generate content that corresponds with the state of the avatar in the game.

ACKNOWLEDGEMENTS

The authors wish to thank Jan IJzermans for his contribution to the concepts of Game Audio Lab, as well as the feedback for this article. Furthermore, the authors would like to express their gratitude to the co-researchers of the Music Design Group as well as the five students that participated in this project [2]. They also would like to mention Jakko Terborg and Rik Nieuwdorp for their work on the basic principles of composite variables in earlier projects.

REFERENCES

[1] The Music Design Group under Jan IJzermans consists of two research groups: Adaptive Music Systems Research Group (http://adaptivemusicsystems.hku.nl/) and Music Design Processes & Multidisciplinarity.

[2] The five students that developed the game lab are Arjen Schut, Mark Doeze, Maurice Alberts, Ramon Kerstens and Rogier Habraken.

[3] A. Brandon, *Audio Middleware* (part 1, 2 and 3), MixOnline Website (2007). Retrieved November 15, 2008.

[4] A. Schut, *GSTOOLKIT*, Utrecht School of the Arts, KMT. Unpublished MA-thesis (2008).

[5] *Half-Life 2*, Valve (2004).

[6] *Valve Source SDK* http://developer.valvesoftware.com/

[7] *Open Sound Control* http://opensoundcontrol.org/

[8] *Oscpack* by Ross Bencina http://www.audiomulch.com/~rossb/

[9] R. Kerstens, *Dynamic Ambiences and Atmosphere in Computer Games - a framework*

for dynamic sound design, Utrecht School of the Arts, KMT. Unpublished MA-thesis (2008).

[10] S. Huiberts & R. van Tol, *IEZA: A Framework For Game Audio*, Gamasutra, (2008). Retrieved January 23, 2008, from http://www.gamasutra.com/view/feature/3509/ieza_a_framework_for_game_audio.php

[11] Nonlinear sound design demonstrations http://adaptivemusicsystems.hku.nl/gameaudiolab/

[12] CPS by Bonneville http://cps.bonneville.nl/

[13] L.J. Paul, *Video Game Audio Prototyping with Half-Life 2*, Interactive Futures (2008). http://cfisrv.finearts.uvic.ca/interactivefutures/IF07/?page_id=31 and videogameaudio.com