VORPAL: A Middleware for Real-Time Soundtracks in Digital Games

Wilson Kazuo Mizutani (Author), Fabio Kon (Supervisor)
Computer Science Department
Instituto de Matemática e Estatística, University of São Paulo
São Paulo, Brazil
{kazuo,kon}@ime.usp.br

***Abstract*—Digital games are a form of real-time, interactive audiovisual media. However, developers do not give soundtracks the same attention they do to graphics and animation. The vast majority of games employ simple sample-based looped tracks for background music and triggered tracks for sound effects, severely limiting the design space of sound. In the masters thesis summarized in this paper [1], we propose a more flexible approach using procedural audio. The solution is supported by interviews with professional sound designers and academic literature. We also demonstrate its viability through the implementation of *VORPAL*, a game audio middleware, and *Sound Wanderer*, a proof of concept game that validates the technology.**

***Keywords*-digital games; dynamic audio; adaptive music; real-time soundtrack; audio middleware; game soundtrack;**

I. Introduction

A typical digital game consists of a software application we interact with in real-time through inputs – keyboard, mouse, joysticks, touch, etc. – and outputs – video and sound. All of these work together to compose an experience aimed at the entertainment of the user [2]. The real-time interaction, in particular, poses many technical and artistic challenges to developers, since there are endless possible performances the playing of a game can incur and only narrow time frames to work with during the execution of the application. In the case of sound, digital signal processing (DSP) is an effective technique employed to control audio dynamics in such time-constrained situations [3]–[5].

A. Motivation

While much is known and studied about how to produce high-quality, real-time graphical presentations for games, little is discussed about how the sound and its underlying technology catch up to that experience, besides the minimum necessary to play the right track at the right time. In the absence of dedicated tools, the workflow for adding a specific sound to a game is as follows [6]:

1) Sound designer play-tests the game and determines a new or different sound it should have;
2) Composer or sound designer creates (re-purposes, mixes, etc.) a computer file with a sample of the required sound;
3) Sound designer explains to programmers when and how the sample is supposed to be played during gameplay;
4) Programmers implement the new sound playback into the game code;
5) A new version of the game with the added sound becomes available;
6) Go back to (1).

There are two main shortcomings to this approach. The first is that the feedback cycle is too long, passing through at least two different development areas in a team. The sound designer depends on the full completion of the process to finally establish whether the improved soundtrack meets the desired experience – that is, the fact that sounds must be implemented *ad hoc* into the game turns the programming section of the team into a bottleneck for the sound design process. The second issue is the need for step (3). Programmers have likely no previous experience with sound design, nor do sound designers with programming. When the task is transferred from the designers creative process to the programmers actual coding, interdisciplinary differences may hinder communication, potentially increasing the number of feedback cycles required in the sound design process.

This problem is not exclusive to soundtrack in game development: the same happens with image textures and 3D models used for graphical representation. However, game development technologies are usually much more prepared to support these asset pipelines using *data-driven design* [7]. This technique relies on an intermediate representation – which is neither code or the assets alone – to represent how each asset fits into the game. Often, dedicated tools with graphical user interfaces (GUIs) are developed to assist in the creation and management of these data files [5]. Hence, a more productive approach to the sound design process in a game would be to use a dedicated middleware to design the soundtrack *and* integrate its content into the game. This way, commonly required features could be grouped into a single tool, allowing software reuse between applications.

The idea itself is not new, and a few game audio middleware products are already in use in the industry as commercial off-the-shelf solutions (COTS). Nonetheless, all of them offer sample-based audio support only (see Section II). While such method has proven itself versatile enough

Tecnology	License	Sound Representation	Work Abstraction
iMuse	Patented	MIDI-based	Sequences + conditional jumps
Wwise	Commercial	Sample-based	Tree structure
FMOD Studio	Commercial	Sample-based	Time-lines + conditional jumps
Elias	Commercial	Sample-based	Layered track loops
Pure Data	Modified BSD	Procedural	Data flow programming

Table I
COMPARISON BETWEEN THE STUDIED SOUND TECHNOLOGIES.

to ship games with sufficiently engaging experiences, it still limits the design space of sound since one cannot easily "break" audio samples apart to perform a more fine-grained composition of the soundtrack. Which, in turn, means there is space for original contributions in the field of game soundtrack technology using alternatives to sample-based audio. In the case of the presented thesis, we propose procedural-based audio and demonstrate its possibilities and viability.

B. Objective

The goal of the presented thesis was to design and implement a procedural-based audio approach to game soundtracks through a dedicated open source middleware, as a more empowering alternative than *ad hoc* implementations and a more flexible approach than current COTS solutions, based mostly on the proposals from [6]. We named the middleware *VORPAL*, a recursive acronym for *VORPAL is an Open-source Real-time Procedural Audio Layer*, and licensed it under the *Mozilla Public License 2.0*[1].

We validated *VORPAL* using two methods. First, we gathered system requirements from both literature on game soundtracks, interviews with professional sound designers active in the field, and analysis of COTS solutions already available. After developing the middleware based on these requirements, we verify it actually satisfies them by using it on a proof of concept game called *Sound Wanderer*.

C. Contribution

By implementing and validating *VORPAL*, the summarized thesis contributes to the field of real-time soundtracks in games by demonstrating a procedural-based approach is not only possible and viable, but is able to produce experiences which would be unfeasible with traditional sample-based solutions.

D. Text Organization

This text is structured as follows. Section II explains how we planned and executed the masters project, including studied literature and the gathering of system requirements for the design of *VORPAL*. Then, Section III presents the *VORPAL* middleware architecture. Section IV describes the proof of concept game *Sound Wanderer*, which verifies the system requirements of *VORPAL* are met. Finally, Section V discusses the results and main findings of the presented thesis.

[1]https://www.mozilla.org/en-US/MPL/2.0

II. METHODOLOGY

The methodology we used in the masters project was divided in three phases: mining of system requirements for a game audio middleware from literature and industry, design and implementation of the middleware, and validation of the technology through a proof of concept game. Here we focus on the first phase, since the others are covered by the remaining sections of the text.

We started the first phase by reading both specialized and academic literature to collect requirements for soundtrack technologies in games. Our investigation revealed the importance of dynamic, adaptive, and responsive soundtracks to the engagement of players [8], [9]. It also explained how these effects have been implemented in the past [10] and how they could improve [6]. These last studies were of particular relevance to the thesis. Collins [10] lists the main kinds of dynamics a game soundtrack can have, serving as a basis for the system requirements of *VORPAL*. At the same time, Farnell [6] defends that procedural-based audio is a more productive and efficient alternative to sample-based audio, which led to the core feature of our proposed middleware being support for procedural-based audio. These and other findings from the literature are detailed in [1, Chapter 2].

During the next part of the first phase, we interviewed three sound designers active in the field of real-time soundtracks. Two of them work for Brazilian game companies, and the third specializes in interactive sound performances, a field very representative of what happens in the soundtrack of a game. We describe the conclusions from each interview in [1, Section 4.2], which, together with the requirements from Collins' work [10], merged into the a list of eleven system requirements for game audio middleware, which can also be found at [1, Section 4.2]. Here, we highlight requirements number 1, *Independence from programmers*, and 3, *Real-Time Controlled Synthesis*, which were core guidelines during the second phase.

To close the first phase, we performed a comparative study (see Table I) of currently available solutions for dynamic soundtracks in games, including both COTS middleware tools – *iMuse* [11], *Wwise*[2], *FMOD Studio*[3], and *Elias*

[2]https://www.audiokinetic.com/products/wwise
[3]https://www.fmod.com

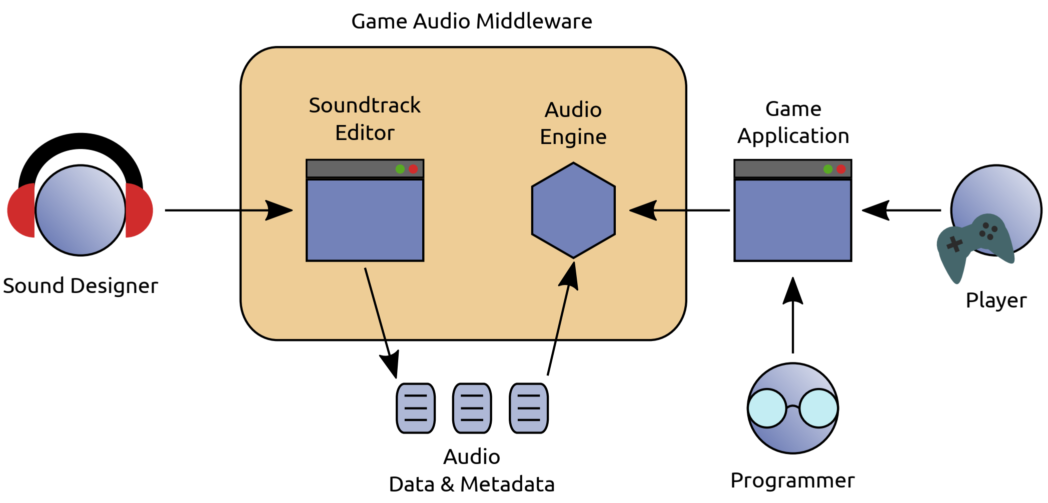

Figure 1. Generic game audio middleware

Studio[4] – and an open source but simpler alternative – *Pure Data*[5]. The motivation behind the inclusion of that last tool was Farnell's work [6], [12]. Further discussion can be found in [1, Section 3.4].

III. The *VORPAL* middleware

During the second phase of the masters project, we took the system requirements and the design of already published tools to design our own game audio middleware, *VORPAL*. Given the need for procedural audio support, and the short amount of time and participants we had, we chose to base our middleware on *Pure Data*, since it provides DSP technologies that simplify considerably the implementation of procedural audio features, and it is an already established technology in the research field of computer music. Based on this decision and the architecture of previous game audio middleware solutions, we divided the software architecture of *VORPAL* into two parts: an audio engine and a soundtrack creation kit. Figure 1 illustrates this reference architecture.

The audio engine is implemented as a `C++` library – since it is the most used language in game development [5], [13] – which can be linked to a game or game engine to produce the actual playback of the soundtrack during gameplay, applying any real-time dynamics prescribed by the sound designer. It provides an Application Programming Interface (API) for programmers to play individual soundtrack elements and assign them three-dimensional positions for spatialization effects. The game is also responsible for polling the API every frame to allow real-time controls over the soundtrack.

The soundtrack creation kit provides the sound designer with tools to determine the dynamics of the soundtrack. It exposes an abstraction layer over *Pure Data* through which

[4]https://eliassoftware.com/elias-studio
[5]https://puredata.info

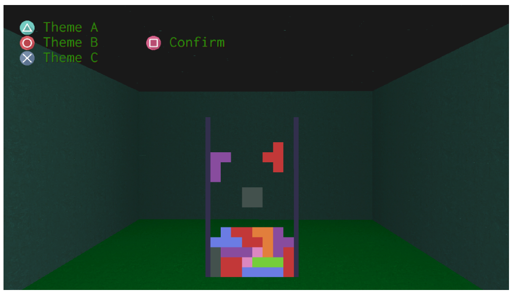

Figure 2. Screen capture from *Sound Wanderer*, during the first part of the game. Here we see a totem which references *Tetris* (*Nintendo*, 1989), playing a Markov-chain variation of its main theme

the sound designer can build and experiment with the rules and behaviors the sound of the game should follow. A thorough explanation of the respective implementations can be found at [1, Chapter 5].

IV. Proof of concept game: *Sound Wanderer*

Sound Wanderer is part of the third phase in our methodology and was developed in partnership with Dino Vicente de Lucca [14], one of the sound designers we interviewed during the first phase. We tried to emulate a real game development scenario where programmers (us) worked together with sound designers (them). We developed the game using the open-source engine Godot[6], demonstrating that our middleware can indeed be integrated with game engines as explained in Section III. The game itself followed no genre in particular, choosing a more experimental design in its experience where we could showcase *VORPAL*'s features.

A complete description of the game can be found in [1, Section 6.1], but we divided it in two parts. During the

[6]http://godotengine.org/

first, the game provides a 3D first-person control to the player, where they can explore a building and interact with it in a way similar to a museum or exposition. Figure 2 shows one of the parts players can interact with. They are faced with different totems referencing classic games and some of their music themes. The totem itself can play three different themes, but only one is the right one, and the player has to guess which. The challenge for them was that the themes were not played in their original composition, but in a procedural Markov-chain variation which randomly imitated the melodies of the original note-by-note passages. This would not be possible with a sample-based approach, where individual notes are too fine-grained to control. After completing three of these puzzles, an exit opens for the second part of the game.

During the second part of *Sound Wanderer*, the game changes to a 2D top-down perspective in an abstract, psychedelic environment. We designed an exploratory experience that would take the most out of the features of *VORPAL*. For instance, we placed four "wardens" spread over the space, each of which dynamically added a new track to the background music when approached. We recorded a gameplay video of the entire game as part of the final thesis presentation[7].

V. Conclusions

With *Sound Wanderer*, we demonstrated that *VORPAL* meets all the system requirements collected during the first step of the methodology and, most importantly, that using procedural-based audio in games is a technically viable alternative to current COTS sample-based solutions. The Markov-chain implementation in *Sound Wanderer*, in particular, shows that the level of control provided by *VORPAL*empowers sound designers to explore unconventional and creative uses of sound in games that would not be possible with sample-based audio. The implementation, however, was not without its faults. Using *Pure Data* as a basis for our soundtrack creation kit might have helped reduce development time, but its learning curve remained steep, even with support from the computer music research community. It also revealed a number of technical issues which affected the audio engine. When we discuss this in [1, Chapter 7], we leave as future work a re-implementation or altogether new production of procedural-based game audio middleware with more focus on usability.

During the execution of the project, we published one international conference article [15] together with a live demonstration of *Sound Wanderer* [14]. The source code for both *VORPAL*[8] and *Sound Wanderer*[9] are available on public repositories.

[7]https://youtu.be/7ON_gnZrHJ0

[8]https://github.com/vorpal-project/vorpal

[9]https://gitlab.com/Kazuo256/sound-wanderer

Acknowledgment

The masters project was supported by Brazil's National Council for Scientific and Technological Development.

References

[1] W. K. Mizutani, "Vorpal: A middleware for real-time soundtracks in digital games," Master's thesis, Instituto de Matemática e Estatística, University of São Paulo, São Paulo, Brazil, 1 2017.

[2] J. Schell, *The Art of Game Design: A Book of Lenses, Second Edition*. A. K. Peters/CRC Press, 2014.

[3] T. Engel and F. 5, "A technique to instantaneously reuse voices in a sample-based synthesizer," in *Game Programming Gems II*, M. DeLoura, Ed. Charles River Media, 2001, pp. 521–524.

[4] K. Weiner and D. Ltd., "Interactive processing pipeline for digital audio," in *Game Programming Gems II*, M. DeLoura, Ed. Charles River Media, 2001, pp. 529–538.

[5] J. Gregory, *Game Engine Architecture*. A. K. Peters/CRC Press, 2014.

[6] A. Farnell, "An introduction to procedural audio and its application in computer games," Aarhus University, Tech. Rep., 2007. [Online]. Available: http://cs.au.dk/~dsound/DigitalAudio.dir/Papers/proceduralAudio.pdf

[7] S. Rabin, "The magic of data-driven design," in *Game Programming Gems*, M. DeLoura, Ed. Charles River Media, 2000, pp. 3–7.

[8] N. Scott, "Music to middleware: The growing challenges of the game music composer," in *Proceedings of the 2014 Conference on Interactive Entertainment*, ser. IE2014. New York, NY, USA: ACM, 2014, pp. 34:1–34:3. [Online]. Available: http://doi.acm.org/10.1145/2677758.2677792

[9] E. Matos, *A Arte de Compor Música para o Cinema*. Brasília, DF, Brasil: Senac, 2014.

[10] K. Karen Collins, *Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design*. The MIT Press, 2008.

[11] LucasArts, "Method and apparatus for dynamically composing music and sound effects using a computer entertainment system," United States Patent 5315057, 1994.

[12] A. Farnell, *Designing Sound*. The MIT Press, 2010.

[13] R. Nystrom, *Game Programming Patterns*. Genever Benning, 2014.

[14] W. K. Mizutani, D. Vicente, and F. Kon, "Sound wanderer: An experimental game exploring real-time soundtrack with openda," Demonstration at the 12th International Symposium on Computer Music Multidisciplinary Research, 2016, São Paulo, 2016.

[15] W. K. Mizutani and F. Kon, "An extensible and flexible middleware for real-time soundtracks in digital games," in *Proceedings of the 12th International Symposium on CMMR*. The Laboratory of Mechanics and Acoustics, 2016, pp. 175–182.