Real-Time Procedural Risers in Games: Mapping Riser Audio Effects to Game Data

S. de Koning*

Breda University of Applied Sciences, Monseigneur Hopmansstraat 2, 4817 JS Breda, the Netherlands

Abstract

The application of audio to a nonlinear context causes several issues in the game audio design workflow. Mainly, industry standard software is not optimised for creating nonlinear audio, and a disconnect exists between the game's timeline and the audio timeline or grid. Standard middleware solutions improve the audio development workflow, but also cause new issues, such as requiring users to switch back and forth between different software before being able to test a sound in its context. This research applied procedural riser audio effects to a walking simulator horror game to explore an alternative to game audio development. The research investigated which are the essential parameters of the riser, and how they can be implemented in Unreal Engine based on the game's state and events. Interviews have been done with game audio designers, programmers, and leads/directors. The interviewees were shown a prototype, developed for this research, that allows the user to design a riser, control how it should adapt, and implement it in Unreal Engine. Analysis of the interviews indicates that modulating the volume and pitch of the riser over multiple layers are the most essential parameters. These parameters can best be modulated based on the player's current position, the elapsed time, and the amount of action over time. Furthermore, the data indicates that the approach put forward in the research is a viable and relevant solution to the stated issues and worth further investigation.

Keywords: Procedural audio; Riser; Adaption; Implementation; Workflow; Audio

1. Introduction

Adapting parameters of audio in real-time, instead of playing cutup pieces of the audio based on game state and events, could be a practical and efficient game audio design method. Colin Walder, Code Lead Audio & Localisation, states: 'By making a connection between the music and the game action we can effectively tell our audience what to feel' [1]. There is a discrepancy between audio being inherently linear, as it can only exist over time and therefore cannot be paused, and games being nonlinear. Because of this discrepancy, certain issues arise in audio design and the audio design workflow. Examples of these issues are industry standard software that is not optimised for creating nonlinear audio, visual and interactive context not being available while developing the audio, the limitations for a composer using audio adaption based on dividing the audio in different parts and layers, and a disconnect between the game's timeline and the musical timeline or grid. These challenges are inherent to the linear nature of sample-based audio being connected to real-time gameplay and will influence a player's experience of the game. To tackle some of these issues, different middleware solutions exist. However, on top of not solving all the stated issues, middleware programs cause some issues of their own, such as requiring its users to

* Corresponding author.
E-mail address: sdk.stijn@gmail.com

switch back and forth between a DAW, a middleware solution, and a game engine before being able to test a sound within its context. This research revolves around the question: What are the essential parameters to control a riser audio effect in a real-time video game, and how can they be implemented into Unreal Engine (UE) based on the game's state and events? The aim is to solve some of the issues that emerge from sample-based audio, by creating a system for dynamic audio, and to demonstrate the validity of this this approach with a specific test case. All relevant material used in this research is available at sdkoning.com/PF/RTRResearchMaterial.pdf.

To contextualise the research to a specific case, the implementation of riser effects in a horror environment is examined. Composer Mikael Baggström defines risers as: 'Sounds that continuously increases the energy and tension build up.' [2]. Risers are an interesting type of sound to investigate because their most important timing is at the impact point at the end of the sound's build-up. More standard sounds, such as piano notes or gunshots, have a very immediate impact at the sound's start. This makes risers difficult to use in an adaptive manner, because it is hard to start a riser without knowing when the impact is going to be. Currently, riser audio effects are prevalent in Hollywood audio, which may make them relevant for games as well. Also, risers are an effective low CPU and memory cost solution to get a lot of auditory impact and are applicable to both sound design and music. To further scope the research the adaptive risers are specifically tested on a horror environment in which the player moves around freely, and a jump scare occurs at some point. This test level has a limited number of external influences, is relatively simple to make, and the horror genre fits the focus on suspense of riser audio effects.

2. Theoretical background

2.1. The standard game audio workflow using middleware solutions

The current standard approach of implementing audio in games is with the use of middleware solutions, described by Böttcher and Serafin: 'Audio middleware is an interface between software, whereby the interface design is intended to reduce programming requirements for the sound designers, while allowing for connectivity with the game engine.' [3]

Guerrilla's Decima engine applies an integrated audio system instead of using a middleware solution. In a talk about this system, audio designer Anton Woldhek and audio programmer Andreas Varga cite their motivations for this approach to include optimising the workflow, emancipating the audio discipline, and instantly being able to hear work in-game [4]. Allowing audio to be tested in-game in a matter of seconds, reduced idea to game time and removed the programmer from the creative loop.

On top of using middleware, expressivity can be maximised by having sound designers and developers venture into each other's territories, and providing sound designers with real-time tools improves their ability to communicate audio to the listener in a more direct way [5]. Furthermore, real-time audio has a powerful influence on a game's interactivity, as immediately receiving audio feedback can be paramount to spatial and empathic immersion [6].

2.2. Dynamic audio design methods

A commonly used method in game music design, is to divide the audio into vertical and horizontal elements using reorchestration and resequencing [7]. Resequencing is transitioning between different pieces of audio, while reorchestrating applies different layers that fade in and out on top of each other. By applying different methods of reorchestration and resequencing, audio systems can be designed based on their required adaptiveness. New possibilities are generated by dividing the audio in stems and/or layers and applying a ruleset based on game data to what needs to be played and when. This reduces the music's repetitiveness, on top of which fewer audio assets are required and storage space can be saved. Adapting audio based on game action also allows for the use of music as a gameplay element, to add to the immersion. In Risk of Rain 2 [8], the player travels through levels by finding and activating a teleporter, which emits a vertical audio layer to the music that increases in amplitude as the player approaches [9]. Using these adaptive composition techniques, audio can be linked to game events while still conforming to an audio-based grid that adheres to tempo and measure. However, it can also cause a disconnect between the game action timeline and the audio timeline. Furthermore, when using reorchestration and resequencing, increasing audio's adaptability, limits the composer as musical fragments must be shorter and adhere to more rules and can get in the way of the player's flow [10].

2.3. Procedural audio in games

The term 'procedural audio' can be used to refer to hyper adaptive audio systems that generate audio on a note-by-note basis in real-time, but also for any audio that adapts based on external factors. In this paper, the more standard adaptive audio techniques are referred to as adaptive or nonlinear, and hyper adaptive audio systems are referred to as real-time or procedural.

Paul Weird, sound designer of No Man's Sky [11], states that by applying procedural audio techniques, memory can be saved, and sound designers can obtain new tools and ideas for applying less linear and more dynamic sound design [12]. Weird considers procedural audio useful when there is an issue of scale and to avoid repetition. A few downsides of procedural audio stated by Weird are that variety does not get perceived as much when everything constantly changes, that it can take a lot of time and budget to make, and that it requires clear communication between programmers and sound designers.

A 'lack of meaningfulness' is generally seen as a risk when using procedural audio. Weird touches on this problem: 'It is almost like, its treating sounds like a function, rather than a creative emotive element.' To address this, Weird added a 'human element' by analysing performances on a digital instrument made for Wwise. Mini Metro [13] applies serialism, a composition technique that uses sequential sets of data on different parameters working together to generate music [14]. Game data is combined with externally authored data, which consists of a ruleset that composer Freeland applies on their designed sounds, to again address the lack of a 'human element'. Rise of the Tomb Raider [15] uses a percussion system that applies a data driven system which analyses composed midi tracks to generate variations in real-time. By analysing externally authored MIDI data, again the issue of procedural music not feeling 'handcrafted' is addressed [16].

Kent Jolly, audio developer for SPORE [17], mentions another risk with procedural techniques being that the sampled audio often used for these techniques can sound unemotive [18]. This was addressed in SPORE by only using very short types of sounds. Every other example discussed also mainly uses shorter sounds. Also in SPORE, to prevent the audio from becoming distracting, after a certain time threshold the audio gets 'simpler' and less in your face.

2.4. Riser audio tools

To map out the elements that risers can consist of, their different uses, and to help define them, common riser production plugins' reoccurring features and notable differences have been examined. The following tools have been selected based on their popularity and how much they differ from their alternatives: Whoosh Designer [19], Gravity [20], Whoosh [21], Rise & Hit [22], and The Riser [23]. Risers generally consist of two parts: the build-up and builddown, or more commonly known as the attack and release. In between the attack and release is the impact. The only plugin that did not provide some form of impact sound is The Riser. The Riser is also the only tool that used synthesis instead of samples. Every analysed tool's description or manual mentions the plugin to be created for designing cinematic type sounds, which supports the statement in the introduction that risers are prevalent in Hollywood's film industry. Most extensive control is provided over the riser's volume and pitch. Furthermore, panning and filtering are often emphasised. Recurring functionalities are allowing the user to link the timing to the audio grid, multiple options for setting duration, layering, and extensive modulation functionality.

2.5. The Shepard scale

The Shepard scale is a prime example of risers and their usefulness, as well as a potential solution to keep risers going indefinitely. In an article about the application of Shepard tones in modern media, Rapan explains:

> By managing the amplitudes of each octave in a Gaussian bell-shaped envelope, the frequencies in the extremes are not perceived: frequencies at the lower end enter softly, and the upper ones disappear in a similar way. This means that the middle frequencies are heard fully. These frequencies are very clear and always moving up or down, but the ones at the last octave slowly disappear. This is done in order to prevent our ears being aware of arriving at a limit, but they do perceive the resulting endless rise or fall. [24]

2.6. Tools development and testing

When designing a product that will add to a larger ecosystem, it is important to keep this ecosystem in mind during development [25]. It should also be investigated whether it is of importance to collect concrete data or to build a theory. Furthermore, not only should it be tested if the tool is something its target audience would want to use, but also if the developed tool itself is easy enough to use [26].

To gain insight in his experience with procedural tools development, technical artist and procedural tools developer at Ubisoft Paris Twan de Graaf has been interviewed for this research. On top of saving time by being able to generate output more quickly, De Graaf mentioned that procedural tools can play a significant role in avoiding human error. He also listed a few common traps in procedural tools development. Firstly, he said to avoid developing all encapsulating tools. 'Fixing one thing can break 3 other things.' He also explained it makes finding bugs more difficult. Furthermore, De Graaf stated that he generally keeps parameters in the background first, because it is often a trap to expose too many parameters: 'It often turns out that you are not going to use 80% of them.' When asked about when a tool is finished, De Graaf replied: 'Basically never. […] They are done until somebody asks for a new feature.'

3. Methodology

3.1. Research method

Data has been gathered by interviewing audio developers in the field about a prototyped tool that implements real-time procedural risers. Quantitative data gathering has been waived because the perception and knowledge of audio of the general game audience can vary widely. Professional audio designers, unlike players, have a general understanding of audio and its practical uses in games. Furthermore, proving the method can enhance the experience for an audience without providing a method of applying real-time risers, would reduce the direct industry relevance of this research [27]. Developing a tool allows the interviewees to experience the real-time audio design method instead of having to speculate about the approach. Since the research is about exploration and understanding as opposed to attempting to validate a theory, the results should provide an indication based on the opinions of professionals.

Each interviewee has been questioned based on the same script, which includes questions about the background of the interviewee and their ideas about the topic, an introduction of the research, a first round of questions, a brief showcase of the tool, and a final round of questions. The interviews were conducted by video call, facilitating interviews around the world. The tool has been showcased by the interviewer, as sending a tool to interviewees to try it out themselves was out of the scope of this research, and having interviewees try out the software themselves could shift their focus to the software's user experience instead of the viability of the proposed method.

The collected data has been transcribed with Otter [28] and analysed using inductive thematic analysis with MAXQDA [29]. Grouping the data based on different themes served to obtain a clear overview of the answers and their relations to one another, to be interpreted to answer the research question.

3.2. Prototype tool

The uses of the prototype include building audio, implementing audio, making audio a gameplay element, and the sonification of physical elements or events. The source code for the prototype is available at https://github.com/StijndeK/RTR and an overview video can be found at https://sdkoning.com/PF/RTRShowcase.html. A more modular approach was preferred, to account for future innovations in the field and so that it can more easily be applied to other types of audio. To address the risk of missing a 'human element', control over the sound and adaption of the riser has been provided.

As concluded in Chapter 2.4, risers generally consist of three dynamic elements: the attack, the impact, and the release. Most emphasis is put on the attack as risers are often used to build up to something. Risers are made by modulating parameters of audio in order to first build up the audio in intensity during its attack and then decrease in intensity during its release.

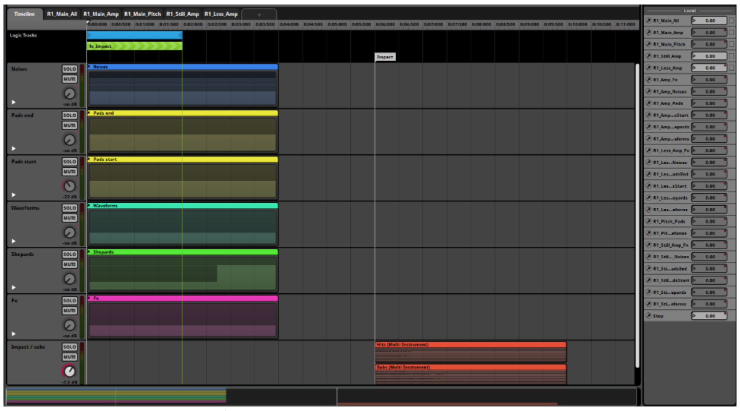
Three main factors are important to control the modulation: the modulation intensity curve, the speed in which the audio is modulated over the curve, and the range over which to modulate. These three factors should be set based on the sound and the audio parameter that is modulated. For example, modulating the pitch of a-tonal noise does not have any effect. Based on their auditory impact as well as on how much they occurred in the analysis of existing riser plugins, amplitude and pitch have been selected as the essential parameters to modulate.

As live synthesis was out of scope for this research, looping samples have been used. Riser samples were deconstructed to static loop-able samples without any modulation and their pitch and gain were modulated during playback over a multitude of layers. Based on the literature review a classification of five layers was constructed: noise, pads, impacts, Shepard scale, and FX.

To adapt the riser to game data, three types of modulation, that are combined to reach the eventual output, have been designed. The first is called position modulation and is modulated by the position the riser needs to reach. For the designed test level this is the player's distance to the impact point, relative to the distance between the start and impact point. The second is called action modulation and decreases the riser's intensity based on the amount of game action happening. The action modulation can be divided in two aspects: having the riser fall in intensity and reaching the lowest point of intensity where no more modulation is possible. During the lowest point of intensity, only minimal static sounds will be played. A Shepard scale is used here to still have the riser feel like it is building in intensity. The third type of modulation is time modulation. As in the example of Spore in Chapter 2.3, the audio becomes less intense after a certain amount of time, to prevent the audio from becoming too distracting.

To decide on the specific type of level to test the risers on, different test levels have been designed and compared, and based on this the case of a walking simulator horror game with a jump scare has been selected. The horror genre lends itself well to the intensity of risers and this case has little external factors that could influence the data such as enemy AI or a combat system. As not placing any other sounds in the context game could feel unnatural and would not accurately represent how well the riser works within context, the other sounds in the level can have an influence on the experience, which is taken into consideration during data analysis. To quickly develop the test level, a finished graduation project that fits the designed test case called Project Rookery [30] has been used. It should be noted that results based on this game can differ from results based on publicly released commercial games.

To first test the design and gain insight in the difficulties of applying the method in current standard middleware, the real-time adaptive risers have been implemented in the test level using the commonly used middleware solution FMOD [31] (Fig. 1). Designing and implementing the procedural risers using FMOD was inefficient, due to the need to manually set parameters and curves, where this process could be automated. In addition, as FMOD only provided a list of the parameters and their current value, there was only little overview. The eventual result within the test level was satisfactory, indicating the method of adapting parameters to suffice.



R1_Main_All	0.00
R1_Main_Amp	0.00
R1_Main_Pitch	0.00
R1_Still_Amp	0.00
R1_Less_Amp	0.00
R1_Amp_Fx	0.00
R1_Amp_Noises	0.00
R1_Amp_Pads	0.00
R1_Amp...sStart	0.00
R1_Amp...epards	0.00
R1_Amp...eforms	0.00
R1_Less_Amp_Fx	0.00
R1_Les...Noises	0.00
R1_Les...adsEnd	0.00
R1_Les...sStart	0.00
R1_Les...epards	0.00
R1_Les...eforms	0.00
R1_Pitch_Pads	0.00
R1_Pit...eforms	0.00
R1_Still_Amp_Fx	0.00
R1_Stil..._Noises	0.00
R1_Sti...adsEnd	0.00
R1_Stil...dsStart	0.00
R1_Sti...epards	0.00
R1_Sti...eforms	0.00
Stop	0.00

Fig. 1. Screenshot of the FMOD main event and its parameters (right side).

To decide on an approach to developing the tool’s prototype, common approaches to game audio tools have been examined and their main advantages and disadvantages have been listed and compared. As mentioned in Chapter 2.6, how the tool fits in the current ecosystem should be considered. The examined options include middleware or game engine plugins, game engine editor tools, and a middleware-based approach. The middleware-based approach applies a middleware editor used by the audio designer to design the audio and its adaption, that is then integrated in the engine using a plugin. Because of this research’s exploratory nature, there is a high likelihood that features will have to be adapted or extended upon. The tool should also be easy to integrate into the current audio workflow to mediate the risk, described in Chapter 3.1.4, of the tool not being viable because it takes too much time to implement in an audio designer’s workflow. Based on the examination of different approaches, a middleware-based approach was selected. The middleware-based approach is not the fastest approach, but it is the most modular, least restricted, it allows for quick implementation in any engine, and because it is based on current middleware solutions audio designers are used to the approach.

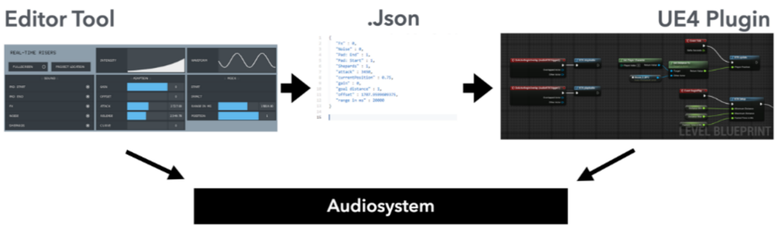


Fig. 2. A visualisation of the dataflow of the prototype.

The developed prototype consists of four elements (Fig. 2). The Audiosystem is used to play and modulate the audio. As with standard middleware, the editor tool and the engine plugin use the same low level audiosystem. In the editor tool (Fig. 3) the audio designer can design the riser, set settings on how the riser should adapt to game data, and play the riser by mocking game data. In the SOUND tab (Fig. 3 A), the user can select which layers should play. In the ADAPTION tab (Fig. 3 B) all settings on how to adapt the audio can be controlled. The MOCK tab (Fig. 3 C) can be used to play the riser within the editor tool. Visual feedback is provided with an intensity curve and a waveform visualiser (Fig. 3 D). Control over the three different modulation types is provided at the bottom of the screen (Fig. 3 E). The editor tool exports the user's settings to a .json file that is imported by the UE4 plugin. For the integration in UE4, two actors have been developed, one for the start and one for the riser's impact position. The riser can be implemented by dragging these to the correct positions in the level. Alternatively, blueprint nodes can be used.

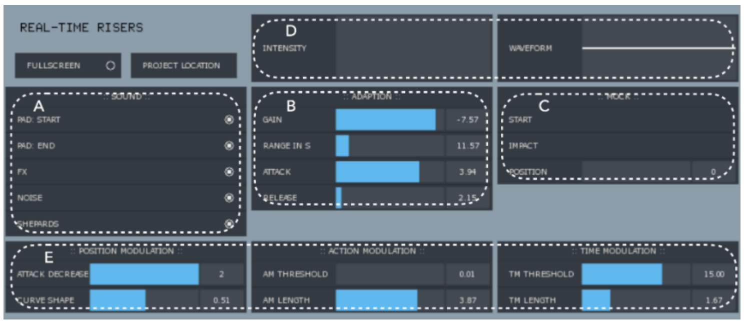


Fig. 3. The editor tool divided into various components.

3.3. Pilot study

To test the designed research method, an initial test interview about an early version of the prototype was done with audio director and composer Jonathan van den Wijngaarden. Some interview questions were edited or removed because the interview took ten minutes too long, and they were not as relevant to the research question. Valuable insights were gathered, and other then taking too long, the interview went smoothly. Based on these results, it seemed likely that the collected data would suffice.

4. Results & discussion

4.1. Data coding

Based on the gathered data from nine interviews, I created 48 codes and subcodes, divided into six topics (Fig. 4). ‘Parameters and Modulation’ are all codes that answer the part of the research question regarding what parameters to modulate and how to adapt them to game data. ‘Viability’ discusses the value and industry relevance of the proposed method. ‘Procedural Audio’ and ‘Background Information’ both give insight into the interviewee’s background. ‘Pipeline and Workflow’ includes every code about how to integrate the method in an engine. Finally, ‘Other Relevant Data’ collects other codes that do not fit the previous topics.

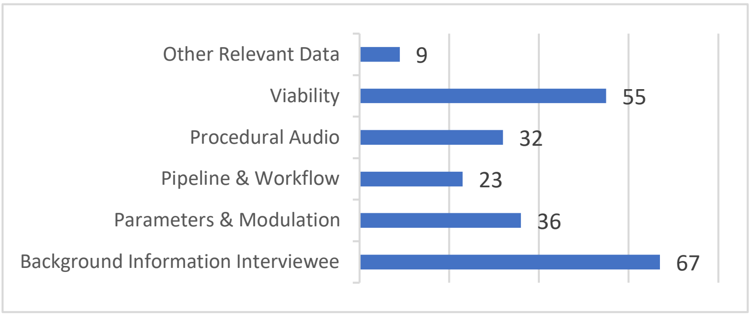


Fig. 4. Code density overview with the number of codes per topic.

4.2. Participants

Interview participants were based in East Asia, Western Europe, and North America. Table 1 provides an overview of the participants interviewed, their years of experience in game development, and their role titles. The role titles have been divided in four categories. The term audio designer was used for people who describe themselves mainly as being a sound designer or composer, which are the people who develop the audio assets and possibly implement them as well. The title audio programmer was used for people who focus on any game audio related programming. The category audio direction/lead was used for interviewees who work in managing and leadership type roles. Because audio directors or leads often have extensive experience in audio design or programming, they have been categorised based on whether they are more experienced in design or in programming. Due to differing career trajectories, the exact experience of participants within each category can vary widely.

All participants indicated that they enjoy experimenting and agreed the space for it is mainly based on available time and budget. All participants recognised the issues related to game audio as stated in this research. When asked about the advantages and disadvantages of procedural techniques, all participants agreed they can be technically demanding and are not always required. The main advantages mentioned are that procedural audio allows for faster reactivity and that it can automate much of the design process. A few interviewees also pointed out that by being able to react more quickly, game audio design can move more towards movie-like sound design.

Table 1. Overview of interviewees

Interviewee	Years of experience	Role
Davis, Will	20 <	Audio lead/director designer
Duquesne, Clément	5 – 10	Audio designer
Fournel, Nicolas	20 <	Audio lead/director programmer
Fuesslin, Florian	15 - 20	Audio lead/director designer
Hays, Tom	20 <	Audio lead/director designer
Huguenard, Charlie	10 - 15	Audio programmer
Smit, Tom de	> 5	Audio designer
Vera, Bogdan	5 - 10	Audio programmer
Wijngaarden, Jonathan van den	10 -15	Audio lead/director designer

4.3. Riser parameters and modulation based on game data

Eight out of nine interviewees agreed with the parameters and modulation used in the prototype. This supports the analysis of existing riser plugins in the literature review (in Chapter 2.4). The interview with Duquesne did not contain sufficient data on the parameters to form a clear conclusion of his opinion on this subject. All participants did name parameters to add, and four of the interviewees specifically indicated that there are many parameters to add. Also, two interviewees said that a riser does not have to be a new sound but can also be the adapting of already playing sound. Audio programmer Vera stated: ‘Think of it like the riser itself does not have to be just the sound that is rising. […] It is the gesture of the riser that can be applied to many things.’ Based on these results, it can be argued that the riser’s parameters in the prototype of volume and pitch modulation are the basic parameters for a riser to adapt to game data, at least for the specific test level used in this research.

All participants except Duquesne agreed with the way the parameters are modulated based on game data in the prototype. This indicates that modulation based on action, time, and action over time can be effective game data to control the riser’s parameters. Duquesne found the riser did not rise enough at the end. Huguenard said: ‘The abstraction of position in motion or action, that makes sense.’ Huguenard, De Smit, and Davis raised the question of what would happen if the player started to behave very unpredictably. Fuesslin argued players tend not to behave unpredictably in this test case. About unpredictable player behaviour, Vera mentioned: ‘if they are trying hard to break your riser, it is their fault. The average person really will not, because it is a video game anyway. I can go really close to a texture and look at it and it is going to look like it is low res, because it is a video game.’

4.4. Pipeline, workflow, and user control

Four interviewees stated that the tool’s middleware-based approach works well. Fournel emphasised it works well as it saves time, and Hays noted there might be difficulties integrating the tool in an existing middleware: ‘I think you could come close, but its underlying intent is different.’ The other interviewees thought integrating the method into an existing standard middleware or engine would be a better approach. It is notable that two of the four interviewees who found the prototype’s approach sufficient also mentioned integrating the tool into current standard solutions. This supports the research outcome that audio developers prefer tools to be integrated directly into their game’s engine. In addition, all four interviewees that stated the prototype’s approach works well are categorised as audio lead/director. This could be because people in leadership positions are less focused on the practical workflow, but it could also be argued they may be more experienced with adapting their workflow. All audio designers mentioned they would prefer the tool to either be in the middleware or the game engine. Audio programmers showed greater awareness of the challenge of implementing the prototype into their existing engine or middleware, but they also recognised that audio designers would want it to be integrated in this way. Future research should be done to integrate the prototype directly in an engine or middleware.

Audio designers generally wanted more control over the sound design. Fournel stated: 'I think where you want to give options is more on the creative side and less on the implementation side.' Fournel also mentioned that this type of tool provides new controls that most audio designers are not currently accustomed to having. Vera supported this by stating: 'there is a tension when it comes to sound design people not being in control of the same parameters they are used to. [...] You have to teach them a whole new technical skill when it comes to procedural. It is almost like a procedural sound designer is a different role than a sound designer.' The use of the prototype would either require sound designers to learn a new skillset, or for the prototype to provide more control over the riser's sound design.

4.5. Viability and industry relevance

It can be argued that the research is relevant and viable as eight out of the nine interviewees stated that the method has potential. Some interviewees were very convinced, and others agreed that it has potential, but also mentioned it may have some issues and conditions. Five interviewees said they are curious how well it performs for edge cases, indicating that although they believe the approach to be viable, there are certain issues that have not yet been addressed in the prototype. The prototype's viability is supported by all interviewees either mentioning examples where they would have liked to, or could have, applied it in the past or mentioning potential future use cases. Six interviewees mentioned a use case for which they would have liked to apply the method in a game they have previously worked on. The other interviewees said that their type of projects would not have required it. Seven interviewees put forward potential future use cases for which they would find the tool interesting or for which they thought it might work well.

Duquesne was the only interviewee who was not positive towards the viability, as he stated: 'I am so familiar with Wwise that I think I would still go to Wwise […] And also, I did not have many risers in my games. [….] One of the reasons that I am not very attracted to specific audio middleware is that I have the competence to do audio tools that answer to specific needs.' Duquesne also said: 'if it is not important for your game, then you do it in another simpler way or you just do not do it if it is not that important.' Duquesne's more negative response to the viability of the prototype can be explained by his ability to program himself, not finding the modulation sufficient, being used to a workflow with Wwise, not having used risers much before for games, and wanting full control over the audio effect.

4.6. Validity, replicability, and relevance

Interviews with audio developers with different types and amounts of experience from different parts of the world provided a broad perspective on the research question. Information on the interviewees and their backgrounds is provided and all used material, including the codes, interview scripts, and the prototype tool, is available online[1]. Furthermore, asking questions based on a proof-

[1] All relevant data has been made available at: https://www.researchgate.net/publication/352020351_Real-Time_Procedural_Risers_in_Games_-_Research_data

of-concept prototype, reduces the amount of speculation, and applies the research to an example that is directly relevant to the industry.

Because the interviewer was also the one who created the interview and prototype, the participants may have been more willing. Furthermore, since interviewees were asked about their general impression, were encouraged to mention anything they might think of, and questions were contextualised to the background and experience of each interviewee, not every question was formulated in the same way or asked in the same order. This is consistent with the research's qualitative nature, but it should be noted that the order and way the questions were asked may have influenced the interviewees answers.

The results indicate an interest in the proposed method as well as that it can be a viable industry application, which supports the industry relevance of the research. Eight of the nine participants indicated that the proposed method is viable. Furthermore, all participants recognised the stated issues.

4.7. Ethical Concerns

Even though diversity among the interviewees was focused on when reaching out to professionals and people around the world have been interviewed, there is a lack of diversity among the interviewed professionals, as they are predominantly white and male. This can firstly be caused by the general lack of diversity in the industry, but this is also a reflection of wider issues of access to technology for women and minorities, and this is rooted in deeper systemic issues of sexism, racism, and capitalism in society. Other potential explanations are that minorities in the industry might not be confident to do interviews, they might be frequently asked for these types of interviews because they are part of a minority, or they might not want to be asked based on their gender or other characteristics.

4.8. Summary

The analysed data indicates that the parameters and modulation used in the prototype are the essential parameters to control a riser. The parameters the prototype uses are volume and pitch modulation over five layers, with control over the modulation's seek speed and curve. They can be adapted to game data using modulation based on action, time, and action over time. The real-time riser effect can be efficiently implemented using a middleware-based approach, although integrating into middleware or an engine is often preferred, especially by audio designers. Furthermore, the results indicate the method is a viable approach, which improves the research's relevance and validity. The diverse backgrounds of the interviewees, the contextualisation of the research question using a prototype, and the replicability of the research also improve the validity of the research.

5. Conclusion and further directions

This thesis set out to explore potential improvements to the standard game audio workflow by tackling various reoccurring obstructions, using the scoped case of riser audio effects for a walking simulator horror game. Research into existing work confirmed there to be room for optimisation in the game audio workflow, highlighted some successful solutions in the past, provided insight in the most important features of existing riser audio tools being gain and pitch control, and provided insight in a potential approach to keeping risers going for a long time by using Shepard scales. The prototype proposes a method of integrating audio based on adapting parameters of the audio itself, instead of triggering different pieces or layers of audio based on what is happening in the game, with a quick and easy to use integration into Unreal Engine. Professional audio developers with varying roles in the field have been interviewed based on a prototype that applies this method to reduce speculation.

The analysed data indicates the essential parameters to control a riser in a real-time video game are the riser's gain and pitch over a multitude of layers. However, a lot of other parameters were named that sound designers would prefer to also control, such as panning and filtering the frequency spectrum. These parameters can be efficiently modulated based on action, time, and action over time, where every modulation type has a separate range and threshold, and every layer has a separate modulation curve and range. The real-time riser effect can be efficiently designed and implemented using a middleware-based approach consisting of a standalone tool with a user interface that extracts a datafile, which is read by a game engine plugin that can be used to link the riser to game data. Using this approach, the tool is not restricted by the limitations of common middleware. However, further research could explore different approaches to the tool, because a high number of interviewees mentioned they would prefer the tool to be integrated in standard middleware or the game engine. The data also indicates the prototype's approach to implementation in Unreal Engine 4 works well, i.e. using two game objects, or with the alternative approach using blueprints. Furthermore, the data gathered showed overall positivity towards the viability of the method, also for other game cases and audio sound effects. Future research could explore applying the method to other audio effects and games. The diverse backgrounds of the interviewees, the contextualisation of the research question using a prototype, and the replicability of the research speak to the validity of the research. However, user testing the tool with audio developers and user testing the resulting game audio with players could also be valuable future research to gain more insight in the method's viability and how to apply it. Also, four of the interviewees mentioned the proposed method allows for making audio less reactive and therefore allows closer alignment with the techniques used in film and other linear media, which is another interesting field to explore in further research. This study indicates that adapting the pitch and gain of multiple layers of a riser based on time, action, and action over time, implemented with an automated pipeline, may be a viable solution to current game audio design obstructions and is worth further investigation.

Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] C. Walder, Synchronizing Action-Based Gameplay to Music, in G. Somberg (Ed.), Game Audio Programming 2, Taylor & Francis Group, Boca Raton, 2019, pp. 332–347.

[2] M. Baggström, Sound Design – Create Riser FX for Transitions, retrieved November 2020, from https://www.skillshare.com/classes/Sound-Design-Create-Riser-FX-for-Transitions/1618082930.

[3] N. Böttcher, S. Serafin. A review of Interactive Sound in Computer Games: Can Sound Affect the Motoric Behaviour of a Player?, in K. Collins, B. Kapralos, H. Tessler (Eds), The Oxford Handbook of Interactive Audio, Oxford University Press, Oxford, 2014.

[4] A. Varga, A. Woldhek, The Next-Gen Dynamic Sound System of Killzone Shadow Fall, 2014, retrieved October 2020, from https://www.gdcvault.com/play/1020559/The-Next-Gen-Dynamic-Sound.

[5] S. Kohata, An Interactive Sound Dystopia: Real-Time Audio Processing in NieR: Automata, 2014, retrieved October 2020, from https://www.youtube.com/watch?v=BrUQdd96qzk&t=28s.

[6] M. Haggis-Burridge, Four categories for meaningful discussion of immersion in video games, 2020 extracted June 2021, from https://www.researchgate.net/publication/340686774_Four_categories_for_meaningful_discussion_of_immersion_in_video_games.

[7] C. Huguenard, Note Based Music Systems, in G. Somberg (Ed.), Game Audio Programming 2, Taylor & Francis Group, Boca Raton, 2019, pp. 307–331.
[8] Risk of Rain 2 (video game), Hopoo Games, 2020.
[9] S. de Koning, Composing for Games as a Film Composer, 2019, retrieved October 2020, from https:// sdkoning.com/PF/ComposingforGamesasaFilmComposer.html.
[10] M. Gordon, DOOM: Behind the Music, 2017, retrieved October 2020, from https://www.youtube.com/watch?v=U4FNBMZsqrY.
[11] No Man's Sky (video game), Hello Games, 2016.
[12] P. Weir, The Sound of No Man's Sky, 2017, retrieved October 2020, from https://www.youtube.com/watch?v=zKJ_XuQjjiw.
[13] Mini Metro (video game), Dinosaur Polo Club, 2014.
[14] R. Freeland, Serialism & Sonification in Mini Metro, 2014, retrieved October 2020, from https://www.youtube.com/watch?v=FgV4hSfsl00&t=37s.
[15] Rise of the Tomb Raider (video game), Crystal Dynamics, 2015.
[16] P. Lamperski, B. Tahouri, Real-Time Procedural Percussion Scoring in 'Tomb Raider's' Stealth Combat, 2016, retrieved October 2020, from https://www.gdcvault.com/play/1023215/Real-time-Procedural-Percussion-Scoring.
[17] SPORE (video game), Electronic Arts, 2008.
[18] K. Jolly, A. McLeran, Procedural Music in SPORE, 2008, retrieved October 2020, from https://www.gdcvault.com/play/323/Procedural-Music-in.
[19] Whoosh Designer (audio DAW plugin), Zero-G, 2014.
[20] Gravity (audio DAW plugin), Heavyocity, 2016.
[21] Whoosh (audio DAW plugin), Tonsturm, 2014.
[22] Rise & Hit (audio DAW plugin), Native Instruments, 2014.
[23] The Riser (audio DAW plugin), Air, 2014.
[24] E. Rapan, Shepard Tones and Production of Meaning in Recent Films: Lucrecia Martel's Zama and Christopher Nolan's Dunkirk, in D. Power, S. Deutsch, LK. Sider (Eds), The New Soundtrack, Edinburgh University Press, Edinburgh, 2018, pp. 135–144.
[25] D. Aliakseyeu, User Experience Evaluation in Industry, Guest lecture presented in Breda University of Applied Sciences, Breda, 2020.
[26] D. Gaiduk, How to Do Usability and UX Testing for Mobile Apps, 2019, extracted October 2020, from https://medium.com/uxreality-blog/how-to-do-usability-and-ux-testing-for-mobile-app-211f92f3cd6d.
[27] A. Burls, D. Gray, M. Kogan, Salutogenisis and coaching: Testing a proof of concept to develop a model for practitioners, International Journal of Evidence Based Coaching and Mentoring, Vol 12 (2) (2014) 41-58.
[28] Otter (transcribing tool), Otter.ai, 2016.
[29] MAXQDA (qualitative data analysis tool), QSR International, 2020.
[30] Project Rookery (video game), graduation project University of the Arts Utrecht, 2020.
[31] FMOD (middleware), Firelight Technologies (1995)