AI-Driven Sonification of Automatically Designed Games

Sara Cardinale, Michael Cook and Simon Colton

School of Electronic Engineering and Computer Science, Queen Mary University of London

Abstract

Music and other sound cues are used to support, enhance and change the experience of playing every type of digital game. In this paper we explore the *sonification* of game states, the direct translation of game events or situations into music. We present three strategies for allowing AI to convey information to the player using sonification, creating a pleasing soundtrack while also affecting the play experience. We extend this to automated game design and suggest that sonification could be embedded into automated game designers to directly influence the design process.

Keywords

Generative Music, Procedural Audio, Automatic Game Designers

1. Introduction

Sonification is the act of using sound or music to convey non-aural information, either for functional or aesthetic purposes. Early examples of sonification that are often given are the Geiger counter, which translates radiation levels into an auditory warning, making it easier to hear both the presence of radiation, as well as the intensity. Sonification is also used as a game mechanic or narrative device in games such as *Alien Isolation*, *In Other Waters*, or *S.T.A.L.K.E.R.*

Automated game design (AGD) is the science and engineering of AI systems that model, participate in or support the game design process. This can include tools which help users explore design work; models of game design theories or frameworks; and systems which design games autonomously. A common goal for AGD research is expanding our idea of how AI can influence game design, and finding ways for AGD systems to create new kinds of experience [1], or solve new kinds of design problem, using novel approaches [2].

Composing music for videogames is a complex creative task, and has been approached in a variety of ways, from linear soundtracks [3] through to complex adapative or generative works that dynamically compose accompaniment to player actions [4]. The composition of music is thus as much part of the game design process as artistic direction or narrative design, and therefore a valid topic for automated game design research to consider. In fact, music offers a unique opportunity for an automated game designer to communicate emotions, knowledge and atmosphere to the player, through a careful combination of musical knowledge, sonification strategies, and game AI analytics.

In this paper we present the results of our sonification experiments working with Puck, an automated game designer. We report on preliminary experimentation sonifying Puck's games, and considering different ways in which information about a game can be sonified, for both emotive and communicative purposes. We also comment on how difficult implementing each of these strategies is: how much it is affected by a specific game design; how much musical knowledge is required; and how expressive the resulting music is. We show that sonifying game states can yield a range of exciting new musical approaches for games and propose that, in the future, automated game designers such as Puck could be adapted to design their own bespoke music composition systems for each new game they design.

2. Background

2.1. Puck

Puck is an automated game designer, described in [5], released as a downloadable app that can be run locally on any computer. Each version of Puck keeps a record of what it creates, allowing it to build an understanding of its design space over time, and maintain a creative history of how its work has developed. Puck also works on long timescales, taking days or weeks to finish a game design, and switching between different projects. These features aim to improve the 'presence' of the system, in the computationally creative sense, by allowing people to develop an appreciation of the system and anticipate its creative process. Puck's creative process is constantly visualised and can be watched.

Puck uses a combination of evolutionary search and partially-exhaustive content generation to slowly explore a design space, using past results to adjust its future search decisions, but retaining all results whether good or bad. Evaluation of games is achieved using AI agents

The Experimental AI and Games Workshop at AIIDE
s.cardinale@qmul.ac.uk (S. Cardinale);
mike@possibilityspace.org (M. Cook); s.colton@qmul.ac.uk (S. Colton)

which play the game in different ways and at different skill levels. The performance of these agents is then analysed by Puck and combined with other analytical tools to create a measure of how promising or interesting Puck considers the game to be.

Puck is built in a modular fashion, allowing for new capabilities to be added and removed dynamically. In [6] the authors consider how the automation of game feel design can be added to Puck's workflow through the use of a pre-existing tool for generating game feel effects called Squeezer. In this paper, we explore new areas for which no existing tool exists, namely the design of dynamic and emotive soundtracks that are suited to a previously unseen game's design. Our focus for this paper was to explore what possibilities exist for sonifying game states based on the same kinds of analytical process that Puck already uses to evaluate games for quality.

For our experiments in this paper, we looked at two different games. As in Johansen' study of game feel effects in Puck, we chose one automatically designed game and one human-designed game, to provide a balance of perspective between Puck's games (which may not be commercial-quality) and established games that have been refined over decades of play. As one of our goals is to enable Puck to design musical systems for any game, studying human-designed games is important from a co-creative perspective, as well as studying how Puck can create musical systems for its own games.

2.1.1. Antitrust

Antitrust is a game designed by Puck in 2021 and described in [5]. It is played on a 5x5 board by two players. Players take it in turns placing pieces of their colour on the board, meaning a board tile location can have three states: empty, or with a piece of either player colour. At the beginning of a player's turn, if there are any rows of four or more pieces of the same type (of either colour) they are removed from the board. When a player plays a piece, if the board is full the game ends, and the winner is the player with the most pieces on the board. Figure 1 shows a screenshot from Antitrust.

2.1.2. SameGame

SameGame is a game designed by Kuniaki Moribe in 1985, and a popular game for amateur developers to clone and develop variants of. It is typically played on a 10x10 board for a single player, but many variations exist and Puck's version is played on an 8x8 board as this is the maximum board size Puck can currently represent. The board is initially filled with a random collection of pieces of four different colours. When a player clicks on a tile, all contiguously connected tiles of the same colour are destroyed (assuming at least three tiles are connected). Pieces then fall down and gather together to fill in gaps, and the player scores points proportional to the number of tiles destroyed. The game ends when the player can no longer destroy tiles, or the board is cleared.

Table 1
NRT Rewrite rules for major and minor chord starting points. In each case, the starting chord is $\{a, b, c\}$.

NRO	Major	Minor
R	$\{a, b, c+2\}$	$\{a-2, b, c\}$
P	$\{a, b-1, c\}$	$\{a, b+1, c\}$
L	$\{a-1, b, c\}$	$\{a, b, c+1\}$
N	$\{a, b+1, c+1\}$	$\{a-1, b-1, c\}$
N'	$\{a-2, b-2, c\}$	$\{a, b+2, c+2\}$
S	$\{a+1, b, c+1\}$	$\{a-1, b, c-1\}$

2.2. Neo-Riemannian Theory

Neo-Riemannian Theory (NRT) comprises methods for analysing chord progressions composed using *triadic chromaticism*, in which the progressions are not restricted by tonality. This means that the music can change in ways that are unexpected to the listener, providing a wide range of emotional effects. Such chord progressions can be challenging to analyse with conventional music theory, which mostly relies on keys and modes to understand the musical context and the development of harmony and melody. Analysing a piece of music through the lens of NRT involves identifying which NRO (Neo-Riemannian Operator) or sequence of NROs (a *compound NRO*) has been applied to a triadic chord to produce the next in the progression, if any. In order to facilitate the application of NRT to generative music, Cardinale and Colton [7] rationalised NRT in terms of rewrite rules based on the original NROs, as per table 1.

NRT is well-suited for analysing film and videogame soundtracks, as this musical genre often uses chromatic chord progressions to quickly respond to on-screen or in-game events [8]. For example, the sudden and dramatic appearance of a character can be immediately matched with a powerful change in chord progression. Furthermore, the relationship between chords described by NRT is well-suited to be implemented in generative systems to compose associative film and videogame music that follows a visual media narrative. The formalisation of NRT as an analysis technique was carried out by Cohn [9] where a mathematical description of NRT was put forward. Moreover, referring to film music, Lehman [8] mapped *compound* NROs to emotional/situational changes in film narratives, as per table 2, which we draw upon later.

Table 2
Association of NRO sequences and emotional and/or situational scene elements [8].

Compound NRO	Emotion/Situation
LP	Antagonism
L	Sorrow, loss
N	Romantic encounters
PRPR	Mortal threats, dangers
RL	Wonderment, success
NRL	Suspense and mystery
RLRL	Heroism (Lydian)
NR	Fantastical
S	Life and death

2.3. Related Work

As discussed earlier, sonification has been used widely as a tool for data comprehension, accesibility, and creativity. Games research has also leveraged sonification to explore new applications to the medium. For example, *SoniFight* [10] creates additional aural cues for games to increase accessibility for visually impaired players. This utility software allows players to add aural cues for meaningful gameplay information, such as how many seconds are left in the round, player's health, amount of ammunition, and the player's location in the playfield. SoniFight uses *watches*, a pointer chain that finds the value of interest (e.g. player's health) and its data type, and *triggers* to play the audio cue based on the watch value. Players can share their SoniFight configurations for games, allowing for collaboration between users. While our approach here is not built with accessibility in mind, it could be an interesting extension of this work to consider it in the future.

Game AI research has also applied sonification to game content generation. For example, *Sonancia* [11] blends level architecture and audio, choosing and mixing sound to create the soundscapes of a level and create an aural experience that can be adapted to any level. Sonancia focuses on creating frightening and tense soundscapes. The sonification system builds the sense of tension as the player progresses through the level, ensuring that player engagement is maintained throughout the gameplay by subtly choosing and changing instruments, so that the music does not become distracting. Sonancia uses two algorithms, one to choose instruments and one to arrange the way that sounds are played based on the current intensity. The results of this tool showed that it can consistently create unique soundscapes and it can create a sense of narrative progression within the level.

Our approach has similarities with Lopes et al.'s work, in that one of our objectives is to provide an aural context for the narrative arc of gameplay. However, we aim to provide general sonification techniques that can be adapted to a range of game designs, from single-player arcade games to two-player strategy games. Additionally, rather than aim for a designed tension arc, our sonification strategies adapt to the current state of the game, led by the player or players. Our work can therefore be seen less as a tool for a designer creating a specific experience, and more as an accompaniment or enhancement to the game's natural flow.

There is also a well-defined space of audio-only games, or games with strong aural components that can be played with little or no visual information. This includes games designed to be played through movement or other indirect sensors, such as *Zombies, Run!*, *J. S. Joust* or *Bounden*. It also includes games designed with blind or visually-impaired users in mind, such as *The Vale* [12], which uses binaural sound and conveys all necessary information to play the game purely through audio. While this is not the same as sonifying game information through music, it does show how games can fully rely on aural information for their design. Our work focuses on augmenting visual games for the time being, but the possibility of applying this to fully audio-driven games in the future holds a lot of promise.

3. Sequencer-Based Sonification

We initially experimented with sonification by building a system that would continually produce music sonifying the state of the game, rather than individual player activity. Inspired by games such as *Chime Sharp* [13], and by grid-based MIDI controllers such as the Novation Launchpad, we treated Puck's game board as a one-track *sequencer*. The sequencer's 'track' starts at the top-left board position (as viewed by the player) and playing left-to-right, top-to-bottom. When the sequencer head reaches a new board position, if there is a game piece on the board at that location it plays a note, sample or other sound depending on the type of piece. When the sequencer head reaches the end of the board, in the bottom-right, it resets back to the top-left.

Supporting the sequencer, we added a basic ambient backing track, with the sequencer head progress matched to the BPM of the backing track. Because player activity does not directly affect the music at the moment of play, the sequencer and the speed of play are unrelated to one another, which we hypothesised would allow the music to remain at a sufficient distance from the player to avoid interfering with gameplay. We added a small visual indicator to the board to show the location of the sequencer head, so the player could more easily identify the progress of the music. Figure 1 shows a screenshot of Antitrust being played in Sequencer mode, with the yellow tile on the second row showing the location of the head, which gently fades in and out as it moves between

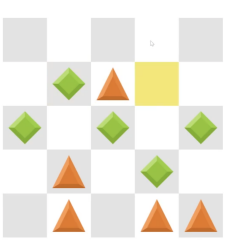

Figure 1: A screenshot of Antitrust with a sequencer. The yellow tile indicates the sequencer's current position.

grid locations.

Our initial testing with this prototype suggested two effects to us: first, although we intended there to be distance between the soundtrack and player actions, we found the music nevertheless had a subtle impact on our play choices, encouraging the player to avoid grouping too many pieces in a small area, and filling in gaps where no sounds were being produced to even out the disruption of sounds in the full loop. Secondly, we observed that in a game such as Antitrust, where the objective is area control, the sequencer's output gently indicates which player is 'winning' in the sense of having overall board control. The generated music flows between emphasising one player's pieces or the other.

3.0.1. Implementation

Of the three strategies we present in this paper, sequencer-driven sonification is the easiest to implement within an automated game designer, as it requires no contextual information about the game itself and can, in theory, be implemented in any game. However, although it can be implemented with no additional information, some games suit it better than others. *Antitrust*, with its relatively small board size and sparse piece coverage, has a naturally slow buildup of pieces, and every piece is precisely placed by a player, making its sequencer output relatively atmospheric. *SameGame* by contrast starts with a randomly filled board of the maximuim size allowable by Puck, which means the sequencer output has less meaning and is more cacophonous. Thus although this is a one-size-fits-all approach, it is not always the best choice.

4. Heuristic-Driven Sonification

This approach uses heuristic functions that measure properties of the game's flow and progress to modify an ambient soundtrack accordingly. This approach is customised more precisely to a particular game design, but as a result requires more preparation to use.

We define a *progress heuristic* as a simple measurement of a game's proximity to an end state. More formally, we could define such a heuristic as a function which maps a game state to a number in the range $[0, 1]$, where 1 indicates a game state that has reached completion (a win, loss or draw)[1]. Progress measured this way is not monotonic increasing, in that progress can be undone and games may move further away from completion. Progress heuristics are also not an exact reflection of how close a game is to completion. Instead, what we aim to achieve with a progress heuristic is a measurement that is close to the average player's *approximate sense* of how close a game is to completion.

As an example, consider *Antitrust*. Antitrust's game end state is the board being full of pieces, therefore a useful progress heuristic would measure the number of pieces on the board, and divide by the number of spaces on the board. However, pieces can be removed from the board if rows of 4 or more are made, and this happens often during the game, therefore a game that is close to completion can be reset to an earlier stage of progress. Nevertheless, the 'fullness' of the board does correlate with an observer's sense of how close the game is to completion.

We tie the progress heuristic for the current game to the background sound in our sonifier. In our experiments we use the progress heuristic to drive the BPM of the background track, making it faster as the game nears completion, updating it each time a player makes a move. This could be tied to other aspects of the music, however, including the overall volume, the mix of different tracks, or the selection of instruments. Multiple heuristics could be made, tracking different game features (such as the progress of individual players to winning, with each controlling the mix of competing instrumental tracks, for example).

4.0.1. Implementation

Implementing this approach for an automatically designed game is not entirely straightforward, as a progress heuristic must be created that suits the game design. For example, the progress heuristic mentioned above for *Antitrust*, i.e., how close the board is to being full, is the exact opposite of an appropriate progress heuristic for *SameGame*, where the game board begins filled up. In the case of Puck, we are able to automatically select progress heuristics from a catalogue of pre-written functions based on the end conditions for the game – for example, an ending condition based on a player making three-in-a-row would track the length of the longest row of pieces. However, automatically generating bespoke progress heuris-

[1]The definition of 0 would vary depending on the game in question, as the opposite of a game's end state is not necessarily a game's starting state.

tics, that take into account the complex dynamics of the whole game design, is a more challenging task. We discuss this in Future Work.

5. NRT-Driven Emotional Sonification

This approach uses game events as triggers for changes within a generative musical composition. The sonification system receives contextual information about changes to the game state, and translates this into emotional cues which are mapped to chord transitions via Neo-Riemannian Theory rewrite rules, as described above.

5.1. Evaluating Game Events

Unlike the previous two sections, which sonify the overall flow of the game, this approach directly responds to in-game events. For our purposes we restricted our system to respond to player actions only, namely tapping the board to place pieces in *Antitrust*, or to destroy pieces in *SameGame*. However, this approach could be easily generalised to respond to other in-game events.

When a player takes a game action, we need to be able to provide some context to the sonification system about how the soundtrack should respond. We focused on evaluating the quality of the move made, so the player (and any observers) can gain an awareness of how good or bad each move is perceived to have been as the dramatic arc of the game continues. To evaluate a particular move as good or bad, we use Puck's in-built MCTS agent to play every potential move available to the current player and rank them according to how good it expects the move to be. When the player selects a move, the move's overall rank is then sent to the sonifier, so it can respond with an appropriate shift in emotion in the soundtrack.

Although the same MCTS system is used to both compute opponent moves and evaluate move quality, the AI opponent does not always take the move considered 'optimal' because it is playing at a lower level of skill than the AI agent used to evaluate move quality. As described in [5], we vary the skill level of an AI MCTS agent by changing the computational resources provided to it. In our prototype sonification example we use a high-skill MCTS agent to evaluate moves, and a medium-skill AI to play the game.

5.2. NRT-Driven Synthesis

In this approach, we utilise NRT-driven sonification of player moves: given the ranking of each player's moves, we sonify significant moves (either good or bad) played across the course of a game. During our initial experiments, we did this by recording a full gameplay session with move analysis and composing the music post-hoc, sonifying the first, best and worst move made across the whole game.

In the previous examples, we made use of samples to play sound. For this sonification approach, we created a tool which implemented the generative rationalisation of NRT as described in [7], to provide emotional adaptation of the music based on the quality of the move made by the player. Our tool, *GENRT*, allows both planned, scripted and dynamic music generation using NRT transforms. This can be used to generate background music for videos, using a cue-sheet, or to respond with dynamic music to accompany live gameplay. Additionally, GENRT can produce various styles of music by allowing the user to control instrumentation, tempo, elements of rhythm, aspects of the chord changes allowed by generative NRT, and how a voice-led melody is produced above (or below) the chord sequence. In particular, GENRT allows music to be generated in a series of episodes, each with different specifications for the music production. The specifications are blended over a series of bars, using sliding probabilities, instruments overlapping with sliding volumes and averaging of parameters. The blending of generative specifications leads to a fairly smooth blending of the music produced. In particular, it is possible to have episodes with increasing tempo, which can subtly increase perceived tension.

In addition, we extended Cardinale and Colton's rationalisation of NRT by implementing re-write rules for *suspended* (*Sus2* and *Sus4*) and *augmented* chord types and adding them to our GENRT tool. This extension improves the ability of the tool to convey emotion, providing different types of chords and therefore different musical colours. For instance, Sus2 and Sus4 chords do not contain the third scale degree to classify them as major or minor, giving them a neutral sound. Furthermore, the lack of the third degree and its replacement with the second, in the case of Sus2, or the fourth degree, in the case of Sus4, creates a small cluster of two intervals close to each other (1^{st}-2^{nd} or 4^{th}-5^{th}). This cluster calls for an aural resolution to the third scale degree to finally reveal the chord's key, thus creating tension when, or if, there is no resolution. Expanding the palette of chords GENRT composes with enables the generation of a broader and richer collection of emotional music accompaniment.

GENRT produces background music for Puck gameplay videos as a chord progression, by starting with a given chord, then repeatedly randomly choosing a compound NRO with some user-imposed constraints, and applying this to produce new chords. Bass, melody and percussion lines are added to correspond to the chords, with fairly simplistic techniques. To make this background music bespoke to a Puck game playthrough video, partic-

ular compound NROs are used to substitute the random ones at key points in the video narrative. These substitutions are chosen according to table 2 to appropriately reflect the game change: the RL (wonderment) compound NRO is used to choose the chord played when the first move occurs; RLRL (heroism) chooses the chord for the best move; and PRPR (danger) is used for the worst move. For post-hoc accompaniment production, the user provides a text file of cues, specifying the timestamps and nature of the most important moves in a game video. In future, the evaluation Puck makes of a player's move will be used to drive the music generation during live gameplay.

We originally found the background chord sequences to be too dramatic for the emotional chord changes to be audible. Hence we added a constraint that during background music generation, the random compound NROs chosen must produce chords where the tonic, third and fifth are all within a given, *fixed* key. In addition, the application of an emotional compound NRO forces the change of the fixed key to the one specified by the chord produced. We found that this approach makes music where the dramatic change was clear and sustained over the duration of a few following chords. Additionally, using the episodic nature of GENRT music production, we increased the tempo slowly over two episodes spanning most of the video. We also used the episode specification to slowly change the bass from a soft synth sound to a louder, more piercing cello line, and added tick-tock style percussion to further subtly increase tension in the music. To synchronise the emotional chord changes with the video, we implemented an exhaustive technique which tries all possible chord durations and percentage decrease in duration from the slow to the fast episode.

We have provided a recording of a game of *SameGame* with an accompaniment produced by GENRT, which demonstrates our NRT-driven approach.[2] The emotional chord changes are synced to the three important moves (first, best, worst) to within an average of 0.3 seconds.

6. Future Work

In this paper we primarily focus on using AI techniques such as game state evaluation as a driver for sonification strategies designed by a human expert. We plan to develop a system that is able to understand what would be an appropriate sonification strategy for a given game. It might take into consideration game design choices such as the type of board, the game's rules and what bad or good turns consist of. This system could be implemented as a module within Puck or another automated game designer to create soundtracks for new AI generated games and compose music that is specific to the game's design, its gameplay and rules.

In future, we aim to empower automated game designers to create games with a sonification-first approach, rather than selecting sonification strategies after the fact. For example, designing a game where playing well leads to more harmonious, interesting or melodic output. This could lead to the emergence of new types of game where player behaviour is subconsciously led by the sonification strategy, gently showing the player optimal strategies, or even fully teaching the player the game simply through sound cues.

We are interested in the sonification of automated game design process, the process that an AI such as Puck goes through when creating a game. This would allow the sonification of the game designer's ideas, and allow the audience to hear the music develop and change as it encounters good or bad ideas. For example, we would like to create a soundtrack to Puck's design process which sonifies the steps that it goes through to develop a new game, mapping or associating certain features and aspects of the design process to musical elements like BPM, modality and instrumentation, moving from ambient to more fast-paced styles. This is especially interesting as Puck designs abstract puzzle and strategy games, and sonifying the environments, game pieces, and rules it chooses can lead to wide musical variety of styles, keys and instruments within the soundtrack. It could also provide a way to add what is known as *framing information* to the creative process. Framing is a computational creativity term for an AI's ability to communicate its creative process to an audience [14]. This would allow for a non-textual framing, with music that conveys the direction of the creative process, how well it is progressing, and whether unexpected things are happening.

Puck's games are similar to genres of game that are typically simple visually, including abstract strategy games and casual puzzle games. These games are played on a single screen, with basic but eyecatching visuals that are easy to read at a glance. While these games can be aesthetically pleasing and excellent examples of design, they represent only one type of game experience.

In our approaches in this paper, we connect music cues and outputs to discrete and easily-identified aspects of gameplay. However, sonifying the player experience in a more complex game such as a 3D open-world adventure game would require a much deeper understanding of the game's design, context, and intended player experience. While there are many examples of specific instances of dynamic, adaptive or context-sensitive music for videogames, there are no examples of systems which can design such generative music systems themselves

GENRT is in an early stage of its development, and there is much work to be done to make the music it produces not just appropriate for a particular game, but also

[2] Available at tinyurl.com/genrtdemo. Playback is muted by default - the unmute button is in the top-right of the video.

to exhibit high levels of musicality and diversity over the games it works for. We plan to eventually make it available as a plugin for a major game engine such as Unity or Unreal, to make it easily available for people to make bespoke soundtracks for games and/or for automated game designers to develop music-generation systems which run as the game is played.

7. Conclusions

In this paper we have described three approaches to sonifying games in the design space of *Puck*, an automated game designer. Each of the three approaches surfaces a different kind of information to the player, at different levels of prominence, and with differing levels of complexity in their implementation. All three approaches can be easily reapplied to other games in Puck's design space, meaning an automated game designer could be extended to apply these techniques to games as they are designed.

Music composition, sound design, and game data sonification are currently unexplored aspects of automated game design. We believe we have shown there is great potential in this space not simply to make compelling *static* soundtracks for games, but to use music and sound as a new generative layer through which the AI can communicate and shape dynamic and complex experiences. We are excited to expand this further, and apply our work in Neo-Riemannian Theory to build more expressive composition systems. Our next step is to study user responses to these sonification strategies, and better understand not just the emotional and personal response to the music itself, but to also study the measurable effect the sonification may have – consciously or otherwise – on the behaviour of the players experiencing it. We are looking forward to exploring the expressive and creative strengths of music composition in an automated game design setting, and building tools that open up this power to game designers of all kinds.

Acknowledgments

This work has been funded by UKRI and EPSRC as part of the "UKRI Centre for Doctoral Training in Artificial Intelligence and Music", under grant *EP/S022694/1*, and by the Royal Academy of Engineering Research Fellowship Scheme. We would like to thank the anonymous reviewers for providing helpful feedback, which improved this paper.

References

[1] G. Barros, M. Green, A. Liapis, J. Togelius, Data-driven design: A case for maximalist game design paper type: Position paper, Proceedings of the 9th International Conference on Computational Creativity, 2018.

[2] N. R. Sturtevant, N. Decroocq, A. Tripodi, C. Yang, M. Guzdial, A demonstration of anhinga: A mixed-initiative EPCG tool for snakebird, in: Proceedings of the Sixteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2020.

[3] C. Larkin, Hollow Knight Soundtrack, 2017.

[4] P. Weir, No Man's Sky Soundtrack, 2016.

[5] M. Cook, Puck: A slow and personal automated game designer, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2022.

[6] M. Johansen, M. Cook, Challenges in generating juice effects for automatically designed games, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2021.

[7] S. Cardinale, S. Colton, Neo-riemannian theory for generative film and videogame music, in: Proceedings of the International Conference on Computational Creativity, 2022.

[8] F. Lehman, Film music and neo-riemannian theory, Oxford Handbook (2014). doi:10.1093/oxfordhb.

[9] R. Cohn, Introduction to neo-riemannian theory: a survey and a historical perspective, J. of Music Theory (1998) 167–180.

[10] A. Lansley, P. Vamplew, C. Foale, P. Smith, Sonifight: Software to provide additional sonification cues to video games for visually impaired players, The Computer Games Journal 7 (2018) 115–130.

[11] P. Lopes, A. Liapis, G. N. Yannakakis, Sonancia: Sonification of procedurally generated game levels, in: Proceedings of the International Conference on Computational Creativity, 2015.

[12] F. Squirrel, The vale: Shadow of the crown, 2021.

[13] S. Curran, Chime sharp, 2016.

[14] J. Charnley, A. Pease, S. Colton, On the notion of framing in computational creativity, in: Proceedings of the Third International Conference on Computational Creativity, 2012.