Sonancia: Sonification of Procedurally Generated Game Levels

Phil Lopes, Antonios Liapis and Georgios N. Yannakakis
Institute of Digital Games, University of Malta, Msida, Malta
louis.p.lopes; antonios.liapis; georgios.yannakakis@um.edu.mt

Abstract

What is the most efficient approach for orchestrating the design of different facets of creativity within the domain of digital games? How can creative elements brought from level design effectively be coupled with audio in order to create tense and engaging player experiences? In this paper the above questions are posed through the sonification of procedurally generated digital game levels. The paper details some initial approaches and methodologies for achieving this core aim.

Introduction

Digital games are interactive and multi-faceted experiences relying on visuals, audio, gameplay and narrative (Liapis, Yannakakis, and Togelius 2014). Procedural content generation (PCG) has become progressively more popular in both commercial games and academia (Togelius et al. 2011), not only as a means of combating content creation burdens, but also integrated into a game's design. Sound generation is not often the focus of PCG research, downplaying its potential in enhancing immersion (Collins 2013), evoking player mood (Scirea, Nelson, and Togelius 2015) and further personalizing the experience.

This paper introduces *Sonancia* as a first prototype towards the inclusion of procedural audio in generated games. *Sonancia* is a system built for generating multiple facets of horror games, with the intention of creating tense and frightful experiences, similar to the game *Amnesia* (Frictional Games, 2010). Audio plays an important role in such games as it attempts to evoke specific moods of unease, which are built in the mind of the player without direct visual representations. In the current *Sonancia* prototype discussed in this paper, the game's level architecture and soundscape are generated. Levels take the form of a haunted mansions' rooms and are procedurally generated via genetic search, while audio snippets are placed and mixed based on the room setup. In that regard, audio generation uses the level architecture to inform which hand-authored sound will be placed in which room, leading to *multi-faceted computational creativity* (Liapis, Yannakakis, and Togelius 2014) as one generated component (levels) influences the other (audio). As *Sonancia* is only the first prototype of a larger system spanning the generation of more facets and more influence between the different generative processes, this paper discusses the successes and limitations of the current framework.

Background

In this section we review game industrial applications and academic studies that are related to *Sonancia* and its underlying framework.

Procedural Content Generation

Digital games have often relied on PCG techniques in order to generate virtual content on demand. Games such as *Minecraft* (Mojang, 2009) and *Borderlands* (2K Games, 2009) rely on constructive PCG techniques for map and weapon generation respectively. Within academia, work on PCG has been extensive and has ranged from complementing industry-dominant constructive PCG approaches (Togelius et al. 2011) to tailoring generated content to specific player experiences (Yannakakis and Togelius 2011). PCG has also been used for assisting designers, in tools such as *SpeedTree* (IDV Inc., 2009), *Sentient Sketchbook* (Liapis, Yannakakis, and Togelius 2013) and *Tanagra* (Smith, Whitehead, and Mateas 2010). However, rarely has the PCG community dealt with audio, with a few notable exceptions (detailed below). While PCG researchers focus on visual and interactive aspects of games, game audio is important in creating immersive experiences for players (Collins 2013; Gasselseder 2014).

Games such as *AudioSurf* (Dylan Fitterer, 2008) and *Vib Ribbon* (Sony Entertainment, 2000) procedurally generate levels driven by characteristics of the background music (e.g. beats per minute or musical intensity). These games fall into the rhythm game genre, where game mechanics are closely tied to the rhythm and sound of the background music. *Sonancia* attempts to apply the inverse approach by distributing sound through a level in order to create dissimilar experiences, as sound does not necessarily consist of "music" solely but of a soundscape that exists within that generated world (Garner and Grimshaw 2014; Collins 2013).

Game Facet Blending

The medium of digital games is a combination of different types of artistic artefacts, that complement and work in conjunction to create an emergent interactive experience. This can be seen as a blend of different artistic artefacts (Lopes and Yannakakis 2014) representing dissimilar creative facets/domains in digital games (Liapis, Yan-

nakakis, and Togelius 2014). Six different creativity facets have been identified in games: visuals, audio, narrative, ludus, level architecture and gameplay. This paper will explore specifically the blend of level architecture with audio.

Game facet blending has been used rather extensively, although not specifically referred to as such in the literature. Studies with the *Angelina* system (Cook and Colton 2011) already explore the sequential facet creation of two different game facets, specifically level architecture (i.e. the level layout and player/enemy placement) and ludus (i.e. rule-sets), for automatic game generation. *Game-o-matic* (Treanor et al. 2012) blends narrative and game-play via mapping verbs and nouns to specific game-play mechanics. Through this blend of mechanics and narrative concepts, interesting and provocative themes can potentially emerge.

Specifically related to the audio facet, *AudioInSpace* (Hoover et al. 2015) blends both game-play and audio, within a side-scrolling space shooter that evolves its shooting mechanics based on the music playing in the background. The music is either pre-selected by the user or is procedurally generated via artificial evolution. Other works have concentrated in generating audio for narrative foreshadowing in games (Scirea et al. 2014) or specifically to generate music that relates to certain moods (Scirea, Nelson, and Togelius 2015), which could potentially be applied for level sonification. This paper, instead, focuses on blending level architecture and audio, putting an emphasis on methods that have the capacity of successfully sonifying unseen (procedurally generated) levels.

Sound in Games

It can be argued that most games already display a form of "procedural sound", where player actions determine what sounds or music the game should play (e.g. players firing guns in the background of a multi-player shooter) or through the simple actions of non-player characters that inhabit that virtual world (Garner and Grimshaw 2014). This work however argues that this is a limited and narrow way of coupling sound with game levels and new approaches could potentially be developed to better orchestrate the two with the specific aim of increasing the player experience and immersion. Collins (2013) argues that the visual aspects in digital games tend to be the player's principal focus during play, while sound tends to work at a more subconscious level, which also suggests why players tend to remember the sounds of play much easier than the visuals after play has finished. Several studies also suggest that picking the appropriate sound (i.e. what to play) and placing it at the appropriate moment (i.e. when to play) can significantly enrich the player experience as diegetic and non-diegetic sounds have a tendency of helping players immerse themselves into the virtual world (Gasselseder 2014).

This is a core reason why more audio solutions need to be explored in game facet generation. This paper brings forward methodologies in order to answer the specific problems of "what to play?" and "when to play?" in procedurally generated levels.

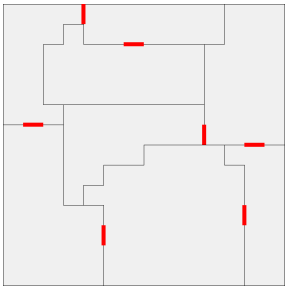

Figure 1: Generated manors consist of a set of rooms and interconnecting doors. Thin (black) lines represent walls, while thick (red) lines represent doors.

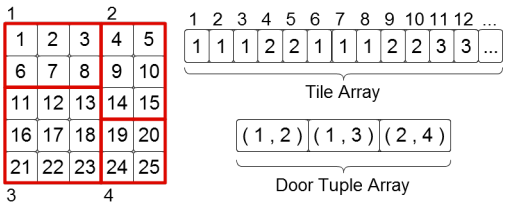

Figure 2: (Left) A phenotype of a 5x5 map with 4 rooms. (Right) The genotype representation of the first 12 values of the phenotype on the left, where the index is the spatial location of the tile and the integer value is the room identifier. An array of tuples represent the available connections between rooms.

Level Generation

In this section the level generation algorithm of *Sonancia* is described. Then initial results of the 3D rendered levels of *Sonancia* are shown. Finally a brief overview of the efficiency and empirical values obtained from the algorithm are presented.

Algorithm Description

In the initial version of *Sonancia*, levels consist of "haunted mansions" containing multiple rooms separated by walls and doors, which interconnect (see Figure 1).

Due to the previous success of search-based PCG approaches (Togelius et al. 2011), it was opted to use a genetic algorithm (GA) using roulette selection, but no recombination (Mitchell 1998). The level representation consists of an array of integers, where each index value of the array is the spatial positioning of the tile in a map and each integer value is the associated room that the tile belongs to. An array of tuple objects, containing 2 integers informs which rooms are connected between them (see Figure 2). This representation allows the GA operators to repartition rooms of a map easily, by simply changing the integer value associated to the specific tile index.

The fitness function evaluates the shortest path of each individual between a start and a goal room (see Figure 3), and gives higher fitness to individuals whose path goes through as many rooms as possible. Fitness is calculated as follows : $f = R - P$, where R is the reward function that sums 10

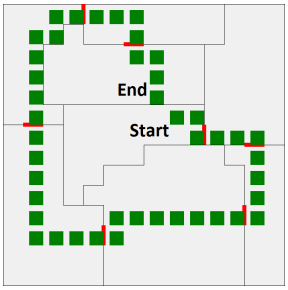

Figure 3: High fitness values correspond to individuals whose shortest path is the one that goes through as many of the chambers as possible. The fitness of the example level illustrated in the figure is 80.

to the fitness value for every room traversed by the shortest path; P is the penalty function that sums 10 for each room that does not have at least one connection to another (i.e. at least one door) discouraging empty unusable rooms. Fitness is calculated in this fashion because the path that players must take in order to progress through the level (i.e. critical path), affects player experience largely and, thereby, allows the sonification system to distribute sound throughout the level so that player experience is affected according to level design affordances. The only level constraint is a limitation to the number of maximum rooms, by which fitness of 0 is given to chromosomes containing more than 9 rooms.

The current system implements three different mutation operators, but no crossover, as it disrupts the chosen representation too heavily. The wall shift mutation retracts or pushes a random wall between two rooms of a map. The divide room operator picks the largest room in the map and divides it in half creating two new smaller rooms. Lastly the door mutation picks two adjacent rooms at random and adds a door if none exist between rooms; if a door exists between rooms it is removed. Only one door can be associated between the same two connecting rooms. There is a chance that each gene will use wall shift mutation (50%), door mutation (50%) or divide room mutation (15%); the disruptiveness of the divide room mutation is offset by its lower chance. As a safety measure, once all mutations are applied to an individual, a flood fill repair function is applied to each room of the level. This is done in order to keep the consistency of room representations and that no rooms with less than 5 tiles are created; if such are generated they are assimilated to a randomly chosen neighbouring room.

The genetic parameters consist of a population of 100 individuals with a maximum run of 100 generations. Elitism is set to retain 3% of the best population for the next generation. The genetic algorithm is initialised by cloning a pre-determined symmetrical level to form a population.

3D Rendered Levels

In the current version of *Sonancia* a 3D level rendering option exists. Figure 4 illustrates the level generated in Figure 1 as a 3D render. In order to render 3D levels *Sonancia* uses the *Unity 3D* (Unity Technologies, 2005) game engine.

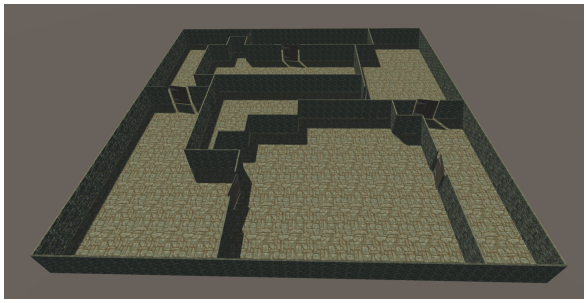

Figure 4: Unity 3D render of Figure 1

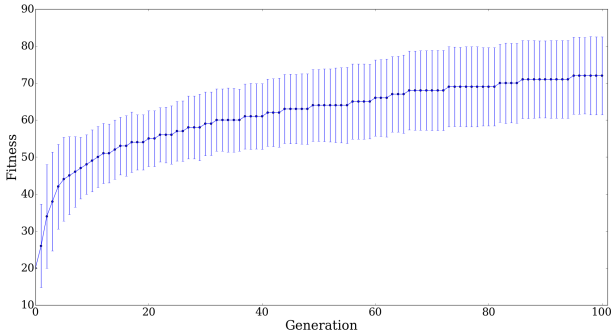

Figure 5: Average fitness value and standard deviation of 100 runs of the level generation algorithm.

Further improvements such as the addition of lighting and texturing are planned to be added into future iterations of the *Sonancia* system.

Level Generation Results

All parameter values were obtained empirically through testing the algorithm over 100 runs each. Figure 5 depicts the average fitness value of 100 runs across 100 generations using the parameters and operators described in the previous section. The GA gradually converges to higher fitness values.

Figure 6 shows different levels generated by the GA. For the current prototype these are favourable results, and provide varied types of levels for testing our proposed sonification system. During experimentation it was observed that the number of rooms averages around 6 to 7, commonly with a linear progression from start to end, which was expected. A limitation of the current level generator, is that horror aesthetics for level architecture are not being considered, even though some interesting side-effects emerge (i.e. narrow corridors and dead-ends). Further iterations of *Sonancia* will take these aesthetics into account (e.g. lighting and texturing).

Level Sonification

Level sonification consists of choosing and mixing sound to simulate the soundscapes of a virtual environment and create an audio experience that is tailored for any level. *Sonan-*

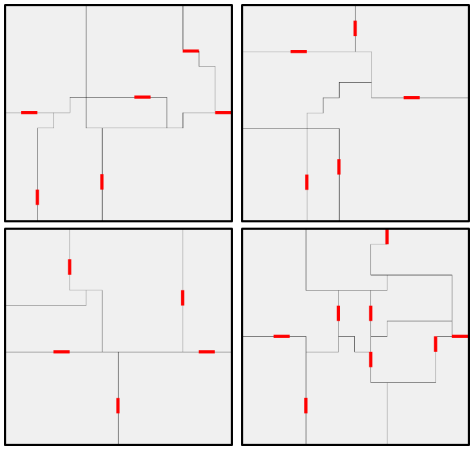

Figure 6: Four different levels generated by *Sonancia's* level generation algorithm. with corresponding fitness values of 70 (top left), 60 (top right), 60 (bottom left), and 90 (bottom right).

cia specifically sonifies levels with the objective of creating horror and suspenseful soundscapes. In this section of the paper we describe the level sonification of *Sonancia*, which consists of two main sub-systems: 1) how audio is selected and 2) how new audio is played in a procedurally generated level.

Audio Playback

Sonancia sonifies sound according to a linear progression of suspense and the critical path of the level. Suspense requires specific structures and progressions (Cheong and Young 2008), where a sense of unease slowly builds as players progress through a level. Subtle sound selections and mixing control is necessary to keep players immersed and engaged during their playing experience.

To make level sonification possible *Sonancia* has direct access to a soundbank of around 50 human authored recordings. All recordings have an average length of 4 seconds, and are of standard digital game WAV file format (2 channel at 44.1kHz 24bit). For *Sonancia* to be able to pick what sound to play, each individual file is represented by different parameter values detailing the specific sound: *instrument*, *note*, *octave* and *intensity value*. The instrument, note and octave parameters detail, respectively, what instrument is playing, at what note and in which octave. This information is necessary for *Sonancia* to mix different sounds available in the soundbank. Finally the intensity value consists of an empirical measure obtained from human listeners on how "intense" that sound is. In future work this parameter will be obtained by the system itself through a crowd-sourced model (Shaker, Yannakakis, and Togelius 2013) capable of ranking sounds by intensity. In the current version we base this value on human empirical values in order to test the capabilities of *Sonancia*.

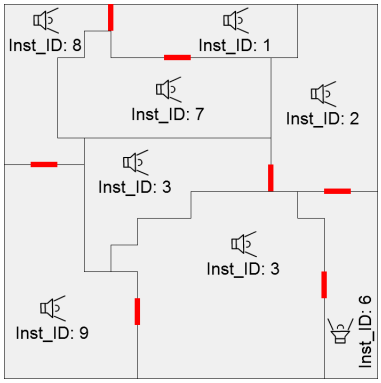

Figure 7: The instrument-based sonification distributes different instruments across the rooms of a generated level. In the example above 8 different instruments are placed in the available rooms.

Sonification Algorithms

Sonification algorithms are specifically related to how sounds are picked and distributed throughout the generated level. Picking what sounds are going to be loaded into memory, ready to be played at specific points in the game is an important part of increasing the audio fidelity and avoiding breaking player immersion. The system currently implements two different sonification algorithms: the *instrument-based* and the *intensity-based* sonification.

The instrument-based sonification first chooses sound files based on the instrument and gives higher probability to instruments who have not been previously chosen. Similar to the roulette wheel selection, a uniformly-distributed random value determines which instrument is selected, and subsequently which recording as well. Each instrument is given an equidistant interval, that is further split by the different recordings of that specific instrument. This interval may vary depending on the total number of instruments available. Once a selection is made, the selected recording is removed from the roulette, and the interval of the instrument is reduced by 50% and re-distributed equally among the instruments who have not yet been selected. If an instrument has no more recordings it is removed from the roulette. This algorithm ensures that no file is selected twice, even though instruments can potentially be selected more than once. Each instrument is associated to a specific room, to offer a different experience as players progress through the level (see Figure 7).

The intensity-based sonification rearranges the way sounds are played according to the level's critical path and intensity value, following a linear progression. On that basis, lower intensity sounds are played at the beginning of the level, while higher intensity sounds are played further down the path as players progress through the level (see Figure 8).

Real-Time Mixing Algorithms

The mixing algorithms consist of rules on how the system mixes the different sounds selected. While the sound distribution approaches described earlier choose what sounds

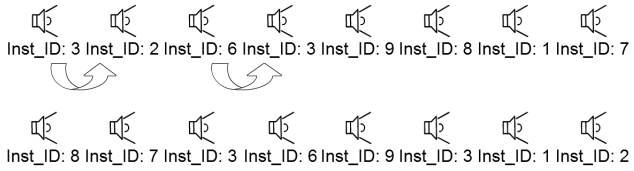

Figure 8: Intensity-based sonification: the particular order of intensity (top figure) is sorted (lower to higher intensity) according to the intensity value of each sound file (bottom figure).

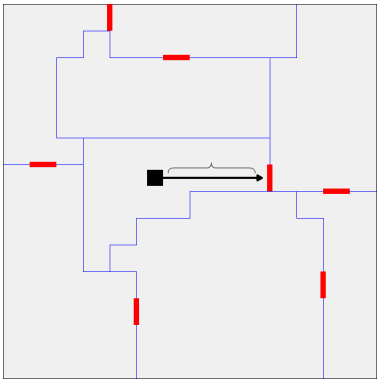

Figure 9: As players move towards the next room the volume of the sound associated to the next room will gradually increase. Once a player enters the new room, the new sound will be played in its entire volume, while the other room's sound will start to decay.

should be played, mixing decides when to play them. For this kind of algorithm to produce satisfactory results it is necessary to have real-time information of the game state (e.g. where the player is currently) in order for *Sonancia* to know when to play or mix a specific sound during play.

The current version of the system has all the selected sound files looping in the background. The distance mixing rule will increase or lower the volume of sounds depending on the distance of players between adjacent rooms (see Figure 9). Players approaching a different room will gradually start hearing the sound from the adjacent room until trespassing, where the full volume of that room is played. The sound of the previous room will gradually decay as the players progress further. With this mixing rule it is possible to hear a mixture of different sounds depending on the player's location. This way some sounds are heard in the background (or in a distance) whereas others are heard in the foreground depending on the player's position with respect to the level and the layout of the different rooms.

Results

To showcase the potential of *Sonancia*, we provide a few examples available online, which consist of different runs from start to finish of a player running through a sonified level[1]. Initial results seem rather positive. During internal testing level soundscapes were consistently unique thanks to the instrument sonification algorithm, with the sound adding a different dimensionality to the experience (even in a 2D environment). The distance mixing rule in conjunction with both sonification algorithms was also surprisingly effective, as the way sound was mixed and played felt like a narrative progression through the level.

The biggest limitation of the current system is specifically in the usage of sound file recordings. The audio fidelity of *Sonancia* depends on the quality of these recordings. Another side-effect is the loss of a degree of control and manipulation of the sound played. In sound design this is known as granularity; the more granular a sound is the easier it is to manipulate. Because sound files are music renders of around 4 seconds long, the system has less flexibility in influencing the sound signal in such a way that it sounds pleasing to the end-user. Programmatic music, through *Pure Data* or MIDI can potentially be a way of increasing granularity, but the audio fidelity will decrease significantly as the quality necessary to render sound in real-time will cause significant latency. Also proprietary tools which allow us to create a higher quality and granular sound are unfortunately unusable if the system is to be shared with the community, as it requires to run in real-time along with the sonification tool. The solution we found around this problem was to record each instrument at a different note and octave.

Future Work

Future work will consist mainly of improving and polishing the *Sonancia* system on several fronts. The intensity value parameter will be re-worked in order to provide a more accurate value mapping to the valence/arousal circumflex model of affect (Russell 1980). To realize this, an artificial neural network model will be created and trained through a crowd-sourced methodology using human listeners to accurately classify different sounds according to their "intensity". This will be integrated into *Sonancia*, so that users may add their own custom recordings in order for the system to classify and use it for sonification, similarly to the DARCI system (Norton, Heath, and Ventura 2010).

Improvements will be made on the mixing and sound distribution algorithms. Additional mixing rules are currently being developed in order to improve player experience, such as a time rule that will force the system to mix different sounds if the player spends more than a threshold amount of time in a specific room, to either push players forward and progress through the level or add some variability to keep the experience fresh.

Once the system is fully ported into Unity, it will be necessary to create new distribution functions for the placement of diegetic sounds in 3D environments. This will require more spatial information from the 3D level and the audio file (i.e. Ambient or Sound FX). To successfully port sonification, some additional work will also be required. The native Unity sound system contains its own set of specifications that differ from the current native solution. The advantage, however, is that it contains a multitude of sound solutions that could potentially aid with sound and signal processing. Lighting will also play a role in later iterations of the system

[1] http://goo.gl/mEbQU7

and will also influence sonification based on the mood that a specific room is trying to convey.

The system controls volume specifically through pure data. The reason for this is extensibility in later iterations of the system more options and more specific sound effects will be implemented such as reverb for different room sizes and high pass/low pass filters for simulating sounds coming from adjacent rooms. Pure data is an open source software, which allows for both signal modification and programmatic musical composition. The distance mixing rule directly controls the parameters that are within the pure data patch (i.e. a pure data program file) to modify the original signal so it can adapt the audio volume to the current game state. More audio modification parameters will be added into the pure data patch. Unfortunately a limitation of the pure data system is that no more than one single patch can be loaded at a time, however integrating different parameters that affect the audio signal in a single patch appears to be a possibility we aim to investigate. This will also entail exploring more granular audio solutions, in order to create high fidelity sound with more options for the sonification algorithms.

Finally the evaluation of the quality of the level generation and subsequently the level sonification system will be measured through physiological manifestations that map to affective states such as anxiety and stress. Different types of sonification parameters will also be tested in order to study how sound should be distributed in order for the experience to be more enjoyable to players. Additionally, the ordering of audio could become more sophisticated than a gradual linear increase of intensity; for instance, the ordering could be based on a dramatic curve model (i.e. steady increase until apex, decrease) or its reverse.

Conclusions

This paper proposed *Sonancia*, an early prototype for the sonification of procedurally generated levels, specifically for the horror digital game genre. The paper described the genetic algorithm used for procedural level generation, and the initial sonification methods used for the creation of level soundscapes. Although the system is still in its early stages of development, preliminary results of sonified game levels appear (and sound) to be aesthetically pleasing and promising. Further improvements will be made to both the level generation algorithm (i.e. lighting and texturing) and level sonification algorithms, such as improving the audio playback, sonification and real-time mixing algorithms.

Acknowledgments

The research is supported, in part, by the FP7 ICT project C2Learn (project no: 318480) and the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

Cheong, Y.-G., and Young, R. M. 2008. Narrative generation for suspense: Modeling and evaluation. In *Interactive Storytelling*. Springer. 144–155.

Collins, K. 2013. *Playing with sound: a theory of interacting with sound and music in video games*. MIT Press.

Cook, M., and Colton, S. 2011. Multi-faceted evolution of simple arcade games. In *CIG*, 289–296.

Garner, T., and Grimshaw, M. 2014. Sonic virtuality: Understanding audio in a virtual world. *The Oxford Handbook of Virtuality* 364.

Gasselseder, H.-P. 2014. Re-scoring the games score: Dynamic music and immersion in the ludonarrative. *IHCI 2014* 1–8.

Hoover, A. K.; Cachia, W.; Liapis, A.; and Yannakakis, G. N. 2015. Audioinspace: Exploring the creative fusion of generative audio, visuals and gameplay. In *EvoMusArt*. Springer. 101–112.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *FDG*, 213–220.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2014. Computational game creativity. In *ICCC*, volume 4, 71–78. Springer.

Lopes, P., and Yannakakis, G. N. 2014. Investigating collaborative creativity via machine-mediated game blending. In *AIIDE*.

Mitchell, M. 1998. *An introduction to genetic algorithms*. MIT press.

Norton, D.; Heath, D.; and Ventura, D. 2010. Establishing appreciation in a creative system. In *ICCC*, 26–35.

Russell, J. A. 1980. A circumplex model of affect. *Journal of personality and social psychology* 39(6):1161.

Scirea, M.; Cheong, Y.-G.; Bae, B. C.; and Nelson, M. 2014. Evaluating musical foreshadowing of videogame narrative experiences.

Scirea, M.; Nelson, M. J.; and Togelius, J. 2015. Moody music generator: Characterising control parameters using crowdsourcing. In *EvoMusArt*. Springer. 200–211.

Shaker, N.; Yannakakis, G. N.; and Togelius, J. 2013. Crowdsourcing the aesthetics of platform games. *CIG* 5(3):276–290.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *FDG*, 209–216. ACM.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *CIG* 3(3):172–186.

Treanor, M.; Blackford, B.; Mateas, M.; and Bogost, I. 2012. Game-o-matic: Generating videogames that represent ideas. In *Proc. of the third workshop on PCG in Games*, 11. ACM.

Yannakakis, G. N., and Togelius, J. 2011. Experience-driven procedural content generation. *Affective Computing, IEEE Transactions on* 2(3):147–161.