UNIVERSITY OF CALIFORNIA
RIVERSIDE

A Progressive-Adaptive Music Generator for Videogames (PAMG): an Approach to Real-Time Algorithmic Composition

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Music

by

Alvaro Eduardo Lopez Duarte

September 2023

Dissertation Committee:
Dr. Tim Labor, Co-Chairperson
Dr. Ian Dicke, Co-Chairperson
Dr. Aaron Seitz
Dr. Shlomo Dubnov

The Dissertation of Alvaro Eduardo Lopez Duarte is approved:

_____

_____

_____

Committee Co-Chairperson

_____

Committee Co-Chairperson

University of California, Riverside

To Elizabeth and Lionel

ABSTRACT OF THE DISSERTATION

A Progressive-Adaptive Music Generator for Videogames (PAMG): an Approach to Real-Time Algorithmic Composition

by

Alvaro Eduardo Lopez Duarte

Doctor of Philosophy, Graduate Program in Music
University of California, Riverside, September 2023
Dr. Tim Labor, Co-Chairperson
Dr. Ian Dicke, Co-Chairperson

Automated methods of musical composition have a broad range of history and techniques. Throughout the last seventy years, digital technology solidified the field adding stochastic computational methods and machine learning models, multiplying both efficiency and possibilities. Lately, procedural music generation shifted from ‘possible’ to ‘commercially viable’ driving a new wave of services and products.

Videogames use procedural methods to modify audiovisual conditions in real time. Game developers currently use pre-produced music/sound design audio sequences, aiming for sound quality, balancing storage space and variety. In this paradigm, real-time manipulation is limited to adaptive mixing procedures (e.g., rules for overlapping, EQ, and effects), and recombination (e.g., random or sequential containers, stretching/pitch shifting, fragmentation). Those are standard options in popular sound design middleware platforms (FMOD and Audiokinetic’s Wwise) designed to allow adaptation, transformation, and multiplication of existent material according to gameplay. Nevertheless, recurring audio clips (especially music) are recognizable. In extended

gaming sessions, the resulting repetition is unpredictably tiring, and reduces the musical storytelling support through generalized reappearance.

Although generative adaptive music has been implemented in select cases, it is not part of current commercial game development pipelines. While algorithmic note-by-note generation can offer interactive flexibility and infinite diversity, it poses significant challenges such as achieving human-like performativity and producing a distinctive narrative progression through measurable parameters or variables.

In this study, I introduce the Progressive Adaptive Music Generator (PAMG) algorithm, which uses parameters derived from gameplay variables to produce a continuous music stream output that transitions seamlessly between moods/styles and progresses among tension/complexity levels. I cover methodological bases found in the literature, identify implementation challenges, sample PAMG produced material, and present a test scenario that includes a trial videogame and a preliminary comparative perceptual experiment. The test shows that players tend to prefer PAMG over conventional music implementation in a number of capacities although results are not globally conclusive. It additionally suggests particular priming effects that occur in connection with incidental generative game music perception.

Lastly, I assess current experiment and PAMG's limitations and future directions, and present a debate about authorship in the upcoming generative-AI context.

Table of Contents

List of Figures

1. Introduction

1.1. A composer's perspective on automated tools for music creation

The generation of music, a (mostly) composer's task, has employed a range of tools, techniques and skills whose framework habitually depends on the intended output. Analytic and stylistic feature classifications, for example, have been references for music content creation even before computer models.

Although a composer's personal methods and philosophic approach may vary, the final piece gives hints of recurrent methodologies, personal priorities, and signature gestures used in its conception. Recurrent patterns in music construction can be associated with preferred stylistic goals, whether those are commissioned, programmatic, or explorative. In this context, a myriad of methodologies and tools have been developed throughout time to arguably —and subjectively— optimize the musical result within an expected aesthetic. Examples can be found on composition exercises framed on stylistic features, composition techniques ranging from improvisation to replication, and also automated methods for material suggestion, analysis, and management. In most cases, these practices involve the use of collections of relative musical parameters that frame standardized style shapes.

Composition processes are conscious logical methodologies that depart from simple elements grouped into more complex structures, in a similar way algorithms work. Basically, abstraction levels in musical features and structures acquire a role in the

compositional modus-operandi stage associating sound events hierarchically into more complex and semantically meaningful musical concepts—in other words, music composition is algorithmic by nature (Rowe, 1993).

These compositional practices normally are subscribed to serve a range of human purposes that include ethical and existential principles (i.e., identity framing, style representation, personal expression, entertainment, among others (Novak & Matt, 2015)), and also have adapted their methodologies to keep up with contextual[1] and/or industrial production requirements—including regulations and definitions of copyright and authorship. As a subset of composer practices, computation methods for music creation, including any level of automation, offer partial solutions for determined and defined technical problems inside that labor, fulfilling sections of the production pipeline. Even with current text-to-music algorithms, there is a human with a purpose triggering the process whose skills in managing the tool using a range of parameters will reflect in how fit the result is to the purpose.

Hence, a common task is to translate compositional methods to logical algorithms. The use of algorithmic rules, a basic old system for example, defines and creates a musical lexicon, even if those rules include stochastic principles. Deep learning models create a

[1] David Cope (2000) and Gerard Nierhaus (2009), among others, overview a number of methodologies used throughout history as algorithmic methods to create music within stylistic frameworks. These include examples as old as the isorhythmic motet, the rota (wheel) or the Musikalisches Würfelspiel (musical dices), to mention a few, which generate new music from combinatorial procedures, a pre-designed material, and rules for interaction.

set of mathematical relationships that act as a compositional framework by extracting statistics from a music corpus to output imitation. Analogously, human composers use auditory and/or performing experiences in their creations, either consciously or unconsciously.

Contemporary composers have questioned themselves about the tools they use either regularly or in experimentation (Mazzola et al., 2011). Within the commercial music composition paradigm, producers and composers tend to evaluate these techniques in terms of efficiency and adequacy, especially to address intended media. For instance, in the interactive art field, which includes real-time performance structures such as videogames, specific features such as responsiveness, and support for multiple possible timelines[2] are required. For this reason, game music composers adjust their techniques to fit not only storytelling style and aesthetic, but also music modularity and functionality within gameplay implementation.

The perspective of a tool that implements compositional logic in real-time may appeal to composers or music designers that want to address a multi-linear output. It potentially allows them to establish transition frameworks or interpolation methods directly into music features like shapes, instrumentation, ranges, gestures, and harmonic tension, among others.

[2] In this work I refer to *multi-linear* storytelling in music as an extension of multi-linear narrative described by Bembeneck, 2013: "The difference between a uni-linear narrative and a multi-linear narrative is that the first only ever has one possible path. In a multi-linear narrative though, a single story can be told in a variety of ways."

Additionally, the idea of automated music software—like any AI these days—puts forward many questions about authorship, disruption of the composer's labor, structures for licensing/regulation, and more. In general, it will add nuance to the 'aesthetic-source' pointer (e.g., artistic origin) in combination with the production means, especially in commercial schemes.

1.2. Motivation

1.2.1. Game enhancement

In game programming, structures such as characters, environments, actions, and events are connected through algorithms that react to gameplay variables. Light rendering is the result of light, textures, and material colors interacting in real-time and analogously diegetic sound playback reproduces the environmental acoustic properties. If the outcome is verisimilar within game diegesis, storytelling, narrative, and mechanics immersion is fostered and the gameplay experience may be extended.

Gameplay, from the player's perspective, is the result of all the elements working concurrently to produce an experience. Engagement and immersion usually include a balance and variation of surprise, satisfaction and challenge (Abbasi et al., 2017). As these qualities shift and define narrative sections at gameplay, multiple levels of emotional support and their transitions are required in music underscoring. A growing body of literature and technologic solutions explore the possibilities and methodologies of interactive music that features wider interpolation range and transition possibilities

reinforcing a continuous and immersive experience for the player (see section 2). Among these approaches, real-time generation of music is thought to achieve higher responsiveness than using longer segments, comparable to how buffer size in digital audio processing correlates to latency. Smaller music modules as, for example, a musical note would achieve deeper interpolation possibilities for gameplay than conventional, clip-based[3] game underscoring, adding adaptive properties and variance. A parametric music engine would provide tight responsiveness and transition possibilities to states, actions, situations, or characters. Additionally, because real-time generative properties[4] mostly produce music variation, the changing music stream provides: a refreshing variance in music material, an active information layer from the game story to the player.

Although the advantages of procedural music seem to pair well with multi-linear narratives such as videogames, the reasons for its lack of wide-spread adoption deserve attention. Investigating the actors involved, the tools and pipelines of game developing should shed light on this issue. Game designers, audio programmers, sound designers, and game music composers already have an idea of the role AI, adaptive, and generative music plays in game development, the influence it can have on their jobs, and any pros and cons of its implementation[5]. Additionally, exploring conventional methods of music implementation and software solutions involved will potentially expose ways to introduce

[3] A digital file, either MIDI or audio commonly used in different Digital Audio Stations (DAWs). In the current work, the term 'clip' will be used to reference audio clips as pre-recorded audio segments loaded in a platform or a middleware engine for game audio implementation.
[4] See section 2.2.
[5] This assertion stems from an industry exploration detailed in section 2.5.

PAMG. Is it possible that game music is good enough as it is now? Are the actors involved in game design hopeful, skeptic, or scared of automatic music?

From the perspective of gamers, how important is actually a music that adapts granularly to their game? Is this distinction only noticeable by heavy gamers or experienced musicians? Although there are reports of music repetition as a dis-engaging factor in gaming, what kind of experiment can be designed to provide information about satiation due to repetition or relief thanks to re-generation? Is music with a low rate of repetition actually promoting engagement?

AI as a global phenomenon is being rapidly introduced into many aspects of our lives as these lines are written. Many music composers, or music content producers are now forging an idea of the impact generative models will have on their labor. The music commission framework, or 'composing for hire', as a common way to delegate and transfer copyright an authorship may be directly affected in the short term. In addition, interpretation and formulation of legal definitions to implement copyright protection may also need a revision. Are there ethical responsibilities to be assumed when creating an automatic model of music generation?

My stance on authorship is applied practically in the present model by using an algorithmic method—interactive/behavioral (Wooller, R., Brown, A. R, et al., 2005)—instead of machine-learning, since the input is a direct choice on music features used as

parameters, as opposed to statistical properties extracted from a training set. Despite these issues, technically, any digital tool designed to generate music material would enable humans (and perhaps other beings such as animals) with a broader range of music skills/background to make music.

Likely, I will not answer all the inquiries posed here and others will surface, but these issues and the future directions of this project will be discussed at length.

1.3. Objectives and research question

The main question that guided the current research is:

Addressing interaction in a videogame, how does the present Progressive-Adaptive Music Generator (PAMG) model design compare to conventional, pre-recorded, clip-based implementation (CBI)?

The comparison comprises two aspects:

I. Efficiency in music implementation to address the challenges of style interpolation, responsiveness, and variance posed by the nonlinear structure of a videogame.
II. The gamers' subjective perception at gameplay in relation to their backgrounds.

The central objective of this project is the development of PAMG. It features parametric streaming properties in real time. The model produces a malleable and continuous music flow, employing melody, harmony and percussion agents. As opposed to machine learning paradigmatic need for extensive training datasets, it uses a multidimensional music-feature setup that reacts progressively and on-the-fly to input parameters.

The testing objectives are:

- To develop a suitable trial game in which the real-time capabilities of generation and interpolation can be tested.
- To develop, assign, and implement a multi-parameter control and interpolation system that sends information about selected videogame variables in real time to PAMG.
- To evaluate how this particular model compares to CBI in the same game, in terms of implementation efficiency and game experience.

1.4. Scope

The project is an exercise in algorithmic composition addressing the task of multi-linear, parametric, real-time music generation. It comprises design and demonstration of the PAMG algorithm which translates compositional ideas[6] into logic. Also, it involves parameter I/O design for implementation into interactive setups to create reactive music

[6] Importantly, the model generates music using theory, tonal management, and tuning systems included in the Western music system, since most videogames and a significant amount of generative music research are developed within that paradigm.

progressively. It links and develops on methods studied in the algorithmic composition literature, and discloses the reasons to use, modify, or put them aside. It includes testing its real-time multi-parameter control through a gameplay experiment where the system is evaluated against a conventional game music implementation.

Lastly, it includes data analysis, limitations, and future work. This includes a view outside game development where I address the debates about authorship under the perspective of generalized use and commercialization of automated music services and their regulation in the near future.

Concretely, this project comprises:

- The PAMG algorithm design and explanation. It is built using the Max/MSP environment, and includes a developing interface design. Its musical output is conventionalized using Western tonal music behavior, and a limited set of four full range orchestral MIDI instruments: strings, piano, woodwinds, brass, and cinematic percussion. This selection intends to fit common commercial game music paradigms.

- A review and justification of the methods employed in PAMG design, their roots on the literature, and an overview of similar examples in the field of generative music.

- A discussion of common and current game music implementation methods, connected software solutions, and their place in game production to identify their limitations and a possible place for PAMG.

- A test setup, consistent of:
    i. a First Person Shooter (FPS) game designed to use variables that include 3D location and game events/conditions to control the musical parameters in PAMG,
    ii. a clip-based music implementation (CBI), and
    iii. a questionnaire in which participants play the game and compare the two experiences (PAMG and CBI) and answer questions about their music/gaming background.

- The concluding section includes data analysis, a discussion of the test and PAMG development possibilities, and their current limitations. Additionally, it includes a debate of the perspectives of automated music into authorship.

- Lastly, it includes accompanying PAMG-produced material that comprises:
    i. video recordings of gameplay sections, CBI-PAMG comparative transitions, and a walkthrough of PAMG parameters modified in real-time by game events,

ii. PAMG music pieces and also music recordings resulting of gameplay interactions during the test,

iii. a compiled Windows executable game using PAMG music implemented as music clips (CBI), and

iv. a compiled Widows executable standalone PAMG application.

2. The field of algorithmic generative music

This is an overview of the global field of algorithmic composition that mentions studies and approaches that inspired PAMG design, going from the general idea of music generation to the particular and specialized subsection of generative adaptive music for videogames. It is not intended to be a historical model survey, for which Nierhaus (2009) or Zhao et al. (2022) for example, offer a thorough overview.

2.1. Algorithmic composition in connection with PAMG

The use of algorithms, a sequence of instructions that lead to solve a problem in the same way each time it is followed, has been employed before the age of computing to make music (McLean & Dean, 2018). Cope (1996) includes the *combination* of musical elements into a piece as a crucial component of his music algorithm definition and model in *Experiments in Musical Intelligence EMI*. Hiller and Isaacson (1959) are usually credited as the first to utilize a computational model with random-number generators and Markov chains (Norris, 1998) to create their famous *Illiac Suite* (1957). Although these two models use different approaches, they employ numerical symbols (besides other codes) to represent notes in a similar way PAMG does. However, they also employ music as input for transformation in EMI and analytic feature acquisition through 'probability theory' i.e., Markov models by Hiller and Isaacson (Ames, 1987). PAMG, on the other hand, uses relatively measurable music parameters as constraints for music material generation. This fits the category of *knowledge based systems* (Papadopoulos & Wiggins, 1999), in which explicit structures or rules produce a consistent aesthetic result. The

structures used by PAMG are tied to a *grammar*, as another category distinguished by Papadopoulos & Wiggins (1999), which has been extensively employed to construct harmonic progressions (e.g., Johnson-Laird, 1981, Steedman, 1984, Cope, 1991, 1996).

2.1.1. Multi-agent systems (MAS)

MAS are distributed and autonomous systems with perception and action abilities (Tatar & Pasquier, 2019). MAS can be useful in modeling and designing musical creativity because it involves independent and coordinated entities, similar to a music ensemble. PAMG is a multi-agent systems that employs agents in charge of musical feature creation in an ensemble-like network. These features are melody, harmony, rhythm, and orchestration. Additionally, there is a structure that distributes I/O in the agent system and helps communicating the different algorithms (see Appendix A: Algorithm Description, 6. MPIC).

2.1.2. Melody creation and accents

The start of the process is a clock that activates pseudo-random number generators (PRNG) (Luby, 1996), which are used for melodic material creation and recalling through *seeding*. This methodology has been used extensively through stochastic processes and Markov chains by composers (Ames, 1989) due to seemingly low complexity. Although it is possible to alter the weights directly, Markov chains were not employed in PAMG since it implies a statistical input, either from a corpus or synthetic. This can still be adapted to improve stylistic replication (see chapter 6). In the meantime,

double seeded *random walk* (Pearson, 1905) algorithms are employed for melody generation and other sequences that require range and step parameters (see Appendix A, 2.3.1.).

Melodic accents (Thomassen, 1982) are onsets perceived with more importance than others in the sequence, due to higher sound level, operations in the time domain such as delays, and in some cases a relationship with higher pitch. Listeners also expect a 'periodicity' which relates to meter. In PAMG an offset in the *velocity* parameter gives the accent (see section 2.3.6.). Accents in a loop are assigned to beats (in-pulse onsets), and delayed or anticipated beats which builds the rhythmic structure of the main melody to be shared throughout the system. The pattern also is assigned to pitch-contour nodes in the *dcontour* mode (see section 3.2.3.).

2.1.3. Tension, complexity and progression

An important feature taken in account in PAMG design is the need for progressive transition from simple to complex music structures. This is driven by quantization of *tension* and *complexity* mostly in connection with the Western tonal harmony system (Christensen, 2002), which fits incidental music for games' commercial languages. The following are some roots in the literature that influenced its design.

2.1.3.1. Tension

Attempts to measure tension in music have evinced many challenges (Lerdahl, 1996, Farbood, 2012). In fact, a definition that encompasses musical tension in general has been puzzling (Ruiz Marcos, 2022). It establishes a link to music psychology (Granot and Eitan, 2011) and maps expectations of conclusion or continuation that sectionalize form (Huron, 2006, Bigand and Parncutt, 1999). These approaches tend to gravitate towards tonal harmony, in which tension has been related to dual concepts such as *stability/instability* and *consonance/dissonance*, and *tension/relaxation* (Lerdahl and Krumhansl, 2007). This also connects to tonal hierarchy in chord sequences that travel within tonal space (Krumhansl, 2001). Tonal progressions have been characterized by rising stress towards a climatic section, which is associated with building tension, and then a resolution which is related to a decrease of tension (Navarro-Cáceres et al., 2020). As an extension of GTTM (Lerdahl and Jackendoff, 1983) a function of the "tonal pitch space" (Lerdahl, 1988) is to calculate "the psychological distance of any pitch, chord, or region of a given reference point" (Lerdahl, 1996, p. 322). The system provides levels of tension increasing from stability at *Octave level*, *Fifth level*, *Triadic level*, *Diatonic Level*, and *Chromatic level*, each of them with a pitch class set that includes the previous and adds the required pitch classes. In PAMG, the levels are distributed to add one pitch per level to complete 12 levels per chord. The sequence of pitches added with increasing dissonance also has been associated with historical music periods (Dahlhaus, 2014, Jensen & Hebert, 2016), in which use of dissonance has different levels of

sophistication[7]. It is also constructed using perceptual affinities based on spectral distance (Helmholtz, 1877, Parncutt, 1989), although it varies depending on the chord function within the progression (Kostka, 2013).

2.1.3.2. Progression complexity

Complexity estimation is a recurrent topic in Music Information retrieval (MIR). Some studies have found links between complexity and cultural music evolution, preference, and arousal (Vitz, 1964, Heyduk, 1975, Berlyne, 1971). For tonal harmony, harmonic rhythm, harmonic dissonance, and harmonic evolution were considered as the main components of complexity values (Temperley, 2004). PAMG employs a network that handles transitions within chord pools organized by levels in a similar way David Cope (1996) uses an Augmented Transition Network (ATN) to concatenate musical material. ATN in Cope's method uses segments from its 'learning' stage to establish the material and the possibilities of transition or concatenation, while PAMG uses chords transition probabilities established mostly by the Western tonal system (Dahlhaus, 2014) with no pre-composed pattern matching. The harmonies are used both as chords and as ordered pitch collections that provide a distributed tonal grid for melody.

[7] Jensen and Hebert (2016) provide a statistical method based on Chroma analysis developing their own: the Jensen Chroma Complexity (JCC), and analyze music from a 70-year period of popular songs. However, in a deeper spectrum, Dahlhaus (2014) compares and articulates the systems of Helmholtz, Rameau, Schenker, Riemann, Hauptmann, von Oettingen, Sechter, and others. From these analyses, gradual incorporation of less-consonant pitch classes in stylistic composition idioms from historical music periods are evident until it reaches atonalism in a 12-tone system in the twentieth century.

2.1.3.3. Temporal complexity

Perception of temporal patterns has been associated with the notion of an internal clock (Povel & Essens, 1985). This clock tends to be a *hierarchical clock* in which the pulse interval rarely is the shortest value and the time unit continuously adapts to the perceived sequence. *Accented* events in music affect the calibration of that value, which ultimately conforms into a steady clock (Thomassen, 1982). Shmulevich and Povel (1998) propose a complexity score for the best internal clock (a decision of what grouping works better) that accounts for the number of ticks that coincide with unaccented pulses and with silence. This will give higher scores to syncopated patterns in comparison to non-syncopated. Similarly, Lerdahl and Jackendoff (1983) offer a hierarchical framework of complexity levels based on perception. It involves cognitive processes that parses music material through segmentation based on regularities and relationships between resulting parts. Rhythmically, patterns containing a higher number of sub-symmetries are considered less complex or simpler than patterns with few *sub-symmetries* (Toussaint & Trochidis, 2018). Other measures of complexity include the pairwise variability index (Toussaint, 2013) and the Keith's measure of complexity (Keith, 1991) that connect sub-symmetries with metric grouping of 2 and 3 as the minimum rhythmic unit. Euclidian patterns (Toussaint, 2010) were considered, but a structure for progression works better with hierarchical models. Using these considerations, a pattern with one sub-symmetry per *beat* (in this case, one onset per beat, or a pattern that follows the pulse) is regarded as the simplest version of a rhythmic pattern in PAMG rhythm generation algorithm. The rupture of those steady structures comes easily by *displacing* or *moving* onsets, one at a

time, to increase complexity progressively. Evidently, the direction of such relocation and the chosen onset play a role, so alternating direction and dividing pulses on *even* and *odd* to apply anticipation/delay prevent the pattern progression from becoming simpler again. As structures of 2 are less complex than structures of 3 (Hasty, 1997), displacing beats by half a pulse (8^{th} note) would be simpler than a quarter pulse (16^{th} note), which are the available values in the current model. The displaced pattern is then the base accent structure when adding onsets in the progression, in a similar way terminal branches in Lerdahl and Jackendoff (1983) are tree structures.

2.1.4. Algorithmic orchestration

Computer aided orchestration mainly addresses symbolic methods and sample-based methods (Carpentier et al., 2012). In sample-based methods, the most common task is timbral information gathering from instrumental audio samples to structure instrument combinations that address perceptual requirements (Esling et al., 2010, Psenicka, D. 2003). Although playability (Collins, 2000)—measures for human performance—is important in algorithmic composition, it is only addressed in terms of constraint solving (Laurson and Luuskankare, 2001) as current task is machine performance. Nevertheless, rules that address instrumentation play an important role in the idiomatic result. Assigning constraints for instruments' range, recombining instruments in succession, and harmonizing and assigning instruments to the melody agent's output are the primary tasks in the PAMG orchestration agent. Some basic orchestral functions are actually prepared by the source agent such as keeping larger gaps between voices in lower registers and

managing texture thickness through voices placing (Gilreath & Aikin, 2004). Current design has a limited set of specific instrument rules since it manages full-range instrument families, and also it is expected to modularly manage other virtual sample/synth instruments with minimum setup to address further style development (see future directions in chapter 5.)

2.2. Adaptive generative music for videogames

Music has an important role on immersion in media in general, but particularly on videogames (Jiulin Zhang & Xiaoqing Fu, 2015). Immersion is characterized by a focused attention, thoughts and goal-targeted actions (Jennett et al., 2008, Wood et al., 2007, Ermi & Mäyrä, 2007). Immersion enhancement through music is related to the way it supports emotions, plot/narrative changes, environmental mood, activity pace, aesthetic consistency, and thematic unity (Vorderer & Bryant, 2006). If the videogame presents a multi-linear narrative, music adaptability to changes boosts immersion and engagement at gameplay (Sweet, 2015). In fact, players perform differently in response to stimuli from dissimilar music (Wharton & Collins, 2011). Likewise, sound effects contribute to immersion and engagement (Grimshaw & Schott, 2008).

Adaptive, interactive, or dynamic music in videogames refers to alteration possibilities in the music flow based on a control input, making it an efficient fit for interactive, multi-linear narratives (Plut et al. 2022, Hutchings & McCormack 2020, Lopez Duarte 2020, Elmsley et al. 2017, Scirea et al. 2016, Brown 2012). Properties of "procedural composition" defined by Collins (2009) such as controlled real-time, rule-based evolution

also imply a level of adaptability. Procedural content generation in games addresses game content creation with "limited or indirect user input" (Togelius et al., 2011a, p.6). Additionally, conventional implementation methods based on pre-recorded segments, re-orchestration, randomization, and re-mixing (see 2.4.) also achieve adaptive properties.

The generative property, used among others by Brian Eno working with SSEYO[8] to describe music that always changes, has been incorporated into the analysis and perspectives of algorithmic music (Wooller, R., Brown, A. R, et al., 2005). Generative aspects are usually divided in: *Linguistic/Structural*, where analytic theories such as generative grammars support generation rules (Lerdahl and Jackendoff, 1983, Cope, 1991, Chomsky, 1956); *Interactive/Behavioral*, resulting of a process with no music input (does not transform pre-made music) (Rowe 1993; Lippe 1997, p 34; Winkler 1998); *Creative/Procedural*, resulting from setup processes configured by the composer[9]; and *Biological/Emergent*, music that relays on processes that guarantee non-deterministic results (Biles 2002a). PAMG features linguistic/structural algorithms. Its music is the result of interactive/behavioral processes with no music material input, and creates music material through creative/procedural algorithmic processes set by the user. PAMG utilizes non-seeded pseudo-random number generators to produce variance in features like velocity and slight variance in phrase resolutions which may point to a

[8] https://intermorphic.com/. Accessed 5/1/2023.
[9] http://www.inmotionmagazine.com/eno1.html. Accessed 8/15/2023.

biological/emergent property. However, seeding is used to recall sequential patterns and, in general, pseudo-random generators are deterministic by nature.

In an attempt to formalize a generative music definition in videogame paradigms, Plut et al. (2022), characterize it as “having systemic autonomy from the game logic”, to avoid covering single-piece tied to single-event cueing within the concept. For this writing, I address generative music as a process in which events parsed in a MIDI-like format are generated in real-time. A typology of generative music in videogames described by Plut and Pasquier (2022) is used also to describe PAMG features in section 3.

There is a growing body of generative adaptive systems for videogames (e.g., Plut et al., 2022, Hutchings and McCormack 2020, Plans & Morelli, 2012, Elmsley et al. 2017, Scirea et al., 2016, among others). Generative music in videogames has a number of entries that fit within the current definition. A more inclusive list of significant examples (that include horizontal and vertical arrangements of pre-composed material) has been elaborated by Plut and Pasquier (2020). Based on PAMG typology (see section 3.), some examples of games using generative music in a similar way are:

Ghost-Writer (Robertson et al., 1998) is a music generating system that uses affect and highlights tension levels in its generation algorithms. It follows a sequence of three tasks: first, it creates an over-all structural form, second, it generates rhythmic patterns using poetic syllabic accents, third, it chooses a triad type among the four tonal possibilities

(augmented, diminished, major, and minor). Finally, it generates random melodies using the constraints laid down by the previous processes. Instruments are then assigned based on the tension level. Similar to PAMG, it uses mostly the author knowledge, constrained stochastic methods, and tension as drivers for generation.

A notable example of evolutionary music (Vico et al., 2021, Miranda & Biles, 2007) would be Spore (2008). It is the only system known to implement a modified version of PureData[10] to automate music generation. The system uses a multi-agent approach to generate independent musical phrases employing seeded random processes and parameters driven by game state changes. As its reaction is not tied to linear processes, it may be classified as linear as opposed to adaptive. However, it can apply real-time DSP effects. In Spore, the system uses more generative procedures during segments with lower quantity of game processes to avoid higher consumption of resources[11]. As in PAMG, there is a multi-agent structure, a series of parameters for music generation, and seeded preudo-random generators for new and recurrent material.

Engels et al.'s (2015) music generation system uses a Markov chain to encode musical events that occur at the same time into a state, increasing the number of simultaneous voices the system can play. The system also segments musical sections using a support vector machine (SVM) based on pitch, duration, timbre, and volume. A Hidden Markov

[10] https://puredata.info/. Accessed 8/11/2023.
[11] https://www.gdcvault.com/play/323/Procedural-Music-in. Accessed 4/3/2023.

Model is used with chords as hidden states to avoid clashes between individual voices, and these chords can be provided externally or tagged automatically by the system. Although PAMG does not use Markov chain models to generate chord grammars, it uses chords as guidelines for melodies and other aspects of phrasing.

Anthony Prechtl (2016) created a generative music system that closely adapts to the level of tension during gameplay, and tests it in his abstract horror game, Escape Point. The system uses a Markov chain to compose a chord progression and adjusts the probabilities of each chord based on the game state and tension level. There are 12 parameter sets that control the generation of music, with four parameters altering the music performance. PAMG also controls music generation with parameters but they are independent. Prechtl found that experienced players preferred the generative music over linear composed music and that all players found the generative music more intense and exciting. Skin conductance responses supported these findings, providing subjective and objective support for the strengths of generative and adaptive music.

Although the Adaptive Music System (AMS) (Hutchings & McCormack, 2020) alters pre-composed music based on affective mapping taken from the game state, it offers generative properties. AMS uses a combination of rule-based algorithms, genetic algorithms, and a Recurrent Neural Network (RNN). It extends a model of affect with 6 categories: happiness, fear, anger, tenderness, sadness, and excitement. AMS uses a "spreading activation" model to connect game state and affective data. It uses a multi-

agent approach with three agent roles: harmony, melody, and a percussive line—a similar approach to PAMG. The harmony agent builds a chord progression, while the melody agents alter pre-composed music using a rule-based approach. AMS has been evaluated using real-world games and has been integrated into an open-source Zelda clone and the Real-time strategy game StarCraft II. A correlational analysis found that AMS slightly increases player immersion.

The Audience of the Singular (AOTS) (Plut, 2017) is a generative system that uses primarily modified Markov chains. It learns symbolic music representation from a corpus of 30 pieces of videogame music from the 1980s and 1990s. AOTS uses a variable-order Markov model for its core horizontal generation and creates four versions of five musical lines, starting with the bass line. The system then composes a primary and secondary melodies in addition to two harmony lines based on probabilities from previous notes, learned melodic contours, and the bass line. AOTS also has a game built around it but the system can function as a stand-alone music instrument, a similar approach to PAMG.

The Agate (AGMS)[12] system addresses composition and arrangement with both horizontal note granularity and vertical instrument assignation, like PAMG. The system uses a rule-based algorithm with both symbolic and audio musical representation. It organizes its music in libraries and rule sets associated with moods provided and attached to a game state externally. Agate generates music with a set of rule-based constraints on

[12] https://www.gdcvault.com/play/1012710/An-Adaptive-Generative-Music-System. Accessed 4/3/2023.

random generation—like PAMG—while the arrangement task uses constrained random generation of short musical phrases provided by the user. Agate creates ambient music by playing short musical phrases using constrains of activity level at random times over the generated soundscape.

AMEE (Hoeberechts et al., 2009) is a real-time music generation system that can create musical compositions based on desired emotional characteristics. Although it is applicable to videogames, its characteristics are designed in general for real-time interactive systems. It has an adaptive methodology and an application programming interface written in Java for external use. However, it currently lacks smooth transitions between different musical selections, and its audio quality is limited to MIDI files. The system offers an interactive way to modify musical parameters in real-time, but changes in the output only occur after the generated musical block completes its playback. PAMG also presents a UI to change parameters in real time, but does not use emotional mapping directly.

barelyMusician (Gungormusler et al., 2015) uses a system architecture based on Hoeberechts et al.'s pipeline structure for real-time music production, and is designed to reflect a typical real-life musical performance. The main component of the architecture is the *Musician* algorithm, responsible for managing and communicating with other components. The *Sequencer* serves as the clock of the engine, tracking highly grained audio pulses and sequencing beats, bars, and sections. This function is similar to the

Clock in PAMG. The audio events are hierarchically designed using common sectional forms. The *Ensemble* component creates the song structure using its generators (macro, meso, and micro) which it passes to the *Performers* module to generate actual sequences of musical notes. *Performers* produce the next sequence, initially using abstract structures of notes that can be modified adaptively before being played. The *Conductor* agent transforms the generated note sequence according to user-determined musical parameters interactively, and the meta information is converted into actual notes and written into the musical score. Finally, each instrument generates the audible output by playing all the notes in its performer's score. PAMG controls act directly on agents' parameters which act on top of generative algorithms, as opposed to the method of transforming music after generation present in barelyMusician.

2.3. Generating change from external parameters

A range of transition methodologies have been developed to address the need to switch between stylistic properties based on continuous (or sudden) changes in the videogame. Cutajar (2020) details usage and examples of the most referenced techniques: abrupt cut transition, crossfading, stingers, horizontal re-sequencing, and vertical re-orchestration. In his evaluation of what makes successful binary transitions (i.e., between two themes), Cutajar's analysis mentions: gradual change rate, smooth fitting, and subtlety. He also develops a method for transitions in generative music based on Markov chains and viewpoints (Conklin & Witten, 1995).

As game development went mostly in the direction of clip-based implementation, these transition techniques are now common tools in conventional adaptive implementation software solutions[13]. They are mainly based on the concept that instantaneous gameplay events can be used to trigger transition rules for orchestral layering and segment re-sequencing[14], besides abrupt cut, crossfading and stinger playback. Despite the fact that this system falls short in style development and variance compared to generative models[15], it has been able to handle adaptive music features. As they are designed around a *source* and a *target* piece, their approach is a 1-dimensional musical transition (i.e., connects only two themes), even if the target is variable. In this paradigm, transitions can use the rules and methods mentioned and also specific transitional music segments—i.e., stingers—as a sort of 'bridge'. Sweet (2015) categorizes transitions into *zone* and *event-triggered.* The following are considerations for zone transitions in conventional implementations:

[13] Like Audiokinetic's Wwise, FMOD, Elias Studio, etc.
[14] Sweet (2015) explores several composition techniques that improve concatenation of musical segments in which loops can be seamless and randomly re-sequenced. He also discusses re-orchestration, a technique in which voicing layers for each musical segment i.e., tracks can be added/subtracted and in some cases exchanged with other segments to achieve variety and change.
[15] Music material reappears in CBI, while in generative models it is constantly generated which gives a higher range of variation. Collins (2007) talks about listener fatigue, a phenomenon discussed also by Sweet (2015) that involves reports of players rejecting the reappearing music segments, especially if they sound the same and are back to back. More on the effect of repetition in 2.4.

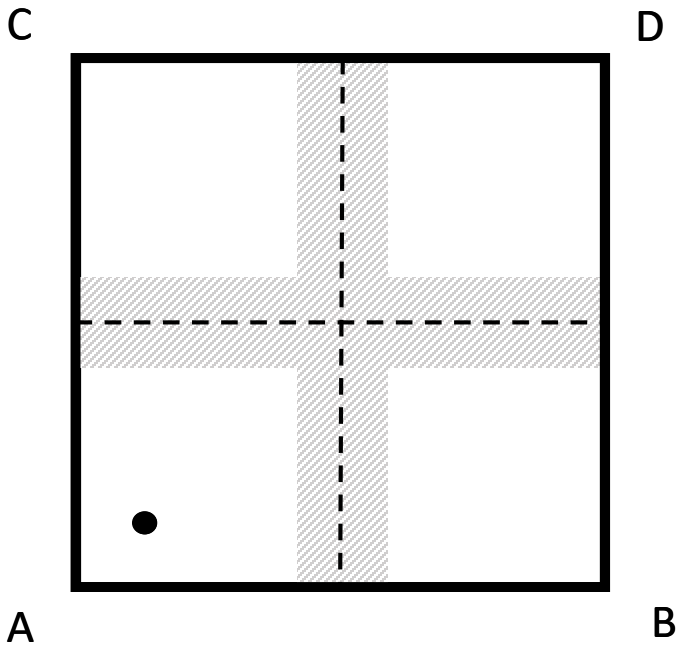


Fig. 1. 2D transitional regions.

The figure exemplifies a square, 2D map game level. By current design, the corners are assigned to a mood that is supported by its respective music theme A, B, C, and D. In conventional implementations, if the player (i.e., the black dot) walks through the transition area (i.e., shaded pattern) the rules in place should start the transition process to another theme. This region is arbitrary, and can be a single boundary (dashed line). In any case, this transition model always connects *two* themes.

Fig. 1, 2-dimensional transitional regions suggest consideration of some particular cases: *what happens if the transition is triggered by the player's location and the player decides to go back to the previous region?* And *what happens if the player is in the center of the graph?*

Cuatajar (2020) solves the first situation by using a system of transition sub-regions that adds weight to the closest theme. The use of a transitional region accounts for a *direction* that triggers the corresponding theme weight, or in the case of CBI, a transition method/segment depending on the target theme. For the first question, as player's

direction can change in the middle of the transition, any process started should be reversible or able to re-transition to a new target *before* arriving to a new musical theme. This transition design may also address the four-point intersection in the middle, but is still a binary decision system that needs a transition design to address variable theme transitioning.

Within the same tools, a transition design based on layers (vertical re-orchestration) that are introduced in the mix using location may address the limitation of the above binary transition, allowing multiple themes to intervene:

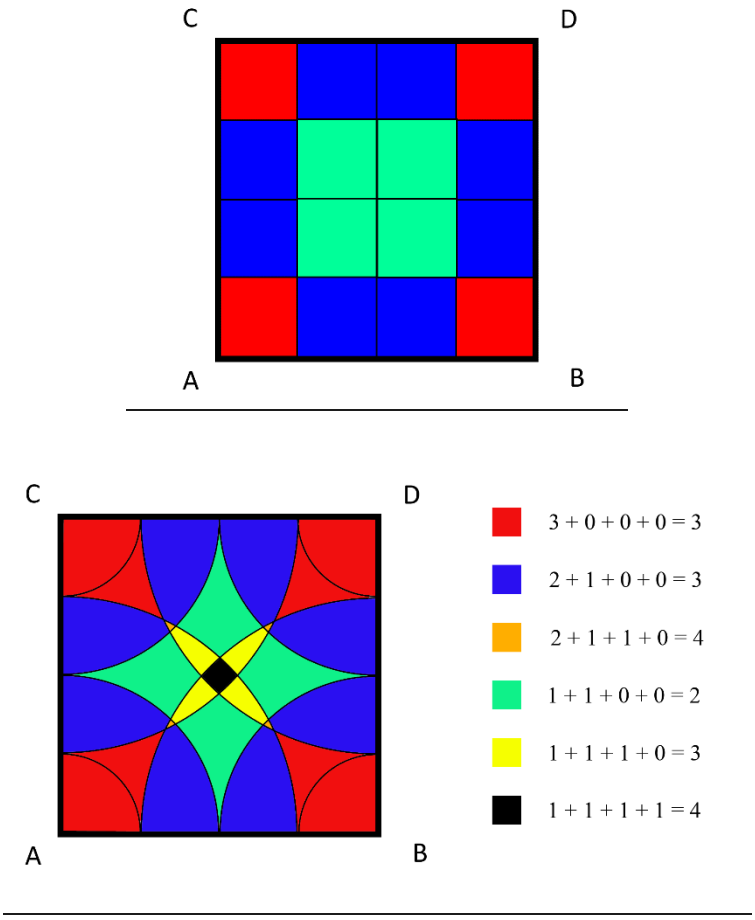


Fig. 2. 2D Transition through layering

Two distinct designs: **Top**: Constructing squared areas growing from the corners and assigning a maximum of 3 tracks per theme, the color coded regions are: *red* is the theme in its purest version, using all its tracks and none from other themes. *Blue* uses 2 tracks from the closest theme, 1 from the closer neighbor (vertical or horizontal), and 0 from the furthest diagonal theme. *Green* uses 2 tracks from the closest theme and 1 for each of the other three themes for a total of five tracks. **Bottom**: As the squared representation may not depict the actual influence of a punctual object placed in the corner, another distance-based representation to the corners will form three concentric regions per theme and the resulting intersections to generate a richer structure. For simplicity, this example continues with four regions with 0 to 3 tracks maximum for each theme. Using the same color code *red* is the 3-track theme, *blue* uses 2 tracks from the closest theme, 1 from the closer neighbor (vertical or horizontal), and 0 from the furthest themes. *Green* uses 1 for each neighbor and 0 from the furthest. New regions appear in *orange* with the same 2 tracks from blue, 1 for each neighbor, and 0 from the furthest, *Yellow* as the intersection of three 1-track regions, and *black* as the intersection of all the 1-track regions.

Fig. 2 shows a system of regions in which tracks are layered depending on the player's proximity to the corners, using squared (a) and circular (b) structures. Although the areas

are arbitrary divisions of the label area (arguably, more suitable subdivisions including an empty region can be imagined), it is an example of natural occurrences in overlaying. The last one (b), offers more varied combinations and tends to be easier to code since it is based on linear distance to the corners, as a contrast to the squared, established Cartesian areas (a). For this system to work, it is essential that the themes share musical features such as tempo, meter (or its multiple), and similar harmonic/tonal progression aspects whenever superposition occurs. The difference in track sums per region (2-4 tracks in (b) and 3-5 tracks in (a)) stands out. This variation range increases if more tracks are used, requiring mixing adjustments to be included in the transition rules.

A combination of these techniques is able to produce effective transitions[16] but the resulting overlapping needs to be carefully tested to avoid intended loudness increase/decrease. This system inspires CBI in section 4, in which layering is used for one of the location themes and for the action variations.

Besides re-orchestration, horizontal re-sequencing—i.e., scheduling of musical cues (usually following the meter grid) as a response to game events—is used in combination to layering in CBI to trigger track transitions at zone boundaries.

In generative music, the problems of music segment transition can be addressed in a more continuous way. Wooler and A. R. Brown (2005) explore the concept of *morphing* in

[16] Cuatajar (2020) finds *smoothness* and *coherence* as the main features of a successful transition.

composition practice, in music parameters or features, and in historic/stylistic music analysis. From their analysis, morphing itself is a pattern-merging process that generates interpolated material for transitions. They also define interpolation as “the technique of estimating appropriate transitional values between known points.” p. 5. This approach is used in the PAMG interpolation module (see Appendix A, 6.).

Hutchings and McCormack, 2020, Lykartsis et al. 2013, and Zentner et al., 2008 for example, address styles, moods, and musical features in relation to affects, and propose generative interpolation techniques. Although the studies utilize music-affect relationships for transition, the current PAMG method differs in that multi-parameter preset interpolation affects generation constantly. In other words, most of the time the musical result is an interpolation between moods or styles.

Besides interpolation, transition methods that can be used in generative paradigms include i) *cyclostationary*, (Slaney et al.,1996) which introduces small changes within a repetitive sequence, ii) interleaving, which alternatively selects elements from music inputs (Oppenheim, D. V., 1997), and iii) weighted averaging, which involves two Markov chain weighted selections based on the transition position (Wooler and A. R. Brown, 2005).

2.4. Game music testing

Testing music for videogames has been focused on measuring its impact in immersion (Klimmt et al., 2019), support for emotions and affections (Plut et al., 2022, Hutchings and McCormack 2020), and valence/arousal (Reuderink et al., 2013). Additionally, there is a possibility of estimating human reactions through physiological measurement (Prechtl, 2016). In the realm of automatic music generation, listening tests mostly deal with one or more musical excerpts for which participants rate or rank quality. Modified Turing-like tests (Turing, 2009), in which participants estimate if the music is generated by a machine or human (Hadjeres et al., 2017, Thickstun et al., 2018; Donahue et al., 2019), select excerpts from a pair, or rates them in an attempt to gauge synthetic music quality.

Although these approaches have deep foundations in thorough experimentation, to test PAMG performance at gameplay I adopted a comparative perspective, designed to gauge how and if the difference between clip-based music implementation and PAMG is appreciated by players. It takes into account the participant's cultural background (Eerola et al., 2006) as an important influence in a listening experiment.

One of the advantages of a generative model is its ability to generate musical variance[17] exceeding what is achieved by clip-based implementation. This would address listener fatigue (Collins 2007). Schellenberg et al., (2008) explore liking of music as a function of

[17] Variance is used as a way to describe variation around a central thread, similar to *development* in music.

exposure. They also point to the role of expertise and stimuli complexity as a factor for arousal potential (Orr & Ohlsson, 2005). An inverse U curve was found showing an increase in familiarity as a result of the first iterations, and then an onset of "satiation" at around 8 exposures for focused stimuli and 32 for incidental. The ranges are longer if the stimuli are longer and more complex, and the arousal potential decreases as the listener expertise increases. Referencing a number of other observations that include an account of complexity as an influence in the inverted U liking curve, they note "The peak occurs after relatively few exposures for simple stimuli and/or focused attending, but much later (if at all) with more complex stimuli and incidental exposure".

As the type of listening in a videogame is incidental and the current themes in the CBI are around 2-minutes long, the combined exposure to test the onset of satiation (i.e., negative liking) at gameplay would be 64 minutes—if a theme is repeated back to back the whole time. As the effect of repeated exposure is lessened by techniques like layering and also other themes are introduced, a dedicated experiment able to test music satiation should include around 80 minutes of gameplay only on CBI. This constitutes a limitation discussed in section 6.

Questionnaires are built using mostly Likert scales (Likert, 1932), and can be seen in Appendix E.

2.5. Why has generative/AI/algorithmic music not taken over game music?

It is easy to recognize why a non-linear storytelling medium like videogames would benefit from non-linear music scoring (Sweet, 2015). It has been addressed by researchers in the last three decades and many theoretical and practical models/tools exist as a result [18]. Generative adaptive, as a subset of multi-linear music, is considered as the most flexible system to implement non-linearity due to the real-time parameter accessibility linked to in-game variables—which would make it fit the category of procedurally generated game elements. Among the many advantages cited (e.g., reducing literal repetition, granular adaptability, context-specific scoring, etc.), generative adaptive paradigms enable players to recognize themselves as agents of change and variety, and also offers the possibility of transmitting real-time contextual information in advance of gameplay events execution. Arguably, immersion is fostered[19] by stimulating player's curiosity, or refreshed by performing less unintentional repetitions throughout longer and numerous audition sessions.

For these reasons, it is not immediately clear why such a small percentage of current games use generative adaptive music scoring instead of many. To be fair, strictly linear

[18] Examples of systems that address multi-linear clip playback are iMuse 1991, DirectMusic 1996, Elias 2019, Psai 2016 and other game-specific models detailed by Plut and Parquier, 2020. Brown 2012, Scirea et al. 2016, Sweet 2015, Elmsley et al. 2017, Lopez Duarte 2020, Hutchings and McCormack 2020, and systems such as Melodrive 2017, Metacompose 2017 address generative adaptive models.

[19] Hutchings and McCormack 2020, and Scirea et al. 2016 have test results that support the idea that generative music reinforces immersion. However, it is done with their systems and more large scale testing is needed. The present research also has shown some results in this subject (see section 4.2.3.4.).

music has been fading out in the last decades[20] in connection with the increase in memory and CPU capabilities, which allows larger clips and more reactive rules. Still, to find out why generative music is not wide spread, other game developing and production dynamics should be considered, such as the investment risk, for example.

Akin to film music production, comprehensive funding in game development will translate to top composers, known performers/orchestras, and experienced, well-equipped sound engineers that are assumed to yield high quality music clips. Low commercial risk has been represented by prestige authors and professional recordings. Either with music libraries or other outsourced production and licensing methods, this clip-based production pipeline has become common in all budgets, propelled by game development tools that facilitate CBI in both sound design and game music. CBI methods allow games to have a full range of pre-produced sound, including licensed music. To achieve an equivalent generative output, real-time synthesis, mix and effects would use more hardware resources such as CPU processing and data bandwidth than CBI. A known limitation in audio clip assignment is the low music variability and responsiveness. Although procedural sound design (Farnell, 2010) and multi-linear adaptive music require more work, game composer tools such as Elias Studio or Psai Unity plugin and middleware solutions such as FMOD[21] or Wwise[22] currently offer a range of tools for adaptive CBI.

[20] A resourceful blog https://splice.com/blog/adaptive-music-video-games/ summarizes the history, and Sweet 2015 offers an explanation of techniques and tools developed throughout videogame history.
[21] https://www.fmod.com. Accessed 5/5/2023
[22] https://www.audiokinetic.com/en/. Accessed 5/5/2023

Some techniques employed by composers, music implementers, and sound designers include:

- Assigning mix parameters to real-time gameplay variables like in sound design[23]. These include but are not limited to levels, Q/frequency in filters, threshold/ADSR/ratio in dynamic processes, pitch-shift/tempo-change, and acoustic effect properties.

- Random clip playback from a selected pool with the possibility of avoiding immediate repetition, which can be combined with transitional sections to connect segments.

- Sub-mix section layering using overlap-able instrument clips. Diverse combinations of instrumental sets not only result in a range of variety but also offer several levels of intensity and mood variation assignable to game conditions and events.

- Progression designing in which harmony follows a continuous form and supports overlapping. This harmonic segment composition may also include especial transitional progressions. Although time consuming, crafting music segments that connect with each other and with themselves, and that work harmonically when played together, benefits music modularity.

[23] For example, in Audiokinetic's Wwise, RTPC (real-time parameter controller) can assign any float/integer game variable, through a range and curve set in a Cartesian mapping graph, to any mix parameters.

Procedural sound design, as predictable evolution of current clip-based paradigms, has been proposed and implemented in physics-informed phenomena such as real-time geometry-generated reverb (Fırat et al., 2022). Meanwhile in visuals, procedural lighting systems already interact and render results based on materials, source, point of view, and game conditions. Automatic music generation in games, however, has scattered examples, mainly because it requires significant pipeline arrangements and programming work[24]. Despite being innovation advocates, game developers know the implications of altering or creating new production models, and have faced the uncertainties and downsides of improvisation. If a game prospect represents an outstanding and distinctive game through mechanics, narrative, and aesthetics, developers come up with a feasible pipeline. There, tasks are itemized for an attainable production timeline. If the added work of, for example, automatic music programming—which is not in conventional pipelines—and game parameterization is reflected in game uniqueness and sales, the project might be considered viable. For those seldom cases, the result tends to be a music-centric game[25] mostly non-generative, and in an even smaller subset, complex unique systems and workflows devised by their programmers have achieved generative adaptive results[26]. If that extra work of designing a music engine was taken over by a software tool that readily fits into the current pipelines (i.e., by offering game engine

[24] This subject was addressed in interviews with Guy Somberg (https://www.linkedin.com/in/guy-somberg-2a72b79/details/experience/), author of the Game Audio Programming book series, and Aleksandar Zecevic (https://www.linkedin.com/in/aleksandarzecevic/details/experience/) Senior Audio Designer\Audio Director at Electronic Arts (EA) for more than 15 years.
[25] e.g., Rez Infinite, Beat Go, and others, besides the music rhythm genre that includes Guitar Hero and similar games.
[26] For a comprehensive spectrum of adaptive and/or generative music games please see Plut and Pasquier (2020) table 1.

compatibility, scalability, ability to compile and build within the game) then a higher number of non-music centric games may employ automatic music.

Up to the present, there is a limited supply of commercial tools for generative music in general game development, or software solutions to simplify its implementation in conventional game audio production.[27]

Although generative-adaptive music and sound design are feasible and desirable, hardware resources have been occupied and kept for video acceleration and other game mechanics. In part, CBI has had enough quality to provide immersion while using low computing power. Audio engines stream from RAM and storage devices, and the bandwidth would depend on simultaneous voices but, even with a high count, it is several orders of magnitude lower than video processing. Music can employ several layers but rarely beyond twenty simultaneous tracks in playback. Sound design often surpasses that, but in practice, crowded environments are avoided not only to save bandwidth but also to control noise. Some effects such as filters and reverbs may use additional CPU resources but their share is monitored and kept to a minimum.

In comparison, synthesized/procedural sound would definitely consume more resources than conventional clip playback by executing more real-time calculations. Generative

[27] Recently, several startups are rising funds for automatic composition technologies such as DAACI, AIVA, Amper, Jukedeck, and many others. They also foresee implementation solutions for videogames.

music may not use much more CPU than CBI, but if the system uses instrument samples, simultaneous playback grows to sound design levels, and RAM usage also increases. If instead of samples it uses synthesis, additional processing power needs to be allocated (synthesis can be light for modern CPUs but added sources and effects may increase its load significantly). In general, current multi-thread processing and bandwidth growth may accommodate for those modest additions, while methods for memory reduction like compression and SSD streaming would have to be evaluated for each platform.

Machine-learning music generators are also a possibility. Current constraints point to a pre-trained model that is able to interpolate among styles e.g., Magenta's MusicVAE[28] (Roberts et al. 2019). As neural network models have reduced real time modification features, the use of embeddings may cover the need for malleability using game parameters. Implementation in videogames may need tools to train or swap pre-trained models, monitor resources usage, and test for unforeseen results. Once the models are trained and the interpolation among styles or moods is working, real-time rendering possibly continues to be the most computationally expensive task, although still less than video.

In general, material capability to include intelligent generative models is already there and is increasing. Doubts can surface in terms of licensing frameworks for deep learning models—already actors and content producers are licensing their signature styles for

[28] https://magenta.tensorflow.org/music-vae. Accessed 5/7/2023.

replication—and how the result seamlessly connect with the previous model both in production pipelines and in consumer reception.

2.6. PAMG in commercial game development pipelines: How it fits into game development and possible problems

Automatic music systems need to define a role inside game development pipelines. Specifically, their function and value into the connection between the director and/or producer with the composer. That link works on a contractual agreement in which creative material fulfills aesthetic objectives and become a commodity for the buyer, in this case the game producer. Then, the first barrier to overcome is how operationally the work of a composer in an algorithmic system guaranties the director's framework. This involves for example, a redesign of screening and testing sessions. Subsequently, as stated before, an evaluation of commercial risk occurs. This likely places the value of generative music more as a contextual gameplay asset—less as a static recording susceptible of publication like in current soundtracks—and also as a potential creative product: a logical model of musical behavior.

Technically and operationally, a tool that facilitates music generation that integrates with current game developing pipeline models has more chances of being employed than the prospect of assigning programing labor and design time to incorporate such a feature. Successful software solutions currently employed in game design have 'pre-cooked' user interfaces, functions, utilities, and processes that promote interdisciplinary collaboration.

For example, animation artists and modelers normally use a specialized software like Blender, Adobe Substance, Cinema 4D, 3Ds Max, or even the same Unreal Engine/Unity, instead of coding. Integration efficiency depends on compatibility and fewer steps to carry their output into the game. Nowadays, their skills should also include source version control (Git/BitBucket) and online/remote networking implementation using agile software development systems (Jira, Monday, Asana, etc.) Although knowledge about technical implementations and game developing flow structures—besides some programming—is desirable for artists and designers, it is not a requirement in current pipelines. Similar to model design artists, a sound designer uses audio editing/mixing DAWs such as ProTools, Reaper, or Cubase, and in some cases modeling solutions like Max/MSP, Pure Data or any synthesizer to create content. To insert their work into current game developing pipelines they use middleware such as Wwise or FMOD, which not only allow assignation and conditional transformation of audio clips to gameplay events but also compiles the logic into the final game build. Sound designers and music composers require a close contact with the audio programmer which supervises game integration, and whose labor is highly supported by the middleware capabilities (Somberg, 2016). Without middleware, the music/sound design audio clips would have to be implemented directly by the audio programmer. In mid-to-low budget games, engines and middleware are trusted with audio handling and programmers and game designers only expose events and variables to middleware users to work with. Hence, sound designers are in charge of music implementation through middleware in most cases.

2.6.1. Into the pipeline

The audio pipeline has seen many improvements in terms of efficiency, independence and possibilities through middleware. Currently, finding a way to implement PAMG as a plugin for Wwise/FMOD seems like the best solution. Also coded as a game engine plugin (within Unity/UE) is considered as a second option, but since monitoring possibilities for mix and assignments are available in the middleware, the first option is preferred for an eventual built-in software solution[29].

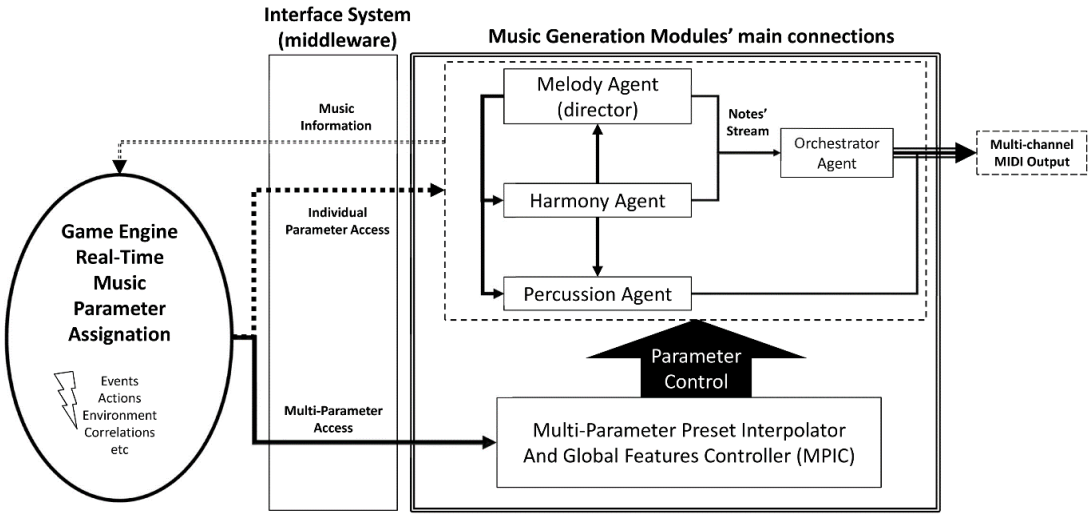


Fig. 3. PAMG implementation diagram.

The game engine sends control values through via middleware to individual generation parameters and to MPIC, and receives music information data that includes MIDI events, beats, tempo, current harmony, and parameter levels.

[29] In conversations with game developers at GDC2023, plugins for the game engines surfaced as viable options for low budget games too, since they have simplified pipelines that do not include audio middleware.

PAMG receives parameter control and generates MIDI-formatted data.[30] Audio output rendering (assigned sampler/synth instruments) may be carried either by the audio engine inside the middleware (Fig. 4) or by platform-native MIDI instruments (Fig. 6).

2.6.2. Two ways to implement PAMG into middleware

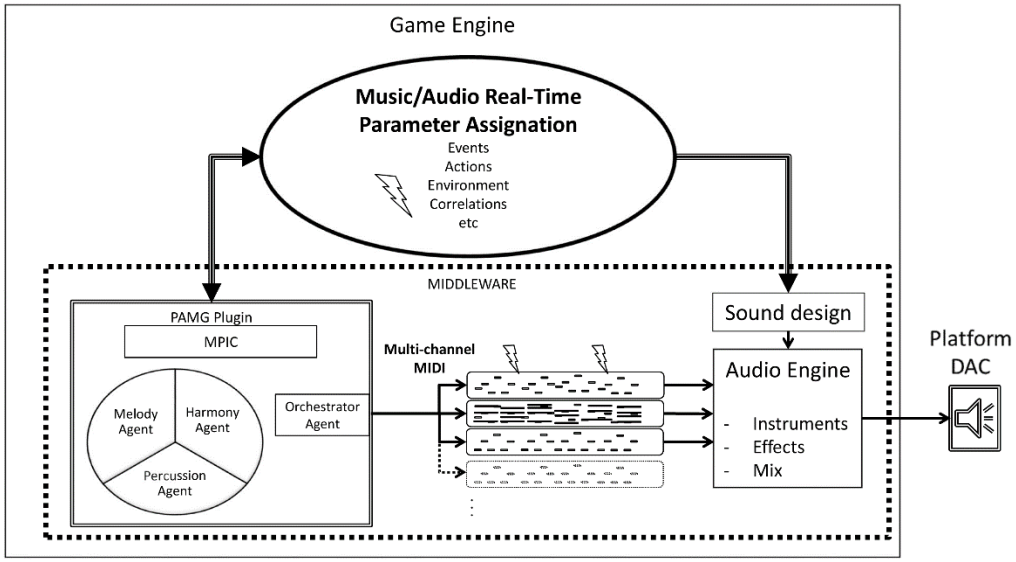


Fig. 4. PAMG using instruments within the middleware.

The first prospect shown in Fig. 4 requires instruments (samplers/synths) either integrated into the compiled PAMG plugin or as a middleware instrument. An all-in-one solution would allow homogenous audio quality, and mix monitored and tested along sound design and voice-over, in addition to resource usage reports through middleware editing

[30] MIDI protocol seems to be a simple and effective solution for music output, although instrument sampling and/or synthesis would be in charge of the final rendering.

features. The final result is compiled into the game, making it a consistent solution across platforms.

In Wwise (middleware), the Synth One synthesizer is able to playback loaded MIDI files and also can be linked to a MIDI input through control surface assignment. Additionally, it is possible to load instrument samples and setup a multi-sample structure for MIDI playback. However, setting up either multiple Synth One[31] synths with their own parameters, and/or multiple instrument samples each one with routings and ranges is time consuming. Currently, only MIDI files loaded as Music Segments[32] or assigned to Music Play-List Containers[33] can be routed to instruments or multi-sample structures, and only one-to-one connections are available. A plugin could be programmed to have MIDI output but until now it has not been done. For those reasons, the inclusion of synth/sampler instrument output from the plugin would be a straight-forward option. In fact, Enzien Audio's "hvcc" or Heavy[34] is a python-based dataflow audio programming language compiler that generates C/C++ code wrappers that allow PureData[35] (Pd) patches to be converted to Wwise plugins. Although objects allowed are limited and support has been suspended since 2018, it is still able to compile a plugin with a custom set of parameters and embedded audio sources. Fig. 5 illustrates a basic music generator

---

[31] https://www.audiokinetic.com/en/library/edge/?source=SDK&id=wwiseobject_source_wwise_synth_one.html. Accessed 5/5/2023.

[32] https://www.audiokinetic.com/en/library/edge/?source=Help&id=what_is_music_segment. Accessed 5/5/2023.

[33] https://www.audiokinetic.com/en/courses/wwise201/?source=wwise201&id=lesson_1_re_sequencing_creating_variation_using_horizontal_approach_using_playlist_containers. Accessed 5/5/2023.

[34] https://github.com/enzienaudio/hvcc. Accessed 5/5/2023.

[35] https://puredata.info/. Accessed 5/5/2023.

in Pd that exposes some parameters in Wwise, and embeds piano samples within the plugin. I explored the possibilities for a MIDI output but the only two plugin categories available—effects and source plugins[36]—are enabled to exclusively receive MIDI. Still, the possibility of building a plugin with MIDI output may exist through Audiokinetic's SDK.

For a complete solution featuring synth/sampler output—a full-fledged version of the Fig. 5 patch—commercial capabilities to be considered besides music generation would include access to different voices and their parameters inside the middleware, which adds programming labor and sample licensing to developing costs.

[36] Effects transform audio, and sources output audio triggered by events or MIDI.

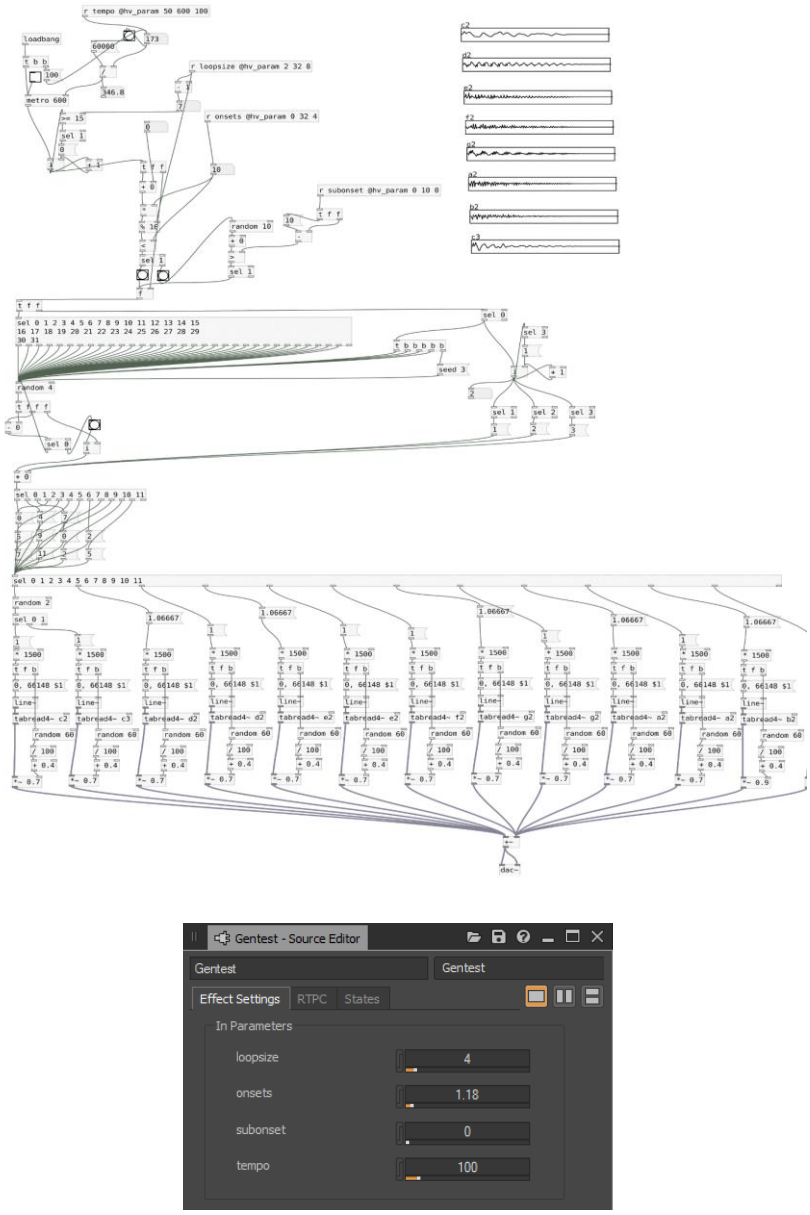


Fig. 5. Pd generator with embedded audio samples in Wwise.

The available Pd objects to use in order to compile the plugin with Heavy is limited. For example [counter], [expr], abstractions, and other functions that could simplify the structure are not supported, requiring a combination of basic objects and then increasing the use of boxes and connections. The patch shown in Fig. 5 (top) is a highly simplified

version of PAMG with one instrument and eight samples of piano. It uses a clock based on [metro], loop size for possible onsets, and a Euclidian rhythm generator with additional onset filling parameter (sub-onset). Then, onsets activate a seeded random generator that distributes them among chords and single notes using three possible pitch sets I, IV, and V. Then the notes are played back using buffers with [tabread4~]. Fig. 5 bottom shows the resulting parameters exposed in the Wwise UI interface.

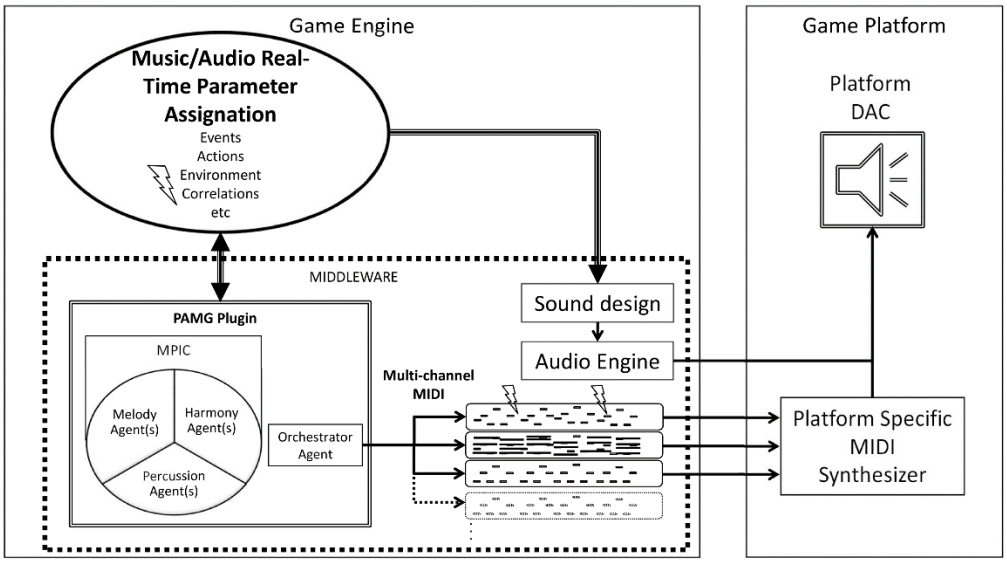


Fig. 6. PAMG using instruments from the game platform.

The second option offers the versatility of platform-available MIDI instruments customized by the final user. For example, in PC platforms, it can be routed to virtual instruments with their custom libraries in addition to MIDI recording software, or even to external hardware instruments through MIDI out ports. However, MIDI instruments may not have an output quality comparable to CBI, and effects and mix levels are an additional setup to manage for final users. Additionally, the current Wwise structure does

not include MIDI out as a compiled in-game feature, although Unreal Engine has this capability. This means that additional coding either in the middleware plugin or in the game engine scripts/blueprints should be done to enable MIDI out port access.

In general, the easiest way to fit PAMG into a game developer's pipeline is by introducing the least changes possible, which translates into readily available compatibility with currently employed technology and business models. A well designed and simplified user interface also is paramount to integrate with current visualization paradigms in middleware and/or game engines. By including PAMG as a sound-design middleware plugin, sound designers will have an opportunity to be in charge of music design through PAMG, making them the music authors.

Through a PAMG, a composer (or user) can come up with a style as a result of algorithmic parametric structure setup used by the engine to generate a range of musical developments or progressions. Then, the music underscoring process would include the design of those setups as aesthetic requirements for game sections, and the technical aspects of generating streams, transitions and interactions are handled by the music engine. The composer will handle the creation of a musical identity while the engine will deploy and implement it.

Game composers, as regular content creators, may incorporate their linear productions for higher-impact music that are in charge of the game's storyline identity (e.g., headers,

openers, and cues) while generative music sections address gameplay. Using PAMG as a middleware plugin has the advantage of allowing conventional and generative music implementations in the same game sound track. This would make important the establishment of aesthetic consistency among the material and the system used.

3. PAMG algorithm

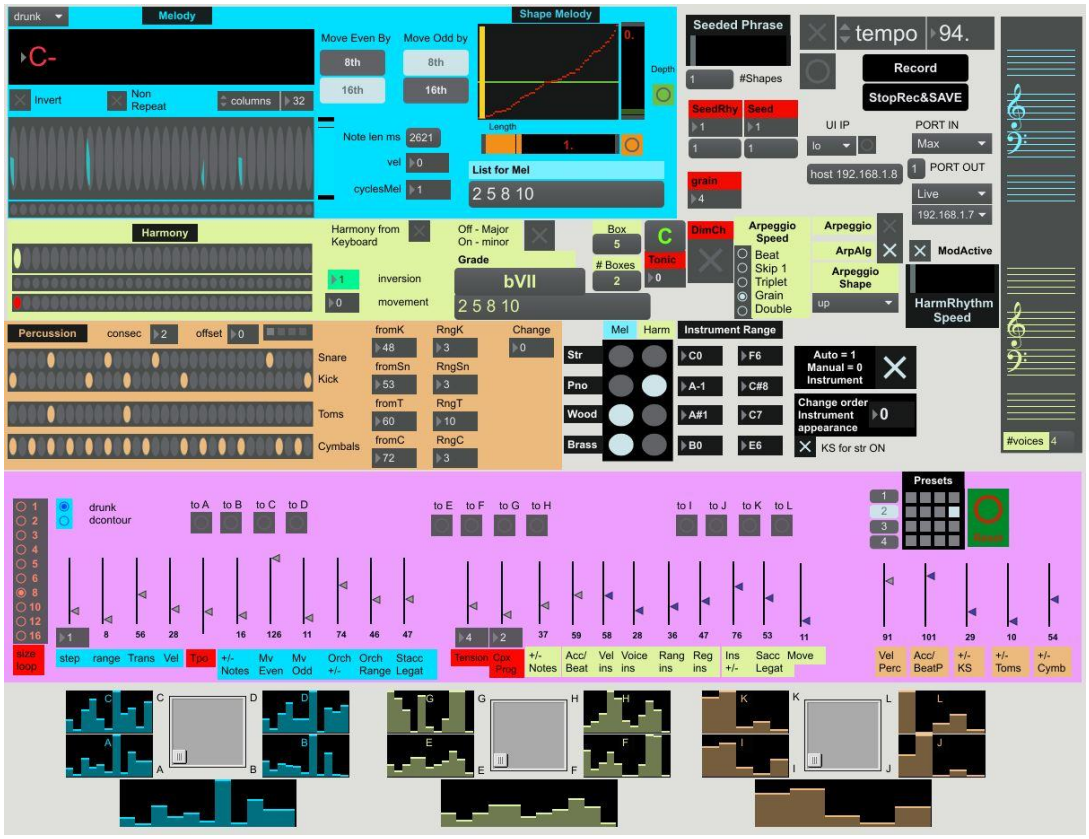


Fig. 7. Progressive-Adaptive Music Generator (PAMG) developing UI

A thorough description of the PAMG can be found in Appendix A.

The architecture used to build PAMG is an *agent network* with granular control over several levels of melodic, rhythmic and harmonic complexities. In addition, parameters belonging to each agent and global system setup can be modulated by incoming gameplay variables.

It comprises a Melody Agent (MA), a Harmony Agent (HA), and a Percussion Agent (PA) (blue, light green, and orange in Fig. 7 respectively)(Appendix A, 2.-5.) Also, visualized in the black matrix ctrl object, an Orchestrator Agent (OA) receives note generation from MA and HA and assigns instrument families to the output, controls instrument range, and automatically makes real-time decisions about instrument appearance. A preset manager and Multi-Parameter Preset Interpolator and Controller (MPIC) (purple, and lower grey XY pads section in Fig 7) have the function of receiving individual parameter control, and interpolate among multi-parameter presets respectively (Appendix A, 6.) Each agent's presets are assigned to the four corners of an XY pad, where the interpolated result is displayed in the respective lower multi-slider display. In the right, a score-like visualization of the generated notes and their range for MA and HA is displayed. Other global controls such as MIDI port selectors, tempo visualization, Harmonic Rhythm speed, Seeded Phrase controls, and Modulation algorithm toggle can be viewed in the upper right section.

3.1. PAMG typology

Using Plut and Pasquier (2020) typology model, PAMG can be an example of some of the proposed categories[37]. The classification should be understood as descriptive rather than restrictive.

[37] In their article "Generative music in video games: State of the art, challenges, and prospects" (2020) Plut and Pasquier offer a complete explanation of their typology, and an overview of 34 adaptive music systems and/or games. In the current section, I use italics to denote the *mode* or *subcategory* that fits PAMG closer within each category.

3.1.1. Musical dimensions

- Generative task:

Composition. The system is in charge of creating new music.

Arrangement. It addresses the recombination of extant elements from melody lines, and chords within chord pools.

Performance. Interpretation is primarily the result of parameter variation on a MIDI event's *velocity* (granular dynamics), and through algorithms like Melody/Harmony/Percussion Change (see Appendix A, 2.5., 3.4., and 4.4.).

- Directionality:

Mixed. PAMG handles *vertical* and *horizontal* directions in music. It creates and manages sequential events and temporal superposition of musical elements. MA for example, handles the horizontal direction of the leading music line, while the HA handles the vertical grid for any note event. MA also handles the 'moved list', a rhythmic pattern that other agents can follow vertically either by supporting it or diverting from it, adhering to the meter hierarchy. HA also handles voicing and range generation which belongs to the vertical direction. OA handles the instrument families and the ranges assigned among them. It also reassigns instruments to provide variance.

- Granularity:

Phrase, *measure*, *beat*, *note*, *chords*, *instrument group*. PAMG uses the *note* to execute the most immediate changes especially in the melody. The *measure* granularity type is

used as a ‘loop size’, which in some cases would not reflect a musical measure. For example, a ‘loop size’ of eight beats can be felt as a measure in 8/4 or two measures on 4/4. Although ‘Loop size’ is the default window used to switch chords by HA, any beat-sized onset in the loop can trigger a change—no chord switch is permitted when the unit is shorter than a beat or a ‘moved’ beat. The *Phrase* granularity type is used to trigger or allow changes in the arpeggiator and instrument family among others. Chords generate a grid for any note performed. They also can be manipulated by adding and subtracting tension which limits the available pitch classes.

- Grid/Groove:

On. The subdivision of the loop size—the ‘grain’—is the minimum event duration with the exception of the arpeggiator output, the ratchet, and onsets in ‘staccato’ mode.

3.1.2. Gameplay dimensions

- Diegesis:

Non-diegetic. The music is not produced within the game world and belongs to the storytelling presented to the player.

- Ambience:

Ambient. The music is not connected to a source.

- Adaptivity/Autonomy:

Adaptive. Although the music generation has more than fifty parameters that can be tied to game variables, PAMG is able to execute changes if left unsupervised or unchanged which adds a level of autonomy from gameplay.

3.1.3. Architecture dimensions

- Generality of the system:

Generic. PAMG is designed to work on a plurality of games. The presets elaborated by the user and the decisions of parameter assignment are specific to a game.

- Generative algorithm:

Rule-based and *stochastic*. Rules in PAMG act as modifiers and constraints for stochastic event generation. Deterministic (e.g., pattern generation) or non-deterministic variance (e.g., velocity generation) are established by seeding/not seeding pseudorandom generators.

- Musical representation:

Symbolic. Inside PAMG, basically two types of symbolic information are shared: numerical (integer and float) and harmonic (see Fig. 11). The numerical symbols cover magnitudes such as MIDI-like formatting (pitch, length, and duration), handle paradigms of organization such as loop onsets indexing, and establishes variance ranges such as

Seeded Phrase generation. The harmonic symbols managed by the chord pools use Western roman numerals (e.g., I, ii, IV, V7, etc.).

- Musical knowledge source:

External. PAMG currently uses algorithmic methods with a number of parameters that affect ranges and behavior. The musical result is unique in the sense that no previous music analysis is employed to structure the model (besides the knowledge and experience of the composer/user).

3.2. Music creation methods overview

In this section, current visualization and the corresponding parameters[38] are explained from the point of view of music generation.

3.2.1. Rhythmic structure creation and progression

The rhythmic development algorithm is based on Lerdahl & Jackendoff (1983) Generative theory, in which the onsets in the timeline acquire a hierarchical place according to its position in the established meter and the accent management. The most simple version of a motive has a superior level in that system (i.e., higher in the tree structure) while the final subdivision of onsets that includes syncopation would be at the end of the tree branches.

[38] Parameters' names will use italics when the name is exactly as displayed by the user interface, or if it is a named information used in the algorithm description in Appendix A.

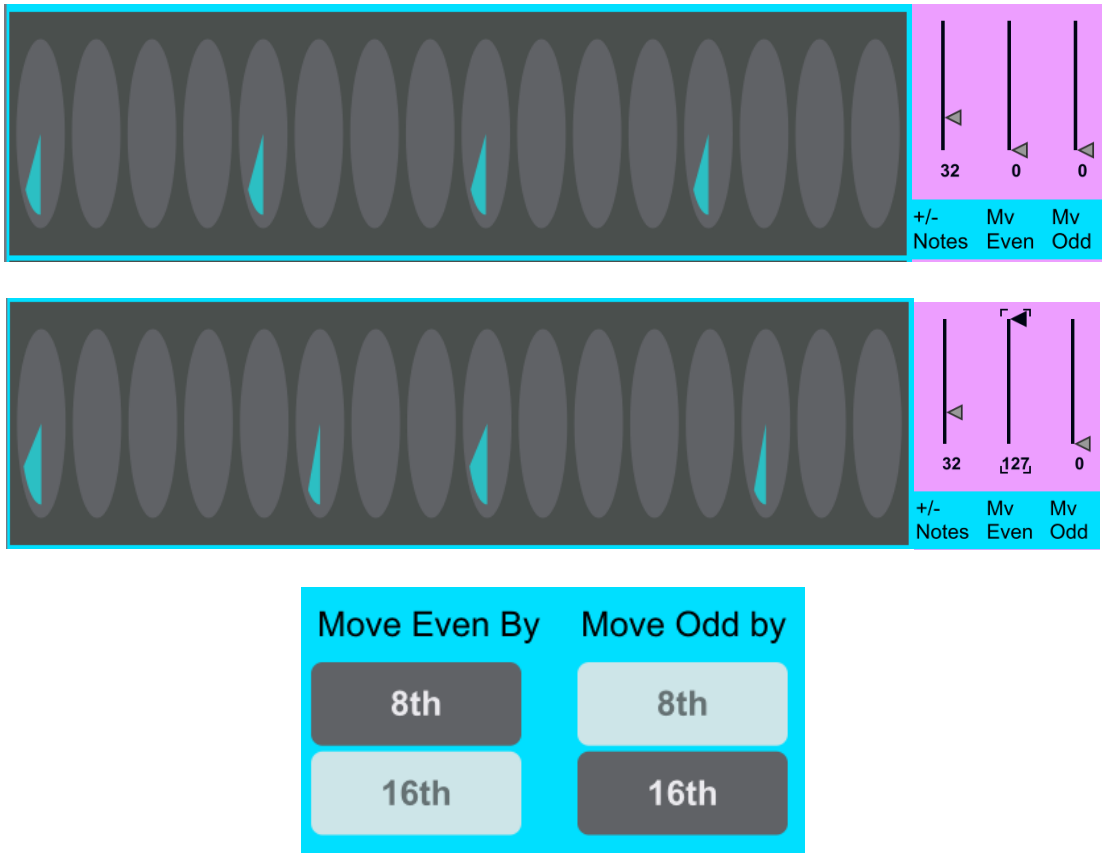


Fig. 8. Odd and Even beats moved and their related sliders.

In the first row, a 4/4 measure with 4-*grain* (16th note) producing main beat, and the corresponding slider positions. Note that the +/- *notes* is not on 0, since it has a number of onsets. From this point down, it will subtract one onset at a time. In the second row, a syncopated rhythm based on the previous onsets. *MvEven* is at maximum meaning that all the even onsets (2 and 4) have been shifted by the amount shown in the setting *Move Even By*. This main rhythm will constitute the *Moved List* and *Beat List* to be used by other agents.

The beat itself is used as a starting point, since it is likely the simplest information that can be carried by a rhythmic pattern. As this 'beat-following' pattern still has the same number of onsets as the meter, the algorithm executes a subtractive operation in the lower section of the +/- *Notes* slider to get the range from 0 to the maximum number of beats within the loop (*Loop Size*). It populates the main beats one by one using seeded random number generators and checks if they are moved as shown in the *Moved List* (Appendix

A. 2.1.) Then, it continues populating in lower levels in the hierarchy, i.e., 8th notes, followed 16th notes one by one until the maximum grid onsets (*Loop Size* x *grid*). In the current design, an algorithm permits a small overlapping in the onset filling algorithm for 8th and 16th note levels in the melody so the obtained patterns are slightly richer—it fills some 16th notes before finishing filling 8th notes. As a result, the rhythmic pattern grows in onsets and complexity on top of an original accent structure[39]. The resulting list also affects the event velocity to preserve the groove and the accent/beat (*acc/beat*) parameter for all agents (Appendix A, 2.1., 3.1., and 4.1.). The algorithm also calculates the distance to other onsets to set the duration parameter as either *legato* or *staccato* (Appendix A, 2.1.3. and 3.1.3.)

3.2.2. Phrasing

The harmonic progression determines the phrase (Appendix A, 3.3) in the sense of departing, acquiring tension and reaching resolution, which constitutes a harmonic cycle. In the melody, pitch generation (either *drunk* or drunk-contour *dcontour*) is seeded in the start of the loop to repeat the gesture whenever the seed is sent. Seeds can be switched for each loop completion, creating a sequence that enables different gestures per loop until the harmonic cycle has terminated (*Seeded Phrase*). This also applies to the *Rhythm Seed* producing greater complexity when more shapes are used.

[39] Accents are onsets whose velocity parameter is proportionally higher than others. In the current model, the onsets that receive this emphasis belong to the beat or *Moved List.*

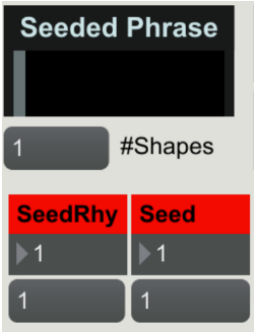


Fig. 9. Seeded Phrase controls.

Additionally, the Change algorithm (Appendix A, 2.5, 3.4, and 4.4) lowers the amount of onsets when there is a harmonic resolution and increases them on dominant sections. Higher values in the Harmonic Complexity parameter (*Cpx prog*) produce a growing number of *Boxes* containing harmonic pitch collections or chord possibilities (see Appendix B) managed by the Chords Pool Manager (Appendix A, 3.3.2.1.) The *Harmonic Speed* parameter adds or subtracts chord-switching onsets within the loop to change the structure of the phrase and making it shorter or longer. The pools and the level assigned can be seen in Appendix B: Chord Pools.

Segmentation in phrasing is reinforced by changes in orchestration (instrument assignments in the OA), change and switch on/off in the arpeggio algorithm, and fluctuations in number of onsets and pitch-class lists depending on dominant/tonic entrances. In addition, an algorithm enables sections with longer endings as ‘breakdown interludes’ disabling harmonic and melodic changes for a number of measures depending on *Cpx prog* and *Loop size* parameters.

3.2.3. Melody contour/shaping

The drunk and drunk contour (*drunk* and *dcontour*) modes (Appendix A, 2.3.1) are modified versions of the Drunk[40] object in Max/MSP. The drunk mode acts basically as the drunk object with the difference of receiving a seed for the sequence and not for the range, although the range is easily set.

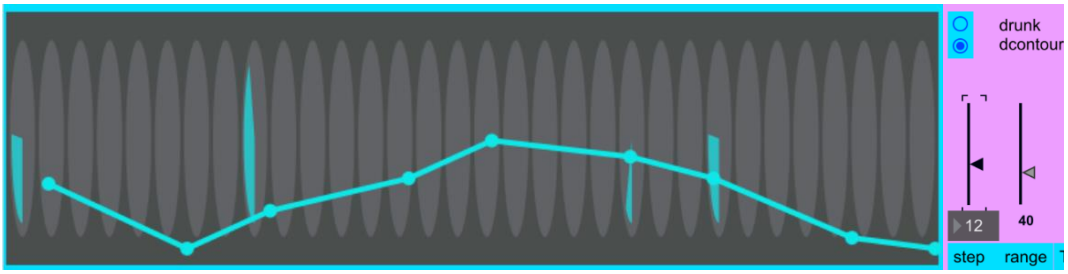


Fig. 10. Drunk Contour (dcontour).

The direction of the melody line is then affected and can be distinguished by the listener when the seed starts the repetition. The *dcontour* mode uses said drunk output to generate a pitch contour per loop. Then, new onsets added will follow the interpolation lines between the *Moved List* onsets provided by the rhythm generation algorithm in MA—the same producing the accent structure. Additionally, a long range shape (Appendix A, 2.4) is able to granularly modify the pitch range from 1/4 up to 8 loops (*Loop Size*). This preserves the contour, and also provides pitch directionality.

[40] Random walk. https://www.britannica.com/science/random-walk. Accessed 5/9/2023.

3.2.4. Harmonic progression

The progressions follow generative grammar procedures similar to David Cope's (1996) Augmented Transition Networks. The system's node values reflect a complexity level. In addition, the complexity parameter value adds nodes and swaps chord options to the phrase. The following is a graph of available chords:

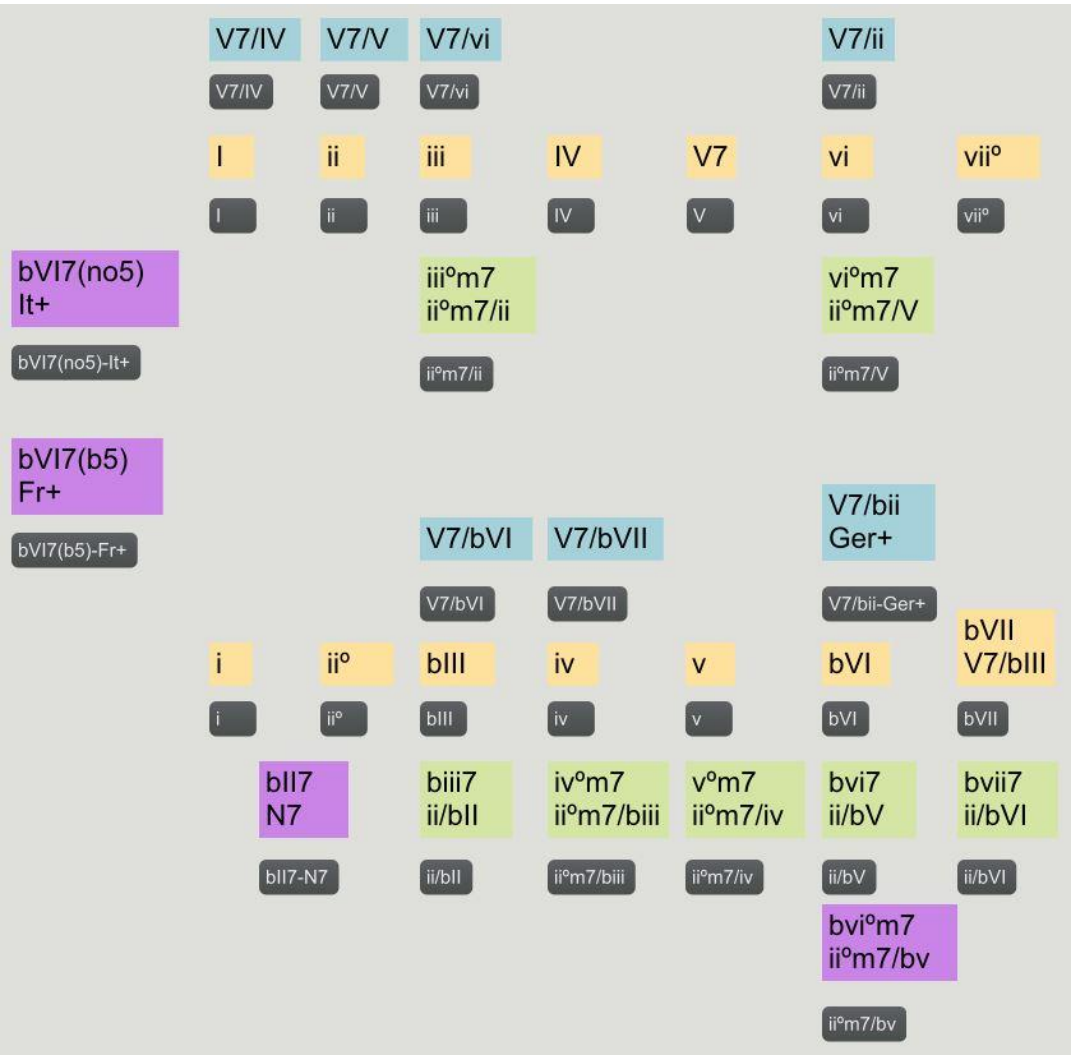


Fig. 11. Chords.

The symbols chosen borrow from traditional Roman numerals used in Western tonal music analysis used first by J.P. Rameau (Christensen, 2002) to describe basic tonal functionality[41]. Chords symbols with yellow background are formed in the major and minor scale. Additional chords formed in the same grade's root occupy the same vertical axis (blue on top and green below), and receive a name closer to its tonal functionality. For example, ii in Major mode can be modified to be a major chord, which in functional terms it is called V7/V (blue background). As the Major and Minor modes will interleave in further complexity levels, the III in Minor is called bIII to distinguish its root from iii in Major, in a similar way v exists in Minor mode and shares fundamental with V for the global root (tonic). bIII with one modification can become minor suggesting a ii grade in an eventual modulation to a bII, in short ii/bII. In general, blue chords are modifications that make them secondary dominant, while green chords are 'secondary subdominant' (either as a ii or a ii° of an eventual tonal path). This symbolic naming is used to remind the possibilities in progression when chords are chosen and determine possibilities for the next node. The purple chord symbols constitute a third level of modification that connects complexity with tension when chromatic pitch additions are included. Among them I included the Neapolitan, Italian and French augmented sixth chord structures (N7, It+, Fr+). The German augmented sixth (Ger+) was considered as a possible V7/bii since its configuration is a dominant seventh chord, hence the blue background. The dark grey

[41] Some differences are: any diminished minor seventh chord, e.g., ii°m7 is seen more often as ii^{ø}7. In general, including 'descriptive' information is due to the gradual inclusion of pitch classes—in the first three tension values it does not include 7th—and the possible directions in harmonic progression. Also, to write the augmented 6th chords I described the intervallic and harmonic relations before It+, Fr+, and Ger+. Another reason to modify original symbolic notation is the limited supply of special characters and plain text tools in Max like superscript.

message (message box) under each colored chord contains the actual symbol employed in the algorithm.

3.2.5. The tension parameter

As discussed in section 2.1., the tension method chosen adds selected upper pitch classes one at a time with increasing dissonance to the harmonic node (chord) completing 12 levels. In the case of I in major (Fig. 12), it starts with conventional consonances—0, 7, 4—, then introduces upper pitch classes in thirds—11, 2, 5, 9—increasing harmonic tension by adding dissonance with the first classes, then adds pitches that belong to the parallel minor pitch collection—3, 10, 8—, and finally adds chromatic dissonances in relation to the root—6, 1. For chords executed by HA, tension is combined with number of voices (*voice ins*) to choose the pitch classes. For example, if the *voice ins* parameter selects 3 voices and the *Tension* parameter is its maximum value, the voices chosen are the right-most in the 12 level: 8, 6, 1 for I (Fig. 12.). In case *Tension* is 7 the voices would be 7, 11, and 2. This adapts the available pitches to the current number of voices to reflect the tension assigned.

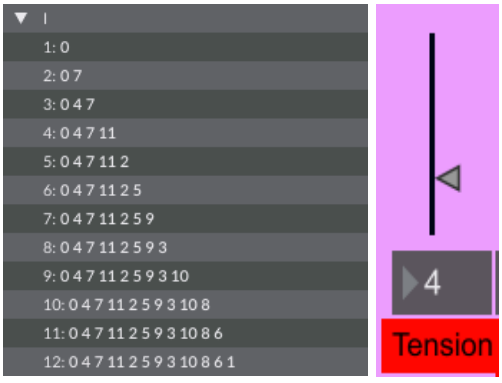


Fig. 12. Tension levels for I in C major.

Pitch class sets per level and per chord are stored in a dictionary. This output is converted to a harmonic grid that serves as a playground for chords and melody. It also feeds the bass algorithm for HA and fluctuates as a result of tension-resolution changes to reinforce phrasing. The complete dictionary can be seen in Appendix C.

3.2.6. Velocity generation

As each agent produces MIDI formatted data (pitch, duration, and velocity), each one has an independent velocity parameter that affects the general loudness of any instrument assigned, and that level serves as a center for fluctuation or 'humanization'. This is accomplished by a drunk object with a variable step and range. It is affected by the main accent pattern (*moved list/beat list*), which increases the value in a range between 10 and 25 (out of 127) on accented onsets, as can be seen in Fig. 13:

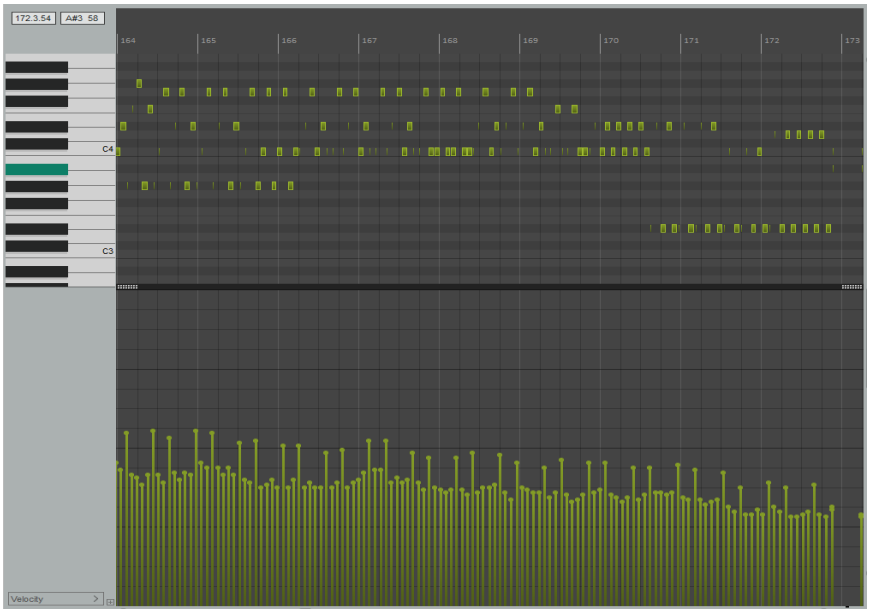


Fig. 13. Velocity generation with accent.

3.2.7. Interpolation

The MPIC module (see Appendix A, 6.) handles I/O for agent's parameters, stores presets for themes, and manages interpolation through XY pads.

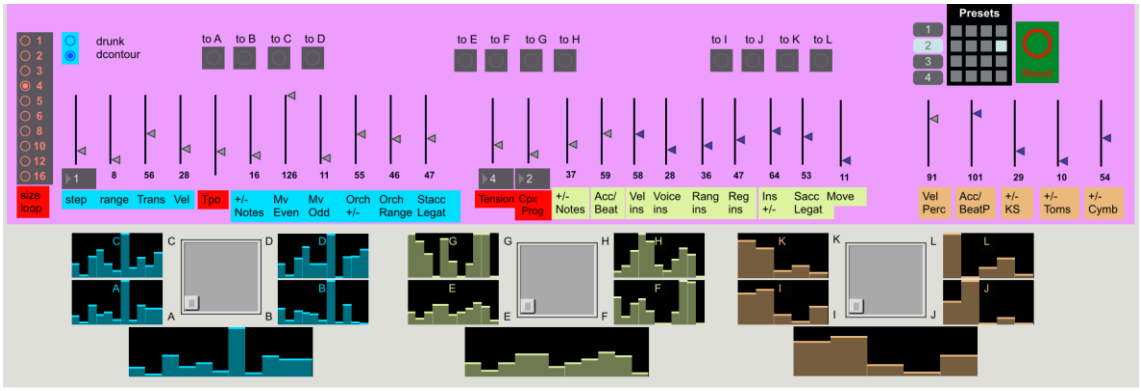


Fig. 14. Multi-Parameter Preset Interpolator and Manager (MPIC).

Parameters per agent are color coded: MA blue, HA green, PA orange. Parameters in red (*Tension*, *Cpx Prog*, *Tpo*, and *size loop*) are managed by the corresponding agent setup but affect global aspects. Presets are organized in rows of 4, and are assigned per agent to a corner of their respective XY pad interpolator (letters A-L), using the buttons in the top. Each corner in the XY pads is associated with preset shown by a color-coded multi-slider, and in the bottom, the interpolated result is shown. In the top right, a *Reset* button restores interpolated values for presets, since parameter sliders can carry an offset when handled individually.

In the current version, it also contains the setup for OSC I/O that can receive data from another user interface like a multi-touch screen on an iPad. This is provisional interface

design for performance that employs a subset of parameters. Selected sampled pieces present in the accompanying Media (see Appendix F) were performed using this layout.

Fig. 15. Performance UI in TouchOSC.

3.2.8. System specifics

The current design is optimized to work with full range instrument sets of Strings, Piano, Wood winds, and Brass. Some algorithms would need to be revised to act in other instrument sets. For example, arpeggios within a brass section are not idiomatic so they are avoided, only ‘piano solo’ allows arpeggio rates of 1/32 note value, and key switches are used for staccato strings (this also is particular to the sampler/bank device). In PAMG description (Appendix A), I excluded technical methods such as math functions, counters, loops, and the like, and user interface (UI) visualization and construction items—unless they serve a purpose in music generation—to focus on the functionality.

4. Evaluating PAMG

This section includes a description of the Trial Game, its origins, game plot, and variables used in the present research to evaluate PAMG. Then, it explains a PAMG-CBI comparative test in which participants play the Trial Game in both configurations and respond a questionnaire about the gameplay experience. This comprises a description and comparison of the methods used to design CBI and PAMG implementation, and the gameplay experiment with its results.

4.1. The Trial Game

A First Person Shooter style template was chosen and a hybrid adventure-action story selected due to the presence of the mentioned characteristics in recent years[42]. This choice aims to test several modes of style interpolation capabilities present in PAMG that vary in reaction time (action vs. location), storytelling (hints about player performance), and instantaneous musical needs (game state changes such as death, pause, and menu).

4.1.1. Origins

The trial game *Music Portals* was designed in the beginning of this research as a test ground for interactive music layering. The idea was to use available tools in Audioketic's Wwise middleware to try assignation of gameplay variables to musical changes in combination with the FPS (first person shooter) Unreal Engine (UE) template in an open world environment. Specifically, its single-level size was used to experiment layered

[42] https://www.statista.com/chart/24700/favored-video-game-genres-in-the-us/. Accessed 5/6/2023.

music crossfade based on distance to map location points. As disclosed in evaluation (see Chapter 5), this earlier implementation experiment did not yield the best musical results.

The current game is an updated version, in which PAMG solves, using generative properties, the problem of style interpolation among multiple map location points. The idea of using a simple scenario for localized testing was replaced by a basic albeit complete, one-level game, with narrative, story, and aesthetic to achieve a real-life scenario for testing. Action and adventure game narratives present in Music Portals and other FPS are desirable to test transition, progression, and responsiveness; a reason to continue developing on top of the existing game. Although developing NPC behaviors and managing lights and textures consumed some time, it was economical in the sense that most of the dynamics and the map are in place already. For example, instead of creating new assets, I modified the factory default mannequins as non-player characters, the first-person gun and arm available on the template, the portal dynamics, and the map from the earlier game. As a compromise between game developing time and simplicity, I settled for a straight-forward story although verisimilar enough to produce an immersive experience that allows the model to demonstrate its potential and limitations. I learned UE programming and also hired a game developer to speed the process of pulling variables. I also developed and implemented conventional sound design in Wwise.

4.1.2. The game plot

You—the player character—are a robot left in a seemingly mysterious world. Your energy is depleting with time. You have a gun that sends what it seems like an incandescent ball. At game start, you are told to ‘find the cube’ which possibly can fix the problem of depleting energy, and is perceived as the main game level goal. Also, you are told that there would be hints to help you find the path. The nightly world seems like an open field with mountains and dirt grounds mostly arid in which you need to adventure before your energy is gone.

You quickly encounter other robots that wander the land. The mystery of ‘friend or foe’ is broken when you notice they chase you. Shooting them seems to affect their speed allowing you to escape, but they do not perish and there are many. They suck your energy if you allow them to reach you. It does not feel good; the action provokes a glitch in your screen, and a chunk of energy is lost.

In the run, you may find a portal. It looks like a door to another world, easy to see through the threshold. When you cross it, your energy is replenished which grants you extra time to continue your journey.

At some point in the level, you may find a lake in which you can submerge and notice that your energy depletes faster at deeper depths. You may see also a big mountain with a path signaled by an arrow and a note saying it leads to the top. As you examine the mountain, thunder and lightning suggests that the path may lead to the goal. The path is

long, since it circles the mountain, and often dark, so you use your light balls to illuminate the terrain. The higher you get, the clearer it is that you approach something important, aided by the music's development. As you approach the top, you clearly feel the change of mood and several blinking lights show you that something big is about to be revealed. There is a red flashing light coming from a crack and you decide to approach, which is supported by dramatic music and an increase of the blinking's frequency. You realize that it is not the cube, but a direct sight of where it actually is. It is not close, but your effort of reaching the top is rewarded by a replenishment of energy.

The search continues. At some point you find another portal that is especially hidden but when you approach the music gradually becomes epic. When you reach it, you read a sign that states that you are getting closer. After crossing, you notice that your options narrow: no space to run free because you have steep mountains on one side and lava on the other. You follow some arrow signs that guide you through a path that crosses the lava and towards a mountain. Then, beyond the mountain, you clearly see the cube which is emphasized by the music. Trying to stay in the path —falling may mean death by lava burn— you get closer to the cube and feel the epic and celebratory music.

By reaching the cube, you get into a 'winning' level. This level only serves the purpose of showing the music transformations in a minimal map with the portals. It is possible to play again by crossing a central door.

4.1.3. The game variables

A list of variables[43] exposed for assignation to PAMG parameters, visible using a switch at gameplay, are outlined below:

1. Player's *world location*.
2. Player's *distance to totems*.
3. Player's *orientation* in relation to goal (camera on pawn rotation in relation to goal in map).
4. Player's *distance* to goal.
5. *Average* of weapon usage (*trigger pull*) in an interval of 1 sec.
6. Density of enemies (NPC) in proximity (*enemies in radius*).
7. Number of *enemies in chase mode*.
8. *Distance to closest enemy* in chase mode.
9. Number of *enemies hit* by player's projectiles in an interval of 2 sec.
10. Amount of player's *energy lost by enemy attacks*.
11. Player's current *energy*.

The totems (*Bright*, *Mysterious*, *Tense*, and *Epic*) are invisible objects with a map location in proximity to the portals, with the exception of Tense. Their distance to the player is sent in real time to PAMG to determine the degree of a preset used to associate a

[43] In this section Italics indicate the game variable or object name used in the design, and the parameter names in PAMG.

'mood'. As the player gets closer to a totem, the interpolation algorithm in PAMG assigns parameter values that are closer to the preset. As the portals are a binary teleport system, totems are duplicated and the distance is calculated as the minimum of the two or three values —Epic has a third instance placed in the goal cube's location. Tense's only instance is at the mountain's peak.

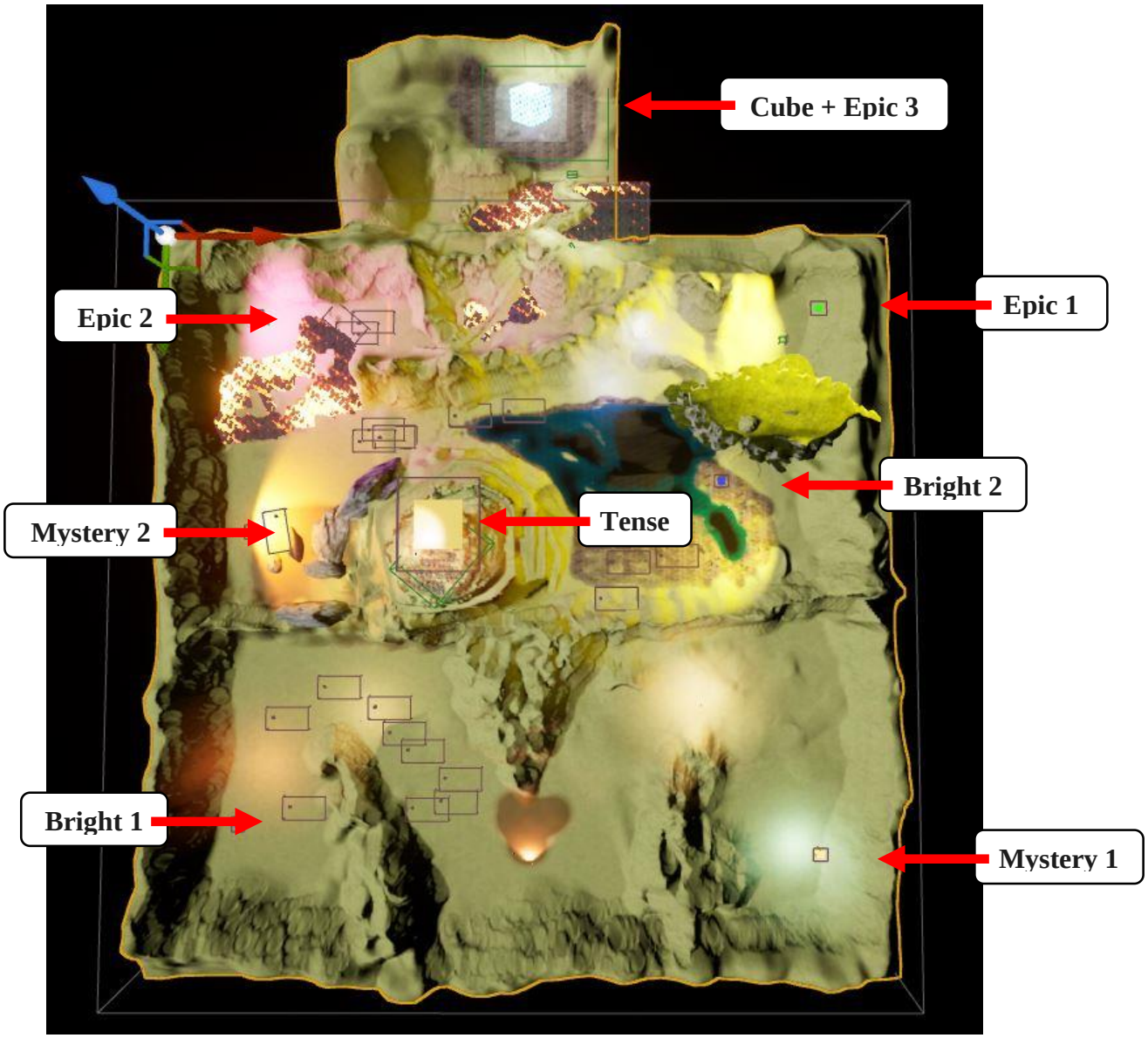


Fig. 16. Map and totem location.

Individual, bipolar parameter control was the selected input mode in PAMG and the output mode from the game. This means that, in addition to the *relative distance to totems* that is assigned to the preset interpolator, it is possible to assign any variable from gameplay to single or multiple parameters, either adding or subtracting from its current position. The variables cited above were thought to be useful in this assignment. After testing, some of them received more importance and additional parameters to modify while others remained unchanged. These are the current assignments:

- Player's *world location*, Z axis:
    a. *+/- notes Mel*. Increases the number of onsets in melody the higher the player is, to put some movement going up the mountain.
    b. *Shape Depth*. Increases the depth of the curve the higher the player is, to shape the melody either ascending or descending.
    c. *Shape Length*. Decreases the length of the curve the higher the player is, to add tension in melody movement.
    d. *Invert*. Finds if the player is ascending or descending to apply the same principle to the melody through the shape.
- *Enemies in radius*:
    a. *DimOn*. When the player is in the radius of an enemy, the diminished filter for harmony is turned on to add an enigmatic sense to the environment.

- *Enemies in chase mode*:
    a. *Vel Chords*. Increases the velocity parameter, to support the tension and action.
    b. *Tension*. Increases to add dissonance supporting a dangerous situation.
    c. *+/- notes Mel* to 0. To allow the chords to depict the action and tension, the melody is muted.
    d. *ArpAlg*/*Arp* off. This allows the chords to maintain ostinato and also intensity.
    e. *Vel perc*. Increases to support action.
    f. *Toms +/-*. Increases the toms onsets also to support action.
    g. *SeedRhy* to random. Avoids same rhythmic pattern when NPCs enter chase mode.
- *Distance to closest enemy*:
    a. *Reg Ins*. Increases the register of the chords to add tension.
    b. *+/- notes Harm*. Increases the number of onsets according to *distance to closest enemy* in chase mode to add movement progressively in an action passage.
    c. *Reset* on non-being-chased. Returns values to default interpolation.

Besides the variables, a number of trigger boxes are placed in the map to provide additional toggle and local functionality:

- Trigger box by Epic (in goal direction):

  Begin overlap:

  a. *Loop size* to 4. Reinforces the epic mood.

  b. *Min/Maj* to Maj. Sets the tonality to major.

  c. *Vel Perc* to max. Adds percussion presence.

- Trigger box by Epic (reverse direction):

  Begin overlap:

  a. *Loop size* to 8. Longer phrases may suggest longer paths.

  b. *Min/Maj* to Min. Sets the tonality to minor.

- Trigger box Pre goal 1:

  Begin overlap:

  a. *Loop size* to 2. Reinforces the epic mood.

  b. *Arp* Off. Allows the chord impact to support epic mood.

- Trigger box Pre goal 2:

  Begin overlap:

  a. *Arp* Off.

  b. *Acc/beat harm* to minimum. Reduces any syncopation to minimum.

- Trigger box Mountain:

  Begin overlap:

  a. *Arp/ArpAlg* Off. In a short distance to the mountain top, the chords are the basis for the dramatic progression.

  b. *Acc/beat harm* to minimum.

c. *Brass Mel* Off, *Piano Mel* On. Mutes the Brass in melody to allow faster idiomatic melodies in piano, woods and strings.

End overlap:

a. *Arp/ArpAlg* On. Returns functionality to default.

b. *Reset*. All parameter values go to the interpolated value.

- Spot light Mountain:

c. *Reg ins*. Calculates the distance to player and applies an increase on register to chords the closer the player is.

Additionally, the game receives and can potentially assign any variable from PAMG. In the current version, I assigned:

- *Beat duration* in milliseconds, used for event synchrony in tempo on dancing animations in the Winning level.
- *First beat* of a measure and *main beat*, used to trigger changes at the same time as the music in dance types.
- *MIDI note-on events for percussion* (except cymbals), trigger lighting in the mountain top.

The game presents many possibilities of musical relationships that unfold during gameplay. Local variables and instant events can easily send parameters by implementing trigger boxes, local executions, or other actors. The implementation of random variations

triggered from the game also proved effective to set user seeds. This prevents repetitive developments, especially noticeable within rhythmic motives.

4.2. Comparative test

In this section, I evaluate PAMG by comparing its implementation with CBI. It consists of two distinct sections:

a) *Comparison of efficiency in music implementation* between CBI and PAMG. The music composition and production labor—which is carried in the experiment by PAMG in real time and PAMG music recordings in CBI—is not part of the present evaluation[44]. Gauging implementation efficiency starts with a description of the process and the methods to achieve design goals. The comparison will address time consumption and found limitations.

b) *Gamer subjective perception test*. The experiment intends to compare the two implementations from the perspective of a player's experience, accounting for background factors.

[44] My experience as a composer suggests that a composer likely would take less time setting up four themes using PAMG parameter presets than writing four orchestral themes in the same instrumental setup, plus editing and mixing. However, there are very effective and competent composers that could challenge that assumption. This comparison, is suitable for further evaluation.

4.2.1. Description of the implementation process

The aim is to provide a comparative observation between two methods of game underscoring in one videogame, aiming to match PAMG's design goals with CBI (see design goals below). The two methods are:

i. Generative, progressive-adaptive music implementation using PAMG being controlled by the game variables in real time. The music authorship belongs to the person that designs the PAMG parameter setup and game variable assignation.

ii. Audio clip-based music (CBI) implementation using Wwise middleware. The music is recorded from PAMG output to focus comparisons on adaptability rather than on music style or mix. Hence, the implementation using Wwise intends to follow the same principles of PAMG implementation—to the extent of middleware possibilities (see limitations)—and the recorded music segments employ the same PAMG theme parameters.

The 'composition' phase in PAMG—before implementation—consists of assigning values to PAMG parameters and store them as presets. These are the four Themes assigned to each XY pad corners in the interpolation module. Although comparing composition efficiency is not part of this evaluation, other automated music generation

evaluations[45] suggest that traditional composition would still win in ‘quality’ while PAMG as other automated models would be efficient in ‘amount’.

The implementation using PAMG consisted of assigning game variables to MIDI controller change messages (cc), and testing results. This is an implementing-testing-tweaking cycle that includes ranges, curves, switches, and so on. The experience of implementing parameters involved setting up methods to send game variables through a MIDI interface from the game engine. Once the MIDI plugin is installed in Unreal Engine, the objects are able to send specified cc values. Several methods were designed to adapt ranges and send messages only when a variable is changed, and scaling the output to a 7-bit range (0-127), also having in account the bipolar property—a value of 63 resets the controller to interpolated value, between 64 and 127 sets a proportional increase in the current parameter value, and between 62 and 0 sets a decrease. PAMG’s message reception must be minimized since the format of some game variables are updated every frame and represented by a floating point number. In a game developer work environment, this labor may consume less time with the help of a programmer that sets the music events and other variables that would be sent to the PAMG.[46]

[45] Examples of subjective perception Turing-like tests (Hadjeres, et al., 2017; Thickstun, et al., 2018; Donahue et al., 2019), and quality perception tests (Huang et al., 2019; Huang, et al., 2017; Hawthorne et al., 2019; Roberts, et al., 2018) suggest that human-made music quality is still generally superior to automated music.

[46] In the current game development, a programmer was hired to expose the required variables (see the trial game description), although I designed the conversion from these frame-updated, float-format variables to 0-127 byte MIDI cc messages sent only when they change or as initialization parameters.

Analogously, implementation on CBI included creating *game calls*[47], which is accomplished from the game engine side. Designing an implementation method in Wwise to accomplish the design goals that PAMG was already able to perform consumed additional time besides the implementation itself. In general, upsetting conventional music implementation methods with higher number of Wwise objects and a wider range of game variables could get closer to the design goals set for the current project, but this would require an even longer timeline for production and implementation. To some extent, this approach is used to obtain progression (e.g., *NPCchase* and *Tense* additional tracks with their own algorithms) but by extending implementation time, CBI efficiency was lower than PAMG. Nevertheless, the question of achieving a close result to PAMG was still present.

4.2.2. Covering PAMG design goals with CBI:

PAMG was designed to accomplish style interpolation and progressive development through parameters, which are included in the Trial Game design as implementation goals. In general, the method used consists of assigning game variables to specific PAMG parameters. These assignments can be seen in section 4.1.3. The following are the implementation goals and their respective methods employed in CBI to get PAMG results as close as possible using Wwise[48]:

[47] The term used in Wwise to assign game variables to its own object system that includes *game states*, *switches*, *parameter control (RTPC)*, and *events*. The names given to assets that belong to these categories in Wwise will be denoted in *Italics.*

[48] *Italics* will be used for method, asset name and proprietary asset categories in Wwise.

I. Implement interpolation among the four environmental themes (*Bright*, *Mistery*, *Tense*, and *Epic*).

Method: *assigning the variable "player's distance to totems" to the switch group "Music" and the state group "Tense".*

The minimum value of four distances to player location—one for each totem—is used to decide which totem (*Bright*, *Mistery*, *Tense*, and *Epic*) is closer. Evidently, the result is transition and/or blending i.e., one source to one destination, or progressive track layering, but not musical interpolation as PAMG performs (see section 5.3.)

The music is available in instrument audio tracks, classified into melodic and harmonic plus percussion for each theme except *Tense*, whose tracks are included in each theme's *music segment* container to blend them progressively.

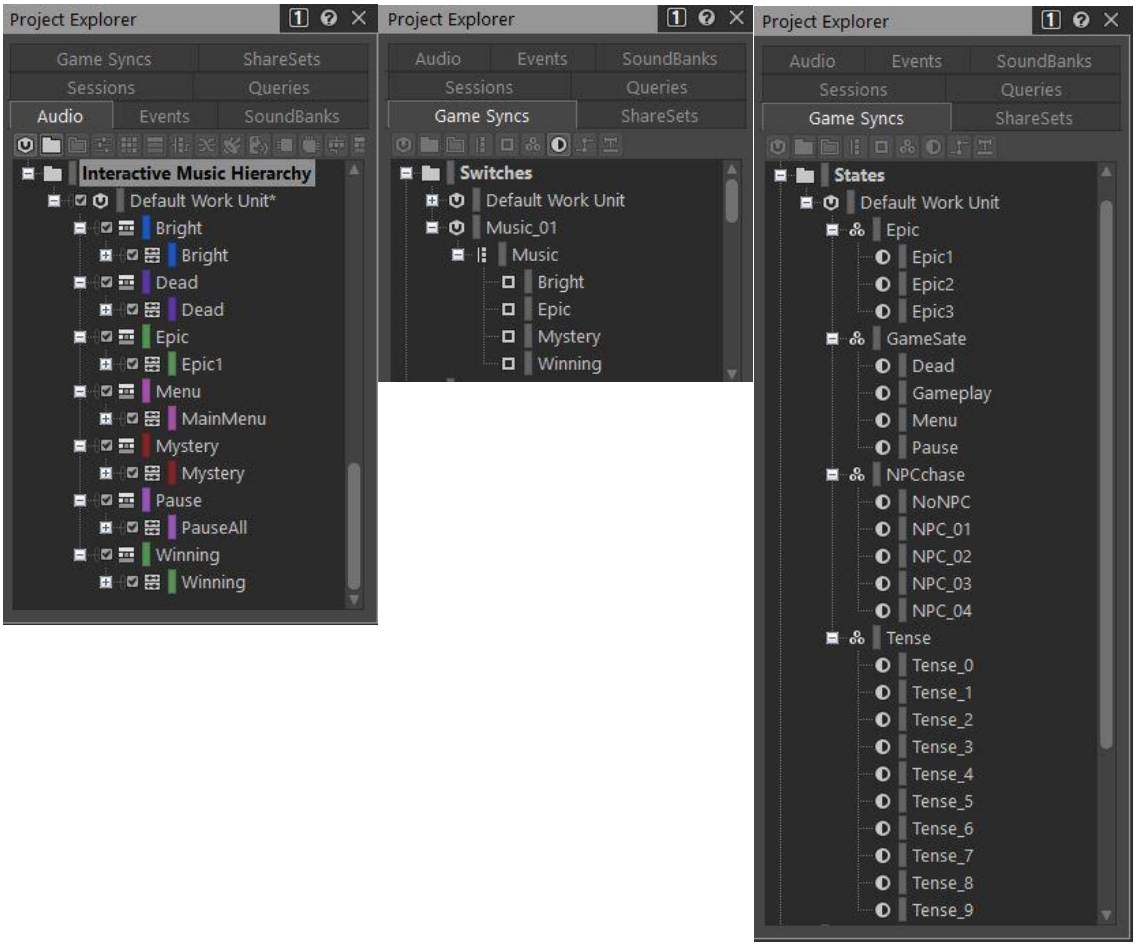


Fig. 17. Music themes (left), game switches (center) and states (right).

Each of the multi-track themes is implemented as a *music segment* wrapped in a *playlist container*, and each track as a *switch track.* The themes can be characterized by ‘active’ (*Bright*, *Mystery*, *Tense*, *Epic*1-3) or ‘static’ (*Menu*, *Pause*, *Dead*, and *Winning*) referring to the design goal of interpolating among them or not respectively. The *Tense* theme is applied as progressive track replacement on each of the other active themes and is tied to the ten states (one per track plus ‘Tense_0’) of the *state group Tense* (see Fig. 17 right). The *switch tracks* can individually use transition settings as a blending method for active

themes, and also can be assigned to ‘Tense_x’ states to replace progressively other theme tracks in a layered blending. The three *Epic* themes are also triggered by a game state that would be switched to ‘None’ in case the music theme switch is not on *Epic*.

II. Implement musical variations for:

    i. Action sections i.e., when the player is attacked by NPCs. They should work in any of the environments.

Method: *assigning the variable “number of enemies in chase mode” to “NPCchase” game state group*.

The *NPCchase* game state group (Fig. 17 right) receives 1-4 NPCs-in-chase-mode variable from the game, which activates progressive track replacement in *Gameplay* themes, one per instrument, increasing tension. For example, with one NPC in chase mode, the original drum track is replaced by the *Tense* drum track. If a second NPC enters chase mode, then the *Tense* wood-winds harmonic track replaces the original wood winds and so forth. This setup accomplishes the required synchrony among themes, and the layering allows four levels of *NPCchase* tension.

    ii. The player gets closer to the cube (goal) within the *Epic* environment.

Method: *assigning the current trigger boxes on Epic to the corresponding Music Switch.*

The *Epic* state group allows each of the *Epic* themes' tracks to use their transition settings when switching to a different state triggered by a game box. This theme has a *switch state* and a *state group* that need to act in combination.

iii. The player gets higher in the mountain (within the Tense environment).

Method: *assigning an average of the variables "player's world location, Z axis" and "Distance to Tense" to the corresponding State group (Tense).*

The *Tense* theme is applied using the progressive track replacing method described in the first design goal, tied to the average of the mentioned variables.

iv. The player is lower in energy and can die.

Method: *applying transposition/tempo change to music correlated to the variable "player's current health."*

Below half player's health, music is transposed up one half-step (100 cents) in correlation to the amount of health lost, and the tempo is increased using a curved RTPC.

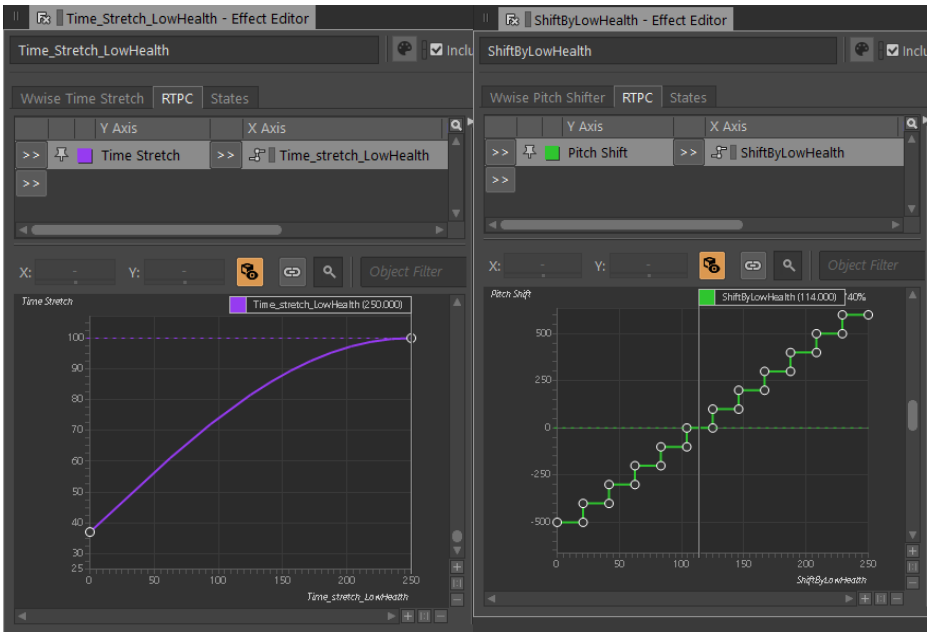


Fig. 18. Time Stretch and Pitch Shift in Wwise.

v. The player dies. Short arrangement/effect.

Method: *assign the Dead event in game to Dead game state.*

The *Dead* theme is played back when the *Dead game state* is called from the game.

vi. A modified version of Epic for the small ‘Winning’ level.

Method: *assigning the begin-play event—Winning level— to the corresponding Music Switch.*

The *Winning* theme is played back when the *Winning music switch group* is called from the game at ‘Winning’ level begin-play event.

vii. Affect current music during the “pause” screen.

Method: *assigning the Pause event in game to the corresponding GameState.*

The respective *Pause* theme is played back when the *Pause State* is called from the game. It uses the *music switch group* and *Tense State* to allow the corresponding tracks to sound when the *Pause* event is triggered.

viii. Create/modify music for the start menu.

Method: *assigning the Start Menu Level begin play event to the corresponding GameState.*

Menu from *GameState* group is triggered whenever the *Menu* screen is loaded in the game.

4.2.2.1. CBI limitations

Although not part of implementation observations, the most salient limitation of CBI is the amount of work needed in the composing stage to accomplish variety and adaptability through randomization and vertical/horizontal layering. In CBI, a composer could design a large number of short segments to increase granularity, which would resemble PAMG closer, but it would be a non-conventional, highly-time-consuming labor (see section 2.3.) For this experiment, I decided to cover the variety aspect by using longer-than-usual

segments (~2 min.) and the same tonality/meter/tempo (A minor/major, 4/4, 98 BPM) to allow easier musical overlapping and distinct cue-points for phrasing.

As the PAMG implementation setup is based on a four-theme interpolation algorithm, a similar approach needed to be found among Wwise tools. There, the switch/state based transition paradigm is limited to act between two themes or tracks (source and destination) allowing for an optional transition segment. A normal use would suggest to activate the themes' *playlist containers* through a *music switch container* event or a *RTPC* that controls gain assigned to distance in gameplay but the musical result was poor: either a deterministic crossfade for all tracks in a theme (theme switching) or overlapped themes in their own volume with unclear musical intention (RTPC mixing). For this reason, the method chosen was to trigger all the music segments through their wrapping *playlist containers*, and set each track to the *switch track* type to access their rules independently. The transition-rules system is designed to use either *playlist* or *music segment containers* as source and destination but it is limited on music tracks (for *switch tracks*, *new transition rule* and *source/destination* are unavailable), while *Exit Source transition values* are still available on *switch tracks* (*Next Beat*, *Next Bar*, *Next Grid*, or *Next Custom Cue*). Using different values in each track allows an instrument-layered and synchronized blend between <u>two</u> themes but not <u>four</u> (the main PAMG interpolation feature). In fact, the method used with *Tense* (switching *State* instead of *Switch*) yielded more sustained track layering and replacement possibilities.

In conclusion, the exercise of designing a CBI is more time consuming and yields less interpolation granularity than implementing PAMG for this particular game music. The complexity in CBI tends to be higher than PAMG since it involves designing, implementing, and testing a high number of transition rules between tracks. In PAMG on the other hand, transitions are implemented by variable value assignation, or instant value sent to parameter(s) or preset interpolation coordinates.

4.2.3. Gameplay Test

The test aims to find information leading to answer the research question:

Addressing gameplay experience, how does the present PAMG model design compare to pre-recorded-music CBI?

In general, the test aims to find if PAMG enhances gameplay experiences within a FPS (first person shooter), if the difference is negligible, or if the experience is better with CBI. Additionally, it attempts to find if and how participant background variables may affect that perception.

The test should gather information about the differences in gameplay experience between CBI using adaptive techniques (see section 2.3.), and PAMG. The range used for comparison is *noticeably better*, *marginally better*, or *no significant difference perceived*. More specifically, the test aims to find videogame players' perceptions of qualitative differences between PAMG music and CBI. Hypothetically, game genre, demographics,

and personal experience in music and/or videogames may play a role in that distinction. The test could find correlations between particular gaming backgrounds (e.g., shooter, strategy, platform, etc.), demographic representations (gender, age, education), or type of involvement with games and music (e.g., occasional to heavy gamers and music/audio consumers), and a preference for one music system over the other. CBI, as explained in previous sections, aims to use current tools for adaptive music implementation in Audiokinetic's Wwise, aiming to present a realistic contemporary videogame scenario.

4.2.3.1. Dependent variables:

The test aims to find if and how the players are able to discern and compare the musical influence of CBI and PAMG in their gameplay experiential qualities such as engagement, emotional reaction, immersion, and curiosity. The dependent variables are then the scores players assign as a comparative measure (*Clearly* more/better/superior, *slightly* more/better/superior, or *no significant difference*) to the two musical experiences provided for gameplay.

4.2.3.2. Independent variables:

- Gaming backgrounds: players may be used to particular videogame genres such as shooter, strategy, platform, fighting, puzzles, etc. The test aims to find any significance in music perception when players are experienced in FPS or not, and if there is any correlation with any other preference.

- Demographic representations: the test may find correlations between gender, age, education, and music preferences.
- Involvement with games: occasional to heavy gamers may have distinctive reactions to game music which reveals a correlation in the testing results.
- Involvement with music/audio: different levels of experience in music/audio may affect subjects' judgment/perception of game music quality. The test aims to find any significance in ear training—i.e., informal/formal training or practice in music or audio—in relation to their perception of game music.
- Music importance in videogames: subjects may have their own priorities regarding videogame music. The test may find correlations between the subject's particular view of music function in videogames and their music preference.

4.2.3.3. Groups:

Half population play *CBI first* while the other half play *PAMG first* to balance the consequence of experience accumulation. Players are told to compare the two experiences knowing that the music will be different in some way. They fill the demographics questionnaire, and then are asked to play between 10 to 20 minutes per game, being allowed to stop at any point in between. Then, they answer the gameplay questionnaire (see Appendix B: Questionnaires). CBI and PAMG use the same game.

4.2.4. Test results

4.2.4.1. Quantitative

In general, results suggest that PAMG had a slightly higher impact[49] on participants than CBI in the current test. The general (all comparative questions) vote counts obtained were:

- CBI clearly more/better/superior (CBI ++) = 35
- CBI slightly more/better/superior (CBI +) = 99
- No significant difference (CBI~PAMG) = 96
- PAMG slightly more/better/superior (PAMG +) = 128
- PAMG clearly more/better/superior (PAMG ++) = 56

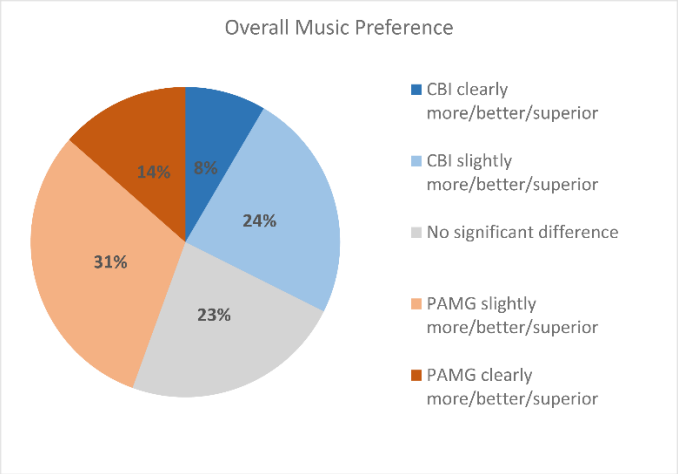


Fig. 19. Overall Music Preference.

[49] Impact here refers to any inclination in the responses towards *slightly* or *clearly more/better/superior* (either "+" or "++", as opposed to *equivalent* "~") in comparative questions between the two implementation methods in the categories of: *emotion production*, immersion/curiosity *fostering*, action-danger/mystery-suspense/adventure-curiosity/tension/epic *appropriateness* and *adaptability*, harmony/melody/percussion *game support*, responsiveness/transitions/variety/emotional transmission/gameplay positive influence/engagement *performance*.

The groups *CBI first* (population that played CBI first) and *PAMG firs*t (population that played PAMG first) displayed a particular trend shown in Fig. 20 and discussed in section 5. It suggests that both models had slightly better scoring when appearing in the second game or slightly lower scoring when appearing first. For example, "CBI slightly more/better/superior" goes from 16% in *CBI first* to 32% in *PAMG first*. Nevertheless, in both groups, "PAMG clearly more/better/superior" is larger than "CBI clearly more/better/superior" (15% vs. 8%, and 12% vs. 9% respectively).

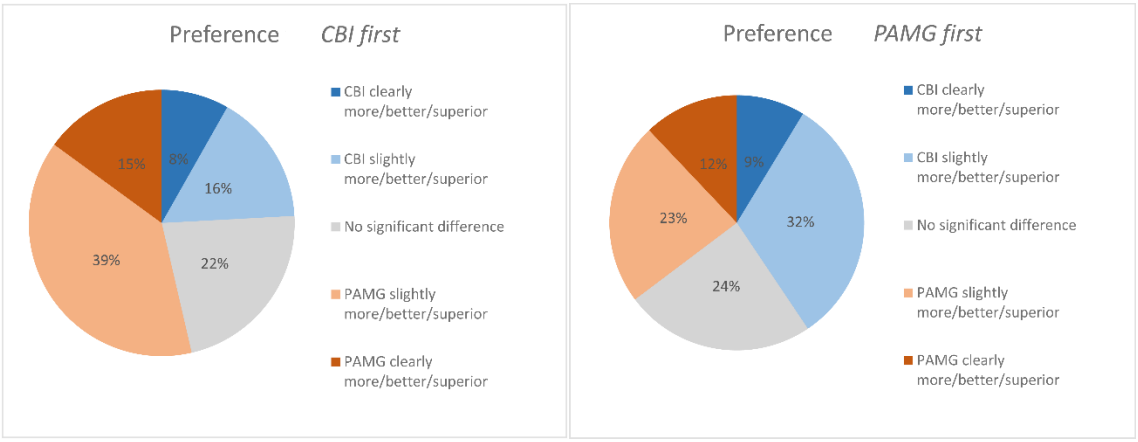


Fig. 20. Preference in both groups (*CBI first* and *PAMG first*).

An analysis of variance is performed in section 5 to find statistical significance in the resulting preferences and possible group interaction.

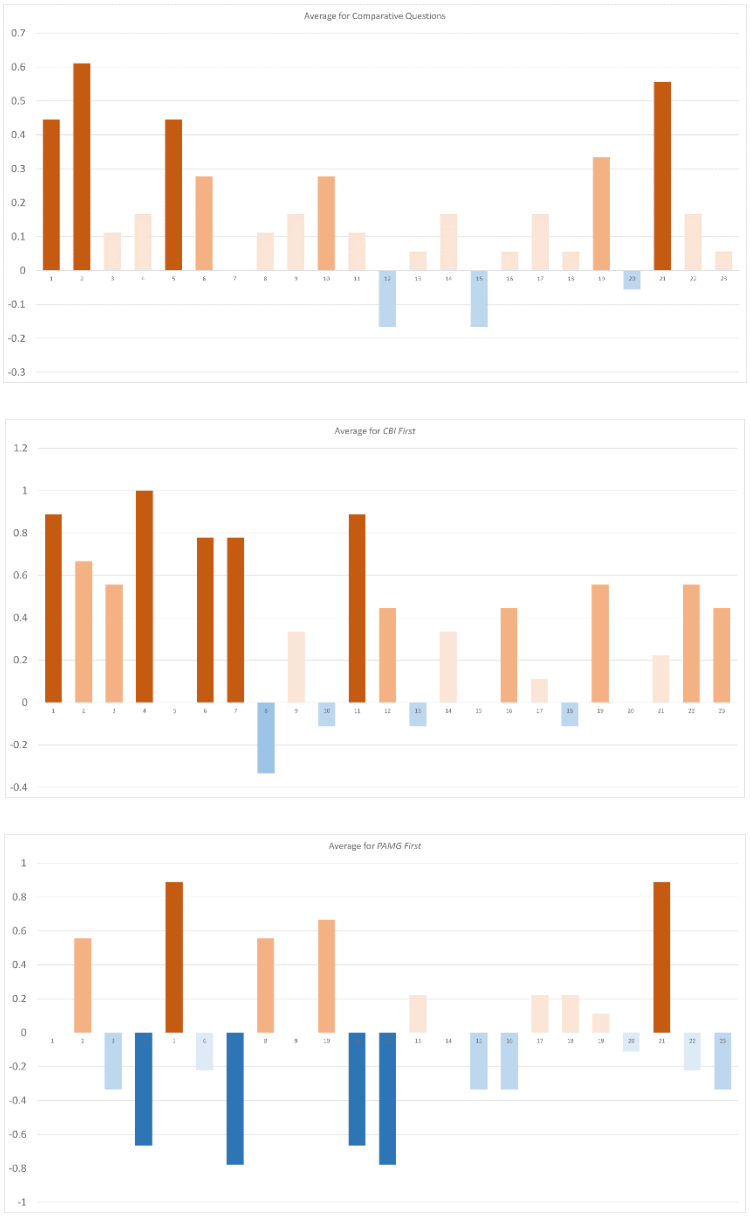


Fig. 21. Average scorings for all comparative questions.

The scorings have a value of -2 = CBI++, -1 = CBI+, 0 = CBI~PAMG, 1 = PAMG+, and 2 = PAMG++ to see easily in the *y* axis the CBI preference average as negative (blue) and PAMG preference average as positive (red). **Top**: Average for comparative questions in general. **Middle**: Average for *CBI first*. **Bottom**: average for *PAMG first*. Positive numbers (red) show the rate of preference of PAMG while negative numbers (blue) rate preference for CBI, darker for strong and lighter for weak results.

The average score for comparative questions suggests a preference for PAMG in the current experiment (Fig. 21 top). Then, detailed results the two sequences of implementation (Fig. 21 middle *Average for CBI-first* and bottom *Average for PAMG-first*) suggest that the preference is skewed to the second musical experience, whether CBI or PAMG. Bar 1 is the question "In your opinion, which game had BETTER MUSIC overall?" in which the middle possibility (music A ≈ music B) was not present. Most bars show a change in the direction of the music heard in the second game **except**:

- 5, *Compare the music stylistic appropriateness in the following game conditions: Action-danger*. The average score grows positively (to PAMG in *PAMG first*) by 0.89 points.
- 8, *Compare the music stylistic appropriateness in the following game conditions: Tension-suspense, eager, edgy, uneasy.* The average score grows positively (to PAMG in *PAMG first*) by 0.89 points.
- 10, *Compare the music support to game situations. In which game did the music ADAPT using a deeper range of possibilities for the following situations: Action-danger.* The average score grows positively (to PAMG in *PAMG first*) by 0.78 points.
- 13, *Compare the music support to game situations. In which game did the music ADAPT using a deeper range of possibilities for the following situations: Tension-suspense, eager, edgy, uneasy*. The average score grows positively (to PAMG in *PAMG first*) by 0.33 points.

- 17, *In which game did the selected musical aspect –percussion– support better any game circumstance in general?* The average score changed positively (to PAMG in *PAMG first*) by 0.11 points.
- 18, *In which game the following music features performed better: Responsiveness to game events and conditions*. The average score changed positively (to PAMG in *PAMG first*) by 0.33 points.
- 21. *In which game the following music features performed better: Ability to transmit emotional information*. The average score changed positively (to PAMG in *PAMG first*) by 0.66 points.

Bars 15 (*In which game did the selected musical aspect –melody– support better any game circumstance in general?*) and 20 (*In which game the following music features performed better: Variety or non-repetitive*) grew from tie towards CBI slightly suggesting that they were deemed poorly in PAMG in both exposures.

Bars 4 (*In which game did you feel that the music got you more curious about the game possibilities*), 7 (*Compare the music stylistic appropriateness in the following game conditions: Adventure-curiosity*), 11 (*In which game did the music ADAPT using a deeper range of possibilities for the following situations: Mystery-suspense*), and 12 (*In which game did the music ADAPT using a deeper range of possibilities for the following situations: Adventure-curiosity*) show the largest change between groups (in the direction of the second music) with a delta of 1.66, 1.55, 1.55, and 1.22 respectively.

4.2.4.2. Qualitative

Players were asked: *Write down in your own words what is your perception of the music in the FIRST (and SECOND) game*. This attempts a deeper understanding of their impressions. Comments on PAMG as second game show allusions to elaboration, instrument variety and distinction, and a couple annotations about being relaxing, mostly in contrast with CBI. Curiously, when commenting on PAMG as a first experience, the remarks tend to highlight a slight dramatic exaggeration, mentioning aspects such as mysteriousness, adrenaline, intensity, tension, scariness, eeriness, and even being faster than the other. CBI comments as a first game refer contrasting issues like being thinner, generic, softer (although there is a mention to louder drums), but also action oriented, while as the second game, support to action and adventure was highlighted as encouraging.

Some examples are:

A heavy gamer that preferred slightly PAMG in their second game, wrote about it:

> "The second game's music had more high pitched sort of piano notes and this kind of relaxed me more. This felt more explorative and less of the action aspect. This music was somewhat calmer than the other music for me."

And about the CBI in the first game:

> "The music in the first game was very intense and the drums to me were very prominent. This made the game feel very action-y and kind of gets the heart racing sort of music."

A heavy gamer that preferred PAMG slightly in the first game wrote about it:

> "The first game's music seemed more tense? Definitely when the other robots were chasing me it felt scarier. I felt my heart beat going up far more in the first than the

second, but that might just be because I already knew what was up with the robots the second time."

And about CBI in the second game:

> "The music felt more epic and less suspenseful here. Also only during this play through did I realize that the music changed when you were being chased by robots. The soundtrack becomes a lot more chill when enemies aren't chasing you."

A heavy gamer that preferred CBI slightly, who scored high in musical experience, and had PAMG in the first game wrote about it:

> "The music started out mysterious and got more intense depending on the area I was in."

And about CBI in the second game:

> "Same as first, music started out mysterious and got more intense depending on the area I was in. I noticed that towards the end near the cube, this version sounded noticeably more triumphant."

The participant that scored the highest in musical experience who marked PAMG music as "considerably better" in the second game wrote:

> "Beautifully written and fitting to the setting. Less maximalist in the sense that it fitted with what was going on-- ambience fitting more of a mystery to explore rather than something to fight against. Music also changed slightly based on location. Flashing lights on the mountain section of the game correlated to the percussive elements in the music, which I thought was interesting and added to the ambience of the game's environment. The music building up to be more and more full the closer that the player got to the objective point also helped to make the game more immersive and the ending of the mission more satisfying and fulfilling."

And about CBI in the first game:

> "Beautifully written but not fitting to the setting or events of the game other than when I was faced with the enemy characters and lava pit. Intense and felt more suited for a boss battle or combat based game while the game itself was more of an exploration game."

An experienced gamer that considered CBI slightly better, and had it in the first game wrote about it:

> "Music A (1st) was the one that really created an immersive experience for moments of risk, danger, enemies approaching... The tension on the player while playing is what defines this music on my opinion. This is the one that I think is the best for this specific videogame."

And about PAMG in the second game:

> "Music B (2nd) was the one that while was not contributing as much as the other one to the immersion experience, helped me to focus better and more on the game and pass it at the end. This is the one that, while not being the best for this game, helped more to focus on the gaming experience and eventually pass the game."

In the relationship between preference and music/gaming background, although some values suggest a relationship between the participant's background and their response, only further testing may establish such correlation.

Five people finished the game listening to CBI in the first game, six finished the game with PAMG in the first game, six finished the game with CBI in the second game, and seven finished the game with PAMG in the second game.

4.2.4.3. Demographics

- Age distribution: 8 participants between 18-21 years old, 2 between 22-30, and 8 between 31-50. Total: 18.
- Gender: 6 women, 10 man, 2 n/a. Same count for Sex: 6 female, 10 male, 2 n/a.
- Gameplay Time in a week: 20+ hours 1 participant, 16-20 hours 2 participants, 11-15 hours 2 participants, 6-10 hours 5 participants, 1-5 hours 2 participants, less than 1 hour 5 participants, and don't play videogames 1.

- Experience on First Person Shooter*: I have never played that kind of games:* 5 participants. *I have played them, but is not my favorite:* 6 participants. *I play them regularly, first and/or third person shooter:* 6 participants. *The shooter is my favorite game category:* 1 participant.

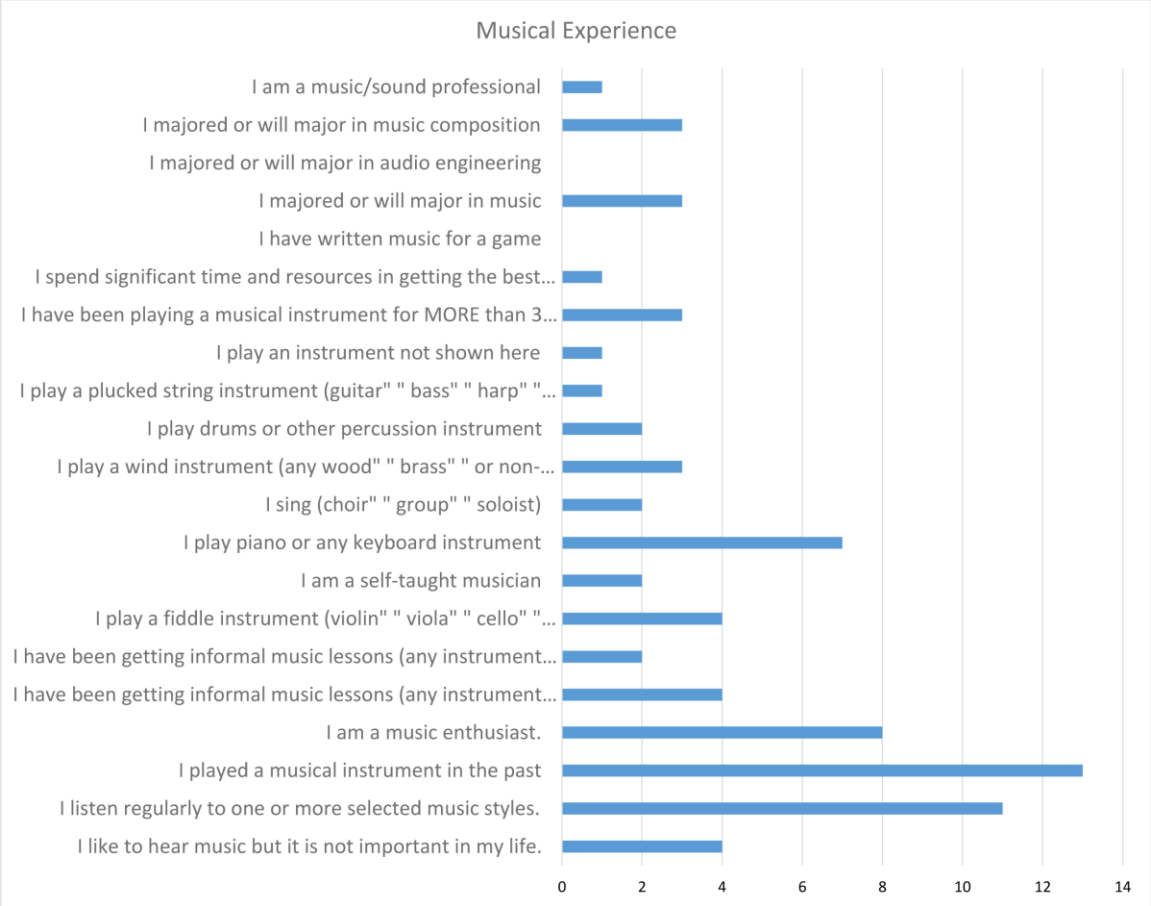


Fig. 22. Musical experience demographics

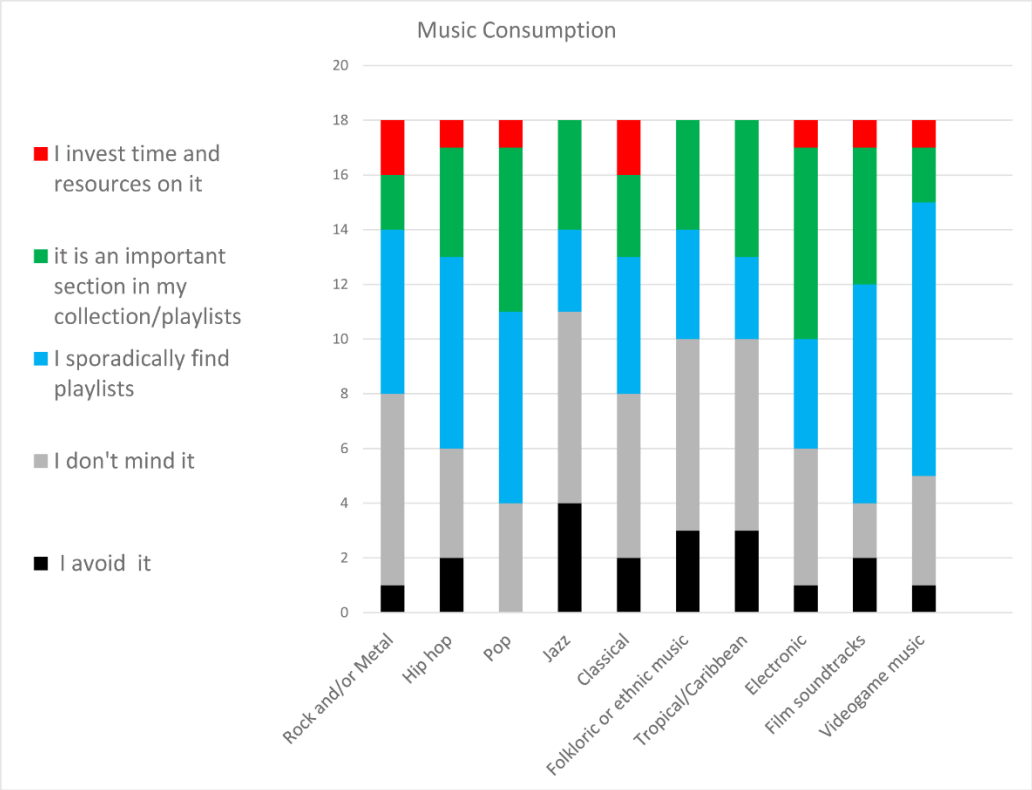


Fig. 23. Music Consumption demographics

- For the ranking assigned to game music features, the following annotations depict the higher and lower ranked: *Support to gameplay mood and rhythm* got rank-1 (most important) 8 times, rank-2 5 times, and rank-3 2 times. *Transmission of emotional content* rank-2 7 times, rank-3 4 times, and rank-1 3 times. *Motivic variety (less repetition)* got rank-8 (least important) 8 times, rank-7 5 times, and rank-6 2 times. *Diversity (more musical themes)* got rank-7 9 times, rank-6 4 times, and rank-5 2 times. *Contribution to immersion (non-distracting)* got rank-8 (least important) 6 times, rank-5 4 times, and rank-4 3 times.

5. Conclusions, analysis, future directions and discussion

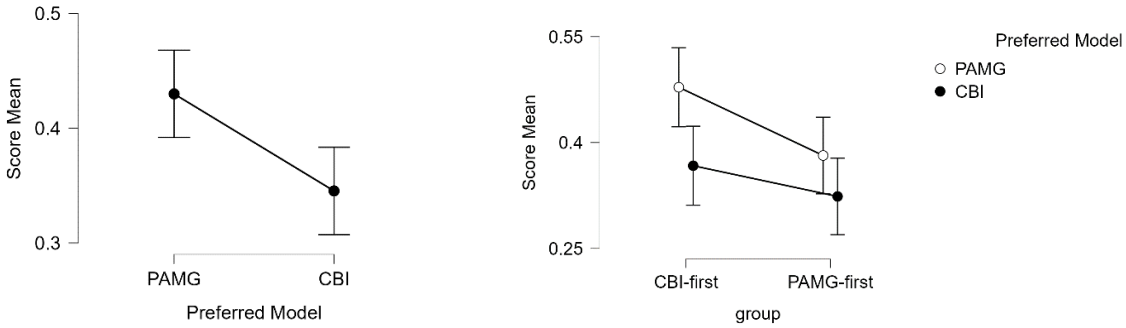


Fig. 24. Preference Scores Mean

The two graphs show the score means obtained for each model for the whole test in the left and for each model by group in right. Each mark shows the standard error interval. The scores are 1 point for *slightly more/better/superior* and 2 points for *clearly more/better/superior.*

PAMG had a higher average score in general than CBI (Fig. 24. Left) in the current experiment and in each group (Fig. 24. Right). Also, a slightly skewed preference towards the second game music is suggested by Fig. 21. These two observations require an analysis of variance (ANOVA) to establish statistical significance and constitute the two main hypotheses presented by the experiment: H_1: There is a significant preference for one model over the other (CBI or PAMG), H_2: There is an influence of the group (CBI-first and PAMG-first) in the preference of a model.

Taking the alpha value as .05, the analysis of variance indicated that there is a non-significant preference of a model in general $F(1, 17) = 2.46$, $p = .13$, in the direction of PAMG. Additionally, the results revealed a non-significant difference in preference of a model between the two groups (*CBI first* and *PAMG first*) showing between-subjects effects: $F(1, 16) = 1.82$, $p = .19$, and within-subjects effects: $F(1, 16) = 2.35$, $p = .14$.

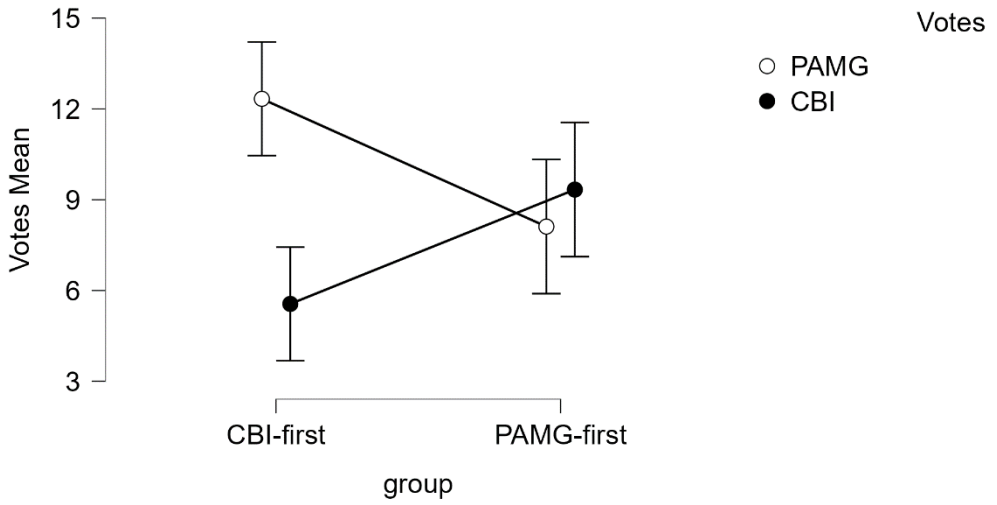


Fig. 25. Votes' Count and Groups

The mean of votes' count per participant that includes all the comparative questions is plotted for each model and each group. This visualizes better the suggestion of a skewed preference for a second music model than the score average in Fig. 24.

An analysis of variance performed in general votes' count in favor of PAMG or CBI per group (Fig. 25) indicated a marginally significant effect of group interaction with PAMG/CBI votes $F(1,16) = 3.79$, $p = .06$ that partially explains the observation of a skewed preference towards the second model.

5.1. Results by question

The following results show the behavior of votes in questions that revealed significant effects ($p < .05$) either within subjects or between subjects indicting a preference of a model or a group interaction:

Using votes' count, the questions *In which game did you feel that the music: provoked emotions in you* ($F(1,16) = 5.44$, $p = 0.03$) and *In which game the following music features performed better: Ability to transmit emotional information* ($F(1,16) = 6.91$, $p = 0.01$), the results indicated a preference of PAMG over CBI independently of the group (see Fig. 26 and Appendix F, Table 1). This suggests that PAMG was noticeably able to evoke and transmit emotions better than CBI.

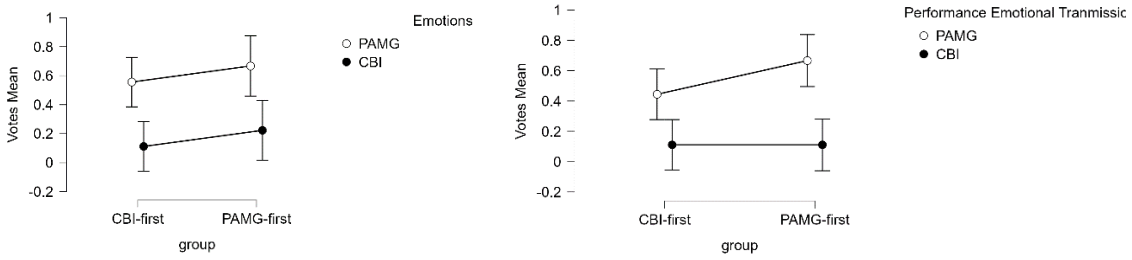


Fig. 26. Questions with significant within-subjects effects.

In the group of questions related to support (see Appendix E, Gameplay Questionaire, question 6b) the analysis found significant between-subject effects ($F(1, 16) = 4.5$, $p = 0.05$), suggesting a difference in how participants of different groups perceive music support by music features. In particular, *harmony* showed a greater difference for each group (see Appendix F, Table 2). Other music features were statistically non-significant.

In Fig. 27, the trend shows mainly an inferior votes' mean for PAMG when is listened first while CBI seems steady in both groups.

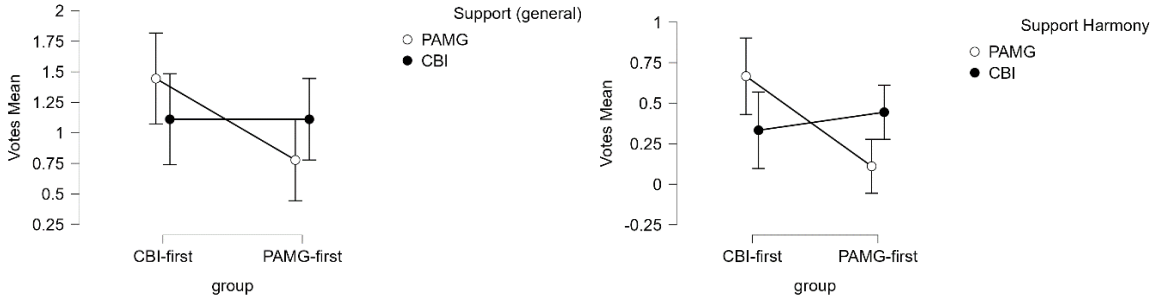


Fig. 27. Support of music features to the game in general

In several questions, the results also reveal significant within-subjects effects for group interaction (see Appendix F, Table 3). In those questions, H_2 is adopted, supporting the trend of participants favoring the second music model.

For the relationship between participants' background and music implementation preferences, the results suggest some correlations within the confines of the sample size, but the statistical value is non-significant. For example, data suggests a correlation of *Music consumption Folkloric-ethnic* to *Appropriateness adventure-curiosity* in favor of PAMG (see 5.1.2.), but the sample is too small to draw conclusions. This invites future research into music background as an independent variable for generative music perception effects. Additionally, current results propose further investigations of the

effects of generative music under different stress conditions and as incidental background.

5.2. Data interpretation

Although the statistical results in general are non-significant to conclusively answer the research questions, gathered data shows a number of trends in favor of PAMG. In this section, an analysis of the combined results between data and the carried qualitative comparison are considered.

5.2.1. Perception of Interpolation vs. Layering

As the tilt towards the second music may be explained by the enlarged focus after the priming effect[50] recedes (see 5.2.4. below), players' descriptions show some depth into the phenomenon. The tendency of perceiving a larger instrumentation, elaboration, and meditative segments are correlated mostly to PAMG in the second game, which supports the enlarged attention idea: once players had a confident vision of game mechanics and possibly had accomplished the goal, they dedicated more perception focus to the background music. Conversely, if they had PAMG during priming, the effect perceived was more in the side of unnerving, exaggerated, faster, intense, scary, and even loud. As a contrast CBI was perceived as action oriented and less 'unpredictable' which may have put some players in a confident mode. The qualitative difference in assessment also

[50] Joe Cutting, Priming in Video Games | Digital Creativity Labs. Retrieved May 7, 2023, from https://digitalcreativity.ac.uk/projects/priming-video-games.html#:~:text=Priming%20is%20often%20linked%20to,teach%20and%20encourage%20good%20behaviour. Accessed 5/8/2023.

points to another effect produced possibly by the layered transitions in recorded themes vs. generative interpolated in PAMG:

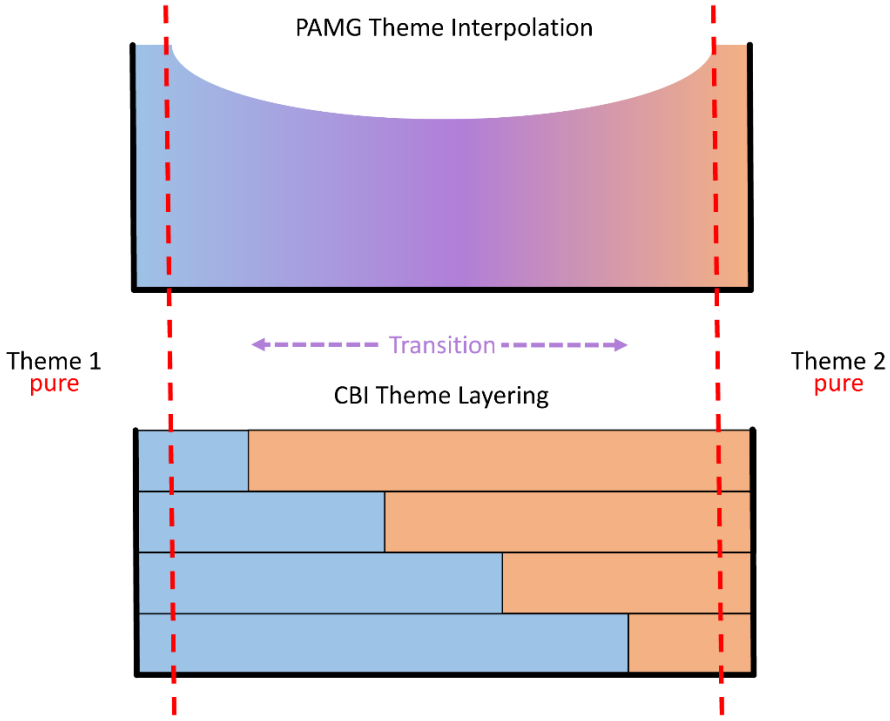


Fig. 28. Theme Interpolation vs. Layering.

In the top, a representation of interpolating two themes (PAMG) and in the bottom transition using layering methods (CBI). The themes in their most pure version can be listened closer to the sides. In the middle, the result of PAMG interpolation is a hybrid theme with feature values averaged in between the themes, while in the CBI layered method, it shares tracks of both themes.

For CBI, themes were recorded from the pure version coming from PAMG presets, which are the corners of the interpolators. The result is that, while PAMG interpolation in the middle produces hybrid material that ultimately is not completely from either theme, in CBI layering the theme's 'purest' material shows up during the transition through track playback.

Layered transitions would continue the ‘pure’ ideas during transition despite being mixed with compatible music material from the other theme, while interpolated themes produce new music in the middle. Although this new resulting material contains complete musical ideas, it has a mixed style that averages *four* values per parameter (four themes) making it more diverse but less consistent. This may be the reason for the observed comments that found some depth, even calm in PAMG after priming, or a wealth of overstated emotions during priming. As a comparison, musical ideas that repeat and overlap using instrument tracks in CBI offer a binary direction—only two themes instead of four—creating a solid and reliable package that is also common in videogame implementation.

This observation attests a higher depth in music variety in PAMG interpolation that is perceived more emotionally during priming, and as aesthetic wealth if attention grows after priming—although this requires confirmation through further experimentation. In general, the experiment suggests a significant difference in how a gamer perceives music depending on stress levels and paradigmatic music implementation within game categories.

5.2.2. Background correlations

Most of the higher correlation indexes seem to associate categories in participants’ background with preferences of PAMG. Participants with higher experience in music tend to prefer PAMG, but the statistical value is non-significant.

Among these results, music *enthusiasts* as a category had more participants inclined to rate better CBI in *appropriateness: action-danger*, *appropriateness: tension-suspense*, *and adaptability: tension-suspense*, while rating better PAMG in *adaptability: adventure-curiosity*.

Four *consumption* categories showed correlation within the sample to PAMG preference: *Folkloric* to *adaptability* and also *appropriateness: adventure-curiosity*, *electronic* to *appropriateness: adventure-curiosity*, and tropical to *appropriateness: adventure-curiosity*.

These results invite further inclusion of the selected music background categories in upcoming tests.

5.3. Test issues and biases

The test unveiled several issues that inform a future full-fledged study. Among them, the most notable are insufficient gameplay time to test exposure to repetition, the reduced number of samples, the variability in gaming background, and the effect of priming and experience in the game in question.

5.3.1. Liking as a function of repetition.

The gameplay time is too short to ensure reflecting the influence of liking vs. repeated exposure (Schellenberg et al., 2008, Madison & Schiölde, 2017). In fact, the number of times players are exposed to a musical idea in the current experiment—i.e., between 4 and 10—may cover only the *upward* section of liking described as familiarity that also

appears to be longer on incidental (i.e., current test) than in focused music tests. This may explain why a significant number of players preferred the ‘repeating’ music over the ‘non-repeating’. Hence, the quantity of exposures for the current experiment would only show a decrease on liking when there is significant gameplay time, such as accruing at least 32 repetitions of a theme—about 80 minutes of gameplay distributed in a game session with as few and shorter breaks in between as possible.

5.3.2. Reduced amount of samples

The reduced number of samples and especially the low statistical significance tend to lead to inconclusive results. Not only is a larger sample necessary but also pre-classification of subject demographics potentially yields more reliable results since groups can distinctly relate to the test category.

5.3.3. Background differences

The large differences in music and/or gaming experience played a role in music assessment. While some players successfully completed the game several minutes before the end of the first session, some did not reach the Epic theme that appears when players are in proximity to the goal, producing bias in their assessment. A way to mitigate this background classification is either to adjust the game difficulty configuration, or to limit the experiment to subjects with a particular gaming experience level. This pre-classification may also be applied to music consumption and preferences, either to

capture a familiar gaming experience or to find traces of engagement across gaming interests/skills.

5.3.4. Game priming and music focus

In all the experiences, gamers take a learning curve that depends on their gaming background and more specifically in FPS experience. They adjust to a new experience starting with a higher focus on objectives and mechanics at gameplay, and then, gradually their attention is able to include more elements such as the background music. In other words, music assessment reliability goes from low to high. It is one of the most salient results: players tend to prefer the music in the second gameplay experience which is consistent with the adjustment period. A method for diminishing this priming effect could be adding a session with no music in the beginning to allow the player to achieve technical acquaintance before the actual music assessment. Similarly, the capacity of a player to retain their impressions until the end of two sessions showed some bias.

5.3.5. Likert-like scales

The Likert-like scales used prominently in the questionnaire produce some bias discussed by Brill (2008). Among them, *central tendency bias*, where participants tend to choose the neutral comparative option to avoid the extremes that would denounce a higher knowledge or understanding of the subject. That is in some way related to the inclination to select answers that portray themselves in a better light, or that fulfill an expectation (s*ocial desirability* and *acquiescence bias* respectively).

5.3.6. Ambiguous instructions

A more robust language and comprehensive instructions may improve biases caused by inaccurate responses in the current experiment:

- A participant with a professional music title and a professorship scored less than a student in music experience because they did not mark all the possible fields. The intention behind this design was to add up the overall music experience score. It was assumed that a participant that marked professional performer and a composer would likely mark in addition any formal music education, and also may considered themselves a music enthusiast. A professional musician would select more marks but, in at least one case, they may have assumed that lower experience fields would be implicit.
- Relating to memory and instructions, a participant made comments about the game but not the music in the written portion, and another mentioned forgetting to pay attention to music in the first test section.

5.4. PAMG observed limitations

5.4.1. Specificity

Although the idea is to produce a model that is versatile enough to be ported from a game to another with minimum setup changes, the resulting behavior centers around a particular style of western tonal orchestral underscoring music. The current orchestration and instrument family management makes style shifting (with instruments) an additional task. In this process, several algorithms need modifications and new ones need to be

incorporated. For example, to produce idiomatic electronic music—besides instrument setup—some modifications include assigning a narrower range to bass, allow less multi-level rhythmic occurrences in kick-snare lines, and assigning other lines to sound effects, to mention a few. Additionally, electronic music often employs continuous changes in their aural properties as a stylistic feature, which means that a capable model should be able to generate them in concurrence with discrete events and phrasing. It is possible to argue that many game styles can benefit from the system's current properties, but changing instrumentation—and style—requires thorough testing and possibly deeper modifications than anticipated.

5.4.2. Phrasing vs. real-time interaction

The current model was designed with the idea of incorporating continuous controllers to a number of music generation parameters. In most cases, they act directly on algorithmic variables to produce real-time changes within the music stream. However, this produces an issue with phrasing. Deployment of musical ideas requires a discrete amount of time that is often tied to rhythmic properties such as meter and tempo. An occurrence in gameplay does not necessarily conform to that rhythmic grid, which means that any associated music variation may look out of place, non-consequential, or discontinuous if the implementation is instantaneous. Besides some methods for 'foreseeing the future' (Plut & Pasquier, 2020), it is possible to apply a certain degree of quantization to variations based on several grid levels (beat, meter, phrase). When the musical idea belongs to a phrase, i.e., a progression, the segment may be too long to offer the

responsiveness for certain game events. Some parameters already have implemented a quantization method that only permits changes at the start of a phrase, or the beginning of a measure (e.g., arpeggio on/off, arpeggio shape change, loop size change, and others). The common property of these parameters is that they make a noticeable change in the identity of a musical idea which may be broken if it does not resolve or arrive at the end of the phrase. The type of parameter that can perform an absolute real-time change, and the methods to apply changes without affecting negatively the phrase construction, are still under evaluation.

5.4.3. High vs. low resolution control for musical changes

Although the original idea of 'smoothing corners' for all musical transitions is useful—and studies as Cutajar (2020) assessed as successful—, in practice music is a discrete form of time domains, just like anything digital. Many ranges are smaller than the 7-bit values provided by MIDI (e.g., usable pitches, melodic intervals, transitions, number of notes, instruments, etc.) and their ratios end up sectionalizing the full range. For example, using the full range of a 128-positions slider for the 12-positions tension parameter and the 9-positions progression complexity parameter ends up having many instances in which changing the slider does not produce an actual music change. On the other hand, when the interpolation mechanism modifies many parameters in a range that includes a change for them, the result may be a steep variation in a small period of time that affects the continuity of the musical idea by truncating or redirecting a phrase in an odd time. In a number of cases, this was mitigated by employing the phrasing-quantization method

(allowing changes only in significant meter positions), and in others implementing a gradual change (e.g., for tempo changes). Nevertheless, besides quantizing transition changes, controlling the amount of discrete changes when several parameters move may regulate the music flow and its variance more efficiently for further development.

5.4.4. Issues of the insertion of PAMG as a product in game development

Although a discussion of a PAMG path into game design was laid down in section 2.4., there are many questions that come up with the prospect of addition of new procedures or change to conventional paradigms. Among them:

- How does PAMG integrate into game authoring systems?
- How does PAMG create and manage a new format of interactive generative music?
- How does PAMG handle different game (and music) genres?
- How does PAMG leverage automation?
- How does PAMG create value by creating new products? What kind of products and what audience may be interested in acquire them?

Finding the answers to those questions generates research projects. Additionally, in consultations with game developers and other industry representatives of music solutions for games[51], two main issues for PAMG were spotted:

[51] Besides the interviews with Aleksandar Zacevic and Guy Somberg (section 2.6), in the frame of GDC2023, a series of interviews with Louis-Etienne Payer of Triptique Audio, Brendan Votano of Animatica Studios, Elliot Callighan of Unlock Audio, Bernard Francois of Preview Labs, Damian Kastbauer and Simon Pressey of Audiokinetic, and Brett Patterson of FMOD gave a thorough insight of

- How does PAMG integrate a composer aesthetic? Or, how does PAMG guarantee continuity between its output and the composer's?
- How does PAMG handle samples and timbre?

The first, reveals a necessity for a method of *addition* instead of *change*. Most game composers have setups with demonstrated efficiency that include instrument setup, format exchange flow, and their favorite custom software for creation, recording, editing, mixing, and delivering. A method that may help to include PAMG in game underscoring without replacing a composer can be a system of analysis that scans and translates statistics on implemented music in the game and outputs parameters and possible progressions paths for PAMG music. It may use machine learning or Markov chain analysis to produce stylistic similarity through parameter setup. Its primary input can be symbolic, possibly in MIDI format to extend the possibilities of timbre (e.g., a signature melody can be assigned to instruments not used by the composer).

The second, as discussed in section 2.6., is a licensing issue. It may be possible to generate timbre through AI scanning in a similar way timbre is generated from a sound sample (Wang et al., 2023), but the sample creator may dispute such employment in the licensing contract. Nevertheless, other idiomatic acoustic textures used often by composers would need a place in the music generation, either as sample management or re-synthesis.

---

competitive, logistic, and marketable aspects of a prospective commercial product in the game development pipeline.

In general, there is still also a possibility of a new practice for a composer using PAMG as an instrument, in which musical ideas are worked and set using the algorithm itself. This would also propose a musical product that is not a static audio but a setup for adaptive generative music with clear authorship assignation.

5.5. Notes on current PAMG game implementation and future game testing

The model is able to conduct independent interpolations by each one of the agents, i.e., by the three XY pads. However, currently a single controller is implemented in the game to place the 2D cursor in the same interpolation location for all agents. Using the PAMG as a performance instrument[52], showed richer developments when the three agents act more independently. For example, while MA and HA are close to Mystery, PA can roam through Tense and Epic producing other kinds of progressions. This division has the potential of deeply diversify the musical output. However, the algorithm to merge/unmerge XYpads locations, which preserves the stylistic properties, has not been designed. In a sense, it has not been needed since the style consistency is supported by single parameter variation which promotes progressions more than the interpolation of moods.

Although many scenarios beyond those currently tested can be subject to trial in an open-world, action-adventure game like the Trial Game, other formats and genres may benefit from generative adaptive music, and also reveal its limitations:

[52] Appendix E contains music pieces using PAMG as an instrument.

- In action sections, sound design events such as explosions, battles, gun fire, and voices tend to occupy most of the aural space. These kind of sounds often are strong and dense thus masking music and adding noise to the resulting auditory space. Those scenarios promote a simpler and in a sense, more formulaic approach to musical support. For example, in the current game the music variation to depict the tension on NPC chase proved to work better by muting MA and preventing arpeggios from HA to allow only chord onsets. It still may sound noisy if many percussion onsets are heard at the same time the player character shoots frequently. This kind of considerations also link music implementation design to sound design. Action might be a component of other genres outside of First/Third person Shooters and sound density also would need to be considered. The following recommendations in other genres use information developed by Sweet (2015) and Aristopoulos (2023).

- **Sandbox games** share the open world features of The Trial game, but being open ended will put the emphasis on resulting music diversity: music material, spectral/timbral distribution, use of register, and contextual support. The player may listen to a large amount of music in a single session causing ear fatigue if a pitch, pitch range, instrument, and/or timbre is employed extensively and continuously. Similarly, conspicuous music developments with high density and complexity may divert the attention from the game to the aural events. As in **Adventure games**, the music should allow the player to be curious. To achieve it, drama and tension should

be employed sparsely and a more predictable musical structure with a touch of enthusiasm and fantasy may help. The premise of these games is to allow the player to exhaustively and limitlessly roam the world. The music should enable and encourage this with some degree of foreseeable/refreshing variation, light progressions, consonance and rewarding happenings linked to gameplay.

On the other hand, although with caution, **Survival and horror** games require suspense elaborated on unpredictability and slowness, tension—using a fluctuating mixture of consonance, dissonance and silence—and drama employed sporadically on emotional storytelling or cue lines.

- **Strategy games**, whether real-time or turn-based, also can benefit from PAMG music variance and progressive transformations to map views and eventual transitions respectively. Map views in many occasions constitute a large section of gameplay time and also are developed in a slow pace. Localized events break the monotony and introduce other game modes/views. Music progressions may be used as information to the player of an impending event, as support for an undergoing circumstance, and/or as a satisfaction reward for an accomplishment, all within the same musical theme.
- **MOBA** (multi-player online battle arena) games pose the problem of the multi-lateral point of view. Games present multi-linear stories, but in this case they also present multi-linear, simultaneous points of view. Clearly the possibility of having one music line created by each gameplay view point is the most plausible. However,

merging storytelling in many situations would be desirable like in team play paradigms. Interestingly, it still should keep a degree of individuality but having the possibility of integration if needed.

- **RPG** (Role-playing games) have a wider range of mechanics, aesthetic, game pace, and storytelling than other genres, leaving the common property of assuming a *role* to structure a range of playing possibilities that seem more open than those in other game categories. Crucially, the player controls a character that uses experience to evolve. This heterogeneity makes common the inclusion of sections and mechanics found in other genres such as strategy, openness, survival, combat, multiplayer and more. Depending on the game section dynamics, many RPG games include a great portion of gameplay that usually occurs in a neutral environment, with a level of surrounding mystery. This is often depicted with slow and inconspicuous music (or diegetic music), occasionally building a stylistic texture that supports the environmental/circumstantial aesthetic. From this basic stable environment, there is a large range of possible developments caused by enemies, informing characters, world transitions, findings, etc. Similar to strategy games, it tests music variance in the main gameplay section, but also would show how the PAMG is able to develop stylistic branches and become dramatic if needed. Interpolation between branches also should be tested to avoid apparent lack of direction.

- **Simulation and Sports** is a varied and large game paradigm. It encompasses simulations of a gamified business, city building, fighting, driving, piloting, in addition to sports. The main environment can vary widely, and the premise of achieving a simulation experience places a contextual framework on the auditory space. For instance, this comprises geographic and age simulations that link to historic stylistic music textures, including present cultural locations. In such games a refreshing variety is appreciated since the gameplay modes or environments may stay unchanged for long periods of time. In some games, the player has been granted a type of control over the music similar to controlling a radio, being able to change radically the music style or turning it off as in real life—although is not exclusive to simulation games. While a similar effect is possible with the PAMG, mainstream music or pre-recorded track playback are not the core of its functionality. To achieve a contrasting style variety, it needs to be versatile enough to handle a wider assortment of instruments and be tested on a range of mainstream styles' idiomatic algorithmic developments. For example, handling electronic music (e.g., EDM for a commercial illustration), it is possible to test if the algorithms currently can use a parameter setup that aligns with the common stylistic features expected. In the current version, the generator does motives and phrases that depart from the iterative nature of EDM, although repetition is within the capabilities. As a core feature in EDM, gradual changes in sound texture achieved by effects, filters, modulation, and other synthesis-related transformations would require a specialized algorithm capable of generating continuous control lines.

- **Puzzlers and party games** may use also PAMG's variation capabilities in a similar way map modes work in strategy games. There may be a higher need of variation and instant reaction in conjunction with sound design to support rewarding events, for example. The density and drama, however, may need to be kept low to avoid overwhelming the player. Small instantaneous changes in register, curious contour shapes, and assigned patterns to game achievements may be useful. As in **Platformer** games, chip-tune sound development and timbre manipulation (controller line generation algorithms) are desired capabilities to work on. For these game genres, variation/repetition balance requires particular attention. In a platformer, for example, since it includes retro aesthetics both in timbre and music structures, catchy melodies with low or no dynamic range and a careful use of repetition are desirable. The possibility of variance and progression within that framework is the real advantage PAMG can bring over conventional music implementation in these game genres.

5.6. Future use of Machine Learning

Not employing statistical extraction from music corpora is a conscious decision for the current model. The reasons are basically the difficulty in developing a system that affects the parameters progressively, and the ethical and legal issues mentioned in the final discussion of this text. However, using machine learning, is a real possibility that can help in several ways. Pre-trained models can provide a richer range of stylistic features,

especially using rhythmic/motivic interpolation. Use of stylistic embeddings may direct idiomatic progressions to transition between simple and complex. Each agent could be a result of independent training and react to each other's output to enrich variance. Within the current system, trained models using parameters' data may provide gesture classification tied to game conditions, player behavior, and discrete events in a more independent manner that the current model does. Regressive models may offer non-linear interpolation possibilities to establish any number of localized style assignations as opposed to a flat 2D area.

5.7. Accessibility

When the model enables users to produce material for their own consumption the music that comes into existence may grow exponentially. Also, with customization possibilities, users will start to relate closely to the output, to the point in which there is an aesthetic expectation fulfilled by the model—like an instrument. This means that with time, automated music models will enable users to make music more efficiently than if they used other tools like instruments, or traditional production software, without the limitation of skill learning. As an example outside AI, many have taken a plunge into software solutions like Garage Band whose intuitive generative tools allowed them to produce finalized music pieces without formal music/music-production knowledge. PAMG also can act as a multiplier of musical output for a music creator and an enabler for an empirical music enthusiast.

5.8. Discussion: The use of automated music tools outside game development and the repercussions of its commercialization in composers' labor

In current music production pipelines, composers are often requested to perform both style replication and quick production of convincing performance sequences ready for publishing. As those tasks involve largely mechanical and technical procedures, automated-music algorithms emerge as a potentially efficient solution. In film and video for example, when montage specificity achieved by temp clips or stock music leads directors to request equivalents, construction and style possibilities tend to be limited to the reference's musical features. This re-elaboration of music structures using genre constraints has occurred often in commercial and popular fields when musicians experiment through the inspiration of a singular piece or author. Analogously, some algorithms now are able to perform style replication using parameters or caption. Private research funding into AI music is surging to fill a potential niche of AI music software in the music production business.

As in other fields, human labor in music faces the impending possibility of replacement by automation. Business models of commissioned music may exploit vague definitions found on current authoring copyright laws in the field of synthetic music. Nevertheless, based on the legal framework, it is possible to anticipate possible scenarios for adaptation, assimilation, and revision of the music authorship concept in light of AI music.

5.8.1. Machine music and human music

Music automation techniques are creative processes originated and in service of the human mind, whether they are rule systems in combination with stochastic distributions (i.e., the current methods employed by PAMG), or statistical imitations of existing material (i.e., machine learning models). Humans are the final quality judges of autonomous systems informed by their own experiences of other human musical structures. In the last seventy years, our ears not only recognize cultural but also synthetic timbral constructions as styles. Digital devices used to perform as musical instruments explore ideas based on numerical possibilities, some of them only playable by machines. Parallel to this first branch of human-machine music, automation methods sought to perform versatile style replication with seamless quality. Despite questions surrounding authorship, machine-composed music aims to compete directly with human creations.

Although rule systems are still viable, recent deep learning developments have propelled research and funding in the direction of statistic replication. Mostly called AI music (although there is a corporation that uses this precise name), machine learning models necessarily employ a training set, which is pre-made material. While this is the source of aesthetics for AI music, rule systems use knowledge-based algorithms to produce style. As most AI models are designed and deployed by computer scientists, evaluations normally are based on statistical validation and Turing-like tests. It has been mostly assumed—especially among musicians—that AI models still cannot compete with human composers in music quality. In that vein, some studies have been investigating bias

against machine creativity. For example, Pasquier et al. (2016) conducted an empirical study to investigate whether listeners hold a bias against computer composed music. They improved upon previous studies by eliminating the Turing test-like condition and attempting to remove any practice effects, and divided their participants into *Informed*, *Naive*, and *Revealed* to test the reaction to acquired knowledge of authorship. Their results suggest that a bias against AI music may exist, but their results were not conclusive and suggested further study with a larger sample size. Hong et al. (2021) tested Expectation Violation Theory (Burgoon et al. 2016) hypothesizing that participants that found synthetic music *beyond* their expectations would rate it higher than those who found it *within* their expectations and vice versa. Their result confirmed the hypothesis. Zlatkov et al. (2023) revived the initiative but again the result did not show significant bias towards AI music. The authors suggest that despite adequate sample sizes, high variability in the data and the ranking method used may have contributed to the lack of conclusive results. They also acknowledge a single-value velocity (limited dynamic range) for computer-composed pieces which in some contexts implies a non-human origin.

The results of these tests may show a need to redirect the effort, perhaps towards music background for AI music assessment. In fact, since this issue dates back to the musical

clocks of the eighteenth century[53], a better depiction of what the participant understands by the concept of music—what do they think music does or can be used for (Novak & Matt, 2015)—may give more light in this correlation. Music made for machines or by machines has had—then and now—a place within the human mind (philosophy, ethics, ontology, human relations and so on), which is far from homogenous.

For future work—more in the side of music quality—formal music analysis inquiring features such as form construction, intertextuality, self-similarity, original motive development, coherence, intention, and performance should play a significant role.
In any case, a commercial niche for AI music output is already here and has raised many questions about how it will affect the livelihood of human composers in the future.

5.8.2. The composer's job

Although composers can work in several capacities, the possible earnings of music licensing, signing a contract, or general commissioning, tend to be more significant when dealing within mainstream or popular entertainment projects. As the industry of music production and business grows and pipelines aligned to the current technology become more entrenched, the labor of composing becomes part of production models ruled by copyright laws, and product management dynamics.

[53] Annette Richards (1999) for example, depicts a similar playground for mechanical music assessment with the works commissioned for music clocks and some curious automatons in the eighteenth century. Not only they had a place already in music consumption but also critics and enthusiasts of all sorts referenced them. In fact, the idea of mechanical performance as possibly having a superior ranges than human performers was acknowledged both by composers, who generated specific content, and also by listeners who had a wealth of reactions tied to their vision of music.

Composition, as any other activity in corporate production, is subjected to simplification, convergence, and automation for the sake of increasing profit margins. With current AI advances, basically any job can potentially be automated, i.e., assumed by machines. AI music generation—and other composer functions—are on the verge of being assimilated into corporate media production because licensing costs and production times will be lowered while multiplying the original material volume. Additionally, the composers' work expectations, modified by the result of increased technological environments, tend to converge with machine-made music as I explain below.

The framework of commercial music is regulated by copyright laws, but the contracts can overrule some of the intellectual property rights generated by a composition. Music composers that are hired by producers to create a musical sequence, arrangement, orchestration, engraving, etc., have a carefully disclosed assignment of copyrights and royalties over a musical piece in their contracts. Although a wide range of agreements is possible (from preserving publishing rights to waving authorship and everything in the middle), the contract rarely discloses stylistic or aesthetic requirements—directors and producers reveal them in meetings and conversations. Projects with different budgets and schedules pose diverse challenges in terms of music creation and production times. Pipelines influenced by current and emerging digital technologies tend to determine, not only the way a composer assumes the labor of creating music, but also the directors' and producers' expectations. Projects in which editors, directors, and producers use musical

references as a framework for musical construction, aesthetic, and/or intentionality, often privilege montage specificity over novelty in music composition.

As an example, film or TV directors often use temporary stock music (temp music) in the editing sessions, contributing to a specific vision of continuity, storytelling, discourse, and aesthetics. Used clips become musical references for which style replication would be safe to perform. Normally, directors allow some freedom to composers but the risk of rejection increases with the degree of divergence from the reference style. This practice translates into a precise audiovisual language but at the same time limits the playground for new compositional ideas such as reinterpretation of moods, and leitmotif assignment. The result is a tendency to use standardized musical moods in contemporary commercial mainstream films, which also is compatible with AI music paradigms. Film composers often are required to solve situations of reconstruction when the film is re-edited after music sync in an extremely fast pace, which is expected due to the on-the-fly editing possibilities offered in current digital platforms. One of the common solutions (besides the ‘magic’ audio editing capabilities of the post-production team) is to replicate standard references, aiming to barely avoid copyright infringement. Kmet (2019) argues that nowadays, composers are prone to produce more fragmentary or modular music clips with less motivic development and less synchronized markers than when “picture-lock” was delivered in the analog era. These musical features seem to address more effectively late post-production requests, as most films now are re-edited several times after the music is written, recorded and mixed.

As it appears, especially in the commercial arena, style imitation serves several purposes such as framing a target audience, or communicating a cultural stereo-typical emotion. When the imitation uses a style-based construction, for example similar instruments, ranges, gestures, tempi, and rhythm, it brings an association with other iterations used in the same vein and with generalized cultural intentions. For example, composers know formulas that would classify music as epic, and its variants for action, drama, love, adventure, passion, fear, evil, etc. In fact, these emotions act as a template for basic temp music used in film editing, and also can be found currently in predesigned 'moods' modes for automatic artificial composition agents such as Amper[54] and Aiva[55].
Within this work dynamic, similarity is an important feature suggested—and in some cases required—to film and game music composers, and many pop music producers. For game music composers, the modularity of the segments, standardized mood, and the least conspicuous transitions conform a production framework, even more than it is for linear storytelling.

To fill the labor of a music composer for media, an AI software should be able to respond to similarity requests made by producers, directors, and video editors. Analogous to music references used by composers, training sets are information whose patterns and statistics are used to generate resembling iterations. Then, similarity is the actual

[54] https://www.shutterstock.com/discover/ampermusic. Accessed 4/5/2023.
[55] https://www.aiva.ai/. Accessed 4/5/2023.

objective and measurable result of AI models. This means that automated music algorithms, as long as they can imitate a provided sample (or interpolate among styles), has a niche in the music production pipeline which explains the significant increase in private funding for AI research in music.

AI composition systems and humans share arguably an exposure to music material that informs and affects our output. Interestingly, human composers have borrowed music material to re-elaborate pieces virtually all the time throughout history, both deliberately and unconsciously[56]. In western societies (and possibly in all history of human art), although this kind of inspired work, quotation, and/or parody is a valid artistic intervention, there are also laws about authorship and intellectual property to address the related capital implications. This blurry boundary of failed attribution is commonly known as plagiarism.

5.8.3. Authorship: similarity, style replication and plagiarism

In the sphere of pop music, there is a rise in legal battles over plagiarism, including audio samples and musical ideas. The method for demonstrating plagiarism in musical ideas are *access* and *similarity* (Stav, 2016), which opens the door for a variety of interpretations. To prove access, it is not enough to point to a previous release, but also to establish a plausible exposure of the material to the plagiarist prior to the alleged re-elaboration. This

[56] In a famous case that came to trial in 1976, Georges Harrison's "My Sweet Lord" was found guilty of "subconscious plagiarism" over the Chiffons "He's so fine" https://performingsongwriter.com/george-harrison-my-sweet-lord/. Accessed 4/5/2023.

means that a natural similarity between pieces that had no influence from one another is conceivable under the law. Hence, plagiarism is difficult to prove when there is no public release of the plagiarized material. Once access is deemed as probable, similarity comes into account. Although technical music constructions can objectively prove it, such as motivic replication by transposition or small modifications, actual cases have shown deep complexities. A quick search through filed claims reveals commonalities between pieces that go beyond conventional and measurable music design to include how they are perceived by regular listeners, i.e. non-experts. This widens the boundaries of similarity to overlap with the connotations of style, but style replication is not regarded as plagiarism by the law.

An important case has set a precedent regarding court verdicts in which similarity is used in close relation to style. Gaye's 1977 song "Got to give it up" won a disputed copyright protection against "Blurred lines" by Robin Thicke (2013), becoming an apparent example of music style copyrighting. Many in the media industry were paying close attention because style replication, as explained before, is a common practice in production, and if such a case becomes normal, many lawsuits may fallow. However, experienced copyright lawyers argue that repercussions on other cases are less than the outcome implied because of trial particularities[57]. This suggests that trial arguments, lawyer skills, deferential testimonies, and evidence made this case unique rather than a

[57] https://mcpherson-llp.com/articles/crushing-creativity-the-blurred-lines-case-and-its-aftermath/. Accessed 5/5/2023.

template for future claims[58]. Nevertheless, there are similar cases in which productions using a referenced music style were disputed as plagiarism.

Style replication, even if it is a tribute or a disclosed inspiration may be liable within the limits of the concept of similarity. As AI music necessarily accesses music material and outputs similarity, there are rising questions about how to apply current legal frameworks or create new ones. In fact, AI has been employed consistently to spot similarity[59] and potential plagiarism[60] in platforms like Spotify and YouTube. Especially, if a single piece or author has been the subject of reproduction with AI, the resulting music products may need a new legal framework and a licensing scheme. Copyright practitioners tend to say that the authors of music material used for AI training hardly can receive compensation for the commercialization of AI products but the law seems obscure about it[61]. Hence, in a number of cases, programmers have been claiming authorship of their models' output with mixed results.

Copyright law has assumed until now that computer methods to create works are tools that humans use. Hence, in many western countries, only human creations can be copyrighted (Ramalho, 2017). This implies that if the authorship is not attributed to the

[58] https://www.americanbar.org/groups/intellectual_property_law/publications/landslide/2015-16/january-february/blurring_lines_the_practical_implications_of_williams_v_bridgeport_music/#ref9. Accessed 5/5/2023.
[59] https://researchrepository.wvu.edu/cgi/viewcontent.cgi?article=6333&context=wvlr. Accessed 5/5/2023.
[60] https://www.musicbusinessworldwide.com/files/2020/11/1_merged.pdf. Accessed 5/5/2023.
[61] https://www.theverge.com/2019/4/17/18299563/ai-algorithm-music-law-copyright-human. Accessed 5/5/2023.

creators of the algorithm, the piece would remain as public-domain. Andres Guadmuz[62](2017) shows an example of a declaration from the United States' Copyright Office in which it will "register an original work of authorship, provided that the work was created by a human being." And also specifies that copyright law only protects "the fruits of intellectual labor" that "are founded in the creative powers of the mind." (1991). The example continues with an assertion that links humans to creative processes in UK copyright law which states: "In the case of a literary, dramatic, musical or artistic work which is computer-generated, the author shall be taken to be the person by whom the arrangements necessary for the creation of the work are undertaken." (Section 9(3) CDPA). This would justify assigning authorship to programmers. In the same article, other cases reveal the importance of labor and artistic decisions in authorship, like the case of Nova Productions v Mazooma Games [2007] EWCA Civ 219, in which the Court of Appeal had to decide on the authorship of a computer game, and declared that a player's input "is not artistic in nature and they have contributed no skill or labor of an artistic kind". In most of the law framework examples examined by Ramalho (2017), the foundation of authorship—and the corresponding copyright protections—seems to be framed in a 'decision/choice'(agency), 'an expression of personality' (pointing to a human origin), and 'an arrangement undertaking' (production and work for hire framework). The main issue is the lack of recognition of an equivalency between AI training and human training. Clearly, going back in the direction of the source of

[62] World Intellectual Property Organization WIPO Magazine https://www.wipo.int/wipo_magazine/en/2017/05/article_0003.html. Accessed 5/5/2023.

knowledge would put authorship in the terrain of Barthes' "Death of the author"[63] (1977) which complicates copyright as a concept. This is exemplified in Fig. 23.

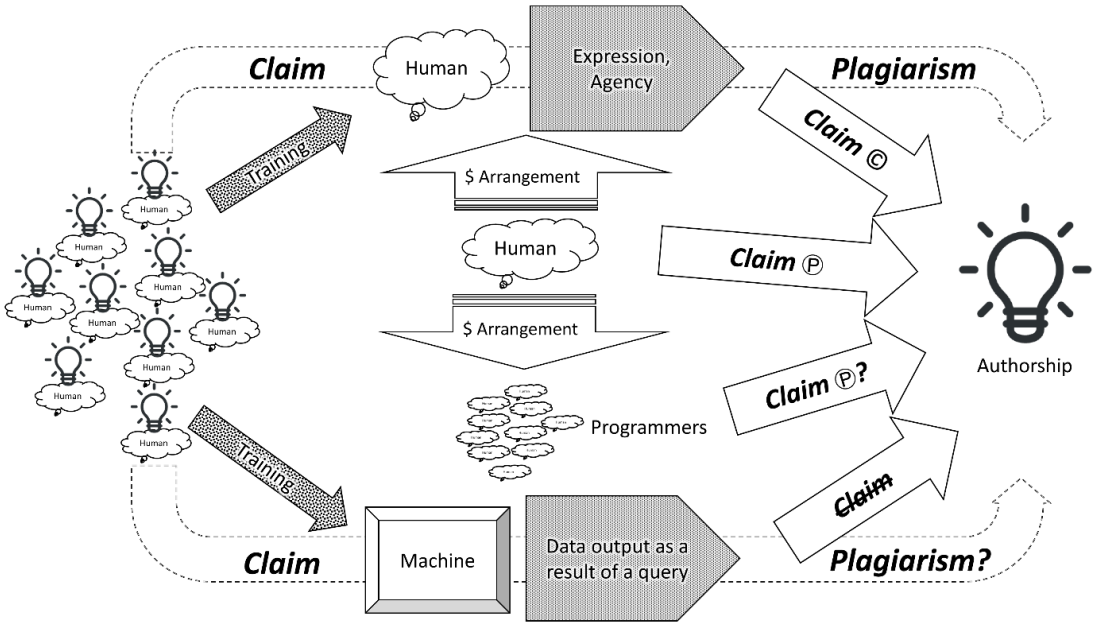


Fig. 29. Authorship.

Humans are usually entitled to claim authorship. However, in current laws the role in creativity of training sets for human or AI works may fall in dispute. In the case of machine-made material, training sets—which are human ideas—are objectively the knowledge source, while in humans that source is subjective. Thus, a human whose idea was a 'training set' for another human idea only could claim a stake in authorship of the

[63] In "The Death of the Author," Barthes argues that the meaning of a literary text cannot be solely determined by the author's intentions or biography, and that readers play an active role in interpreting a text. He also claims that writing is not the creation of a voice, but rather the destruction of every voice, and that literary texts are not original but rather a "tissue of quotations." While this view may seem extreme, Barthes emphasizes the importance of language and the way in which meaning is constructed through familiar words put together in new ways.

later through a plagiarism case. The same process arguably would work for machine-made works. Programmers under some circumstances could claim authorship, although it is also subject of debate.

Arguably, current AI models depart significantly from being a tool since creative decisions are taken by the program, albeit within a training set[64]. More recently, as AI creative works have been accruing significant commercial value, there are more cases to rule by the Copyright Office. In a recent decision, the federal agency refused to grant copyright to Stephen Thaler for an image created with his AI model at Creative Machines. The Copyright Board wrote in its Feb. 14, 2022 ruling: "Thaler must either provide evidence that the work is the product of human authorship or convince the Office to depart from a century of copyright [legal theory]… he has done neither."[65] Summarizing, the boundaries of authorship in the case of AI in music currently are blurry. Taken from current U.S. copyright law, the point of origin should be a human—considered the author—in which case AI music would not be granted copyright. On the other hand, as in U.K. law, it is possible to argue that the programmers provided labor to create the algorithm that ultimately created the piece, which gives them authorship. A third possible case would be to adjudicate copyright to the author(s) of the original pieces

[64] In fact, scholars have written to frame AI decision making as a tool within copyright: J. Bing and T. Harvold, Legal Decisions and Information Systems (Universitetsforlaget; Henley on Thames 1977); G. Sartor, Artificial Intelligence and Law: Legal Philosophy and Legal Theory (Tano 1993); P. Leith, Formalism in Al and Computer Science (Ellis Horwood 1990); and D. Bourcier, L. Bochereau and P. Bourgine, Extracting legal knowledge by means of multilayer neural network: Application to municipal jurisprudence, in *Proceedings of the 3rd ICAIL, Oxford* (ACM 1991), 288.
[65] https://dot.la/creative-machines-ai-art-2656764050.html. Accessed 5/6/2023.

that constitute the training set—since the algorithm not only had access to them objectively but also sought similarity either through a musical or text prompt. This case may be a fertile ground for research since the model can provide a rate of similarity between its output and the training set entries. With these questions, it is not only authorship laws but also commercial music models that need updating. As can be envisioned, models used as music services would take over both the auditory spectrum and the money stream claimed by human musicians today to make a living, which seems ethically flawed. An explosion of music material will rapidly become comparable in size with human music, to the point in which utility or functional music (elevator, waiting room, airport music, etc)—characterized by being artificially generated—may distinguish itself from the human counterpart and require separate regulations. In the meantime, the legal battles are starting. The first appearances of replication have been using timbre emulation: individuals make a song, sing it and then replace the voice by an AI-generated voice of a prominent author. These AI works in most cases currently are made with a combination of AI solutions such as, for example, ChatGPT for lyrics and AIVA for music material[66]. Artists are complaining and have convinced platforms to block AI materials and some others are claiming a portion of the royalties. AI-music creators may

[66] In the spectrum of AI-music generation there is a growing body of resources: https://theresanaiforthat.com/s/music/. Accessed 5/9/2023.

argue that *parody* is legal[67] but debates will follow arguing a breach in purpose and fair use of identifiable material as it applies currently to face pictures or audio recordings for publications. This copyright conflict will be ensued by fund assignment and embracement of AI-replicating technologies by music producers and publishers—those that took the artist side in the first round. In this case, their contracts will include assignment of rights for AI training in the resulting recordings, allowing them to capitalize not only on replication volume consumed, but also on emerging technologies such as artist-generated radio stations and software.

It is possible to describe the intention of an AI user/programmer as style replication or interpolation of existing material. In case of art, an aesthetic decision of an AI user is closer to an editor/curator than an author, as it is seen in current generative text models. In music, there are many algorithms whose creative methods do not use pre-made music material as an input and instead use rules and stochastic principles. In them, the discussion of authorship is closer to current human origin definitions, since aesthetic decisions are taken by the user when choosing the variables and parameters. That is the case with PAMG.

[67] "The fair use exception is governed by the factors enumerated in section 107 of the Copyright Act: (1) the purpose and character of the use; (2) the nature of the original work; (3) the amount and substantiality of the original work used; and (4) the effect on the market value of the original work. Generally, courts are more likely to find that a parody qualifies as fair use if its purpose is to serve as a social commentary and not for purely commercial gain." https://www.law.cornell.edu/wex/parody#:~:text=In%20the%20United%20States%2C%20parody%20is%20protected%20by,use%20exception%20to%20combat%20claims%20of%20copyright%20infringement. Accessed 5/9/2023.

6. References

Abbasi, A. Z., Ting, D. H., & Hlavacs, H. (2017, January 31). *Engagement in Games: Developing an Instrument to Measure Consumer Videogame Engagement and Its Validation* [Research Article]. International Journal of Computer Games Technology; Hindawi. https://doi.org/10.1155/2017/7363925

Ames, C. (1987). Automated Composition in Retrospect: 1956-1986. *Leonardo*, *20*(2), 169–185. https://doi.org/10.2307/1578334

Anders, T. (2021). On Modelling Harmony with Constraint Programming for Algorithmic Composition Including a Model of Schoenberg's Theory of Harmony. In E. R. Miranda (Ed.), *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity* (pp. 283–326). Springer International Publishing. https://doi.org/10.1007/978-3-030-72116-9_11

Aristopoulos, M. (2023). *The game music toolbox: Composition techniques and production tools from 20 iconic game soundtracks*. Routledge.

Berlyne, D. E. (1971). Aesthetics and Psychobiology. New York, NY, USA: Appleton.

Bigand, E. and Parncutt, R. (1999). Perceiving musical tension in long chord sequences. In *Psychological Research*, volume 62(4), pages 237–254. Springer. Available at https://doi.org/10.1007/s004260050053.

Biles, A. (2002a). GenJam in Transition: from Genetic Jammer to Generative Jammer. In *International Conference on Generative Art*, Milan, Italy.

Burgoon, J. K., Bonito, J., & Lowry, P. (2016). Application of Expectancy Violations Theory to communication with and judgments about embodied agents during a decision-making task. *Int. J. Hum. Comput Stud.*, *91*. https://doi.org/10.1016/j.ijhcs.2016.02.002

Carpentier, G., Daubresse, E., Vitoria, M. G., Sakai, K., & Carratero, F. V. (2012). Automatic Orchestration in Practice. *Computer Music Journal*, *36*(3), 24–42.

Collins, Karen (2007). An Introduction to the Participatory and Non-Linear Aspects of Video Games Audio. In: *Essays on Sound and Vision*. Ed. by John Richardson and Stan Hawkins. Helsinki, Finland: Helsinki University Press and Finnish Society for Ethnomusicology. Chap. 4.

Collins, K. (2009). An Introduction to Procedural Music in Video Games. *Contemporary Music Review*, *28*(1), 5–15. https://doi.org/10.1080/07494460802663983

Collins, N. (2000). "Caring for the Instrumentalist in Automatic Orchestration." In Proceedings for Acoustics and Music: Theory and Applications, pp. 32-38.

Conklin, D., & Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, *24*(1), 51–73. https://doi.org/10.1080/09298219508570672

Cope, D. (1991). *Computers and musical style*. Madison, Wis.: A-R Editions.

Cope, D., & Cope, D. (1996). *Experiments in musical intelligence*. Madison, Wis. A-R Editions.

Cope, D. (2000). *The algorithmic composer*. A-R Editions.

Christensen, T. S. (2002). *The Cambridge history of Western music theory*. Cambridge, U.K. ; New York : Cambridge University Press. http://archive.org/details/cambridgehistory00chri

Chomsky, N. (1956). Three models for the description of language. In *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113-124, doi: 10.1109/TIT.1956.1056813.

Cutajar, S. (2020). Automatic Generation of Dynamic Musical Transitions in Computer Games Ph.D. dissertation, The Open University. http://oro.open.ac.uk/69192/

Donahue, C., Mao, H. H., Li, Y. E., Cottrell, G. W., & McAuley, J. (2019). Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training. In *Proc. of the 20th international society for music information retrieval conference* (pp. 685–692).

Eerola, T., Himberg, T., Toiviainen, P., & Louhivuori, J. (2006). Perceived complexity of western and african folk melodies by western and African listeners. *Psychology of Music*, 34 (3), 337–371.

Elmsley, A, Velardo, V, & Grooves, R. (2017). *Deep Adaptation: How Generative Music Affects Engagement and Immersion in Interactive Experiences.* DMRN12+ Digital Music Research Network One-Day Workshop, Queen Mary University of London. https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/30583/ID%3A%2097442919-02C3-424B-9C18-B08D362CCC22.pdf?sequence=1

Engels, S., Tong, T., & Chan, F. (2015). Automatic Real-Time Music Generation for Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, *11*(1), Article 1. https://doi.org/10.1609/aiide.v11i1.12775

Ermi, L., & Mäyrä, F. (2005). Fundamental Components of the Gameplay Experience: Analysing Immersion. *Worlds in Play: Int. Perspectives on Digital Games Research*. Vancouver, British Columbia, Canada.

Esling, P., Carpentier, G., & Agon, C. (2010). Dynamic Musical Orchestration Using Genetic Algorithms and a Spectro-Temporal Description of Musical Instruments. In C. Di Chio, A. Brabazon, G. A. Di Caro, M. Ebner, M. Farooq, A. Fink, J. Grahl, G. Greenfield, P. Machado, M. O'Neill, E. Tarantino, & N. Urquhart (Eds.), *Applications of Evolutionary Computation* (pp. 371–380). Springer. https://doi.org/10.1007/978-3-642-12242-2_38

Fırat, H. B., Maffei, L., & Masullo, M. (2022). 3D sound spatialization with game engines: The virtual acoustics performance of a game engine and a middleware for interactive audio design. *Virtual Reality*, *26*(2), 539–558. https://doi.org/10.1007/s10055-021-00589-0

Farnell, A. (2010). *Designing sound*. MIT Press.

Gilreath, P., & Aikin, J. (2004). *The Guide To MIDI Orchestration* (O. Torres, Ed.; 3rd edition). Musicworks.

Gungormusler, A., Paterson-Paulberg, N., & Haahr, M. (2015). *barelyMusician: An Adaptive Music Engine For Video Games*. http://www.tara.tcd.ie/handle/2262/82637

Granot, R. and Eitan, Z. (2011). Musical tension and the interaction of dynamic auditory parameters. In *Music Perception*, volume 28(3), pages 219–246. University of California Press. Available at https://doi.org/10.1525/mp.2011.28.3.219.

Grimshaw, M., & Schott, G. (2008). A Conceptual Framework for the Analysis of First-Person Shooter Audio and its Potential Use for Game Engines. *International Journal of Computer Games Technology*, *2008*, e720280. https://doi.org/10.1155/2008/720280

Hadjeres, G., Pachet, F., & Nielsen, F. (2017). Deepbach: a steerable model for bach chorales generation. In *Proc. of the 34th international conference on machine learning* (pp. 1362–1371).

Hasty, C. F. (1997). *Meter as Rhythm*, Oxford UK: Oxford University Press.

Helmholtz, H. V. (1877). *On the Sensations of Tone: As a Physiological Basis for the Theory of Music*, Translated and edited by Alexander J. Ellis.

Heyduk, R. G. (1975). Rated preference for musical compositions as it relates to complexity and exposure frequency. *Perception & Psychophysics*, *17*(1), 84–90. https://doi.org/10.3758/BF03204003

Hiller, L., & Isaacson, L. M. (1959). Experimental music; composition with an electronic computer. New York, McGraw-Hill. http://archive.org/details/experimentalmusi00hill

Hoeberechts, M., Demopoulos, R., & Katchabaw, M. (2009). *A Flexible Music Composition Engine* (Patent No. WO2009036564 (A1)). https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=20090326&DB=EPODOC&locale=&CC=WO&NR=2009036564A1&KC=A1&ND=1

Hong, J. W., Peng, Q., & Williams, D. (2021). Are you ready for artificial Mozart and Skrillex? An experiment testing expectancy violation theory and AI music. *New Media & Society*, *23*(7), 1920–1935. https://doi.org/10.1177/1461444820925798

Huron, D. B. (2006). *Sweet anticipation: music and the psychology of expectation*. MIT press.

Hutchings, P. E., & McCormack, J. (2020). Adaptive Music Composition for Games. *IEEE Transactions on Games*, *12*(3), 270–280. https://doi.org/10.1109/TG.2019.2921979

Jennett, C., Cox, A. L., Cairns, P., Dhoparee, S., Epps, A., Tijs, T., & Walton, A. (2008). Measuring and defining the experience of immersion in games. International Journal of Human-Computer Studies, 66(9), 641–661. https://doi.org/10.1016/j.ijhcs.2008.04.004

Jensen, K., & Hebert, D. G. (2016). Evaluation and Prediction of Harmonic Complexity Across 76 Years of Billboard 100 Hits. In R. Kronland-Martinet, M. Aramaki, & S. Ystad (Eds.), *Music, Mind, and Embodiment* (pp. 283–296). Springer International Publishing. https://doi.org/10.1007/978-3-319-46282-0_18

Jolly, K., McLeran, A. *Procedural music in spore*, (2008). https://www.gdcvault.com/play/323/Procedural-Music-in

Keith, M. (1991) From Polychords to Pólya. *Adventures in Musical Combinatorics*, Princeton NJ: Vinculum Press.

Klimmt, C., Possler, D., May, N., Auge, H., Wanjek, L., & Wolf, A.-L. (2019). Effects of soundtrack music on the video game experience. *Media Psychology*, 22(5), 689–713. https://doi.org/10.1080/15213269.2018.1507827

Kmet, N. (2019) “We’ll Fix It In Post”: Digital Editing and the Film Score. Conference paper. *Music and the Moving Image.* NYU Steinhardt.

Kostka, S. M. (2013). *Tonal harmony: With an introduction to twentieth-century music* (7th ed.). McGraw-Hill.

Krumhansl, C. L. (2001). *Cognitive foundations of musical pitch*. Oxford University Press. Available at https://doi.org/10.1093/acprof:oso/9780195148367.001.0001.

Laurson, M., and M. Kuuskankare. 2001. "A Constraint Based Approach to Musical Textures and Instrumental Writing." In *Proceedings of the Seventh International Conference on Principles and Practice of Constraint Programming*, Musical Constraints Workshop. Center for Music and Technology, Sibelius Academy, Helsinky, Finland.

Lerdahl, F., & Jackendoff, R. (1983). *A generative theory of tonal music*. MIT Press.

Lerdahl, F. (1988) "Tonal Pitch Space". *Music Perception*, 5, 315-350.

Lerdahl, F. (1996). Calculating Tonal Tension. *Music Perception: An Interdisciplinary Journal*, *13*(3), 319–363. https://doi.org/10.2307/40286174

Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, *22 140*, 55–55.

Lippe, C. (1997). Music for piano and computer: A description. *Information Processing Society of Japa SIG Notes*, 97 (122): 33-38.

Lopez Duarte, A. E. (2020). Algorithmic interactive music generation in videogames. *SoundEffects - An Interdisciplinary Journal of Sound and Sound Experience*, *9*(1), 38–59. https://doi.org/10.7146/se.v9i1.118245

Luby, M. G. (1996). *Pseudorandomness and cryptographic applications*. Princeton University Press.

Lykartsis, A., Pysiewicz, A., von Coler, H., & Lepa, S. (2013). *The Emotionality of Sonic Events: Testing the Geneva Emotional Music Scale (GEMS) for Popular and Electroacoustic Music*. The 3rd International Conference on Music & Emotion, Jyväskylä, Finland, June 11-15, 2013. https://jyx.jyu.fi/handle/123456789/41586

Madison, G., & Schiölde, G. (2017). Repeated Listening Increases the Liking for Music Regardless of Its Complexity: Implications for the Appreciation and Aesthetics of Music. *Frontiers in Neuroscience*, *11*. https://www.frontiersin.org/articles/10.3389/fnins.2017.00147

Marsden, M., & Ajoodha, R. (2021). Algorithmic Music Composition Using Probabilistic Graphical Models and Artificial Neural Networks. *2021 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, 1–4. https://doi.org/10.1109/SAUPEC/RobMech/PRASA52254.2021.9377235

Mazzola, G., Park, J., & Thalmann, F. (2011). *Musical Creativity: Strategies and Tools in Composition and Improvisation*. Springer Science & Business Media.

McLean, E., & Dean, R. T. (Eds.). (2018). *The Oxford Handbook of Algorithmic Music*. Oxford University Press.

Miranda, E. R., & Biles, A. (Eds.). (2007). *Evolutionary computer music*. Springer. http://dx.doi.org/10.1007/978-1-84628-600-1

Moffat, D.C., Kelly, M.: An investigation into people's bias against computational creativity in music composition (2006).

Navarro-Cáceres, M., Caetano, M., Bernardes, G., Sánchez-Barba, M., & Merchán Sánchez-Jara, J. (2020). A Computational Model of Tonal Tension Profile of Chord Progressions in the Tonal Interval Space. *Entropy*, *22*(11), 1291. https://doi.org/10.3390/e22111291

Navarro-Cáceres, M., Caetano, M., Bernardes, G., Sánchez-Barba, M., & Merchán Sánchez-Jara, J. (2020). A Computational Model of Tonal Tension Profile of Chord Progressions in the Tonal Interval Space. *Entropy*, *22*(11), 1291. https://doi.org/10.3390/e22111291

Nierhaus, G. (2009). *Algorithmic composition: Paradigms of automated music generation*. Springer. http://dx.doi.org/10.1007/978-3-211-75540-2

Novak, D., & Matt, S. (Eds.). (2015). *Keywords in Sound*. Duke University Press.

Orr, M. G., & Ohlsson, S. (2005). Relationship Between Complexity and Liking as a Function of Expertise. *Music Perception: An Interdisciplinary Journal*, *22*(4), 583–611. https://doi.org/10.1525/mp.2005.22.4.583.

Johnson-Laird, P. N. Jazz Improvisation: A Theory at the Computational Level. In *Representing Musical Structure*, Howell, P., West, R., and Cross, I., eds. Academic Press, 1991.

Oppenheim, D. V. (1997). *Interactive system for compositional morphing of music in real-time* (6th ed.). https://worldwide.espacenet.com/publicationDetails/biblio?FT=D&date=19970902&DB=EPODOC&CC=US&NR=5663517A

Papadopoulos, G., & Wiggins, G. (1999). AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects. Proceedings from the AISB'99 Symposium on Musical Creativity.

Parncutt, R. (1989). *Harmony: A psychoacoustical approach*. Springer-Verlag.

Pasquier, P., Burnett, A., Maxwell, J. (2016). Investigating listener bias against musical metacreativity. In: *Proceedings of the Seventh International Conference on Computational Creativity*, pp. 42–51.

Pasquier, P., Eigenfeldt, A., Bown, O., & Dubnov, S. (2016). An Introduction to Musical Metacreation. *Computers in Entertainment*, *14*(2), 1–14. https://doi.org/10.1145/2930672.

Pearson, K. (1905). The Problem of the Random Walk. *Nature*. 72 (1865): 294. doi:10.1038/072294b0.

Plut, C. (2017). *The Audience of the Singular*. Master's thesis, Simon Fraser University, Vancouver, BC.

Plut, C., & Pasquier, P. (2020). Generative music in video games: State of the art, challenges, and prospects. *Entertainment Computing*, *33*, 100337. https://doi.org/10.1016/j.entcom.2019.100337

Plut, C., Pasquier, P., Ens, J., & Tchemeube, R. (2022). PreGLAM-MMM: Application and evaluation of affective adaptive generative music in video games. *FDG '22: Proceedings of the 17th International Conference on the Foundations of Digital Games*, 1–11. https://doi.org/10.1145/3555858.3555947

Povel, D.-J., & Essens, P. (1985). Perception of Temporal Patterns. *Music Perception: An Interdisciplinary Journal*, *2*(4), 411–440. https://doi.org/10.2307/40285311

Prechtl, A. (2016). *Adaptive Music Generation for Computer Games*. Ph.D. Dissertation. Open University, UK.

Psenicka, D. 2003. Sporch: An Algorithm for Orchestration Based on Spectral Analyses of Recorded Sounds. In *Proceedings of the International Computer Music Conference*, pp. 207-210.

Ramalho, A. (2017). *Will Robots Rule the (Artistic) World? A Proposed Model for the Legal Status of Creations by Artificial Intelligence Systems* (SSRN Scholarly Paper No. 2987757). https://doi.org/10.2139/ssrn.2987757

Rep. Clarke, Y. D. [D-N.-9. (2022, February 4). *Text - H.R.6580 - 117th Congress (2021-2022): Algorithmic Accountability Act of 2022* (02/04/2022) [Legislation]. http://www.congress.gov/

Reuderink, B., Mühl, C., & Poel, M. (2013). Valence, arousal and dominance in the EEG during game play. *International Journal of Autonomous and Adaptive Communications Systems*, *6*(1), 45. https://doi.org/10.1504/IJAACS.2013.050691

Richards, A. (1999). Automatic Genius: Mozart and the Mechanical Sublime. *Music & Letters*, *80*(3), 366–389.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., & Eck, D. (2019). A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music (arXiv:1803.05428). arXiv. https://doi.org/10.48550/arXiv.1803.05428

Robertson, J., de Quincey, A., Stapleford, T., & Wiggins, G. (1998). Real-Time Music Generation for a Virtual Environment. *Proceedings of ECAI-98 Workshop on AI/Alife and Entertainment*.

Rowe, R. (1993). *Interactive music systems: Machine listening and composing*. MIT Press.

Ruiz Marcos, G. (2022). *Tension-driven Automatic Music Generation*. https://doi.org/10.21954/OU.RO.000142CF

Ruiz Marcos, Germán. (2022). *Tension-driven Automatic Music Generation*. https://doi.org/10.21954/OU.RO.000142CF

Schellenberg, E. G., Peretz, I., & Vieillard, S. (2008). Liking for happy- and sad-sounding music: Effects of exposure. *Cognition and Emotion*, *22*(2), 218–237. https://doi.org/10.1080/02699930701350753

Shmulevich, I., & Povel, D.-J. (1998). Rhythm complexity measures for music pattern recognition. *1998 IEEE Second Workshop on Multimedia Signal Processing (Cat. No.98EX175)*, 167–172. https://doi.org/10.1109/MMSP.1998.738930

Slaney, M., Covell, M., & Lassiter, B. (1996). Automatic audio morphing. *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, *2*, 1001–1004 vol. 2. https://doi.org/10.1109/ICASSP.1996.543292

Stav, I. (2016). Musical Plagiarism: A True Challenge for the Copyright Law. *Intellectual Property Law*, 25.

Steedman, M, (1984). A Generative Grammar for Jazz Chord Sequences. *Music Perception*, 2:52–77.

Scirea, M., Togelius, J., Eklund, P., & Risi, S. (2016). MetaCompose: A Compositional Evolutionary Music Composer. In C. Johnson, V. Ciesielski, J. Correia, & P. Machado (Eds.), *Evolutionary and Biologically Inspired Music, Sound, Art and Design* (Vol. 9596, pp. 202–217). Springer International Publishing. https://doi.org/10.1007/978-3-319-31008-4_14

Somberg, G. (2016). *Game Audio Programming: Principles and Practices*. CRC Press. https://doi.org/10.1201/9781315368696

Sweet, M. (2015). *Writing interactive music for video games a composer's guide*. Addison-Wesley.

Tatar, K., & Pasquier, P. (2019). Musical agents: A typology and state of the art towards Musical Metacreation. *Journal of New Music Research*, *48*(1), 56–105. https://doi.org/10.1080/09298215.2018.1511736

Thickstun, J., Harchaoui, Z., Foster, D. P., & Kakade, S. M. (2018). Coupled recurrent models for polyphonic music composition. In *Proc. of the 20th international society for music information retrieval conference* (pp. 311–318).

Thomassen, J. M. (1982). Melodic accent: Experiments and a tentative model. *The Journal of the Acoustical Society of America*, *71*(6), 1596–1605. https://doi.org/10.1121/1.387814

Togelius, Julian, Emil Kastbjerg, David Schedl, and Georgios N. Yannakakis (2011a). "What is procedural content generation?: Mario on the borderline". In: Proceedings of the 2nd International Workshop on Procedural Content Generation in Games - PCGames '11. Bordeaux, France. doi: 10.1145/2000919.2000922

Togelius, Julian, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne (2011b). "Search-based procedural content generation: A taxonomy and survey". In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3, pp. 172–186. doi: 10.1109/TCIAIG.2011.2148116

Toussaint, G. T., & Trochidis, K. (2018). On Measuring the Complexity of Musical Rhythm. *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 753–757. https://doi.org/10.1109/UEMCON.2018.8796634

Toussaint, G. T. (2010). Generating “Good” Musical Rhythms Algorithmically. *Proceedings of the 8th International Conference on Arts and Humanities*, 774–791.

Toussaint, G. T. (2013). The pairwise variability index as a measure of rhythm complexity. *Analytical Approaches to World Music*, vol. 2, no. 2, pp. 1-42.

Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the turing test* (pp. 23–65). Springer.

Vico, F., Albarracin-Molina, D., Diaz-Jerez, G., & Manzoni, L. (2021). Automatic Music Composition with Evolutionary Algorithms: Digging into the Roots of Biological Creativity. In E. R. Miranda (Ed.), *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity* (pp. 455–483). Springer International Publishing. https://doi.org/10.1007/978-3-030-72116-9_17

Vitz, P. C. (1964). Preferences for rates of information presented by sequences of tones. *Journal of Experimental Psychology*, *68*(2), 176–183. https://doi.org/10.1037/h0043402

Vorderer, P., & Bryant, J. (Eds.). (2006). *Playing Video Games: Motives, Responses, and Consequences* (1st edition). Routledge.

Wharton, A., & Collins, K. (2011). Subjective Measures of the Influence of Music Customization on the Video Game Play Experience: A Pilot Study. *Game Studies*, *11*(2). https://gamestudies.org/1102/articles/wharton_collins

Williams, D., Mears, J., Kirke, A., Miranda, E., Daly, I., Malik, A., Weaver, J., Hwang, F., Nasuto, S., & Williams, D. (2016). A Perceptual and Affective Evaluation of an Affectively-Driven Engine for Video Game Soundtracking. *Computers in Entertainment*, *14*.

Winkler, T. (1998). *Composing Interactive Music*. Cambridge, Massachusetts, MIT Press.

Wood, R. T. A., Griffiths, M. D., & Parke, A. (2007). Experiences of Time Loss among Videogame Players: An Empirical Study. *CyberPsychology & Behavior*, *10*(1), 38–44. https://doi.org/10.1089/cpb.2006.9994

Wooller, R., & Brown, A. R. (2005). Investigating morphing algorithms for generative music. *Third Iteration: Third International Conference on Generative Systems in the Electronic Arts*, Melbourne, Australia.

Zentner, M., Grandjean, D., & Scherer, K. R. (2008). Emotions evoked by the sound of music: Characterization, classification, and measurement. *Emotion*, *8*(4), 494–521. https://doi.org/10.1037/1528-3542.8.4.494

Zhang, j., & Fu, X. (2015). The Influence of Background Music of Video Games on Immersion. *Journal of Psychology & Psychotherapy*, *5*(4), 191. https://doi.org/10.4172/2161-0487.1000191

Zhao, Z., Liu, H., Li, S., Pang, J., Zhang, M., Qin, Y., Wang, L., & Wu, Q. (2022). *A Review of Intelligent Music Generation Systems* (arXiv:2211.09124). arXiv. http://arxiv.org/abs/2211.09124

Zlatkov, D., Ens, J., Pasquier, P. (2023). Searching for Human Bias Against AI-Composed Music. In: Johnson, C., Rodríguez-Fernández, N., Rebelo, S.M. (eds) Artificial Intelligence in Music, Sound, Art and Design. EvoMUSART 2023. Lecture Notes in Computer Science, vol 13988. Springer, Cham. https://doi.org/10.1007/978-3-031-29956-8_20

Appendix A: Algorithm description

1. **Clock**

   It is the time generator that produces the time grid for the events in general, and it is the start of the playback algorithm. It produces a loop of indexed pulses (Count Loop) sent to agents and other algorithms to react synchronously, which constitute the minimum onset or grain. The loop size in pulses is the result of the chosen number of beats multiplied by the chosen grain (pulse subdivision). Its parameters are *Tempo*, *Size* and *Grain* (minimum subdivision of the pulse). All the Agents act in relation to the loop size in pulses (loop-based index).

2. **Melody Agent (MA)**

   2.1. Melody Rhythm generation

   2.1.1. Beat and Moved Lists

   It creates a list of onsets based on the beat (size / grain) using the loop-based indexes (e.g., 0, 4, 8, 12, in a 16-size loop). Then, it moves or shifts a random seeded amount of the resulting indexes to produce syncopation proportional to the parameters *move even*, *move odd* (even beats are 0, 2, 4, 6, etc., and odd are beats number 1, 3, 5, etc.), and the *value of motion* either a 16^{th} or an 8^{th} note. Also, it is possible to set a *pattern of motion* (forward-backward). Returns the Moved List and Beat List in loop-based index value.

   2.1.2. Additive and Subtractive Filling

   It subtracts from Moved List or adds onsets progressively through a seeded random generator, using incremental subdivisions of the pulse. Its parameter is Melody +/- *onsets*. Returns a list of loop-based index values that are used as onsets, and assigns a duration value

   2.1.3. Staccato/Legato Duration

   It produces a seeded random sequence of binary staccato/legato Boolean values applied to currently available onsets. Then, it calculates the duration of an onset that is either a *staccato* or *legato.* The duration value uses the minimum loop unit (e.g., 1 in a 16-size loop would be a 16^{th} note), and the legato duration is the difference between current onset index and the next one. Its parameter is *Staccato-Legato* value (lower for mostly legato, higher for mostly staccato). Returns a collection of onset values and their durations (a non-zero value for legato, 1 for staccato, 0 for not-onset).

   2.2. Velocity Generation MA

   Using a random-walk sequence, assigns a scaled value for velocity (amplitude in MIDI) using as input the melody *velocity* parameter value. It gives accent — higher value— to onsets from the Moved List, and expands/contracts the range in direct proportion to the amount of onsets (+/- *onsets* parameter value). Returns a MIDI velocity value for every onset.

   2.3. Melody Pitches Generation

   2.3.1. Note Stream Generation

To assign pitches in real time, *drunk* and *drunk-contour* modes are triggered for every onset. Its parameters are *step* for maximum interval, *range* for a total range of action, and a *non-repeat* Boolean.

2.3.1.1. Drunk mode uses a note-by-note random-walk sequence and receives the *seed* to restart the sequence. Returns a pitch value.

2.3.1.2. Drunk-contour mode applies the random-walk sequence to the Moved List onsets and interpolates between them to create a contour with the duration of one loop. The #nodes parameter allows to choose the amount of nodes in the contour whit a maximum of moved list—the same as beats present in the loop. Reacts to the *invert* Boolean parameter to apply a melody inversion based on the mean pitch value. Returns a pitch value.

2.3.2. *Transposition*

Its parameter value adds or subtracts to the pitch value. It also receives transposition offsets from other algorithms to react to phrasing, melody shape, and lack of change in the register.

2.3.3. Harmony Filter

It receives the Pitch Class List for Melody to elaborate a whole range grid where individual pitch values are re-pitch using current harmony. Returns the harmonized pitch value.

2.4. Shape Melody

Using a 2D table, assigns modification values for transposition in the melody in time. Its parameters are *depth* as a value of influence, *length* as a value of loops that it takes to complete, and *invert* as a Boolean (this parameter also affects Drunk-Contour shape).

2.5. Melody Change

2.5.1. Transposition Change

It stores a window of previous pitches and their duration to analyze the range, register and amount of pitches. Then, after a finite number of phrases, it produces a transposition offset.

2.5.2. Phrase Change

Using the chord denomination generated by the Harmony Agent (e.g., I, ii, IV, V, etc.) alters the Melody *+/- onsets* preset to reflect tonal phrase resolutions in rhythm.

2.6. MIDI Event Construction MA

Using the length (staccato will be 1/64, legato the duration value in loop units), velocity and pitch, produces melody MIDI event stream.

Operation

The Count (a loop-based index coming from the Clock) queries the value stored at the rhythm onset collection to check for non-zero values. If and when found, MA produces a harmonized pitch. Then, the duration (either staccato or legato) and velocity values join the obtained pitch to conform to a MIDI event sent to the Orchestration agent.

3. Harmony Agent (HA)

3.1. Harmony Rhythm Generation

3.1.1. Accent/Beat Follower

It uses the Moved List and the Beat list to create an interpolated list of loop-indexed values and uses a seeded random generator to progressively adhere to the Moved list producing syncopation or to the beat list for each onset. Its parameter is *Accent/Beat* value. Returns an interpolated list of loop-indexed values for rhythmic accents.

3.1.2. Additive and Subtractive Filling algorithm (same as in Melody Rhythm generation)

It uses the Accent/beat follower list to add or subtract onsets. Its parameter is Harmony *+/- onsets*, and *seed rhy* for harmony.

3.1.3. Staccato-Legato Duration (same as in Melody Staccato-legato):

It produces a seeded random sequence of binary staccato/legato values applied to currently available onsets. It calculates *staccato* or *legato* duration. Returns a collection of loop-indexed values where 0 means not-onset, 1 staccato, and a duration value means legato. Its parameter is Harmony *Staccato-Legato* value. Returns a collection of onset values and their durations (a non-zero value for legato, 1 for staccato, 0 for not-onset).

3.2. Velocity Generation HA

It uses as input the harmony *velocity* parameter value. It gives accent —higher value— to onsets from the Moved List. Returns a MIDI velocity value for every chord onset or every arpeggio pitch.

3.3. Harmony Sequence Generation

3.3.1. Switch Harmony

It selects from the currently available harmony onsets (chords) an incremental amount of chord changes with a maximum of four changes per loop. It also prevents changes in the last beat to I and from I in the first beat to allow longer resolutions. Its parameter is *Harmonic Rhythm Speed*, currently independent from the interpolator. Returns a collection of loop-indexed Booleans wherein true values trigger a chord switch in the Harmonic Progression selector.

3.3.2. Harmonic Progression Selector

3.3.2.1. Chord Pools manager

Stores and manages harmonic content and progression in a phrase. Using the *Complexity* parameter value, it establishes the number of chord pools (collections of chord possibilities) and selects harmonic content available for each complexity level. The amount of chord-pools and their content changes progressively with the complexity level (e.g., at *Complexity* value 3, there are 3 ordered chord pools from which chords available belong to level 3). Also, for the first four chord pools levels, the Boolean Major/Minor parameter enables chords form the selected category. Each chord pool has a timely place in the progression (i.e., ordered), which means that chord pool # 1 can only be followed by

chord pool # 2, or by # 1 again. This means that more pools are present on higher complexity levels. Also, first pools are present in more levels so they have more level possibilities than the higher pools. The Chord Pool # is used as a reference for other processes to do phrase quantization (allow changes only at the end/beginning of a phrase). The chords in a chord pool are designed to connect to those found in contiguous chord pools, following usual western progression sequences. Currently, the algorithm holds a maximum of eight chord pools per phrase and nine complexity levels. This enables longer and increasingly complex progressions and also responsive level-change reaction in the middle of a phrase. The input is a trigger that executes a chord selection from the following pool using a seeded random generator. It returns a symbol—I, ii, V, etc.—that pairs with the *tension* parameter to query the Chord Dictionary, whose output is the Pitch Class List for harmony and melody that feeds their respective filters. The melody list adds 3-1 levels (i.e., pitches) on lower tensions to provide interpolation possibilities. As a feature, a random generator uses a normal distribution to decide if the progression modulates in next round, when there are dominant chords. The progression may continue with the next box if no modulation is decided or change the root (*tonic* pitch class value) and select Chord Pool #1 for next round.

3.3.2.2. Bass Selector

It provides possibilities for the lowest pitch used in the chord builder (see below) according to the complexity level and the chord pool number. The unit is an index of the chord list #5 for each chord found in the Chord Dictionary. This particular list has five pitch classes (e.g., I: 0, 4, 7, 11, 2). For example, if a I is followed by 1, the chord is going to use as bass the first pitch class of the list #5 which is the root. The number 2 would be the major third (4), and the 3 the fifth (7) in a similar way inversions are classified in the western canon.

3.3.3. Chord Dictionary Query

The Chord Dictionary holds a series of pitch-class lists for every chord, organized in the western tonal music classification (e.g., I, IV, V, vi, vii^{o}, etc.) Every chord key holds a series of lists that incrementally add pitch-classes based on the relative dissonance with the root, using the *Tension* parameter value (1-12). A query based on the tension value outputs a list of pitch classes normalized to C. The *tonic* pitch class value transposes the list to the current tonality (mod 12).

3.3.3.1. Post Filter

The resulting pitch class list passes through a filter that shifts the numbers to match a pitch class set that belongs to non-diatonic collections to introduce a level of dissonance if the *Dim* toggle is activated. The result is a similar pitch class set whose dissonance is instantaneously higher than the original.

3.3.4. Chord builder

Using the pitch class list for harmony that comes from Chord Dictionary (key is the chord and list number for tension), it builds a whole-range list of possible pitch classes. Then, using the *Voices*, *Range*, *Register*, and the bass pitch class selected, it builds a list of pitches used for playback either as a vertical chord triggered by the onset stored on the Harmony Rhythm collection, or as arpeggio. Its algorithm organizes interval classes from larger in the bass to shorter in the higher register to emulate orchestration. Using the *movement* parameter, it shifts the chords in the register going up/down for every onset restarting with the loop. The maximum range depends on the onsets per loop and the range established by the movement parameter.

3.3.5. Arpeggiator

3.3.5.1. Arpeggio Selector/activator/enabler

Using the orchestration configuration (see orchestrator flags), the Melody and Harmony +/- *onsets* parameter values, the *Complexity* parameter value, and the *Size* value, it chooses the rate (1/4, 1/8, 1/8 triplet, 1/16, and 1/32 note), the shape (up, down, and up&down) of the arpeggiator, and also triggers a flip switch (on/off). It conditions the arpeggiator speed possibilities to avoid faster melodies and/or chord onsets, turns it off if brass is solo, and turns brass off if it flips on.

3.3.5.2. Arpeggio Generator

It assigns an arpeggio to the current chord voices, calculates and updates the arpeggio rate using current tempo, and sends the pitches for playback. Its parameters are *switch on/off*, *arpeggio rate* value (1/4, 1/8, 1/8 triplet, 1/16, and 1/32 note), and *arpeggio shape* (up, down, and up&down).

3.4. Harmony Change Algorithm

Using the current chords, it prepares and sends an offset to *Register* and Harmony +/- *onsets* parameter values. The variations are classified in default and dominant. The execution is allowed when there has not been any change in those parameters for a selected number of phrases. It uses a random generator.

3.5. MIDI Event Construction HA

Using the length, velocity and pitch, produces the harmony MIDI event stream.

Operation

Like in the MA, the Loop Count pulls non-zero values from the rhythm onset collection. If and when found, HA produces a chord or an arpeggio. Then, duration (either staccato or legato) and velocity values join the obtained chord pitches to conform the MIDI events sent to the Orchestration agent.

4. Percussion Agent (PA)

4.1. Percussion Rhythm Generation

4.1.1. Accent/Beat Follower (same as in Harmony Rhythm generation)

It uses the Moved List and the Beat list to interpolate and add syncopation for all percussion instruments. Its parameter is *Accent/Beat* value. Returns an interpolated list of onset loop indexes.

4.1.2. Additive and Subtractive Filling algorithm (same as in Melody/Harmony Rhythm generation)

For each percussion instrument, it uses the interpolated list from Accent/beat follower. Its parameter is *Kick/Snare*, *Toms*, *Cymbals +/- onsets*. Returns a collection of loop-indexed values of 0 for not-onset (no duration calculated.

4.1.2.1. Add/Sub for Kick/Snare: It distributes the onsets between the Kick drum and snare drum timbres. It receives a *consecutive* value that sets how many snare drums are consecutive before a kick drum appears. It also receives an *offset* value that shifts the pattern.

4.2. Velocity Generation PA

For each percussion instrument, it uses as input the *velocity* parameter value of the Percussion Agent. It gives accent —higher value— to onsets from the Moved List. Returns a MIDI velocity.

4.3. Ratchet

It decides when and how to do a ratchet for each instrument except the kick drum using a random generator. The decision depends on the *+/- onsets* value for each instrument, allowing execution only on lower values (under 31% for each instrument). It uses a random generator to decide the type of ratchet (64^{th}, triplet 32^{nd} , 32^{nd}, and triplet 16^{th}, note values in four or two 8^{th} note lenghts). It also calculates the velocity line for the ratchet with a random generator added to an ascending value.

4.4. Percussion Change

Using the current chords, it sends a trigger to the percussion instruments to offset their parameters. The variations are classified in tonic, dominant, and default. It shifts the *consecutive* value in the Kick/Snare and *+/- onsets* value on all instruments using a random generator to support phrasing, with a specific bipolar range per instrument.

4.5. MIDI event construction PA

It manages the range for each percussion instrument in MIDI pitch values using *from* and *range* (*from* is the lower MIDI pitch for the instrument and *range* is the interval of pitches that can be used to alternate the timbre in the sampler configuration). Using velocity, pitch, and an arbitrary constant length (in this case 300 ms), produces a real-time sequence of MIDI events sent to a MIDI channel and port.

Operation

Like in the MA and HA, the Count pulls non-zero values from the rhythm onset collection in each percussion instrument. If and when found, a trigger to produce an onset is sent. Then, duration (a constant) and velocity values join the obtained percussion pitches to conform and dispatch the MIDI events.

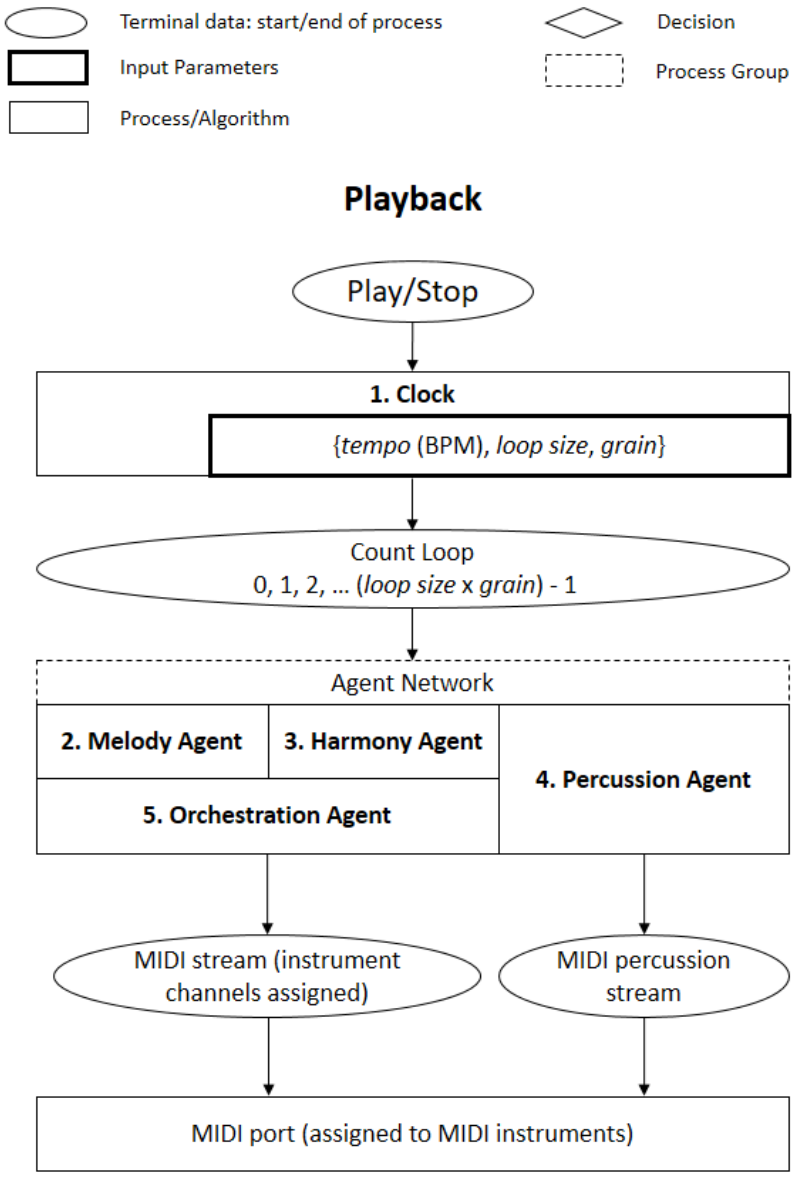
5. **Orchestration Agent (OA)**
   Using a seeded random generator, it manages the instrument family assigned to the Melody/Harmony agents.
   5.1. Instrument adds/subtract:
      Adds/subtract for each agent using the *Orch +/-* parameter for each agent. It distributes the instrument families among the HA stream voices using HA *velocity* (intensity acquired with overlapping), *voices* and *range* parameters.
   5.2. Instrument switch:
      Switches after two phrases with no orchestration changes.
   5.3. Pitch range set the among instrument voices for MA:
      Operates using the *Orch Range* parameter and the Pitch Class List for Melody.
   5.4. Flags send/receive:
      To the arpeggiator algorithm *brass solo* and *piano solo*, and receives arpeggiator on/off.
   5.5. Instrument-family range management:
      Uses pitch range in MIDI pitch values (non-real-time parameter).
   5.6. MIDI event submission:
      It sends the final MIDI events to the MIDI channels and ports specified.

6. **Multi-Parameter Preset Interpolator and Manager (MPIC)**
   Stores and manages most of the agent parameters (I/O) and the multi-parameter presets, and assigns them to the interpolator slots for each agent independently. A group of multi-parameter presets can be understood as a *theme* and each preset as a *style*. The interpolation among preset slots (styles) provides seamless musical transitions among them within a theme. In the example provided, there are four presets per theme, and the interpolation is executed by three XY pads for each of the agents —the preset positions are in the four corners. The cursors' coordinates on the XY pads are used to calculate interpolation. The presets can be elaborated as progressive developments that branch from an original mood. In addition, individual parameters can be offset during playback and will keep their relative position during interpolation, or return to the interpolated value using the *reset interpolator* trigger. In the example, it is possible to save a multi-parameter preset that includes all the agents' parameters in each of the four theme's slots. Then, to assign the theme to the interpolator, the user clicks in the preset row header, populating the four corners of the XY pads with the presets (1^{st}: lower-left, 2^{nd}: lower-right, 3^{rd}: upper-left, 4^{th}: upper-right.)

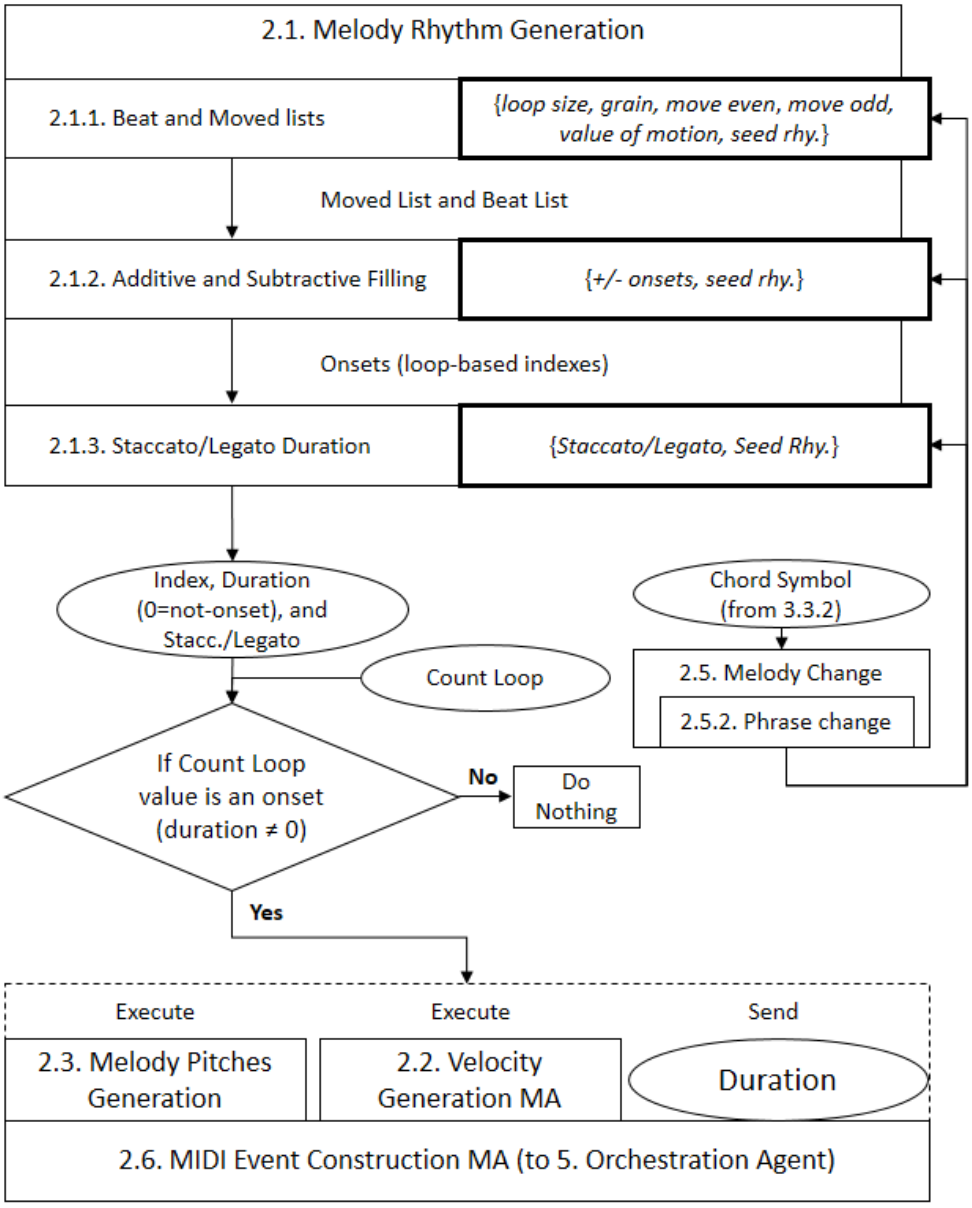
7. **Seeded phrase generator**
   It takes the seed values for rhythm and pitch and adds a seeded random value for every new Chord pool, restarting the sequence at Chord pool #1 (phrase start), to provide a sequence of seeds whose length can change the behavior of the music content for every chord, or provide a level of recurrence by phrase. The length of this sequence is set by the *Seeded Phrase Length* parameter, which is the number of values to be added to the first seed. If the current seed is 0, the random generators

produce non-seeded values (the seed taken is date and time), which is ideal for ‘improvisational’ passages.

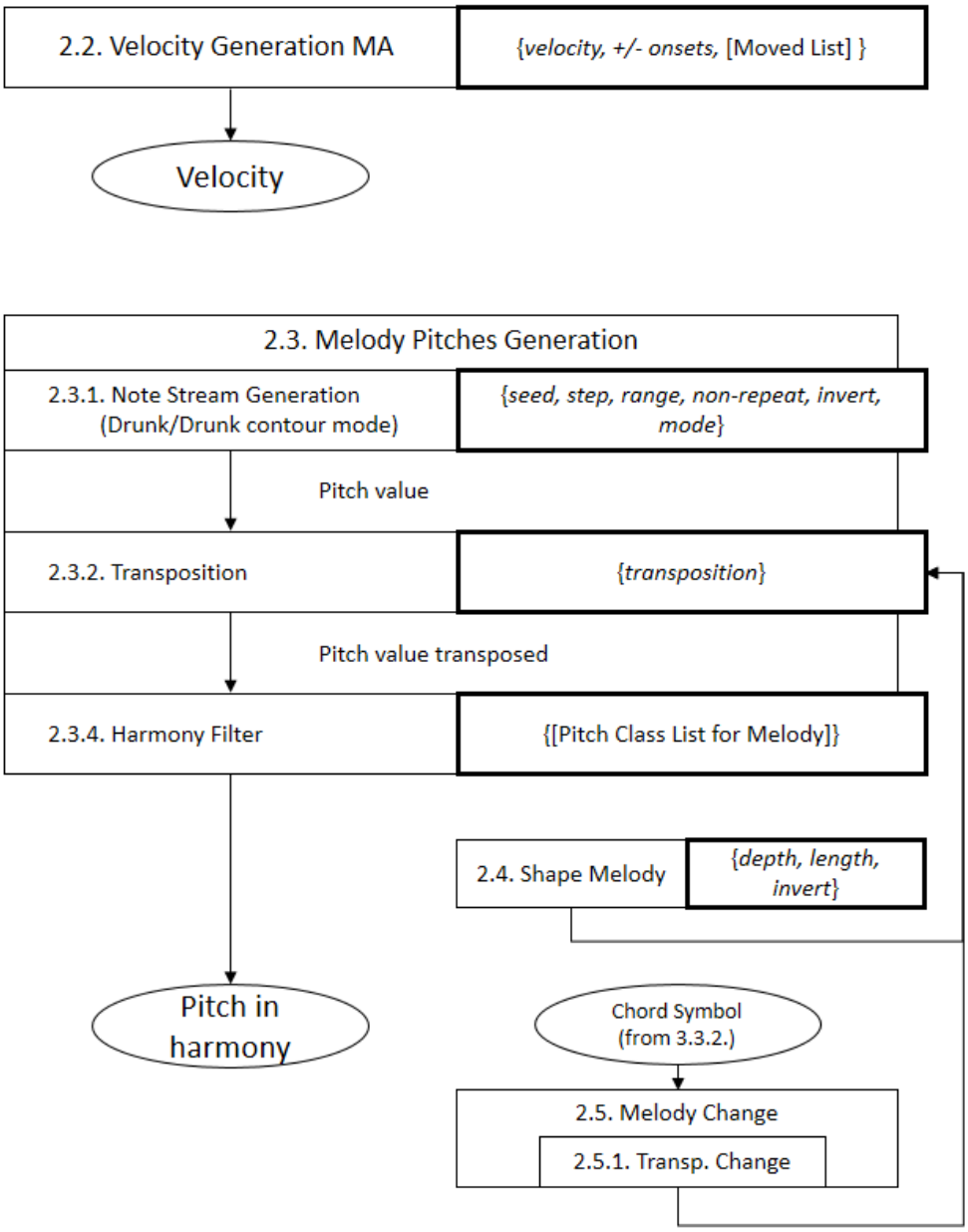
Algorithm Graph:



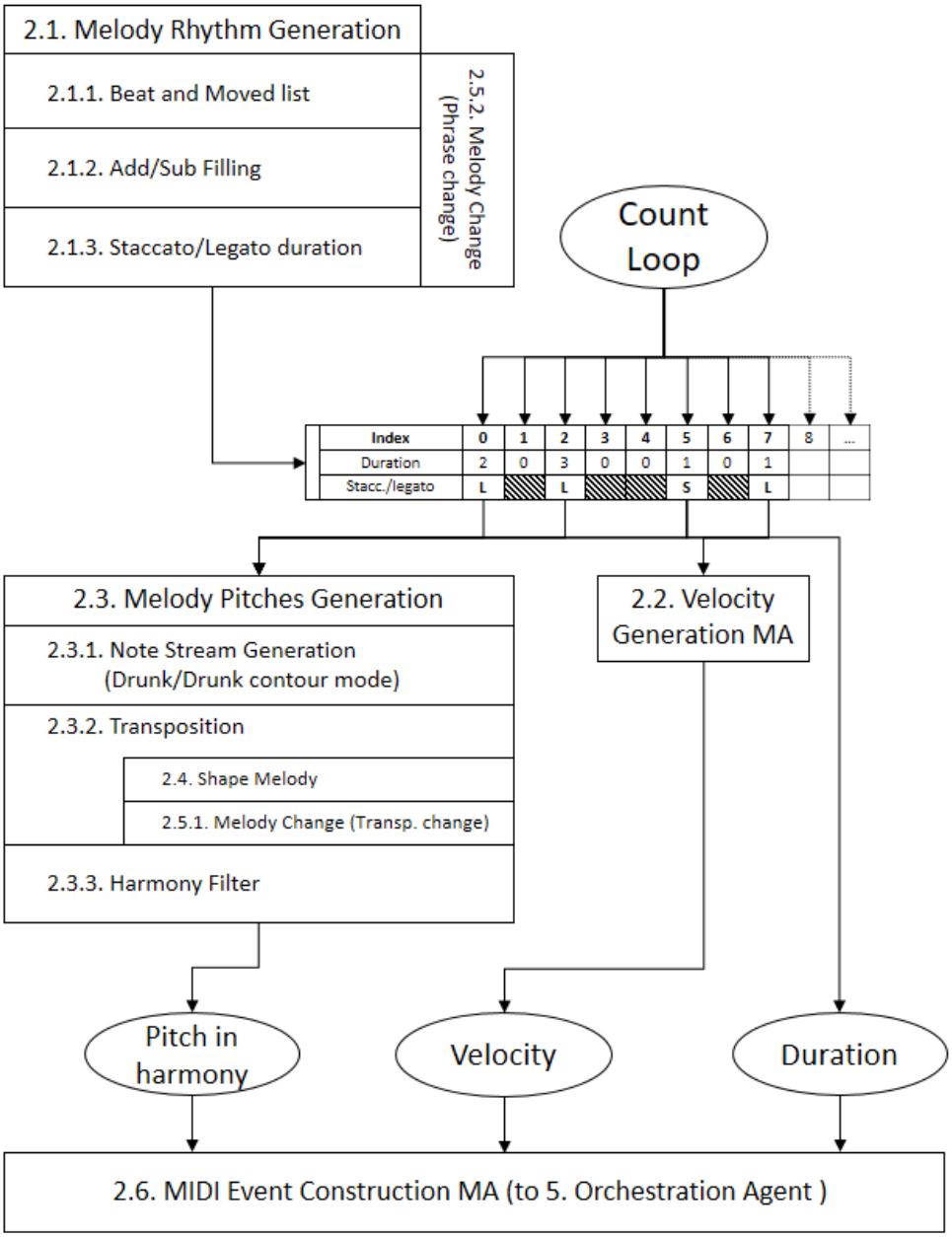
2. Melody Agent (MA): Part 1



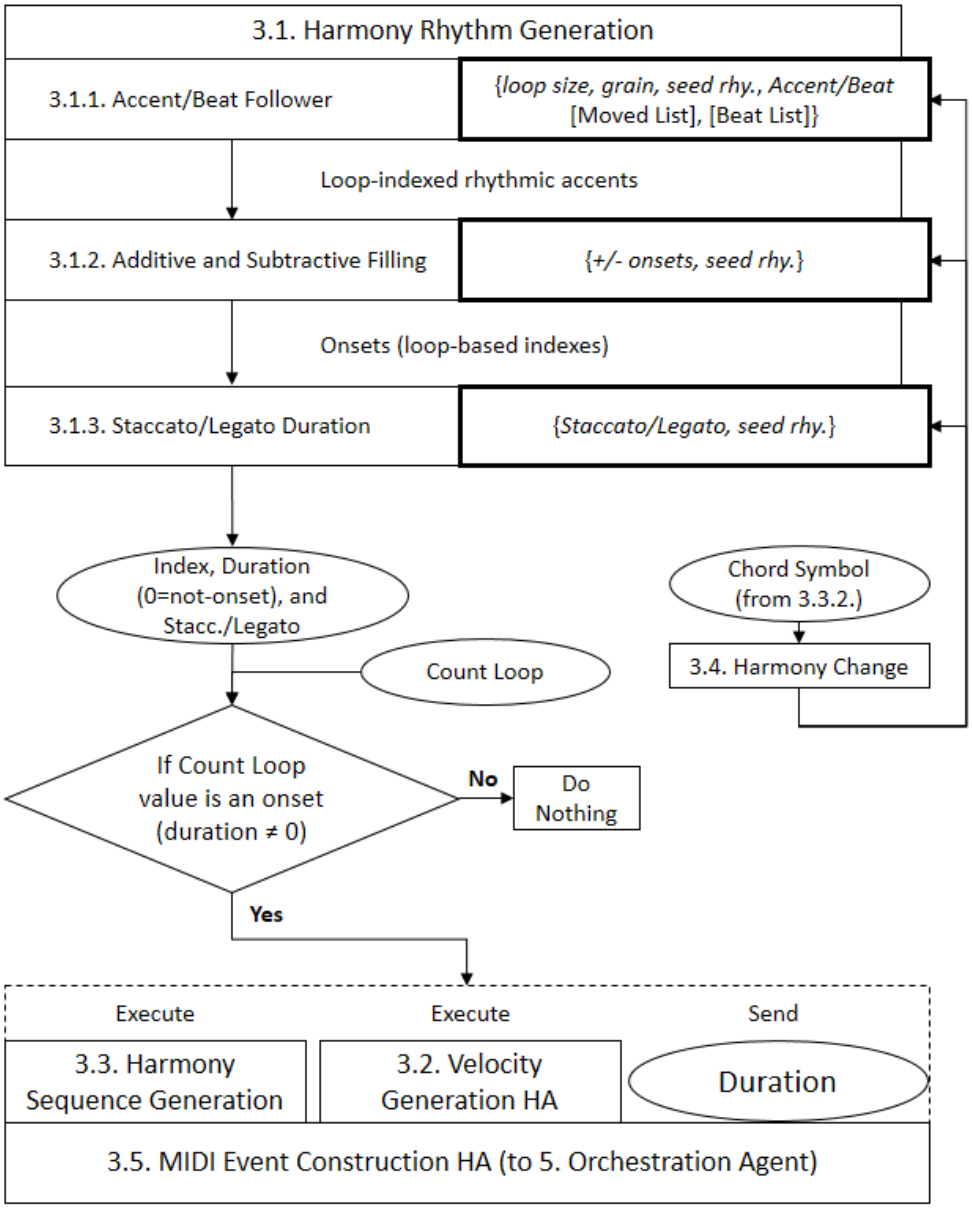
2. Melody Agent (MA): Part 2



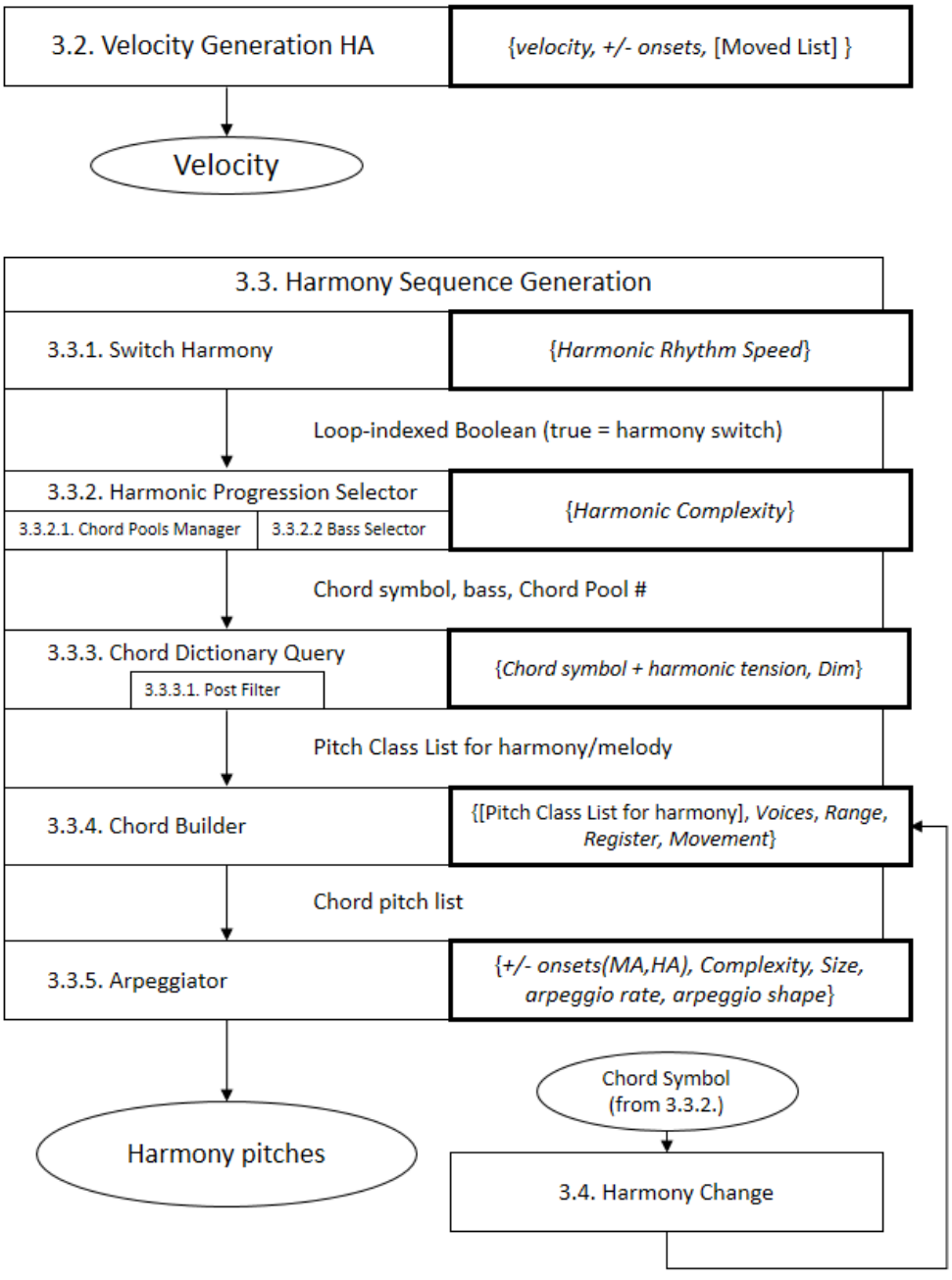
2. Melody Agent: Overview



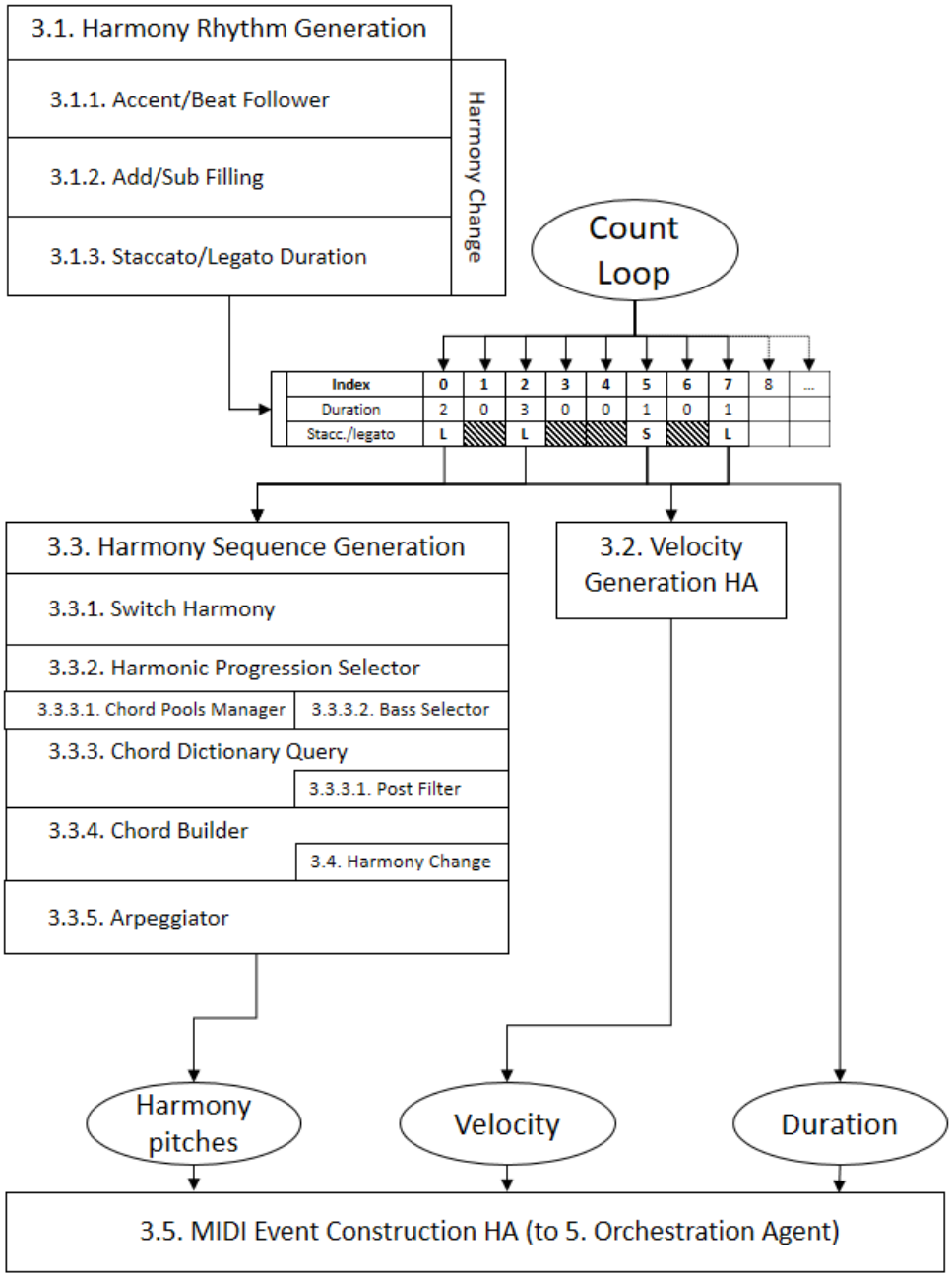
3. Harmony Agent (HA): Part 1

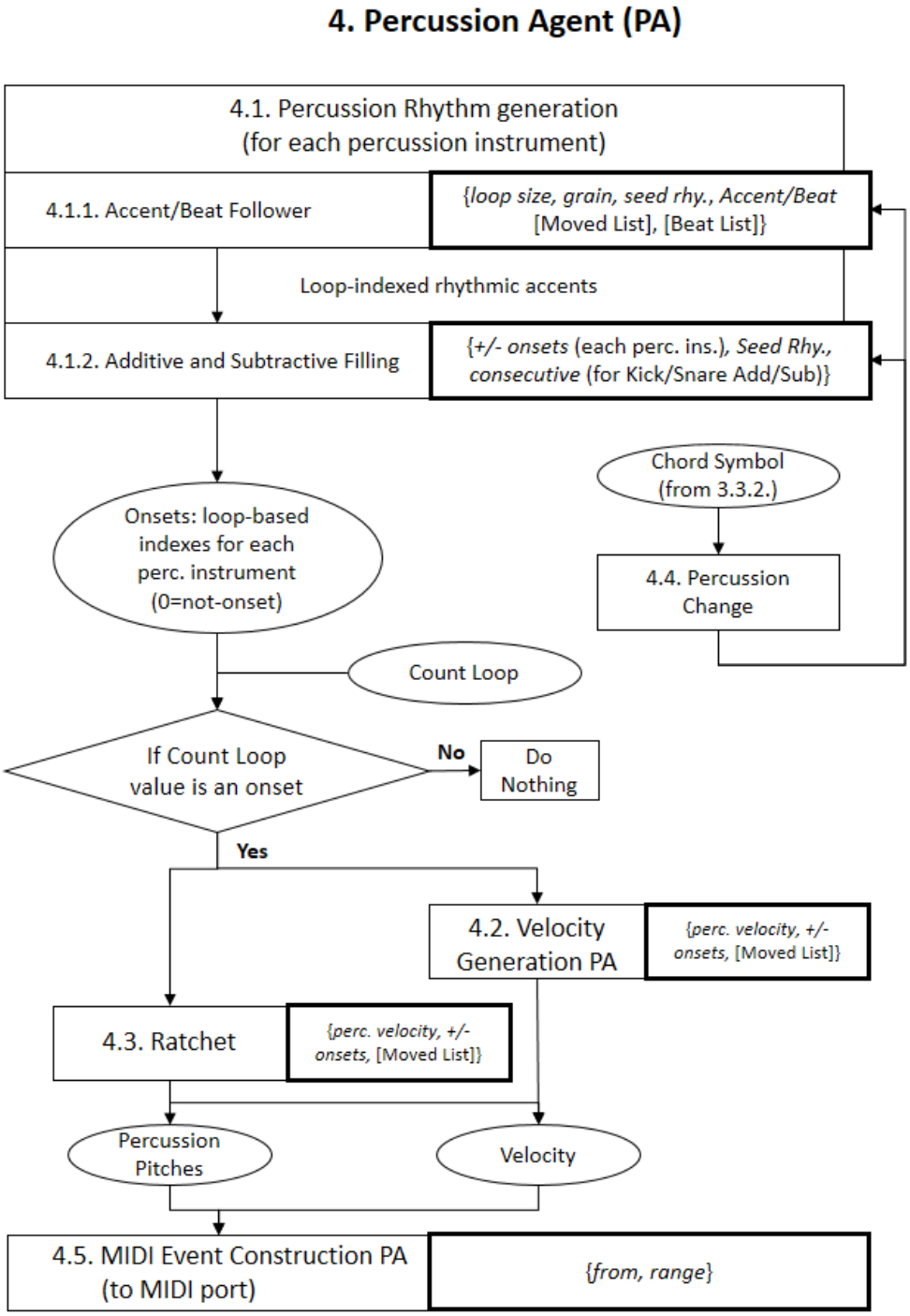


3. Harmony Agent: Part 2

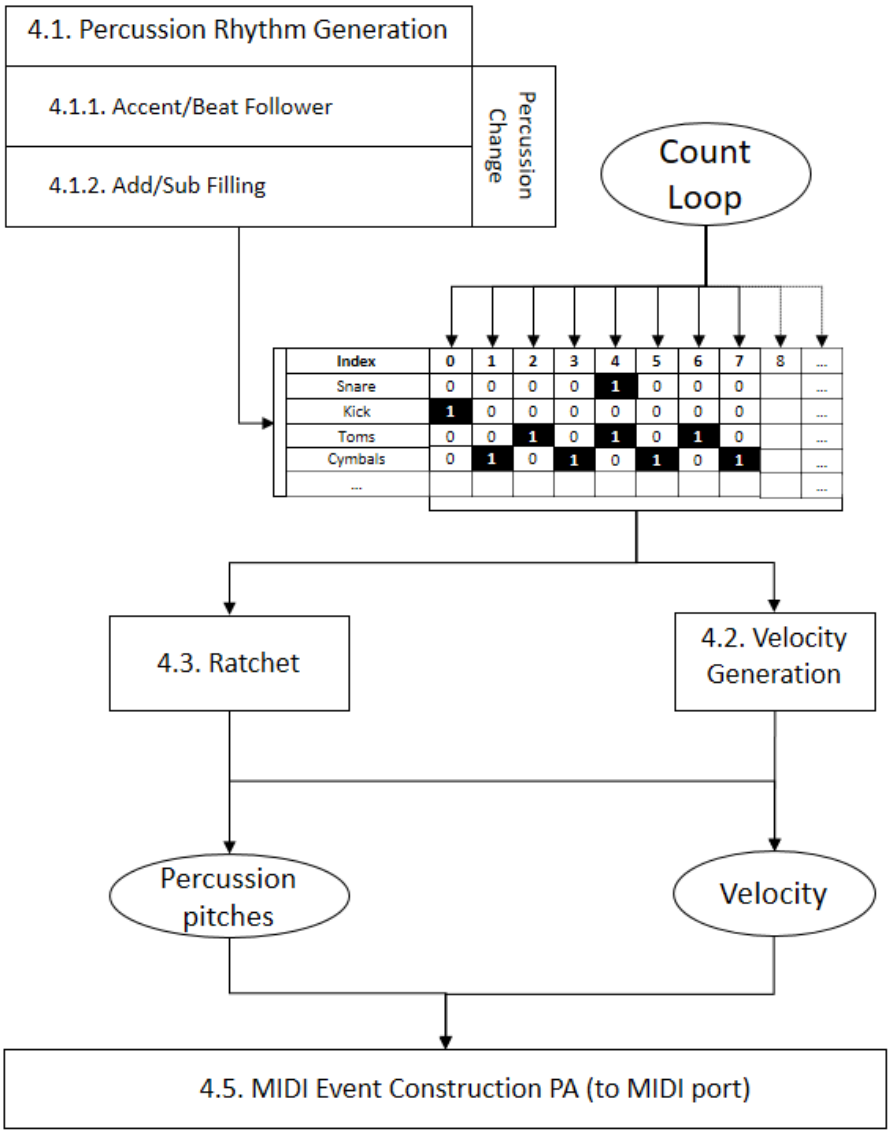


3. Harmony Agent (HA): Overview

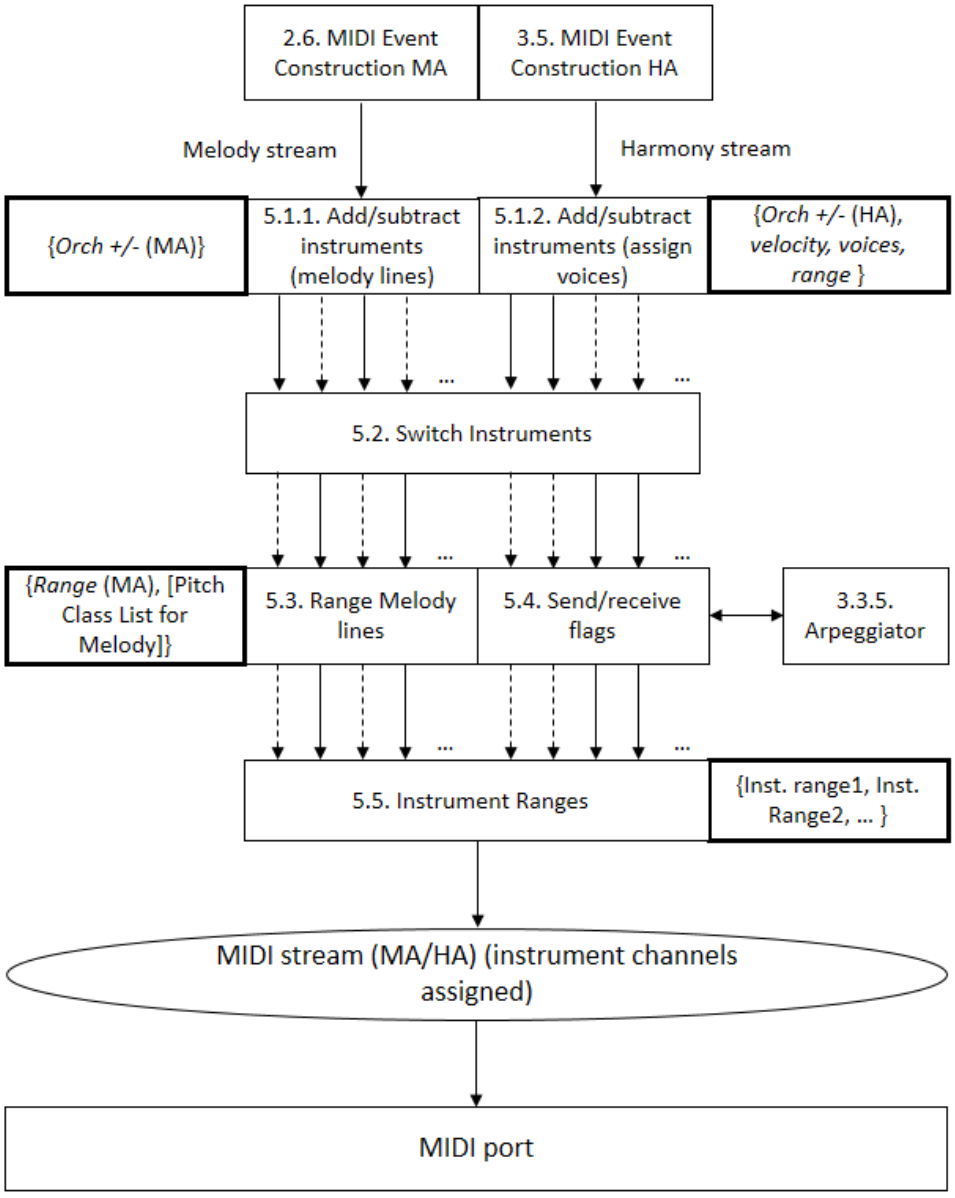



4. Percussion Agent (PA)
4.1. Percussion Rhythm generation
(for each percussion instrument)
4.1.1. Accent/Beat Follower
{loop size, grain, seed rhy., Accent/Beat [Moved List], [Beat List]}
Loop-indexed rhythmic accents
4.1.2. Additive and Subtractive Filling
{+/- onsets (each perc. ins.), Seed Rhy., consecutive (for Kick/Snare Add/Sub)}
Chord Symbol (from 3.3.2.)
Onsets: loop-based indexes for each perc. instrument (0=not-onset)
4.4. Percussion Change
Count Loop
If Count Loop value is an onset
No
Do Nothing
Yes
4.2. Velocity Generation PA
{perc. velocity, +/- onsets, [Moved List]}
4.3. Ratchet
{perc. velocity, +/- onsets, [Moved List]}
Percussion Pitches
Velocity
4.5. MIDI Event Construction PA (to MIDI port)
{from, range}

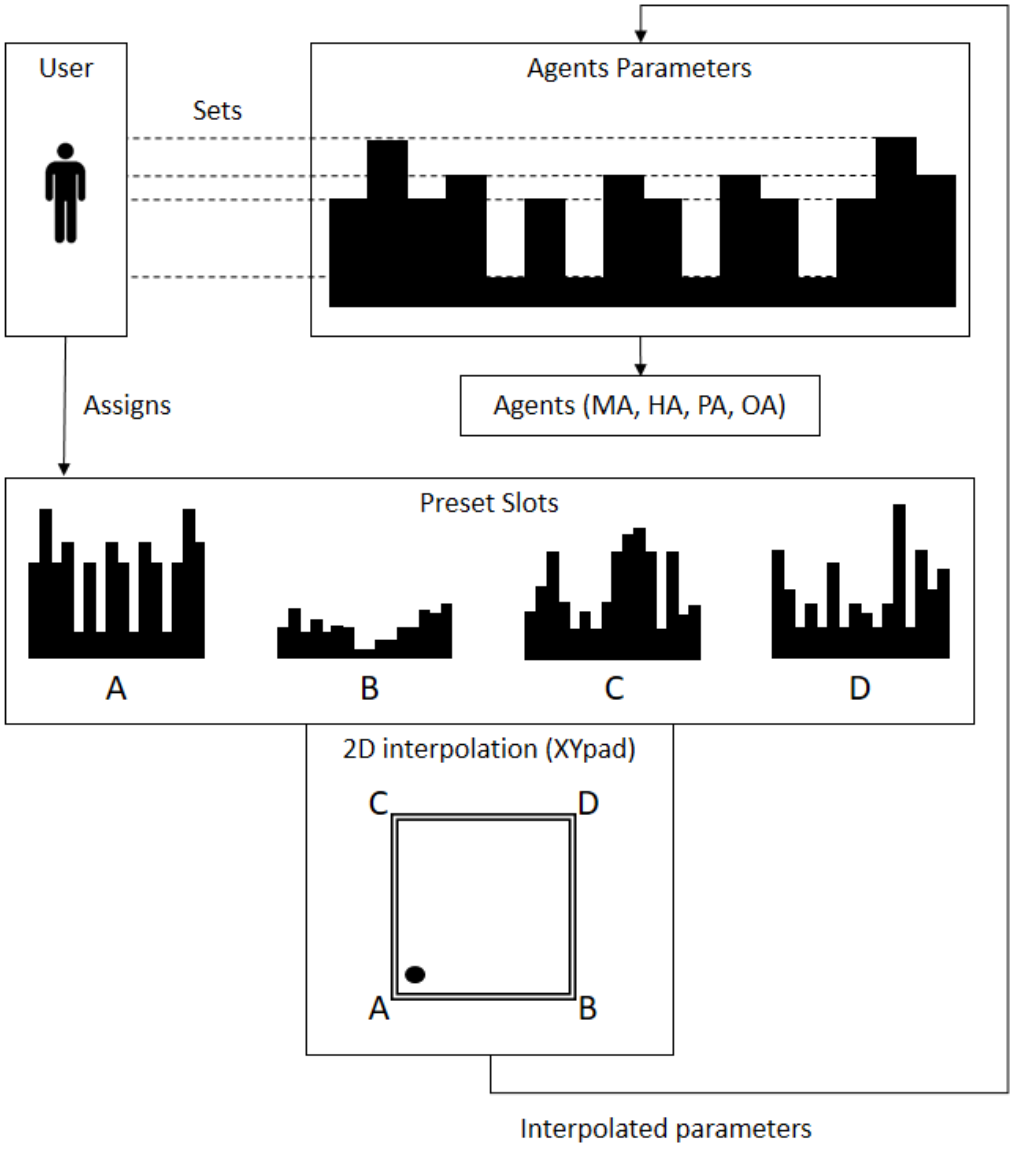
4. Percussion Agent: Overview



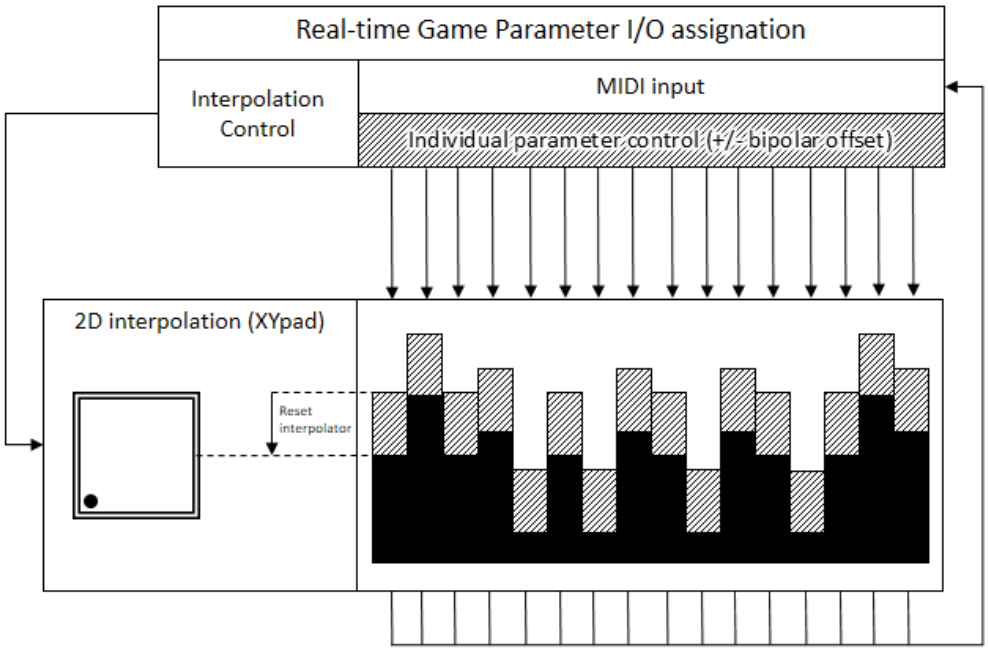
5. Orchestration Agent (OA)



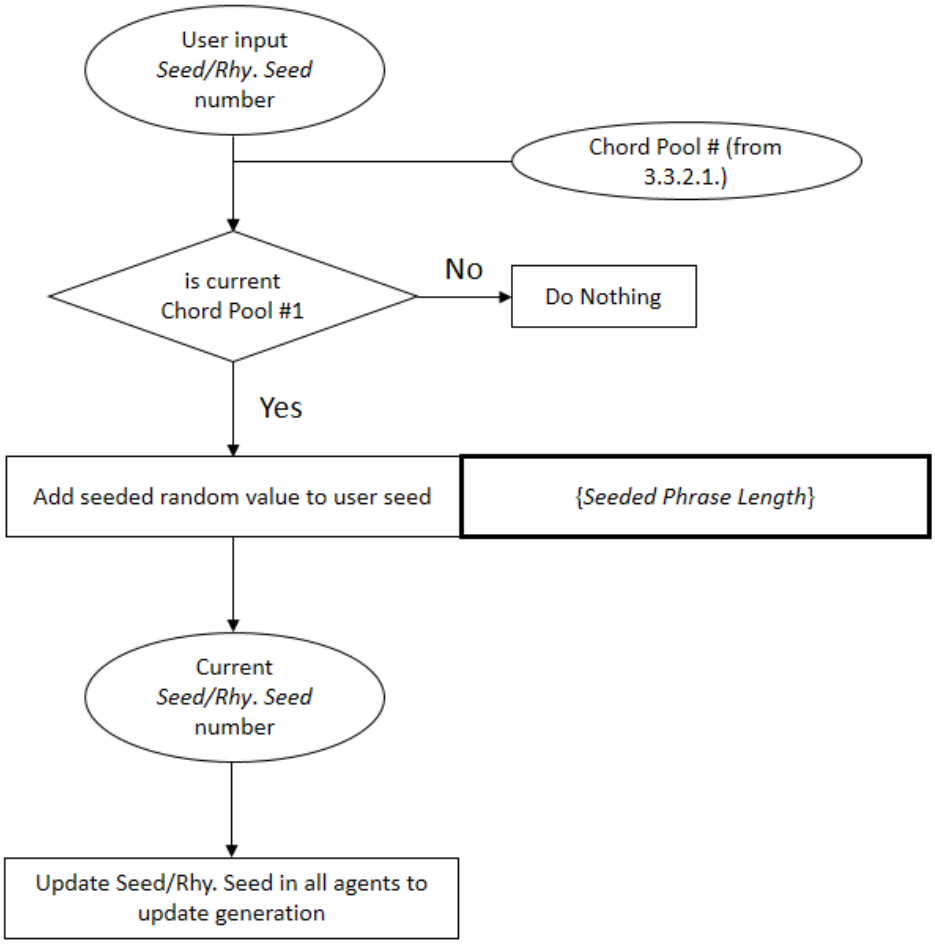
6. Multi-Parameter Preset Interpolator and Manager: Control and Assign



6. Multi-Parameter Preset Interpolator and Manager: Game I/O



7. Seeded Phrase Generator



Appendix B: Chord Pools

	B1	B2	B3	B4	B5	B6	B7	B8
L1	I1, i1							
L2	I1, i1	IV3, iv3, vi2, bVI2, I2, i2						
L3	I1, i1	IV1,iv1 vi1, bVI1						
L4	I1, i1	IV1, iv1, bVI1, vi1, ii1/2, ii°1/2	V3					
L5	I1, i1	iii1, vi1/3, bVI1/3, I2, i2	ii1/2, IV1/2	V1				
L6	I1, i1	IV1/2, vi1/3, bVI1/3, I2, i2	ii1/2, V7/V1/2*, ii°1/2	V1, vii°1/2*, bVII1/2*				
L7	I1, i1	iv1/2, V7/ii1/2/3, V7/IV1/2/3	ii1/2/3*, IV1/2/3*, bIV1/2/3*	ii°1/2/3, V7V1/2/3*, V7/bVII1/2/3*	V1/2/3*, vii°1/2/3*, bVII1/2/3*	I1/2/3*, vi1/2/3*, i1/2/3*, bIII1/2/3*		
L8	I1, i1	V7/vi1/2, V7/ii1/2/3, V7/IV2/3, V7bVI1/2, bIII1/2	I°1/2/3*, bVII1/2/3*, ii°m7/v1/2/3*, ii/bV1/2/3*, ii1/2/3*, ii°m7/biii1/2/3*, ii°m7/bv1/2/3*, bVI1/2/3*, iv1/2/3*	bVI1/2/3*, V7/V1/2/3*, vii°1/2/3*, bII7-N7 1/2/3*	I3, i3	V1/2/3*, vii°1/2/3*, bVII1/2/3*	I1/2/3*, vi1/2/3*, i1/2/3*, IV1/2/3*	
L9	I1, i1	ii°m7/v1/3, ii/bVI1/2, ii°m7/iv2/3, V7/bVI1/3, ii°m7/ii1/3	V7/bVII1/2/3*, bVII1/2/3*, v1/2/3*, ii°m7/ii1/2/3*, V7/IV1/2/3*, ii1/2/3*, ii°m7/biii1/2/3*, ii/bV1/2/3*, ii/bV1/2/3*, bVI1/2/3*, V7/V1/2/3*, V7/ii1/2/3*	V7/ii1/2/3*, ii/bII1/2/3*, ii°1/2/3*, bVI1/2/3*	bVI7(no5)-It+1/2/3*, bVI7(b5)-Fr+1/2/3*, V7/bii-Ger+1/2/3*, bII7-N7 1/2/3*	I3, i3	V1/2/3*, vii°1/2/3*, bVII1/2/3*	I1/2/3*, vi1/2/3*, i1/2/3*, IV1/2/3*, IV2, bVI2, iv3, bIII1/2

The level of complexity is managed by the *Harmonic Complexity* value (L1-L9) and produces a different amount of possible boxes per harmonic cycle. The number (1, 2, or 3) after the chord symbol is the *bass* employed, and can be tied to the *inversion* number (1 = root, 2 = third, and 3 = fifth).

Appendix C: Harmony dictionary

```
{
        "I" :       {
                    "1" : [ 0 ],
                    "2" : [ 0, 7 ],
                    "3" : [ 0, 4, 7 ],
                    "4" : [ 0, 4, 7, 11 ],
                    "5" : [ 0, 4, 7, 11, 2 ],
                    "6" : [ 0, 4, 7, 11, 2, 5 ],
                    "7" : [ 0, 4, 7, 11, 2, 5, 9 ],
                    "8" : [ 0, 4, 7, 11, 2, 5, 9, 3 ],
                    "9" : [ 0, 4, 7, 11, 2, 5, 9, 3, 10 ],
                    "10" : [ 0, 4, 7, 11, 2, 5, 9, 3, 10, 8 ],
                    "11" : [ 0, 4, 7, 11, 2, 5, 9, 3, 10, 8, 6 ],      ’
                    "12" : [ 0, 4, 7, 11, 2, 5, 9, 3, 10, 8, 6, 1 ]
        }
’
        "ii" :      {
                    "1" : [ 2 ],
                    "2" : [ 2, 9 ],
                    "3" : [ 2, 5, 9 ],
                    "4" : [ 2, 5, 9, 0 ],
                    "5" : [ 2, 5, 9, 0, 4 ],
                    "6" : [ 2, 5, 9, 0, 4, 7 ],
                    "7" : [ 2, 5, 9, 0, 4, 7, 11 ],
                    "8" : [ 2, 5, 9, 0, 4, 7, 11, 6 ],
                    "9" : [ 2, 5, 9, 0, 4, 7, 11, 6, 10 ],
                    "10" : [ 2, 5, 9, 0, 4, 7, 11, 6, 10, 8 ],
                    "11" : [ 2, 5, 9, 0, 4, 7, 11, 6, 10, 8, 1 ],
                    "12" : [ 2, 5, 9, 0, 4, 7, 11, 6, 10, 8, 1, 3 ]    ’
        }
’
        "iii" :     {
                    "1" : [ 4 ],
                    "2" : [ 4, 11 ],
                    "3" : [ 4, 7, 11 ],
                    "4" : [ 4, 7, 11, 2 ],
                    "5" : [ 4, 7, 11, 2, 0 ],
                    "6" : [ 4, 7, 11, 2, 0, 9 ],
                    "7" : [ 4, 7, 11, 2, 0, 9, 5 ],
                    "8" : [ 4, 7, 11, 2, 0, 9, 5, 6 ],
                    "9" : [ 4, 7, 11, 2, 0, 9, 5, 6, 1 ],
                    "10" : [ 4, 7, 11, 2, 0, 9, 5, 6, 1, 8 ],
                    "11" : [ 4, 7, 11, 2, 0, 9, 5, 6, 1, 8, 10 ],
                    "12" : [ 4, 7, 11, 2, 0, 9, 5, 6, 1, 8, 10, 3 ]    ’
        }
’
        "IV" :      {
                    "1" : [ 5 ],
                    "2" : [ 5, 0 ],
                    "3" : [ 5, 9, 0 ],
                    "4" : [ 5, 9, 0, 4 ],
                    "5" : [ 5, 9, 0, 4, 2 ],
                    "6" : [ 5, 9, 0, 4, 2, 7 ],
                    "7" : [ 5, 9, 0, 4, 2, 7, 10 ],
                    "8" : [ 5, 9, 0, 4, 2, 7, 10, 8 ],
                    "9" : [ 5, 9, 0, 4, 2, 7, 10, 8, 3 ],
                    "10" : [ 5, 9, 0, 4, 2, 7, 10, 8, 3, 1 ],
                    "11" : [ 5, 9, 0, 4, 2, 7, 10, 8, 3, 1, 11 ],
                    "12" : [ 5, 9, 0, 4, 2, 7, 10, 8, 3, 1, 11, 6 ]    ’
        }
’
        "V" :       {
                    "1" : [ 7 ],
                    "2" : [ 7, 2 ],
                    "3" : [ 7, 11, 2 ],
                    "4" : [ 7, 11, 2, 5 ],
                    "5" : [ 7, 11, 2, 5, 9 ],
                    "6" : [ 7, 11, 2, 5, 9, 4 ],
                    "7" : [ 7, 11, 2, 5, 9, 4, 0 ],
                    "8" : [ 7, 11, 2, 5, 9, 4, 0, 10 ],
                    "9" : [ 7, 11, 2, 5, 9, 4, 0, 10, 3 ],
                    "10" : [ 7, 11, 2, 5, 9, 4, 0, 10, 3, 1 ],
                    "11" : [ 7, 11, 2, 5, 9, 4, 0, 10, 3, 1, 8 ],
                    "12" : [ 7, 11, 2, 5, 9, 4, 0, 10, 3, 1, 8, 6 ]    ’
        }
’
        "vi" :      {
                    "1" : [ 9 ],
                    "2" : [ 9, 4 ],
                    "3" : [ 9, 0, 4 ],
                    "4" : [ 9, 0, 4, 7 ],
                    "5" : [ 9, 0, 4, 7, 5 ],
                    "6" : [ 9, 0, 4, 7, 5, 2 ],
                    "7" : [ 9, 0, 4, 7, 5, 2, 11 ],
                    "8" : [ 9, 0, 4, 7, 5, 2, 11, 3 ],
                    "9" : [ 9, 0, 4, 7, 5, 2, 11, 3, 1 ],
                    "10" : [ 9, 0, 4, 7, 5, 2, 11, 3, 1, 6 ],
                    "11" : [ 9, 0, 4, 7, 5, 2, 11, 3, 1, 6, 8 ],
                    "12" : [ 9, 0, 4, 7, 5, 2, 11, 3, 1, 6, 8, 10 ]    ’
        }
’
```

```
        "vii°" :    {
                    "1" : [ 11 ],
                    "2" : [ 11, 5 ],
                    "3" : [ 11, 2, 5 ],
                    "4" : [ 11, 2, 5, 9 ],
                    "5" : [ 11, 2, 5, 9, 7 ],
                    "6" : [ 11, 2, 5, 9, 7, 4 ],
                    "7" : [ 11, 2, 5, 9, 7, 4, 1 ],
                    "8" : [ 11, 2, 5, 9, 7, 4, 1, 6 ],
                    "9" : [ 11, 2, 5, 9, 7, 4, 1, 6, 8 ],
                    "10" : [ 11, 2, 5, 9, 7, 4, 1, 6, 8, 3 ],
                    "11" : [ 11, 2, 5, 9, 7, 4, 1, 6, 8, 3, 10 ],
                    "12" : [ 11, 2, 5, 9, 7, 4, 1, 6, 8, 3, 10, 0 ]
        }


        "V7/IV" :   {
                    "1" : [ 0 ],
                    "2" : [ 0, 7 ],
                    "3" : [ 0, 4, 7 ],
                    "4" : [ 0, 4, 7, 10 ],
                    "5" : [ 0, 4, 7, 10, 2 ],
                    "6" : [ 0, 4, 7, 10, 2, 5 ],
                    "7" : [ 0, 4, 7, 10, 2, 5, 9 ],
                    "8" : [ 0, 4, 7, 10, 2, 5, 9, 3 ],
                    "9" : [ 0, 4, 7, 10, 2, 5, 9, 3, 11 ],
                    "10" : [ 0, 4, 7, 10, 2, 5, 9, 3, 11, 8 ],
                    "11" : [ 0, 4, 7, 10, 2, 5, 9, 3, 11, 8, 6 ],
                    "12" : [ 0, 4, 7, 10, 2, 5, 9, 3, 11, 8, 6, 1 ]
        }
        "V7/V" :    {
                    "1" : [ 2 ],
                    "2" : [ 2, 9 ],
                    "3" : [ 2, 6, 9 ],
                    "4" : [ 2, 6, 9, 0 ],
                    "5" : [ 2, 6, 9, 0, 4 ],
                    "6" : [ 2, 6, 9, 0, 4, 7 ],
                    "7" : [ 2, 6, 9, 0, 4, 7, 11 ],
                    "8" : [ 2, 6, 9, 0, 4, 7, 11, 5 ],
                    "9" : [ 2, 6, 9, 0, 4, 7, 11, 5, 10 ],
                    "10" : [ 2, 6, 9, 0, 4, 7, 11, 5, 10, 8 ],
                    "11" : [ 2, 6, 9, 0, 4, 7, 11, 5, 10, 8, 1 ],
                    "12" : [ 2, 6, 9, 0, 4, 7, 11, 5, 10, 8, 1, 3 ]
        }
        "V7/vi" :   {
                    "1" : [ 4 ],
                    "2" : [ 4, 11 ],
                    "3" : [ 4, 8, 11 ],
                    "4" : [ 4, 8, 11, 2 ],
                    "5" : [ 4, 8, 11, 2, 0 ],
                    "6" : [ 4, 8, 11, 2, 0, 9 ],
                    "7" : [ 4, 8, 11, 2, 0, 9, 5 ],
                    "8" : [ 4, 8, 11, 2, 0, 9, 5, 6 ],
                    "9" : [ 4, 8, 11, 2, 0, 9, 5, 6, 1 ],
                    "10" : [ 4, 8, 11, 2, 0, 9, 5, 6, 1, 7 ],
                    "11" : [ 4, 8, 11, 2, 0, 9, 5, 6, 1, 7, 10 ],
                    "12" : [ 4, 8, 11, 2, 0, 9, 5, 6, 1, 7, 10, 3 ]
        }
        "V7/ii" :   {
                    "1" : [ 9 ],
                    "2" : [ 9, 4 ],
                    "3" : [ 9, 1, 4 ],
                    "4" : [ 9, 1, 4, 7 ],
                    "5" : [ 9, 1, 4, 7, 5 ],
                    "6" : [ 9, 1, 4, 7, 5, 2 ],
                    "7" : [ 9, 1, 4, 7, 5, 2, 11 ],
                    "8" : [ 9, 1, 4, 7, 5, 2, 11, 3 ],
                    "9" : [ 9, 1, 4, 7, 5, 2, 11, 3, 0 ],
                    "10" : [ 9, 1, 4, 7, 5, 2, 11, 3, 0, 6 ],
                    "11" : [ 9, 1, 4, 7, 5, 2, 11, 3, 0, 6, 8 ],
                    "12" : [ 9, 1, 4, 7, 5, 2, 11, 3, 0, 6, 8, 10 ]
        }
        "V7/bVI" :  {
                    "1" : [ 3 ],
                    "2" : [ 3, 10 ],
                    "3" : [ 3, 7, 10 ],
                    "4" : [ 3, 7, 10, 1 ],
                    "5" : [ 3, 7, 10, 1, 0 ],
                    "6" : [ 3, 7, 10, 1, 0, 8 ],
                    "7" : [ 3, 7, 10, 1, 0, 8, 5 ],
                    "8" : [ 3, 7, 10, 1, 0, 8, 5, 6 ],
                    "9" : [ 3, 7, 10, 1, 0, 8, 5, 6, 4 ],
                    "10" : [ 3, 7, 10, 1, 0, 8, 5, 6, 4, 1 ],
                    "11" : [ 3, 7, 10, 1, 0, 8, 5, 6, 4, 1, 9 ],
                    "12" : [ 3, 7, 10, 1, 0, 8, 5, 6, 4, 1, 9, 11 ]
        }
```

```
"V7/bVII" : {
            "1" : [ 5 ],
            "2" : [ 5, 0 ],
            "3" : [ 5, 9, 0 ],
            "4" : [ 5, 9, 0, 3 ],
            "5" : [ 5, 9, 0, 3, 7 ],
            "6" : [ 5, 9, 0, 3, 7, 2 ],
            "7" : [ 5, 9, 0, 3, 7, 2, 10 ],
            "8" : [ 5, 9, 0, 3, 7, 2, 10, 8 ],
            "9" : [ 5, 9, 0, 3, 7, 2, 10, 8, 1 ],
            "10" : [ 5, 9, 0, 3, 7, 2, 10, 8, 1, 4 ],
            "11" : [ 5, 9, 0, 3, 7, 2, 10, 8, 1, 4, 11 ],
            "12" : [ 5, 9, 0, 3, 7, 2, 10, 8, 1, 4, 11, 6 ]
            }
,           "i" :       {
            "1" : [ 0 ],
            "2" : [ 0, 7 ],
            "3" : [ 0, 3, 7 ],
            "4" : [ 0, 3, 7, 10 ],
            "5" : [ 0, 3, 7, 10, 8 ],
            "6" : [ 0, 3, 7, 10, 8, 5 ],
            "7" : [ 0, 3, 7, 10, 8, 5, 2 ],
            "8" : [ 0, 3, 7, 10, 8, 5, 2, 4 ],
            "9" : [ 0, 3, 7, 10, 8, 5, 2, 4, 9 ],
            "10" : [ 0, 3, 7, 10, 8, 5, 2, 4, 9, 1 ],
            "11" : [ 0, 3, 7, 10, 8, 5, 2, 4, 9, 1, 6 ],
            "12" : [ 0, 3, 7, 10, 8, 5, 2, 4, 9, 1, 6, 11 ]
            }
,           "ii°" :     {
            "1" : [ 2 ],
            "2" : [ 2, 8 ],
            "3" : [ 2, 5, 8 ],
            "4" : [ 2, 5, 8, 0 ],
            "5" : [ 2, 5, 8, 0, 7 ],
            "6" : [ 2, 5, 8, 0, 7, 3 ],
            "7" : [ 2, 5, 8, 0, 7, 3, 10 ],
            "8" : [ 2, 5, 8, 0, 7, 3, 10, 4 ],
            "9" : [ 2, 5, 8, 0, 7, 3, 10, 4, 6 ],
            "10" : [ 2, 5, 8, 0, 7, 3, 10, 4, 6, 9 ],
            "11" : [ 2, 5, 8, 0, 7, 3, 10, 4, 6, 9, 11 ],
            "12" : [ 2, 5, 8, 0, 7, 3, 10, 4, 6, 9, 11, 1 ]
            }
,           "bIII" :    {
            "1" : [ 3 ],
            "2" : [ 3, 10 ],
            "3" : [ 3, 7, 10 ],
            "4" : [ 3, 7, 10, 2 ],
            "5" : [ 3, 7, 10, 2, 0 ],
            "6" : [ 3, 7, 10, 2, 0, 8 ],
            "7" : [ 3, 7, 10, 2, 0, 8, 5 ],
            "8" : [ 3, 7, 10, 2, 0, 8, 5, 6 ],
            "9" : [ 3, 7, 10, 2, 0, 8, 5, 6, 4 ],
            "10" : [ 3, 7, 10, 2, 0, 8, 5, 6, 4, 2 ],
            "11" : [ 3, 7, 10, 2, 0, 8, 5, 6, 4, 2, 9 ],
            "12" : [ 3, 7, 10, 2, 0, 8, 5, 6, 4, 2, 9, 11 ]
            }
,           "iv" :      {
            "1" : [ 5 ],
            "2" : [ 5, 0 ],
            "3" : [ 5, 8, 0 ],
            "4" : [ 5, 8, 0, 3 ],
            "5" : [ 5, 8, 0, 3, 7 ],
            "6" : [ 5, 8, 0, 3, 7, 2 ],
            "7" : [ 5, 8, 0, 3, 7, 2, 10 ],
            "8" : [ 5, 8, 0, 3, 7, 2, 10, 9 ],
            "9" : [ 5, 8, 0, 3, 7, 2, 10, 9, 1 ],
            "10" : [ 5, 8, 0, 3, 7, 2, 10, 9, 1, 4 ],
            "11" : [ 5, 8, 0, 3, 7, 2, 10, 9, 1, 4, 11 ],
            "12" : [ 5, 8, 0, 3, 7, 2, 10, 9, 1, 4, 11, 6 ]
            }
,           "v" :       {
            "1" : [ 7 ],
            "2" : [ 7, 2 ],
            "3" : [ 7, 10, 2 ],
            "4" : [ 7, 10, 2, 5 ],
            "5" : [ 7, 10, 2, 5, 3 ],
            "6" : [ 7, 10, 2, 5, 3, 11 ],
            "7" : [ 7, 10, 2, 5, 3, 11, 9 ],
            "8" : [ 7, 10, 2, 5, 3, 11, 9, 0 ],
            "9" : [ 7, 10, 2, 5, 3, 11, 9, 0, 4 ],
            "10" : [ 7, 10, 2, 5, 3, 11, 9, 0, 4, 8 ],
            "11" : [ 7, 10, 2, 5, 3, 11, 9, 0, 4, 8, 1 ],
            "12" : [ 7, 10, 2, 5, 3, 11, 9, 0, 4, 8, 1, 6 ]
            }
,           "bVI" :     {
            "1" : [ 8 ],
            "2" : [ 8, 3 ],
            "3" : [ 8, 0, 3 ],
            "4" : [ 8, 0, 3, 7 ],
            "5" : [ 8, 0, 3, 7, 5 ],
            "6" : [ 8, 0, 3, 7, 5, 10 ],
            "7" : [ 8, 0, 3, 7, 5, 10, 1 ],
            "8" : [ 8, 0, 3, 7, 5, 10, 1, 11 ],
            "9" : [ 8, 0, 3, 7, 5, 10, 1, 11, 6 ],
            "10" : [ 8, 0, 3, 7, 5, 10, 1, 11, 6, 4 ],
            "11" : [ 8, 0, 3, 7, 5, 10, 1, 11, 6, 4, 2 ],
            "12" : [ 8, 0, 3, 7, 5, 10, 1, 11, 6, 4, 2, 9 ]
            }
,           "bVII" :    {
            "1" : [ 10 ],
            "2" : [ 10, 5 ],
            "3" : [ 10, 2, 5 ],
            "4" : [ 10, 2, 5, 8 ],
            "5" : [ 10, 2, 5, 8, 0 ],
            "6" : [ 10, 2, 5, 8, 0, 7 ],
            "7" : [ 10, 2, 5, 8, 0, 7, 3 ],
            "8" : [ 10, 2, 5, 8, 0, 7, 3, 1 ],
            "9" : [ 10, 2, 5, 8, 0, 7, 3, 1, 6 ],
            "10" : [ 10, 2, 5, 8, 0, 7, 3, 1, 6, 4 ],
            "11" : [ 10, 2, 5, 8, 0, 7, 3, 1, 6, 4, 11 ],
            "12" : [ 10, 2, 5, 8, 0, 7, 3, 1, 6, 4, 11, 9 ]
            }
,           "ii/bII" :  {
            "1" : [ 3 ],
            "2" : [ 3, 10 ],
            "3" : [ 3, 6, 10 ],
            "4" : [ 3, 6, 10, 1 ],
            "5" : [ 3, 6, 10, 1, 0 ],
            "6" : [ 3, 6, 10, 1, 0, 8 ],
            "7" : [ 3, 6, 10, 1, 0, 8, 5 ],
            "8" : [ 3, 6, 10, 1, 0, 8, 5, 7 ],
            "9" : [ 3, 6, 10, 1, 0, 8, 5, 7, 4 ],
            "10" : [ 3, 6, 10, 1, 0, 8, 5, 7, 4, 2 ],
            "11" : [ 3, 6, 10, 1, 0, 8, 5, 7, 4, 2, 9 ],
            "12" : [ 3, 6, 10, 1, 0, 8, 5, 7, 4, 2, 9, 11 ]
            }
,           "ii°m7/biii" :          {
            "1" : [ 5 ],
            "2" : [ 5, 11 ],
            "3" : [ 5, 8, 11 ],
            "4" : [ 5, 8, 11, 3 ],
            "5" : [ 5, 8, 11, 3, 7 ],
            "6" : [ 5, 8, 11, 3, 7, 2 ],
            "7" : [ 5, 8, 11, 3, 7, 2, 10 ],
            "8" : [ 5, 8, 11, 3, 7, 2, 10, 9 ],
            "9" : [ 5, 8, 11, 3, 7, 2, 10, 9, 1 ],
            "10" : [ 5, 8, 11, 3, 7, 2, 10, 9, 1, 4 ],
            "11" : [ 5, 8, 11, 3, 7, 2, 10, 9, 1, 4, 0 ],
            "12" : [ 5, 8, 11, 3, 7, 2, 10, 9, 1, 4, 0, 6 ]
            }
,           "ii°m7/iv" :            {
            "1" : [ 7 ],
            "2" : [ 7, 1 ],
            "3" : [ 7, 10, 1 ],
            "4" : [ 7, 10, 1, 5 ],
            "5" : [ 7, 10, 1, 5, 3 ],
            "6" : [ 7, 10, 1, 5, 3, 11 ],
            "7" : [ 7, 10, 1, 5, 3, 11, 9 ],
            "8" : [ 7, 10, 1, 5, 3, 11, 9, 0 ],
            "9" : [ 7, 10, 1, 5, 3, 11, 9, 0, 4 ],
            "10" : [ 7, 10, 1, 5, 3, 11, 9, 0, 4, 8 ],
            "11" : [ 7, 10, 1, 5, 3, 11, 9, 0, 4, 8, 2 ],
            "12" : [ 7, 10, 1, 5, 3, 11, 9, 0, 4, 8, 2, 6 ]
            }
,           "ii/bV" :   {
            "1" : [ 8 ],
            "2" : [ 8, 3 ],
            "3" : [ 8, 11, 3 ],
            "4" : [ 8, 11, 3, 6 ],
            "5" : [ 8, 11, 3, 6, 5 ],
            "6" : [ 8, 11, 3, 6, 5, 10 ],
            "7" : [ 8, 11, 3, 6, 5, 10, 2 ],
            "8" : [ 8, 11, 3, 6, 5, 10, 2, 0 ],
            "9" : [ 8, 11, 3, 6, 5, 10, 2, 0, 7 ],
            "10" : [ 8, 11, 3, 6, 5, 10, 2, 0, 7, 4 ],
            "11" : [ 8, 11, 3, 6, 5, 10, 2, 0, 7, 4, 1 ],
            "12" : [ 8, 11, 3, 6, 5, 10, 2, 0, 7, 4, 1, 9 ]
            }
,           "ii/bVI" :  {
            "1" : [ 10 ],
            "2" : [ 10, 5 ],
            "3" : [ 10, 1, 5 ],
            "4" : [ 10, 1, 5, 8 ],
            "5" : [ 10, 1, 5, 8, 0 ],
            "6" : [ 10, 1, 5, 8, 0, 7 ],
            "7" : [ 10, 1, 5, 8, 0, 7, 3 ],
            "8" : [ 10, 1, 5, 8, 0, 7, 3, 2 ],
            "9" : [ 10, 1, 5, 8, 0, 7, 3, 2, 6 ],
            "10" : [ 10, 1, 5, 8, 0, 7, 3, 2, 6, 4 ],
            "11" : [ 10, 1, 5, 8, 0, 7, 3, 2, 6, 4, 11 ],
            "12" : [ 10, 1, 5, 8, 0, 7, 3, 2, 6, 4, 11, 9 ]
            }
,           "ii°m7/bv" :            {
            "1" : [ 8 ],
            "2" : [ 8, 2 ],
            "3" : [ 8, 11, 2 ],
            "4" : [ 8, 11, 2, 6 ],
            "5" : [ 8, 11, 2, 6, 1 ],
            "6" : [ 8, 11, 2, 6, 1, 10 ],
            "7" : [ 8, 11, 2, 6, 1, 10, 3 ],
            "8" : [ 8, 11, 2, 6, 1, 10, 3, 0 ],
            "9" : [ 8, 11, 2, 6, 1, 10, 3, 0, 7 ],
            "10" : [ 8, 11, 2, 6, 1, 10, 3, 0, 7, 4 ],
            "11" : [ 8, 11, 2, 6, 1, 10, 3, 0, 7, 4, 5 ],
            "12" : [ 8, 11, 2, 6, 1, 10, 3, 0, 7, 4, 5, 9 ]
            }
,
```

```
        "bII7-N7" :             {
                    "1" : [ 1 ],
                    "2" : [ 1, 8 ],
                    "3" : [ 1, 5, 8 ],
                    "4" : [ 1, 5, 8, 11 ],
                    "5" : [ 1, 5, 8, 11, 3 ],
                    "6" : [ 1, 5, 8, 11, 3, 10 ],
                    "7" : [ 1, 5, 8, 11, 3, 10, 6 ],
                    "8" : [ 1, 5, 8, 11, 3, 10, 6, 4 ],
                    "9" : [ 1, 5, 8, 11, 3, 10, 6, 4, 9 ],
                    "10" : [ 1, 5, 8, 11, 3, 10, 6, 4, 9, 7 ],
                    "11" : [ 1, 5, 8, 11, 3, 10, 6, 4, 9, 7, 2 ],
                    "12" : [ 1, 5, 8, 11, 3, 10, 6, 4, 9, 7, 2, 0 ]
        }
,       "bVI7(no5)-It+" :       {
                    "1" : [ 8 ],
                    "2" : [ 8, 6 ],
                    "3" : [ 8, 0, 6 ],
                    "4" : [ 8, 0, 6, 1 ],
                    "5" : [ 8, 0, 6, 1, 5 ],
                    "6" : [ 8, 0, 6, 1, 5, 10 ],
                    "7" : [ 8, 0, 6, 1, 5, 10, 3 ],
                    "8" : [ 8, 0, 6, 1, 5, 10, 3, 11 ],
                    "9" : [ 8, 0, 6, 1, 5, 10, 3, 11, 7 ],
                    "10" : [ 8, 0, 6, 1, 5, 10, 3, 11, 7, 4 ],
                    "11" : [ 8, 0, 6, 1, 5, 10, 3, 11, 7, 4, 2 ],
                    "12" : [ 8, 0, 6, 1, 5, 10, 3, 11, 7, 4, 2, 9 ]
        }
,       "bVI7(b5)-Fr+" :        {
                    "1" : [ 8 ],
                    "2" : [ 8, 2 ],
                    "3" : [ 8, 0, 2 ],
                    "4" : [ 8, 0, 2, 6 ],
                    "5" : [ 8, 0, 2, 6, 5 ],
                    "6" : [ 8, 0, 2, 6, 5, 10 ],
                    "7" : [ 8, 0, 2, 6, 5, 10, 1 ],
                    "8" : [ 8, 0, 2, 6, 5, 10, 1, 11 ],
                    "9" : [ 8, 0, 2, 6, 5, 10, 1, 11, 3 ],
                    "10" : [ 8, 0, 2, 6, 5, 10, 1, 11, 3, 4 ],
                    "11" : [ 8, 0, 2, 6, 5, 10, 1, 11, 3, 4, 7 ],
                    "12" : [ 8, 0, 2, 6, 5, 10, 1, 11, 3, 4, 7, 9 ]
        }
,       "V7/bii-Ger+" :         {
                    "1" : [ 8 ],
                    "2" : [ 8, 3 ],
                    "3" : [ 8, 0, 3 ],
                    "4" : [ 8, 0, 3, 6 ],
                    "5" : [ 8, 0, 3, 6, 5 ],
                    "6" : [ 8, 0, 3, 6, 5, 10 ],
                    "7" : [ 8, 0, 3, 6, 5, 10, 1 ],
                    "8" : [ 8, 0, 3, 6, 5, 10, 1, 11 ],
                    "9" : [ 8, 0, 3, 6, 5, 10, 1, 11, 7 ],
                    "10" : [ 8, 0, 3, 6, 5, 10, 1, 11, 7, 4 ],
                    "11" : [ 8, 0, 3, 6, 5, 10, 1, 11, 7, 4, 2 ],
                    "12" : [ 8, 0, 3, 6, 5, 10, 1, 11, 7, 4, 2, 9 ]
        }
,       "ii°m7/v" :             {
                    "1" : [ 9 ],
                    "2" : [ 9, 3 ],
                    "3" : [ 9, 0, 3 ],
                    "4" : [ 9, 0, 3, 7 ],
                    "5" : [ 9, 0, 3, 7, 5 ],
                    "6" : [ 9, 0, 3, 7, 5, 2 ],
                    "7" : [ 9, 0, 3, 7, 5, 2, 11 ],
                    "8" : [ 9, 0, 3, 7, 5, 2, 11, 4 ],
                    "9" : [ 9, 0, 3, 7, 5, 2, 11, 4, 1 ],
                    "10" : [ 9, 0, 3, 7, 5, 2, 11, 4, 1, 6 ],
                    "11" : [ 9, 0, 3, 7, 5, 2, 11, 4, 1, 6, 8 ],
                    "12" : [ 9, 0, 3, 7, 5, 2, 11, 4, 1, 6, 8, 10 ]
        }
,       "ii°m7/ii" :            {
                    "1" : [ 4 ],
                    "2" : [ 4, 10 ],
                    "3" : [ 4, 7, 10 ],
                    "4" : [ 4, 7, 10, 2 ],
                    "5" : [ 4, 7, 10, 2, 0 ],
                    "6" : [ 4, 7, 10, 2, 0, 9 ],
                    "7" : [ 4, 7, 10, 2, 0, 9, 5 ],
                    "8" : [ 4, 7, 10, 2, 0, 9, 5, 6 ],
                    "9" : [ 4, 7, 10, 2, 0, 9, 5, 6, 1 ],
                    "10" : [ 4, 7, 10, 2, 0, 9, 5, 6, 1, 8 ],
                    "11" : [ 4, 7, 10, 2, 0, 9, 5, 6, 1, 8, 11 ],
                    "12" : [ 4, 7, 10, 2, 0, 9, 5, 6, 1, 8, 11, 3 ]
        }
}
```

Appendix D: MIDI controllers

By column, it shows: index, name, control change controller (cc), channel, and port.

1	ToBright	0	1	Max
2	ToSad	1	1	Max
3	ToTense	2	1	Max
4	ToEpic	3	1	Max
5	SeedPhrase	5	1	Max
7	LoopSize	7	1	Max
8	Reset	8	1	Max
9	Ply/Stp	9	1	Max
10	Step	10	1	Max
11	Range	11	1	Max
12	Trans	12	1	Max
13	Vel	13	1	Max
14	Tpo	14	1	Max
15	+/-Notes	15	1	Max
16	MvEven	16	1	Max
17	MvOdd	17	1	Max
18	Orch+/-	18	1	Max
19	OrchRange	19	1	Max
20	StaccLegat	20	1	Max
21	Dcontour	21	1	Max
22	Non-Rep	22	1	Max
23	Invert	23	1	Max
24	ShapeRang	24	1	Max
25	ShapeLen	25	1	Max
26	#Nodes	26	1	Max
30	Tension	30	1	Max
31	CpxProg	31	1	Max
32	+/-NotesH	32	1	Max
33	Acc/Beat	33	1	Max
34	VelIns	34	1	Max
35	VoiceIns	35	1	Max
36	RangIns	36	1	Max
37	RegIns	37	1	Max
38	Ins+/-	38	1	Max
39	StaccLegat	39	1	Max
40	MovementH	40	1	Max
41	HarmRhySp	41	1	Max
42	ArpOn	42	1	Max
43	ArpSpd	43	1	Max
44	ArpShape	44	1	Max
45	Min/Maj	45	1	Max
46	DimOn/Off	46	1	Max
47	ArpAlgOnOf	47	1	Max
48	Root	48	1	Max
50	VelPerc	50	1	Max
51	Acc/BeatP	51	1	Max
52	+/-KS	52	1	Max
53	+/-Toms	53	1	Max
54	+/-Cymb	54	1	Max
60	XYmelx	60	1	Max
61	XYmely	61	1	Max
62	XYHarmx	62	1	Max
63	XYHarmy	63	1	Max
64	XYPercx	64	1	Max
65	XYPercy	65	1	Max
70	StrMel	70	1	Max
71	PnoMel	71	1	Max
72	WoodMel	72	1	Max
73	BrassMel	73	1	Max
74	StrHarm	74	1	Max
75	PnoHarm	75	1	Max
76	WoodHarm	76	1	Max
77	BrassHarm	77	1	Max
78	OrchAuto	78	1	Max
80	seed	80	1	Max
81	seedRhy	81	1	Max
82	ArpOnAsy	82	1	Max
108	ResetNow	108	1	Max

Appendix E: Questionnaires

Demographics Questionnaire

Thank you for participating in this study and taking the time to answer these questions. Please, fill out this form prior to your scheduled game experience. If you use a phone, please use landscape mode for convenience.

1. Please select your age group
   - 18-21
   - 22-30
   - 31-55
   - 55 +

2.a Please choose the gender with whom you mostly identify:
   - Woman
   - Man
   - Transgender
   - Non-binary/non-conforming
   - Prefer not to respond

2.b Please select your sex:
   - Male
   - Female
   - Intersex
   - Prefer not to respond

3. On average, how much time do you play videogames in a week?
   - 20+ hours
   - 16 to 20 hours
   - 11 to 15 hours
   - 6 to 10 hours
   - 1 to 5 hours
   - Less than 1 hour
   - Don’t play videogames

4. What is your musical experience? Select any true statement (one or more).
   - I like to hear music but it is not important in my life.
   - I listen regularly to one or more selected music styles.
   - I played a musical instrument in the past
   - I am a music enthusiast.
   - I have been getting informal music lessons (any instrument or voice) for LESS than 3 years

I have been getting informal music lessons (any instrument or voice) for MORE than 3 years
I play a fiddle instrument (violin, viola, cello, contrabass)
I am a self-taught musician
I play piano or any keyboard instrument
I sing (choir, group, soloist)
I play a wind instrument (any wood, brass, or non-orchestral)
I play drums or other percussion instrument
I play a plucked string instrument (guitar, bass, harp, mandolin, ukulele, etc)
I play an instrument not shown here
I have been playing a musical instrument for MORE than 3 years
I spend significant time and resources in getting the best sound quality (equipment, techniques, etc)
I have written music for a game
I majored or will major in music
I majored or will major in audio engineering
I majored or will major in music composition
I am a music/sound professional

5. What are the sources of your music consumption? place sliders accordingly for each category:
Rock and/or Metal (Heavy, Dead/Speed/Black Metal, Punk, Glam, and Progressive, Alternative, Indie rock, and others)
min = 0
max = 4
default = 2
names = I avoid it, I don't mind it , I sporadically find playlists, it is an important section in my collection/playlists, I invest time and resources on it

Hip hop (Alternative, Boom bap, British, Chopper, Hardcore, Freestyle, Turntablism, Trap music, Rap, and others)

Pop (Blues, Soul, 70-80-90s House/Dance/Disco pop, Pop rock/jazz, R&B, Rock'n roll, Country, Ballad, Romantica, Electropop, Bubblegum, and others)

Jazz (Acid, Afro-Cuban, Bebop, Cool, Dixieland, Swing, European free, Free jazz, Jazz blues/funk/fusion, Soul jazz, Gypsy, Big band, and others)

Classical (Renascence/Baroque/Classical/Romantic/Twentieth century orchestral music)

Folkloric or ethnic music (Local/regional traditional from any country or geopolitical location)

Tropical/Caribbean (Cumbia, Salsa, Merengue, Reggaeton, Calypso, Bachata, and others)

Electronic (Ambient, Drum'n bass, EDM, Breakbeat, Techno, Electro, Dub, Electro fusion, House, Deep, Tribal, Industrial, Electroacoustic, Trance, Chill out, Rave, and others)

Film soundtracks (any format originated on a film/TV soundtrack)

Videogame music (any format originated in a videogame)

If the major source of music you usually consume is not listed, please write it below

6. What is your experience with 'Shooter' games (First Person, Third Person)
I have never played that kind of games
I have played them, but is not my favorite
I play them regularly, first and/or third person shooter
The shooter is my favorite game category

7. What is more important in videogame music? Rank the following music scoring features in your opinion by dragging the boxes up or down
Support to gameplay mood and rhythm
Stylistic appropriateness and consistency
Transmission of emotional content
Responsiveness to game events and conditions
Diversity (more musical themes)
Motivic variety (less repetition)
Sound quality
Contribution to immersion (non-distracting)

8. Fill in your three (3) favorite videogames of all time. Characterize each videogame combining the two drop-down features. Then, select the usability you find most accurate in your experience, and select how significant its music is. If there's no game, please fill N/A in the first game.
Videogame Name : text
Videogame Feature 1: dropdown : Action, Shooter, Role-Playing, Sport, Adventure, Fighting, Racing, Strategy, Puzzle, Simulation, Platform: Please Select

Videogame Feature 2: dropdown : Action, Shooter, Role-Playing, Sport, Adventure, Fighting, Racing, Strategy, Simulation, Puzzle, Maze, Side scrolling, Platform, Beat/slash 'em up, Multiplayer online, Battle royale, Music, Tower Defense, Sandbox, Survival: Please Select

Videogame usability: dropdown: It helps me to relax, I use it to kill time, It stimulates my imagination, I like adrenaline, I train my logic, I train coordination, I develop problem-solving skills, I learn to manage resources, It's good for socializing, I like artistic expressions:Please select

Music Importance: dropdown: I like it BECAUSE of its music, Its music is better than average, Its music is good, Its music is secondary , Its music can be turned off with no consequence: Please select

Describe what you like most about the music: text

Gameplay Questionnaire

Game music order

A then B (Music A = g1, Music B = g2)
B then A (Music B = g1, Music A = g2)

1.a What made you stop playing the FIRST Game (g1)? (e.g., run out of time, got bored, got dizzy, passed the game, etc.)

1.b What made you stop playing the SECOND Game (g2)? (e.g., run out of time, got bored, got dizzy, passed the game, etc.)

2. Please, write down in your own words what is your perception of the music in the FIRST game (g1):
3. Please, write down in your own words what is your perception of the music in the SECOND game (g2):

4. In which game did you feel that the music... (++ far more, + more, g1~g2 equivalent)

min = 0
max = 4
default = 2
names = g1++, g1+, g1~g2, g2+, g2++

Provoked emotions in you
Helped you to get more immersed
Got you more curious about the game possibilities

5. Compare the music stylistic appropriateness in the following game conditions (++ clearly more appropriate, + slightly more appropriate, g1~g2 equivalent)

min = 0
max = 4
default = 2
names = g1++, g1+, g1~g2, g2+, g2++

Action-danger
Mystery-suspense
Adventure-curiosity
Tension-suspense, eager, edgy, uneasy
Epic, triumphal.

6.a Compare the music support to game situations. In which game did the music ADAPT using a deeper range of possibilities for the following situations (++clearly more possibilities , + slightly more possibilities, g1~g2 equivalent):

min = 0
max = 4
default = 2
names = g1++, g1+, g1~g2, g2+, g2++

Action-danger
Mystery-suspense
Adventure-curiosity

Tension-suspense, eager, edgy, uneasy
Epic, triumphal.

6.b In which game did the selected musical aspect (melody, harmony-chords, percussion) support better any game circumstance in general? (++ clearly superior, + slightly better, g1~g2 equivalent)

min = 0
max = 4
default = 2
names = g1++, g1+, g1~g2, g2+, g2++

Melody
Harmony-chords
Percussion

7. In which game the following music features performed better? (++ clearly superior, + slightly better, g1~g2 equivalent)

min = 0
max = 4
default = 2
names = g1++, g1+, g1~g2, g2+, g2++

Responsiveness to game events and conditions
Smoothness in music transitions
Variety (non-repetitive)
Ability to transmit emotional information
Influenced positively your gameplay performance
Engagement (the music made you want to play longer)

8. In your opinion, which game had BETTER MUSIC overall?

Game 1 (g1) had SLIGHTLY better music
Game 2 (g2) had SLIGHTLY better music
Game 1 (g1) had CONSIDERABLY better music
Game 2 (g2) had CONSIDERABLY better music

Appendix F: Tables of results

Table 1. Significant within-subject effects

Within Subjects Effects

Cases	Sum of Squares	df	Mean Square	F	p	η^2_p
Emotions	1.778	1	1.778	5.447	0.033	0.254
Residuals	5.222	16	0.326			
Performance Emotional Transmission	1.778	1	1.778	6.919	0.018	0.302
Residuals	4.111	16	0.257			

Note. Type III Sum of Squares

Table 2. Significant between-subject effects

Between Subjects Effects
Support (general)

Cases	Sum of Squares	df	Mean Square	F	p	η^2_p
group	1.000	1	1.000	4.500	0.050	0.220
Residuals	3.556	16	0.222			
Between Subjects Effects						
Support Harmony						
group	0.444	1	1.000	6.400	0.022	0.286
Residuals	1.111	16	0.069			

Note. Type III Sum of Squares

Table 3. Significant group interactions

Within Subjects Effects

Cases	Sum of Squares	df	Mean Square	F	p	η^2_p
Curiosity	0.250	1	0.250	1.385	0.257	0.080
Curiosity ✻ group	3.361	1	3.361	18.615	< .001	0.538
Residuals	2.889	16	0.181			
Appropriateness Curiosity	0.028	1	0.028	0.108	0.747	0.007
Appropriateness Curiosity ✻ group	3.361	1	3.361	13.081	0.002	0.450
Residuals	4.111	16	0.257			
Adapt Adventure-Curiosity	0.444	1	0.444	1.882	0.189	0.105
Adapt Adventure-Curiosity ✻ group	1.778	1	1.778	7.529	0.014	0.320
Residuals	3.778	16	0.236			
Adapt Mystery-Suspense	0.111	1	0.111	0.348	0.564	0.021
Adapt Mystery-Suspense ✻ group	2.778	1	2.778	8.696	0.009	0.352
Residuals	5.111	16	0.319			
Adapt (general)	1.000	1	1.000	0.762	0.396	0.045
Adapt (general) ✻ group	9.000	1	9.000	6.857	0.019	0.300
Residuals	21.000	16	1.313			

Note. Type III Sum of Squares

Appendix G: Media

Please find the accompanying media in this Google Folder link.

1. The Audio folder contains:
    - PAMG&AL_Studio_IIIc_Numb (5/2021):

    This piece was recorded using an old version of PAMG. Arpeggio algorithm, percussion ratchet, harmony speed, small interlude, and variable change for percussion where not implemented yet.
    - PAMG&AL_Studio VIII_In_the_Distance (2/2022):

    This piece had the Arpeggio but not the percussion ratchet, harmony speed, small interlude, and variable change for percussion.
    - PAMG&AL_Studio_IX_Pickaboo (2/2022):

    Same as *In the Distance.*
    - PAMG&AL_Studio_XII_Need_Air (4/2023):

    All current algorithms implemented.
    - PAMG&AL_Studio_XIII_Have_a_Hand (4/2023):

    Same as *Have a Hand.*
2. PAMG_Recordings_Gameplay:

    A selection of several PAMG (music B) recordings from gameplay participants. The place in the sequence for the participant is shown by the letters A and B (if A-B the recording is from the second game and vice versa).
3. The Video folder contains:
    - *PAMG sample 1*:

    A series of transitions recorded from The Trial game using PAMG.
    - *Transitions Comparative CBI-PAMG*:

    A video comparing CBI and PAMG in the same game segments.
    - *PAMG UI Demo*:

    A walk-through of the game and PAMG UI showing real-time interaction.
4. *The_Trial_Game_CBI_Wwise.exe*:

    A Windows compiled version of The Trial game as an executable using Wwise to design CBI.
5. *PAMGcomp.exe*:

    A Windows compiled standalone application of PAMG. To use it, request the activation code upon the password prompt is provided.