SoundEffects
An Interdisciplinary Journal of Sound and Sound Experience

Alvaro E. Lopez Duarte
PhD Candidate in Digital Composition
University of California, Riverside

Algorithmic interactive music generation in videogames

a modular design for adaptive automatic music scoring

www.soundeffects.dk

SoundEffects | vol. 9 | no. 1 | 2020 issn 1904-500X

Abstract

In this article, I review the concept of algorithmic generative and interactive music and discuss the advantages and challenges of its implementation in videogames. Excessive repetition caused by low interactivity in music sequences through gameplay has been tackled primarily by using random or sequential containers, coupled with overlapping rules and adaptive mix parameters, as demonstrated in the Dynamic Music Units *in Audiokinetic's Wwise middleware. This approach provides a higher variety through re-combinatorial properties of music tracks and also a responsive and interactive music stream. However, it mainly uses pre-recorded music sequences that reappear and are easy to recognize throughout gameplay. Generative principles such as* single-seed *design have been occasionally applied in game music scoring to generate material. Some of them are complemented with rules and are assigned to sections with low emotional requirements, but support for real-time interaction in gameplay situations, although desirable, is rarely found.*

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 style through measurable parameters or program arguments. Starting with music generation, I examine conceptual implementations and technical challenges of algorithmic composition studies that use Markov models, a-life/evolutionary music, generative grammars, agents, and artificial neural networks/deep learning. For each model, I evaluate rule-based strategies for interactive music transformation using parameters provided by contextual gameplay situations. Finally, I propose a compositional tool design based in modular instances of algorithmic music generation, featuring stylistic interactive control in connection with an audio engine rendering system.

I. Introduction

In the book *Writing Interactive Music for Video Games: A Composer's Guide,* Michael Sweet (2015) examines the relevance of gameplay variability in music content. As a complement to some examples of interactive videogame music, in the section "Composer Perspective", Bear McCreary[1] asserts: "The more often the audience is exposed to a sonic idea, the less impact it has." (Sweet, 2015, pp. 22).

This kind of back-to-back sequence replay is mostly distinct from the gesture of restatement or re-exposition of music material intentionally. The repeated use of the same musical audio clips potentially stands out as unintentional filling of auditory space. It can cause an arguable diminution in tension and surprise on the listener. In audiovisual storytelling, music often provides emotional context and continuity for timelines in which a repeated segment mostly relates to a narrative intention. In the case of videogames' multi-linear structures, occurrences

 issn 1904-500X

and gameplay cumulative time are larger than in single-line stories, causing the assigned music sequences to reappear constantly under current underscoring techniques.

Following McCreary's claim, repetition not only leads to decreased interest in the music segment, but also potentially to a conscious and deliberate music filter by the player. Although it is not possible to state that all repetition is annoying, changes triggered by interaction are reportedly satisfying. For example, the use of highly repetitive music in *Space Invaders*[2] is supported by an increase in tempo correlated to the remaining invaders, providing change and player's participation.

This suggests that introducing adaptation to gameplay situations, in which the players recognize themselves as agents of change and variety, may improve the overall game experience. Besides reducing literal repetition, it has the potential of enabling transmission of real-time contextual information, given that a continuous appearance of new elements stimulates players' attention throughout longer and numerous audition sessions.

The concept of adaptive music for videogames has been theorized and disclosed in several writings[3], elaborating on concepts such as dynamic or interactive music, and also encompassing the research on responsive procedural sound design[4].

A game is a singular and complex cohesion of systems and aesthetics. Its rules, shapes, textures, sounds, and music respond to specific needs that may not necessarily include an active always-changing music. Some stylistic constructions precisely require repetitive or minimalistic aesthetics. Furthermore, particular categories in games' functional music as stingers and transitions provide information about game-state changes and appeal to players' memories and associations. Hence, short and distinctive segments with low or no variation are frequently needed in any videogame music production paradigm.

In this article, the idea of a player's participation in a multi-linear audiovisual story generation, specifically as interaction in real-time music playback within a videogame, is addressed through the possibilities, challenges, limits, and techniques of automatic music composition.

II. Film-like music production vs. real-time music generation

On the side of repetition, music has had a "signature" role in generating identity. Vintage videogame music, for example, was able to get stuck in the player's memory, causing unique aesthetic associations. This is similar to the use of repetition in pop music's "hooks" and catchy phrases in jingles and commercials. Although this may be useful in establishing a distinctive style and a marketable product, the gameplay music as a subject of multiple auditions may benefit from variation in music content, leaving identity association to consistent stylistic features.

issn 1904-500X

There is a wide range of videogame genres, artistic and narrative needs, and budget sizes. As for most of the services, custom and dedicated original music in a game production demands a larger budget than standardized versions, pre-produced segments, or automatic music. Mostly, comprehensive funding for music production will reflect on hiring experienced and prestigious composers, known performers/orchestras, and highly trained sound engineers running top-notch recording technology in legendary studios. From this budget-range tendency of using film soundtrack production pipelines, a preference for linear segment production clearly emerges – as opposed to a multi-linear, real-time generation of music. More recently, and in the mid-size budget sector, the sound quality – as a main argument in favor of linear music – has been challenged by competitive results achieved by MIDI, combined with high quality instrument samples and/or efficient synthesis and audio rendering systems. This can be the pivot point and the opportunity for a generative engine with high-quality audio output to be considered.

Tackling music repetition

At a more basic level, the prospect of variety in contextual music at gameplay may appeal to heavy players, even if its function is not to provide additional information

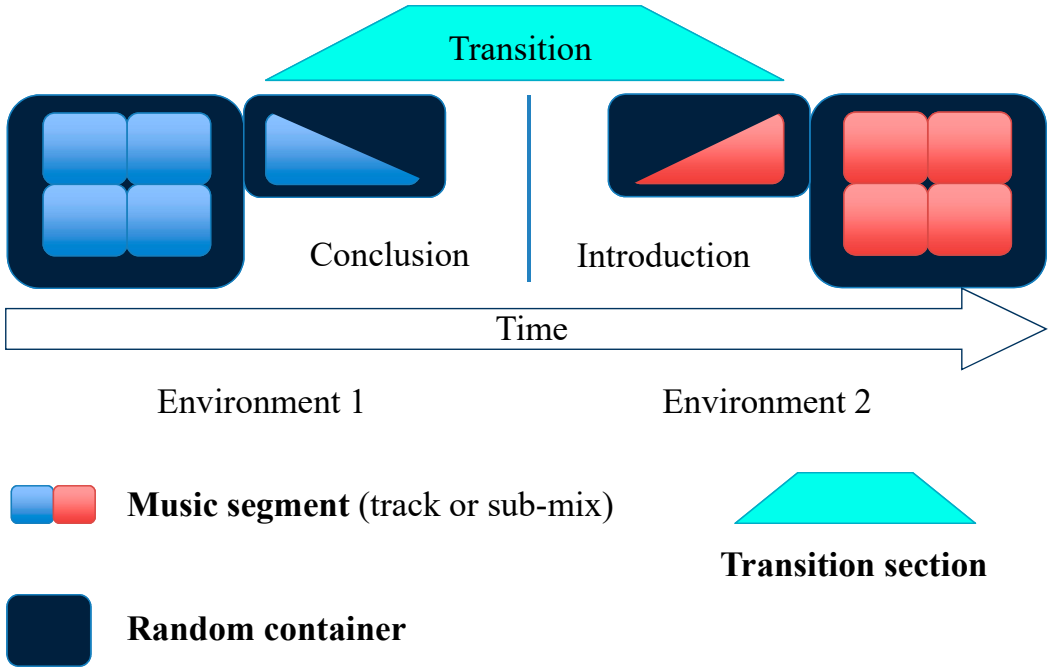

Figure No. 1. Variation through re-combination. In this example of music assignment for videogame environments, instead of a single segment that repeats as soon it ends, there is a pool of segments especially-designed to play randomly and to be overlapped, providing more possibilities. In the event of a transition to a second environment, there is another random container with segments designed to conclude and "cross-fade" with the introduction segments of the new environment.

issn 1904-500X

to gameplay than the one originally established. A representation of this concept for two environments is shown in Figure 1. Current diversification techniques using re-combination of pre-produced music include:

- Random triggering of loop segments using algorithms to avoid immediate repetition. This can be combined with transitional overlapping sections to connect the segments.
- Use of different instrument sub-mix sections. These diverse combinations of instrumental sets not only result in a range of variety, but also offers levels of intensity useful for a range of assignments.
- Designing progressions that follow a continuous form and support several types of harmonic overlapping. Crafting music segments that connect with each other and with themselves, and that additionally work harmonically when played together, benefits the music modularity. For a composer, however, this task may be time consuming and constraining.
- Assigning interactive variables to sound properties for finalized music pieces, segments, or tracks. Parameters coming from gameplay may affect levels, filters, transpositions, dynamic processes, and acoustic effect properties in real time, resulting in a responsive and interactive stream of music.

These techniques usually have been implemented through sound design software solutions such as Wwise and FMOD, requiring production, memory allocation, and playback structure design for more segments than in the single-line contextual music approach.

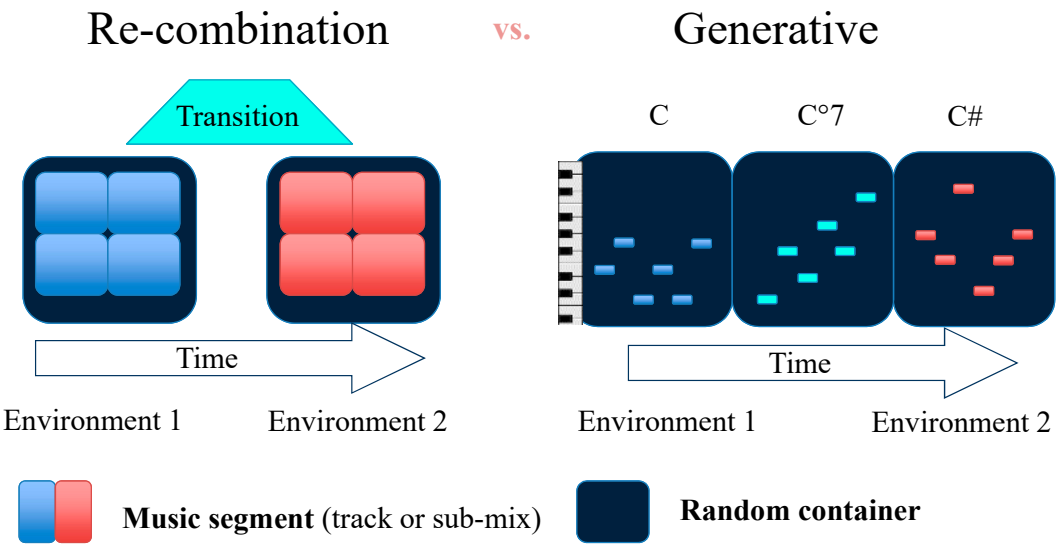

Figure No. 2. Re-combination vs. generative. On the left, a representation of the re-combinatorial paradigm explained in Figure 1. On the right, an example of generative music assigned to environments. In this case, the random container pulls musical notes from a particular harmony. The transition is handled by pivotal harmonies such as diminished chords (e. g. C°7), in their own random container, enabling harmonic tension and resolution.

issn 1904-500X

By using a similar approach, but replacing music segments by notes[5], a random, constrained distribution will present a simple version of the generative property (right side on Figure 2).

For instance, it is possible to employ a limited set of recorded pitches from an instrument exactly as if they were music segments placed in random containers and to activate a subsection equivalent to a chord. Different containers with particular harmonies activated in a designed sequence will produce a generative music stream showing harmonic progressions.

As an interesting example that pioneered the commercial use of generative algorithmic music production (although not for videogames), Tim and Peter Cole developed Koan, a software designed in collaboration with Brian Eno at SSEYO in 1996, currently available as Noatikl or Wotja[6]. It generates music uninterruptedly based on different random distributions. The user can alter magnitude values on statistic probabilities for a number of musical properties such as rules, distribution, ranges, phrase lengths, harmony, mutation factors, and much more. If a music engine for videogames uses a similar strategy, it could easily assign and connect discrete gameplay streams (i.e. real-time parameters procedurally generated) to control music properties.

Currently there are a number of initiatives on automatic music production, some of them marketed as AI music. Among many of them, it is worth to mention Amper[7], Aiva[8], Melodrive[9], and Google's Magenta (disclosed in the neural networks section). Along with an increasing number of individual models and AI music communities, they offer different approaches and uses – from open collaborative repositories for experimental development to proprietary and commercial projects for soundtrack production, custom ambient, and game music generation.

III. Music as numerical patterns:

Throughout numerous examples of computational automatic music generation, from Hiller's *Illiac Suite* (1957) until today, algorithms or rules established by composers have been able to produce distinctive aesthetics. Music content as numeric representations can be the subject of discrete functions and processes to produce new material and/or transitional developments, preserving or producing style. Furthermore, direct generation is also available from randomly generated sequences. The act of choosing the functions, rules, constraints, ranges, and distributions has been an aesthetic decision executed by composers, music designers, or artists in general.

Given the great quantity and depth of modifiable musical dimensions, a relative stability in some of them – even if it is a pattern of variation – constitutes a factor of identity. Several authors working on algorithmic composition or computational analysis elaborate on the relationship between patterns and similarity and how

issn 1904-500X

they may involve one or multiple dimensions, including, but not limited to pitch collections, event duration and occurrence, harmonic and melodic density, dynamics, and timbre. Among them, David Cope, in his book *Hidden Structure: Music Analysis Using Computers*, proposes four principles as a foundation for discrete music analysis: "all music consists of patterns; all pitch patterns can be reduced to scales; all elements of scales have different functions; and all patterns, scales, and functions in music are best understood by modeling their processes" (Cope, 2008, pp. 1). Patterns, scales, functions, and process modeling have been essential elements in both music analysis and algorithmic composition.

The idea of developing "semi-automatic" music generation techniques has been considered even before the advent of the computer. Examples as old as the isorhythmic motet, the *rota* (wheel) or the *Musikalisches Würfelspiel* (musical dices) (Cope, 2000), to mention a few, generate new music from combinatorial procedures, a customized material, and a set of instructions or interaction. Using computers, most automatic music generation methods similarly start with aleatoric processes in a pre-designed corpus and then apply rules or constraints to generate the sequences. Its results are also comparable with common composer's methods that start with an improvisation phase that provides motives and ideas which are later developed through rationalized techniques, theoretical knowledge, and stylistic processes.

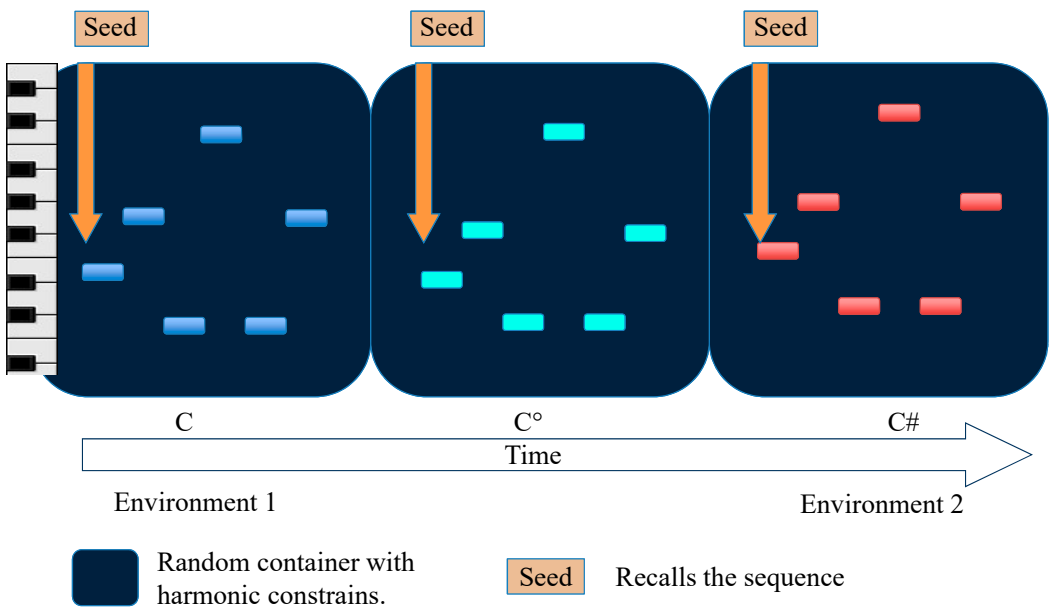

Figure No. 3. Simple generative using seed. This is a small development over the generative principle using random notes from a harmony. As in the previous figure, random containers using a harmony grid provide the stream of notes. In this case, however, the seeding of the PRNG (Pseudo Random Number Generator) restarts the sequence. If the random container has a different harmony assigned, the pitches are reassigned to the closest match. This produces a similar shape, but in a different harmony.

issn 1904-500X

Hence, it may be possible to argue that an intrinsic algorithmic component may be essential for music making.

Creativity in computation – basically any generative process – starts with an algorithm known as the *pseudo-random-number generator* (PRNG). It is used in a wide range of tasks such as lottery drawing, anti-aliasing, text generation, encryption, and more. It was first achieved by sampling external phenomena with measurable levels of entropy like electric noise. The idea of random number generation through an algorithmic approach[10], i.e. without the use of external entropy, has been a computationally economic solution. Besides the property of producing a sequence with no discernible pattern or repetition[11], it features the possibility of recalling a sequence. This type of algorithms –called *Pseudo Random Number Generators* (PRNG) or *Deterministic Random Bit Generators* (DRBG) (von Newman, 1951) – is able to recover a sequence through an initialization value called *seed*.

In music, *single-seed* random generation recalls sequences of events and/or generative properties, making them useful for shape-building.

Thanks to seeding, besides the typical music variety achieved through random generation, distinctive contours reappear supporting music identity. However, the richness of progressive development based on compositional techniques is not present. Adding algorithmic rules to the resulting stream may bring some of those

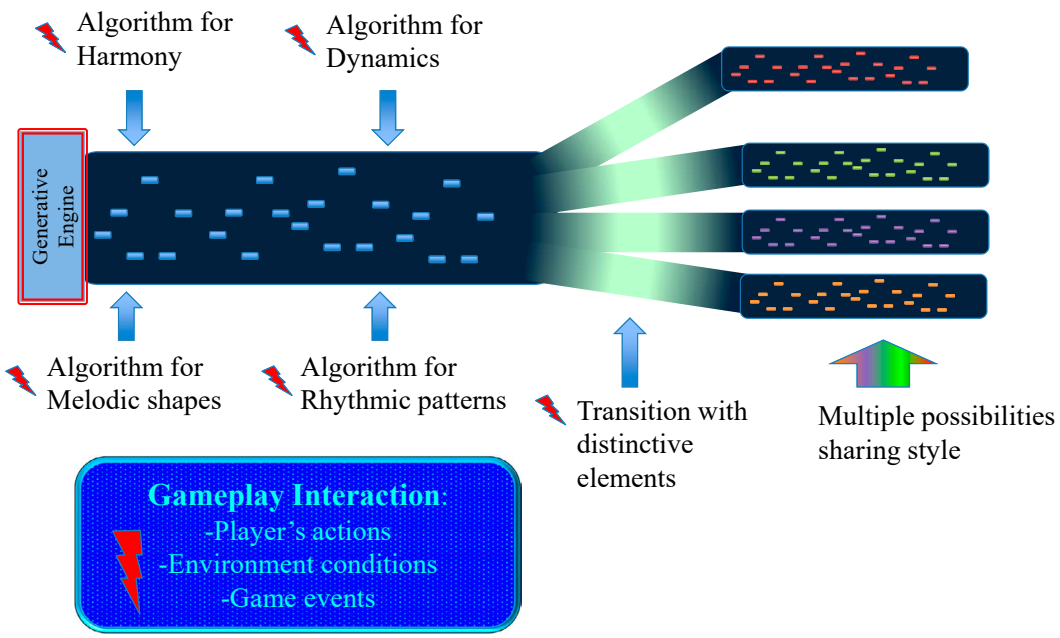

Figure No. 4. Generative + algorithmic. This graph shows a generative engine similar to Figure 3, but adding algorithms to affect the resulting pitch stream. If they feature inputs (red thunder) tied to one or more gameplay events, states, actions or conditions, the music stream is changed while maintaining some construction characteristics. This allows for style support, malleability and variety.

issn 1904-500X

musical developments back. Certainly, applying sequential variations in music properties in conjunction with stochastic distribution management – among other techniques – may improve the music quality.

For computer-aid composition, most techniques can be classified in *generative* processes (e.g. stochastic distributions and random generation) and *algorithmic* strategies (e.g. functions, rules, and deterministic operations) (Ames, 1987). If a system combines both generative and algorithmic procedures, the result potentially offers unlimited variety in music creation *and* pattern and structure development, which then would support style and aesthetic shape. Furthermore, algorithms may be interactive by using conditional and/or progressive modifications through control inputs in real time. This element is particularly useful in videogames, where gameplay situations produced by environments, events, and players' actions constantly generate streams of discrete information suitable to affect parameters of music generation. Features such as harmony, density, range, rhythmic or motivic patterns, and timbre can be altered without interrupting the music flow or the need to overlap other tracks, enhancing the game experience with variety, emotional support, malleability, and responsiveness.

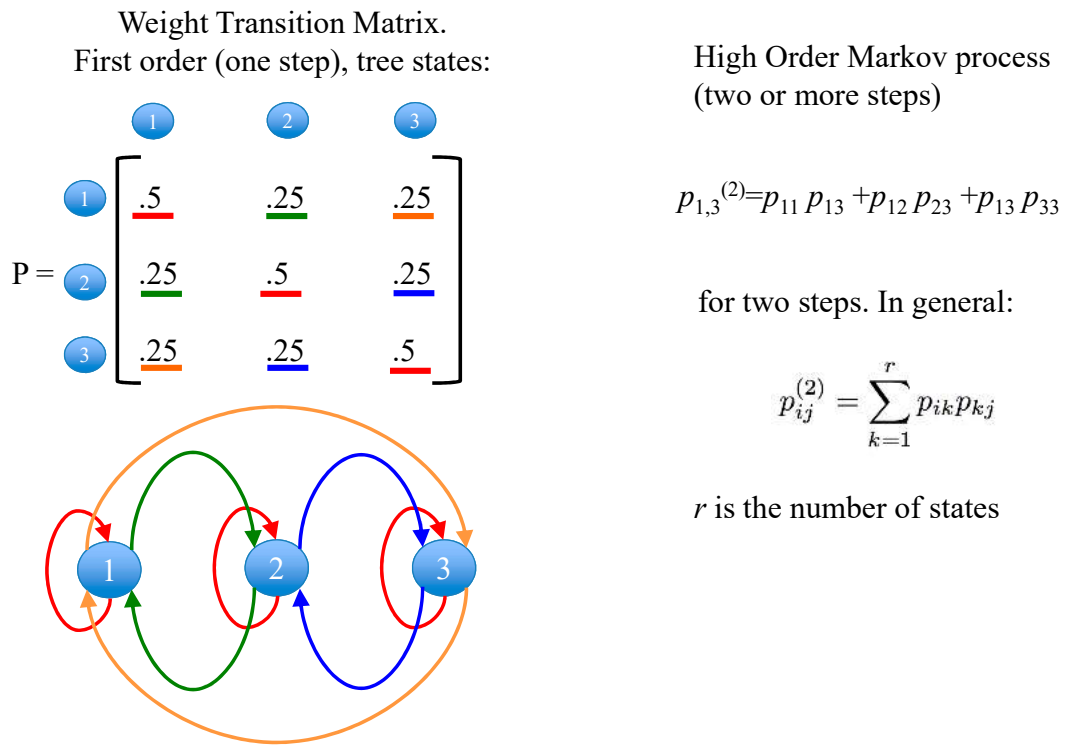

Figure No. 5. Markov models. On the left, a common display of a transition matrix showing the weights representing the statistical possibilities of transition from one state to another. In this example, the states can be seen as the blue nodes in the graph below. Each transition is color coded and the horizontal sum of the weights should be equal to one. On the right, the equation for a second order Markov process (two transitions), and below, a generalization of the second order for any number of states.

issn 1904-500X

As an underlying relation of music with numbers, patterns, and rule frameworks, several models for statistic classification, stochastic processes and probability have been important tools for algorithmic composition. The following is an overview of five salient models[12], their characteristic features in music, and how they could offer interaction aiming for videogame support:

Markov models (MM)

Designed to describe sequences of events in time, discrete MM calculate the stochastic probability for a future state based on previous states (Norris, 1998). Mostly, this probability is stored as a weight coming from statistic information for each transition. Some usual representations are the state transition graph and the transition matrix.

For each transition, there is a value that represents the likelihood of change from one state to other. *High-order Markov processes* are used to calculate more than one transition or step. There are also *hidden Markov models,* in which the internal transitions are hidden, but the observable output symbols are visible.

MM (*discrete* type) have been employed in a variety of music material generation, including melody construction, harmony progression, and other feature series in several Markov orders. From important pioneering works to more recent approaches[13], this model has been one of the most frequently used for music automation and has consequently been adapted in a wide range of ways to suit different compositional needs. It has been used primarily in two ways: to reproduce a style by generating weight tables based on music analysis and to generate new material by designing transition/state structures. In MM, the occurrence of a transition, or a discrete step between states, basically generates a non-cero associated weight. Accordingly, because there is no register for non-existent transitions, the model is unable to produce events that did not happen in the training set. This characteristic creates dead-end situations or sub-sequence loop traps, requiring interpolation procedures or exit algorithms (Nierhaus, 2009). In this model, besides a relatively high randomness in lower orders and overly restrictive output in higher orders, the resulting structure similarity to a training corpus can cause a disconnection with other transition branches and a limited set of results in successive orders (Brooks et al., 1993).

For music generation, weight table design and music parameter assignment are the key to achieve style. Incidentally, a control system that connects gameplay situations with weights' magnitude is able to produce significant changes in music shape, either gradually or contrastingly.

A-life/evolutionary music

Artificial life (*a-life*) algorithms use a cooperative network of entities based in characteristic biological phenomena such as birth, movement, growth, reproduction,

issn 1904-500X

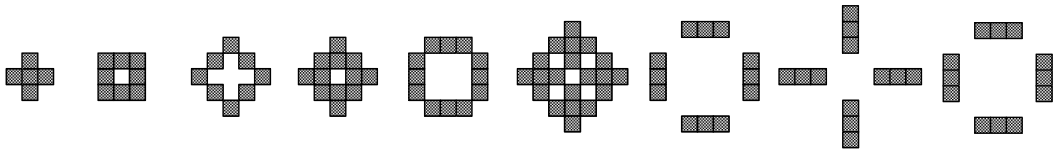

Figure No. 6. Horton Conway's Game of Life.
Rules:
· a cell appears in next round if it has three adjacent cells
· a cell with two or three adjacent cells survives in next round
· no cell appears in other cases.
Following the rules for each round in this Cellular Automata, the figure illustrates the outcome of several rounds departing from an initial state on the left. For every new round, the position of the current active cells is evaluated under the rules, and a new configuration emerges for the next round. The cells can survive and reproduce if the conditions are met. In this example, an oscillating sequence of two patterns is reached after a few rounds, showing one of the most common behaviors (Wolfram, 2002).

and death. A common approach called *Cellular Automata* (CA) models an n-dimensional-space dynamic system in which cells' behavior provide generative information derived from their activity. Their transitions are executed in discrete steps and follow local rules. A cell assumes a state determined by its own and neighbors' states in the previous round. Transition rules apply for all the cells in the n-dimensional space, and the possible cell states should include at least "*active*" and "*inactive*" – which can be associated with "alive" and "dead". Stephen Wolfram, describing theoretical background for CA (Wolfram, 2002), identifies four resulting behavior classes: a simple and homogenous final state, a set of different final states with stable or periodical structures, a random behavior, and a set of pattern-governed structures.

A usual illustration is Conway's *Game of Life* (GoL) (Gardner, 1970) for a 2-dimensional space. As illustrated in a sequence of steps from left to right in Figure 6, it reaches a stable pattern in the last two cell configurations (Wolfram's fourth behavior).

As an example of music implementation, in 1993 Eduardo R. Miranda (Burraston & Edmonds, 2005; Miranda, 2011) applied CA in CAMUS by assigning events to cell changes. It employed a GoL rule system for pitch and duration generation, and a mathematical space as an environment for MIDI channel assignation.

This technique is quite open for rules and assignation which means that the resulting music shape can be completely different with just a slight change. Multi-dimensional, high-resolution spaces and special harmony cell distributions can handle a wider range of possibilities and more interpolated/deeper parameter changes. Patterns of stable oscillation are frequent, highlighting an evident rhythmical usability. In fact, the "step" property (transition rounds), signals an important assignment of a time unit for the whole system, which is reflected in a "grid-like" quantized music. Clusters and discrete glissandos are common musical results, making it suitable for sound design and retro gaming aesthetics.

issn 1904-500X

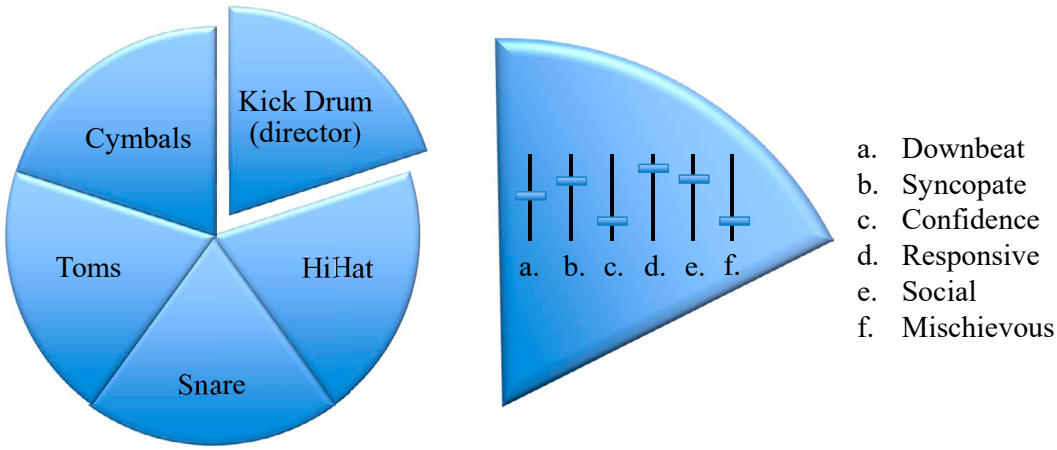

Figure No. 7. Multi-agent models. On the left, a representation of a multi-agent model in which the role of each agent is to generate a line for their instrument in coordination and interaction with other agents. On the right, a sample agent featuring several parameters (Eigenfeldt, 2009) that can alter their "personality", thus affecting the global result.

Multi-agent models

Agents can be understood as a subcategory of a-life systems since they share paradigmatic "living" approaches such as the agent's autonomy, agency, collaborative or social properties, environment awareness, shared objectives, and individual skills. In the multi-agent model, individual autonomous algorithms present distinctive abilities, functions, and goals that are used to interact with other agents or humans to generate a collaborative result. This system models a "social simulation" where entities within a context undertake a task benefiting from simultaneity, specialty, and coordination (Niazi & Hussain, 2011).

In this approach, hierarchical roles as timing, harmonic environment, and tonal direction are common, and assigning musical dimensions to an agent's role define its identity. A multi-agent real-time interactive setup shares several qualities with a human ensemble such as individual agent skills.

Arne Eigenfeldt elaborates on particular human-like characteristics that give life and motion to an improvisational rhythmic virtual ensemble. On *Kinetic Engine* (Eigenfeldt, 2009), he uses agents for percussion instruments with variables assigned to features, including *confidence, social, commitment, mischievous,* and *soloistic.* These features are controlled by a propensity level that uses fuzzy logic algorithms (Novák & Moko, 1999). The feature level configuration in each agent defines its personality. Agent capabilities such as perception of environmental dynamics (including the distinction of its own influence), degrees of autonomy, interactive initiative (social), and responsiveness to stimuli play an important role in collaborative constructions. This suggests that the multi-agent concept may be a natural fit for autonomous per-

issn 1904-500X

cussion ensembles since in human contexts, an individual carries out each instrumental line in the resulting music[14].

This type of algorithm offers the flexibility of real-time interaction with internal and external events in addition to a set of parameters available for external control and potential assignment from gameplay.

Among possible issues, the use of a network of entities, in which each one is able to act with a degree of indeterminacy, tends to multiply the possibilities of events out of range, context, or control. In other words, multiple agent interactions are able to unleash an overwhelming cascade of changes in musical features leading to unexpected results. Therefore, constraints in inter-agent communication lines, algorithms for "self-control" depending on contextual conditions, and/or environmental limiting thresholds may be useful.

Generative grammars

Originating from linguistic models developed by Noam Chomsky (1956, 1968), this technique is a functional stratification system used in syntax theories dealing

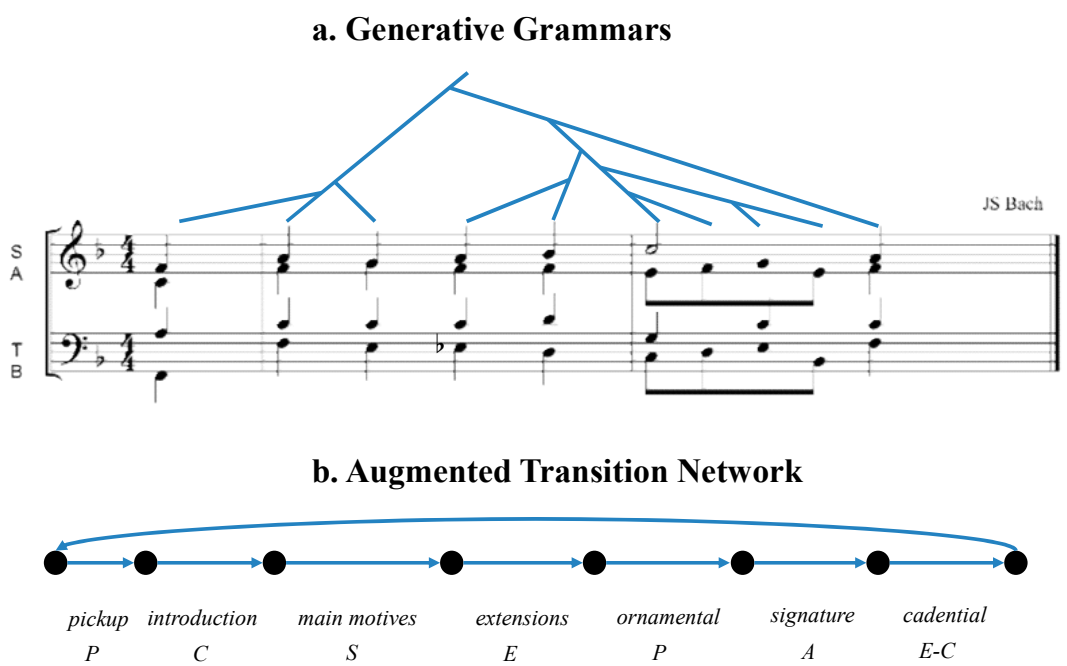

Figure No. 8. Generative grammars and ATN. (a.) A usual tree representation (Lerdahl and Jackendoff, 1983) resulting from the analysis of J.S. Bach's choral Christus, der ist mein Leben, first phrase. The lower levels in the tree represent the final object present in the piece. Going up the tree, hierarchical divisions represent several stages of grouping. (b.) In an Augmented Transition Network (ATN) (Cope, 1996), the musical segments between nodes (arrows) are classified following the rules of SPEAC (Statement, Preparation, Extension, Antecedent, and Consequent). These categories also offer a range of possible connections provided by grammar paradigms. Although Cope uses specialized algorithms for transition, segment selection, and jumps, it is possible to use a Markov weight table to manage concatenation.

issn 1904-500X

with formal sentence structures. In language, it involves expressions constituted by symbol strings and hierarchical rules that bind their structure together as a given phrase. Following these rules sequentially or recursively generates new string sequences that are technically correct in the chosen language or context, although the result is not always semantically meaningful.

In algorithmic composition, this has been a method for hierarchical organization of music material based on Western tonal music functionality that is useful to analyze and model form. Using this precise approach, Fred Lerdahl and Ray Jackendoff (1983) developed a system of generative grammars for Western tonal music referenced recurrently in computational generative grammar studies, including those mentioned in this section (see Figure 8 a).

An important example is a method called *Augmented Transition Networks* (ATN) (Cope, 1996). Based on hierarchical functionality, it is a system of nodes, conditionals, and subprograms to bind segments and groups of segments sequentially, and in some cases recurrently. Cope names his structure SPEAC (*Statement*, *Preparation*, *Extension*, *Antecedent,* and *Consequent*). Roles of music snippets are classified in his example as *pickup*, *introduction*, *main motives*, *extensions*, *ornamental*, *signature*, *or cadential* (see Figure 8 b). This organization allows modularity and exchange with similar segments creating convincing combinations and acceptable phrasing. In this technique, Cope mostly uses existent pieces as the music material to feed the algorithm. Either composed by himself or by well-known composers, the pieces are initially processed by segmentation and classification methods. Then, standard music variation techniques (e.g. transposing, stretching, inverting, and reversing) are applied to the snippets in order to extend the possibilities, and finally, the ATN algorithm constructs one or several pieces by concatenation of material.

ATNs, employing harmony as a variable in conjunction with a note-generating algorithm, easily produce a high percentage of technically correct progressions. This can govern harmonic sequences or voice leading sections, thus producing phrasing. Also, when assigned a value related to tension, it has the potential to receive information from gameplay, modifying the current harmonic and/or rhythmic behavior. In this case, responsiveness may depend on the time interval between nodes in a similar way that sample buffers generate latency in digital audio.

Artificial Neural Networks (ANN)/deep learning

Based in the brain neuron model, AAN (also called deep learning[15]) constitutes a connectionist subsection of the machine learning algorithms. The ANN model is designed to reduce the error between a *hypothesis*[16] and the known value for *regression* and *classification* problems[17]. The process of minimizing the error is regarded as the *learning* property in the field of machine learning. The most frequently used

issn 1904-500X

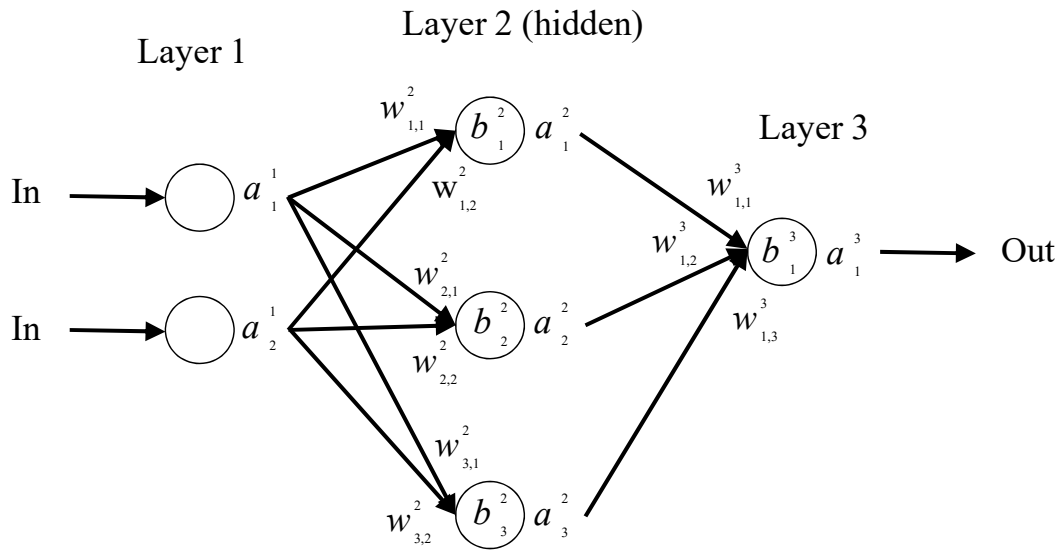

Figure No. 9. Deep neural networks. In the graph, a is the activation function, b is the bias, and w is the weight assigned to each connection represented by an arrow. The nodes are circles representing where the product of the input value and the weight plus the bias is evaluated by the activation function. Whether the value is passed or transformed depends on the type of activation function. The subscript number is an index node in its layer, and the superscript number is the layer index for a and b. For w, the superscript is the layer index and the subscript couple is the node index and the previous node index, respectively.

methods to learn in deep neural networks include the *cost function* and *gradient descent* through *back propagation*[18].

The nodes or neurons have an *activation* function[19] (sigmoid, tanh, ReLu, among others), and the connections are controlled by *weights*, which are updated by the gradient descent to minimize the error. Designs with internal or hidden neuron layers – called *deep learning* models – are known for the good results in multidimensional classification and regression problems.

ANN models require a *training set*, which is a database organized to fit the input. The training may include procedures such as formatting and splitting the dataset into training, test, and validation pools, feeding the system with a labeled output for *supervised* problems or defining the categorical classes for *unsupervised* learning[20], and determining a number of epochs, batches and other system configurations for processing. Cross-validation sets offer a way to evaluate the efficiency of the system by comparing known and obtained results.

The most frequently used environments for music are variants of supervised learning in a classification problem, with a recurrent architecture design (recurrent neural networks or RNN). Most networks use music pitches as the main input and output parameter and check for activation. Some examples include the use of harmony sequences in a similar way that the model is employed for words or letters

issn 1904-500X

in natural language processing. The recurrent model (RNN) learns from sequential patterns through time, which makes it a good fit for music. Structure design, parameter formulation, and a significant training set (musical examples belonging to a style, a genre, or an author) are determinant for the efficiency of the model.

Long short-term memory (LSTM) networks (Hochreiter & Schmidhuber, 1997), a type of RNN, has a structure that allows past events to influence the hypothesis, making it particularly effective for sequential series. Its structure allows the use of weight on significant events in the timeline, making it similar to the way memory works. It features input and output units with a gate mechanism that learns to filter irrelevant content (low repetition) and to "forget" when information is outdated.

Google's Magenta, an open project from the Google Brain team develops machine-learning systems for music and art creation employing LSTM networks through a *TensorFlow* architecture[21]. *Performance-RNN*, one of Magenta's models, uses dynamics and human timing in the training set (Yamaha e-piano competition dataset), allowing a distinctive performance quality not seen in other models. A version in java script is available, allowing a trained model to function in a browser demonstrating how modifications on weights for different notes are easy to apply in real time.

IV. Interactive and modular *Algorithmic Music Generator* (AMG)

The preceding approaches to music generation feature some level of responsiveness – with the main mechanism responsible for adaptive music generation in a videogame. This particular characteristic is the main interest of the present work since it enables interactivity and then participative story generation through music variation. Some distinctive properties from each model would fit specific stylistic constructions better and consequently suit – or more easily support – specific game aesthetics. Hence, choosing, tailoring and customizing models based on game dynamics and needs would be a first step to design a music engine for a particular videogame.

In a more generic approach, using a basic homophonic music standard as a target and accounting interactive and modular possibilities in its structural design, I propose the following structure for an AMG:

- A multi-agent system is responsible for both percussion and global music timing, reacting to variables related to gameplay, movement, and activity. Distinctive instrumental density, grouping probability, meter and tempo management, and other pre-designed features can influence and frame stylistic properties. Algorithms manage levels in "responsiveness", "confidence", "social", "commitment" and "mischievous" – as in Eigenfeldt's example. The multi-agent network reacts to gameplay speed and/or density indexes by assigning their values to the parameters in the agents' personality configuration.
- Two pre-trained performance-RNN modules manage the leading voice and bass lines to take advantage of the creativity and performativity of the model. Train-

issn 1904-500X

ing the model properly ensures stylistic fidelity to the input set, although some additional heuristics must be applied for quantization and timing. Idiomatic instrumental possibilities and ranges should be employed on the training corpus. Interpolation algorithms for weights between pre-trained models potentially provide refreshing variations and a wider stylistic scope.

- A second or third order Markov model adequately manages instrumental ranges and voice quantity/distribution for pads/strings (whose role in homophony is mostly chords and accompaniment). A subset of the sequences produced by the percussion agents manages the rhythmic properties for this instrumental section. Transition probabilities in the MM are connected to custom gameplay parameters (e.g. emotional levels). In this instrumental category, intervals of silence and wider dynamic ranges should be included in the model design to provide phrasing and variety.
- An ATN provides a tonal grid for all pitched instruments. Through generative grammars and stochastic design, this module drives the AMG harmonic progression stream. Reaction to game inputs is enabled by including a categorized level (e.g. emotionality index) for nodes' possibilities in the ATN design. Therefore, parameters coming from any interaction in gameplay events, sections, environmental conditions, and occurrence statistics influence the ATN's decision regarding chords in upcoming nodes. Additionally, selected parameters manage tension through pitch class addition/subtraction which has the particular effect of incrementally blurring the tonal center. Stochastics design controls a range of variation for stable long environmental segments.

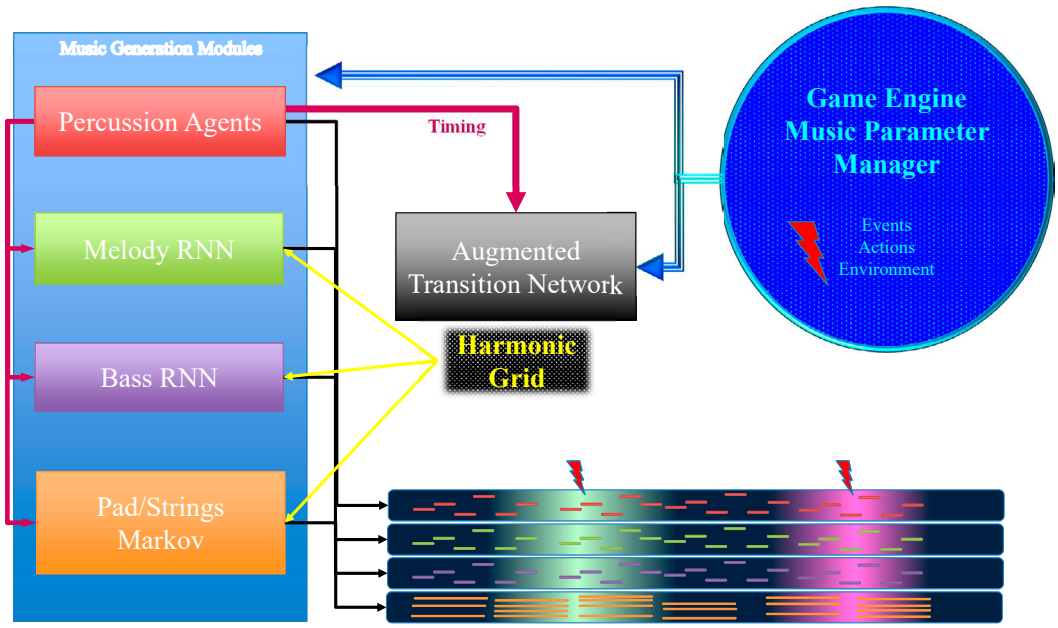

Figure No. 10. Algorithmic Music Generator (AMG). On the left, the modules in charge of musical event generation (percussion agents, melody RNN, bass RNN, pad/strings Markov) are interconnected and responding specifically to timing (red arrow) from the percussion agents. Their stream is shown at the end of black arrows. In the middle, an ATN uses generative grammars to switch a harmonic grid to filter pitches coming from the generators (yellow arrow), and its transitions are activated by the timing. On the right, the game engine provides information about the game state to modify the generators and the ATN. Its changes and transitions show up in the music stream (red thunder).

issn 1904-500X

As a result of the pre-set/recall capabilities on configuration of the AMG parameters and by executing narrow and/or seeded stochastic distributions, similarity with low repetition is possible. Through this approach, the AMG is also able to produce recognizable and almost repetitive gestures for stingers or conclusive sequences.

Quality in performance depends greatly on audio engine management. Employing stochastic methods such as *random walk*[22] algorithms can potentially "humanize" playback. It is achieved by adding the random variation in short ranges to:

- Envelopes in ADSR, volume, effects, and other mixing parameters.
- Timing control, either as a variation in steady-tempo sections or gradual increase/decrease of global tempo.
- Exclusive instrument parameters such as the modulation type/values of synthesizers or sampler's playback configuration levels.

Future work includes implementation possibilities in Unreal Engine and Unity (or any other game development software) and integration with sound design specialized applications such as Wwise and FMOD. A useful strategy would be to place the AMG as a subprogram in the game engine and send music events to a performance system (e.g. synthesizer, sampler) that belongs to the game editor or to the sound

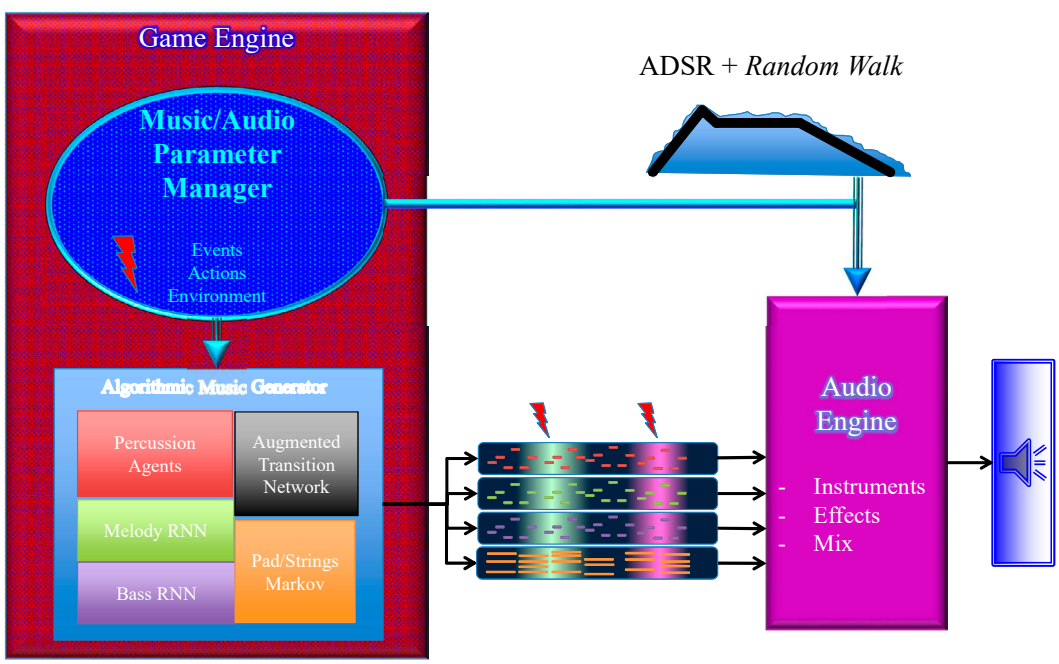

Figure No. 11. Possible implementation of AMG. A section within the game engine is in charge of sending a selection of parameters to the AMG, as shown in figure 10. This section also sends control information to the audio engine parameters passing through a Random Walk shaper. This small change in envelopes for effects, mix, and instruments adds human qualities to the music result. The stream of music in MIDI reaches the audio engine and is rendered in real time using the envelope shaper (ADSR + Random walk).

issn 1904-500X

design *soundbank*[23]. This configuration makes the algorithmic generation independent of the built audio engine.

AMG development should usually involve a pilot version able to receive controls from the game engine. This version ideally is able to test all the parameters coming from gameplay in real time. A pilot designed through third party programming environments (Max MSP/Pd, ChuK, CSound, SuperCollider, etc.) has the advantage of fast structural setting and modular audio implementation. An AMG designed, or transcoded and tested, into the native scripting language of the game engine (usually C++ or C#) should be ready for a built game.

V. Conclusion and final thoughts

The concept of active music generation not only fits into interactive paradigms of videogames, but potentially also benefits other audiovisual or artistic real-time participatory setups. The intended utility is to support a multi-linear narrative interactively and adaptively and as a result extend the output material while preserving aesthetics. The stylistic capabilities of an AMG are determined in general by algorithmic aesthetic decisions and/or by feature learning from a preexistent musical corpus.

To some extent, this tool may be able to generate material even if the human input (either musical or compositional) is reduced or eliminated. In this case, if the machine performs without any guidelines such as a training set, a style-matching parameter setup, or an explicit musical intention, the output will include a large number of random events, resulting in an indeterminate emotional support. Curiously, the act of reducing human intervention or operation to denote a lack of direction could be perceived as a valid aesthetic decision which intrinsically has human origin.

Automatic systems are replacing humans in methodical tasks, but in the field of art production, not only are the origins and primal ideas mostly human (or human + machine), but they are also assessed by humans. This raises an ominous inquiry about the nature of authorship and may need further legal definitions into licensing schemes.

Arguably, formulaic music making by humans or machines has been around for a long time. Styles, genres, and currents have had influential authors and important contributors that rarely create completely new material. An AMG constitutes a streamlined way to perform not only style replication, but also development and variation featuring real-time participation. The idea of a machine producing music material interactively is an opportunity for involvement, re-creation, appropriation and even democratization of art generation processes beyond videogames with the potential of nurturing a wide range of aesthetic auditory experiences.

issn 1904-500X

References

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

Ames, C. (1989). The Markov Process as a Compositional Model: A Survey and Tutorial. *Leonardo*, *22*(2), 175-187.

Brent, R. (2007). Some long-period random number generators using shifts and xors. *ANZIAM J. ANZIAM Journal*, *48*, C188–C202.

Brooks, F., Hopkins, A., Neumann, P., Wright, W. (1993). "An experiment in musical composition." In S.M. Schwanauer, D.A. Levitt (Eds.) *Machine Models of Music*. MIT Press, Cambridge, Mass. ISBN 0-262-19319-1

Burraston, D., & Edmonds, E. (2005). Cellular automata in generative electronic music and sonic art: a historical and technical review. *Digital Creativity*, *16*(3), 165-185.

Chomsky, N. (1956). *Three models for the description of language.* New York.

Chomsky, N. (1968). *Syntactic structures.* The Hague: Mouton.

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

Cope, D. (2008). *Hidden structure: music analysis using computers*. Middleton, WI: A-R Editions.

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

Eigenfeldt, A. (2009). The Evolution of Evolutionary Software: Intelligent Rhythm Generation in Kinetic Engine. *Proceedings of EvoMusart 09 – European Conference on Evolutionary Computing*, Tubingen, Germany, pp. 498-507. Berlin: Springer.

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "live." *Scientific American*, 223, October, 1970.

Hiller, L., & Isaacson L. *Experimental Music* (New York: McGraw-Hill, 1959; re-printed Greenwood Press, 1979)

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735-1780.

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

Miranda, E. (Ed.). (2011). *A-life for music: music and computer models of living systems*. Middleton, WI: A-R Editions.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of machine learning*. Cambridge, MA: MIT Press.

Niazi, M., & Hussain, A. (2011). Agent-based computing from multi-agent systems to agent-based models: a visual survey. *Scientometrics*, *89*(2), 479-499. https://doi.org/10.1007/s11192-011-0468-9

Nierhaus, G. (2009). *Algorithmic composition: paradigms of automated music generation*. Wien; New York: Springer. Retrieved from http://dx.doi.org/10.1007/978-3-211-75540-2

Norris, J. (1998). *Markov chains* (1st public edition). Cambridge, UK; New York: Cambridge University Press.

Novák, V., Perfilieva, I., & Moko, J. (1999). *Mathematical principles of fuzzy logic*. Boston: Kluwer Academic.

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

Sweet, M. (2015). *Writing interactive music for video games: A composer's guide*. Upper Saddle River, NJ; Boston; Indianapolis; San Francisco; New York; Toronto; Montreal; London; Munich; Paris; Madrid; Cape Town; Sydney; Tokyo; Singapore; Mexico City: Addison-Wesley.

von Neumann, J. Various techniques used in connection with random digits. A. Householder, G. Forsythe, and H. Germond (eds.), *Monte Carlo Method*, National Bureau of Standards Applied Mathematics Series, 12 (Washington, D.C.: U.S. Government Printing Office, 1951): 36-38.

Wolfram, S. (2002). *A new kind of science*. Champaign, IL: Wolfram Media.

Zicarelli, D. (1987). M and Jam Factory. *Computer Music Journal*, *11*(4), 13-29. https://doi.org/10.2307/3680237

issn 1904-500X

Notes

1 Bear McCreary's video game credits include Sony's *Socom 4: U.S. Navy Seals*, Capcom's *Dark Void*, and the innovative video game/television hybrid *Defiance*. Other important sound tracks include *Battlestar Galactica, Da Vinci's Demons, Marvel's Agents of S.H.I.E.L.D.* and *The Walking Dead.*

2 Nishikado, Tomohiro. Taito, Japan, 1978.

3 Among others, the two entries by Karen Collins, *From Pac-Man to pop music: Interactive audio in games and new media*, 2008, and *Playing with sound: A theory of interacting with sound and music in video games*, 2013. In the first, some approaches and theories about non-linear music composition, indeterminate audio playback, and interactive or adaptive music for videogames are disclosed. In the second, the author elaborates on the concepts and theories involving sound and image in an interactive scheme such as synchresis, kinesonic congruence, players' audition, co-creativity and performance among others.

4 Especially, in the book *Designing Sound,* Andy Farnell (2010) explains and studies a wide range of sound-design techniques resulting from virtual acoustic conditions, similarly to how physics drive visual rendering in videogames.

5 That is, if the playback system acts as a synthesizer/sampler module and the stream of events feature Midi-style information (pitch, duration and intensity)

6 https://intermorphic.com/

7 https://www.ampermusic.com/

8 https://www.aiva.ai/engine/

9 https://melodrive.com/

10 Algorithmic generation of random sequences by a deterministic process through functions and mathematical operations.

11 Most of the pseudo-random digit generators re-start the sequence after a number of digits. This is called the *period* (Brent, R.P., 2007), and tends to be very long in order to hide any predictable structure.

12 These models constitute the most important field of study in the large field of algorithmic composition, generative music, and artificial intelligence in the last thirty years. Although some of them are far older, the possibilities offered by new hardware produced new results leading to recent approaches and challenges of a higher level such as stylistic and performance generation, genre reproduction, self-similarity, and motivic development among many others.

13 Among many, Xenakis' *Analogique A* (1958), in which MM are employed for density sequences. Also, the previously mentioned *Illiac Suite* (Hiller, 1957) regarded as the first work of computer-assisted composition. A more recent example is J. Chadabe's *M* and *Jam Factory* (Zicarelli, 1987), a software for interactive production of compositional material. A longer list of works using Markov models can be found in "The Markov Process as Compositional Model" (Ames, 1989).

14 In a drum set in which a single performer is assigned to a range of percussion instruments, the idea of including idiomatic limitations such as the possibility of maximal simultaneous hits may also be incorporated.

15 Deep learning refers to multi-layer neural networks. In this type of network, there are three types of layer: input, hidden, and output layer. The hidden layers are composed of nodes that connect input and output layers internally (Mohri & Talwalkar, 2012). The number of neurons and hidden layers is proportional to the capacity of the model to calculate more convoluted (and computationally expensive) hypothetical curves.

16 Hypothesis in *machine learning* refers to the inferred value that the model outputs based on training (Mohri & Talwalkar, 2012).

issn 1904-500X

17 In regression problems, the model predicts a value that normally comes from a continuous function generated in the learning process. In classification problems, the model outputs a label or a category for each input.

18 The cost function calculates the distance between the hypothesis and the expected value, and the back propagation adjusts the weights, starting with the output nodes and going back to the input nodes. The gradient descent is a recurrent function that finds a better value on each iteration (Mohri & Talwalkar, 2012).

19 The activation function sends a value to the next layer depending on the input weight and the type of function. It is basically a decision taken by the node, and the type of output can be a Boolean (activated/deactivated) or a proportional value between 0 and 1.

20 Supervised problems use a labeled training set or a target value that the model uses for training. On the other hand, unsupervised problems allow the model to produce the labels from an unlabeled training set.

21 https://magenta.tensorflow.org/

22 The random walk is a path of random sequential steps. It was described first as "The problem of the Random Walk" (Pearson, 1905).

23 The soundbank refers to a built audio section that encompasses not only the audio samples, but also the algorithms that activate and manage them.

issn 1904-500X