An adaptive music generation architecture for games based on the deep learning Transformer model

Gustavo Amaral, Augusto Baffa, Jean-Pierre Briot, Bruno Feijó, Antonio Furtado

An adaptive music generation architecture for games based on the deep learning Transformer model

Gustavo Amaral
Dept of Informatics
PUC-Rio
Rio de Janeiro, Brazil
gustavoacs99@gmail.com

Augusto Baffa
Dept of Informatics
PUC-Rio
Rio de Janeiro, Brazil
abaffa@inf.puc-rio.br

Jean-Pierre Briot
LIP6 & Dept. of Informatics
CNRS – Sorbonne Université & PUC-Rio
Paris, France & Rio de Janeiro, Brazil
Jean-Pierre.Briot@lip6.fr

Bruno Feijó
Dept of Informatics
PUC-Rio
Rio de Janeiro, Brazil
bfeijo@inf.puc-rio.br

Antonio Furtado
Dept of Informatics
PUC-Rio
Rio de Janeiro, Brazil
furtado@inf.puc-rio.br

***Abstract*—This paper presents an architecture for generating music for video games based on the Transformer deep learning model. Our motivation is to be able to customize the generation according to the taste of the player, who can select a corpus of training examples, corresponding to his preferred musical style. The system generates various musical layers, following the standard layering strategy currently used by composers designing video game music. To adapt the music generated to the game play and to the player(s) situation, we are using an arousal-valence model of emotions, in order to control the selection of musical layers. We discuss current limitations and prospects for the future, such as collaborative and interactive control of the musical components.**

***Index Terms*—video game music, adaptive music generation, deep learning, Transformer, layering, emotion model.**

I. Introduction

Music is essential in video games. It provides an embedding context for the players and complements the scenario [1]. Music can also offer some ways of controlling the player [2]. Meanwhile, a recent observation is that many players replace a game's music by listening to some musical piece of their choice [3]. We postulate that this is because of the absence of enough personalization of the music the game offers. Therefore, we started investigating the possibility of generating personalized music based on player preferences (music style, as defined by a corpus of music samples).

Deep learning techniques are effective in learning a music style from a corpus and generating conformant samples [4]. Various issues still remain on how to customize, control and orchestrate the generation of music in function of the situation (game and player). Because such various open questions, and the fact that generative music for games is still a recent domain with few prototypes (as surveyed in [5] and Section II-A), we believe in the importance of searching for simple models that explore fundamental aspects of music generation for games. More specifically, it becomes essential to look for models that, more straightforwardly, represent the modes of emotion and levels of emotional intensity involved in video game music generation. Also, we must look for deep learning techniques that are especially adequate to the musical narrative. And, perhaps even more importantly, we should seek a model to support instrumental layers used in video game composition (such as the practice of "striping", which is to record orchestral sections separately for future mixing according to the whims of the composer). As film and game music composer Eímear Noone explains: "we might record the strings separately, for example, but we'll compose in a way that the strings on their own provide a functioning piece of music. Then, if our character triggers something in the world, perhaps a battle, we can land the wood winds or brass on top of that to increase the intensity. Each part must be self-contained yet work with others – you need to be able to kick in the brass, kick in the percussion, whenever it's triggered by gameplay." [6]. See also, e.g., [7] for a general introduction to layering.

In this context, instead of looking for alternatives or improvements in the few existing complete models for game music generation (such as the excellent work by Hutchings and McCormack [8], to be analyzed in Section II-B), we decided to explore more straightforward and flexible models for generating and adapting music, based on layering [7]. Also, we believe that our proposed model can facilitate the control and orchestration of music for video games in a collaborative environment.

With the above mentioned principles in mind, after several experiments, we opted for the Transformer architecture [9], because it better captures the long-term structure of music [10].

We thank CNPq (Brazil) for their financial support through Visiting Researcher (PV) fellowship/grant N° 302074/2020-1.

978-1-6654-6156-6/22/$31.00 ©2022 IEEE

In order to model the psychological state of the player as respect to the gameplay context, we decided to select the relatively standard arousal/valence model of emotions [11], and to map it to the control (adaptation) of the generation of music. We associate arousal (i.e., intensity) with the number of active layers (e.g., the system can add a layer with woodwinds or brass to increase the intensity level if a battle starts in the game's world). And the valence corresponds to the emotional modes of the generated music. We also discuss, in this article, future extensions that this simplified approach makes easier to implement. In particular, we want to move towards collaborative and interactive control of the music components generated by the Transformer-based architecture.

The following sections introduce the design, implementation and preliminary experiments with a prototype architecture for generating personalized and adaptive music for games aligned with the above mentioned principles.

II. Background and Related Work

A. Adaptive versus Generative

In [5], Plut and Pasquier present a survey about various approaches and challenges for the generation of music for video games. They consider two primary techniques:

- *adaptive music* (also named *interactive music*), where music is organized in order to be able to react to a game's state [12]. Some musical features (e.g., adding or removing instrumental layers (such as for striping), changing the tempo, adding or removing processing, changing the pitch content...) are linked to game play variables.
  An example of adaptive music is the "Luftrausers" game [13], where the composed music has been split into 3 groupings of instruments, each of which has 5 different arrangements, which a player may select for his avatar (see more details in [5, Section 1.2]).
- *generative music*, where music is not preexisting and dynamically adapted, as for adaptive music, but is generated on the fly. It is created in some systemic way by the computer and is sometimes called procedural music or algorithmic music [14]. The musical content is generated from some model.
  The model can be specified by hand. This was for instance the case for the first piece of music composed in 1957 by a computer (the "ILLIAC I" computer at the University of Illinois at Urbana-Champaign (UIUC) in the United States), and therefore named "the Illiac Suite" [15]. The human "meta-composers" were Hiller and Isaacson, both musicians and scientists. It was an early example of algorithmic composition, making use of stochastic models (Markov chains) for generation as well as rules to filter generated material according to desired properties. The limits are that specifying the model is difficult and error prone. The progress of machine learning techniques made it possible to learn models from examples (in other words, specify a model by extension rather than by intention). All but one of the generative music systems surveyed in [5] are using Markov chains models. Markov models are indeed simpler than deep learning/neural networks models, but they face the risk of plagiarism, by recopying too long sequences from the corpus. Some interesting solution is to consider a variable order Markov model and to constrain the generation (through min order and max order constraints) [16]. The only surveyed game music generation system based on artificial neural networks is Adaptive Music System (AMS) by Hutchings and McCormack [8] and it will be summarized in next section (Section II-B).

Generative music is more general and adaptive than pre-composed composed adaptive music, but is also more difficult to control and more computing demanding. As, noted by [5]: "Another reason that generative music may not have received widespread attention in the games industry is that it is often unpredictable and can be difficult to control. The audio director of "No Man's Sky" game, Paul Weir, notes that generative music was used in the game with an acknowledgment that it could produce "worse" music than composed music." [17]. Actually, that distinction between adaptive and generative music is not that clear, as often systems classified as generative are not completely generative and include adaptation components. This is for instance the case of the Adaptive Music System. Note that it is classified as generative in [5], although its very name claims it as adaptive! In fact we need systems to be both generative and adaptive. We will now summarize it in next section (Section II-B) in order to illustrate some issues and also for its own merits.

B. Architecture of the Adaptive Music System

The architecture of (AMS) [8] is multi-agent and multi-technique:

- the *harmony role agent*, which generates a chord progression, using an RNN (trained on a corpus of symbolic chord sequences, actually an extension of the harmony system from the same authors [18]);
- the *melody role agents* (one for each instrument), which instantiate characteristics (length, pitch, proportion of diatonic notes...) of pre-existing melodies, using an evolutionary rule system (XCS, for eXtended learning Classifier System [19]) and adapting them to the harmony;
- the *rhythm role agent*, which uses another RNN model.

AMS considers a model of 6 emotions: sadness, happiness, threat, anger, tenderness (the most consistently used labels in describing music across multiple music listener studies [20]) and excitement (important for scoring video games), whose selection is triggered by current game state (every 30ms, the list of messages received by the Open Sound Control (OSC) is used to update the activation values). Emotions in turn will modulate the selection among melodies (choosing the melodic theme assigned to currently highest activated concept or affect), with the instantiation of their characteristics being

managed by a spreading activation model. It is a graph of concept nodes, connected by weighted edges representing the strength of the association between the concepts (and is inspired from a semantic content organisation in cognitive science [21]). Activation spreads as a function of the number of mediating edges and their weights. As explained in [8]: "Spreading activation models don't require logical structuring of concepts into classes or defining features, making it possible to add content based on context rather than structure. For example, if a player builds a house out of blocks in Minecraft, it does not need to be identified as a house. Instead, its position in the graph could be inferred by time characters spend near it, activities carried out around it, or known objects stored inside it."

As we can see, AMS actually proposes a sophisticated (and clever) generation model which includes both adaptive and generative aspects (for instance, harmony is generated and melodies are adapted). AMS has been tested in two games: Zelda Mystery of Solarus (MoS) (actually an open-source clone version) [22] and StarCraft II [23].

The comparison of our proposed model with the much more complete and robust AMS architecture is twofold. Firstly, we more straightforwardly represent the emotional intensity in music generation, and secondly, we better support layering music. The simplicity of our approach aims to facilitate future prospecting in collaborative and interactive environments. Furthermore, we use a deep learning architecture (Transformer) better suited to capture long-term coherence in music.

III. Adaptability versus Continuity and other Design Issues

A pure generative approach is some kind of ideal, as it could in principle combine personalization (learnt styles) with real-time adaptation (to the game and players situation). Note that the issue of how to combine various context information, plot, evolution, player(s) situation, etc., including statistics, e.g., average reactivity of a player, into some decision about what is the objective (adapt to current game context, or the opposite, trigger a player to engage more) and how to accordingly adapt the music is still an open issue. It is likely that it should use some aggregation/interpretation rules, as well as multi-criteria decision strategy, within some front end module in charge of mapping events and models from the game up to the control parameters for music generation or/and adaptation.

Using symbolic-level music models as opposed to signal-level music models brings the advantages of higher level manipulation (at the composition level) and less computer resources (although, recent waveform-level models such as WaveNet [24] demonstrated the feasibility of real-time conditioned generation, used for instance for intelligent assistants such as Google Echo or Amazon Alexa). An important and actually difficult issue remains the capacity to generate on the fly music content and be able to adapt it, while maintaining some continuity. (As for a musician improvising in some jazz context, balancing between constructing and following some musical discourse and fitting into the dynamic context, in the first place, harmonic modulations. Also note that music does not necessarily have to adapt immediately to events, as opposed to sound effects.) Recent progress for control strategies for Markov chains and as well for deep learning show promising results. Markov constraints have been proposed as an unifying framework for Markov-based generation while satisfying constraints [25], and has been applied to real-time improvisation [26] and to interactive composition [27]. Challenges for introducing control are somehow harder for deep learning, (as explained, e.g., in [28, Section 10]), but progresses are made, using control strategies such as conditioning (adding some additional input to the neural network in order to parameterize training and generation), e.g., as in [29].

IV. Current Proposal

Although simpler, our current prototype shares some similarity with AMS (see Section II-B and [8]), in that it uses both neural network-based generation and an emotion reference model. In the following sub-sections, we will describe and motivate various aspects and components of the architecture and of the generation process, namely: the general design principles; the curation and pre-processing of the training musical examples; the way music generated is layered; the emotion model chosen to map the game play into some control of the generated music; the mapping discipline; the complete architecture; the implementation; and the preliminary evaluation.

A. Design Principles

After having at first experimented with a recurrent neural network architecture of type LSTM (part of Google's Magenta project library) [30], we selected the Transformer architecture for its ability to enforce consistency and structure, by better handling long-term correlations. Transformer [9] is an important evolution of a Sequence-to-Sequence architecture (based on RNN Encoder-Decoder), where a variable length sequence is encoded into a fixed-length vector representation which serves as a pivot representation to be iteratively decoded to generate a corresponding sequence (see more details, e.g., in [31, Section 10.4]). Its main novelty is a self-attention mechanism (as a full alternative to more classical mechanisms such as recurrence or convolution), to focus on contributing elements of an input sequence. For more details on the architecture, please see the original article [9] or some pedagogical introduction [32]). It recently became popular for such applications as: translation, text generation (e.g., the Generative Pre-trained Transformer 3 aka GPT-3 model), biological sequence analysis and music generation [10].

The proposal by Jeffries for ambient music generation based on the Transformer [33] has also been a source of inspiration.

B. Training Examples

The user may select a corpus of music of its preference (e.g., classical, jazz, techno, ambient…, choosing a more narrow – e.g., of a specific composer or band – or some wider corpus) to

be used as the set of training examples. The music generated will be corresponding to this style thus defined by the user. In the experiment described in this paper, we have chosen a corpus of ambient music, more precisely a Spotify playlist named "Ambient songs for creativity and calm", curated by Jeffries, and containing approximately 20 hours and 165 titles [34].

The compressed audio files (mp3) corresponding to the musical training examples have been uncompressed into wave-form (wav) files and then, by using a pitch detector, to symbolic (midi) files. For the polyphonic transcription to midi files, we used the Onsets and Frames transcription system [35], developed by the Magenta project. It uses both convolutional and recurrent (LSTM) neural networks in order to: 1) predict pitch onset events; and 2) use this knowledge to condition framewise pitch predictions. Obviously, we may also use instead directly MIDI music scores from existing symbolic music libraries.

C. Layering

We consider layers of music, analogous to the production of orchestral music for games [6], with currently up to 4 layers:

- 1st layer, the most conservative and neutral;
- 2nd layer, to add more excitement, e.g., though some additional instrument;
- 3rd and 4th layers, to intensify the immersion and the tension.

These layers are generated from the same learning corpus, but from different seeds (starting sequences) and with different generation parameters (currently, we vary a temperature parameter that controls the determinism of the generation, for some more likely or more unpredictable result), depending on the controlling model (as will be presented in Section IV-E). In addition to this static parameterization of their generation according to the controlling model, each musical layer will be dynamically activated and played (or not) (currently within the Ableton Live platform, a real-time sequencer for live music creation and production [36]), depending on the strategies of the controlling model.

D. Mapping Emotions

In order to have some high-level and human understandable control of the generation by the game play context (game and player(s)), we chose an emotion model, more precisely the arousal/valence model [11], in which an emotion can be mapped using two parameters:

- the *arousal*, which represents the intensity of the emotion;
- and the *valence*, which represents its quality (e.g., if it is positive, negative, neutral...).

In order to simplify our current prototype, we now consider only 9 (discrete) emotions, as illustrated in Fig. 1.

The emotion model is designed as a server receiving control information from the game, in order to be able to work with various games and game values models. The game play information (events) emitted by the game may be about the game situation, player(s) situation, but also from various other sources such as quests, terrains, etc. How to aggregate these various informations is still an open issue for future work (see Section V-A).

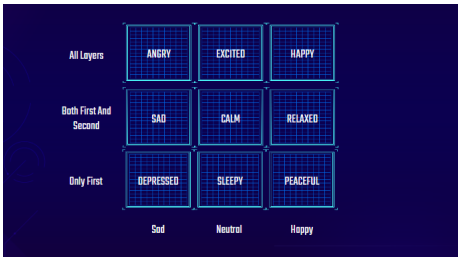

Fig. 1. Strategy/Layer/Emotion model, with the 9 pre-defined emotions based on the arousal/valence emotion model

E. Strategy and Control Model

While planning for the future some more advanced state machine for mapping the emotions into generation control strategies (as will be detailed in Section V-A), in the current prototype we have implemented 9 pre-defined strategies (corresponding to the 9 emotions shown in Fig. 1). For each strategy, different values corresponding to the parameters for the generation: which layers are activated; which instruments (sampled or synthetic sounds, currently selected from some instruments library for ambient music within Ableton Live) are used; and which effects are used. More strategies/types may be added by adding strategy classes to the implementation.

The complete model (Strategy/Layer/Emotion) for controlling music generation is shown in Fig. 1. Current mapping is as follows: the strength (arousal) corresponds to the number of active layers, while the quality (valence) corresponds to the choice of emotional modes of the generated musical components.

F. Architecture

The flow logic of current architecture is as follows:

1) User's client requests a music;
2) The server maps the user feeling through the arousal valence parameters;
3) It fetches, from memory, a song correspondent to the mapped emotion (this optimization is detailed in next Section IV-G);
4) If no associated music has already being generated, it starts the generation;
5) After the music is fetched, it attaches metadata such as instruments;
6) It delivers the request response with the music to the final user;
7) The memory is refreshed.

G. Implementation

To optimize the music generation process, at least one music corresponding to each strategy is saved in memory. The architecture is designed as a server responsible for music generation, for various possible game clients, based on game

engines like Unity or Unreal, or specific ones. In order to automate and scale-up the machine learning life cycle, we have used the Pachyderm platform (pipeline) [37]. For the implementation, we have used Nvidia CUDA development environment for high performance GPU-accelerated applications.

H. Evaluation

Current architecture has been tested with an emulated game model and with music generated from a corpus of ambient music. The complete code as well as input and output examples are available on the following code repository: `https://gitlab.com/music-gen/server`. Arousal valence values have been estimated according to possible moments of the hero's journey and the behavior of the system. We are planning the integration with a real game using Unity.

V. Prospects

A. Game/Music Mapping Model

At present time, input from the game play is limited, but it could benefit from many more parameters and events (e.g., plot situation, player situation, including statistics, e.g., average player reactivity...) and how to aggregate them. And, as opposed to mapping music to player state, we may want to oppose it instead, e.g., if the player is perceived to be showing to signs of abandon, you may want him to try to boost him with some positive music. Last, note that [38] proposes an additional dimension: tension, that you could compute and use to improve the system's emotion mapping.

In addition, as mentioned in Section IV-E, we are planning to substitute current strategy scheme with some more abstract and general state machine model, analog to the AMS spreading activation model (see Section II-B and [8]), in order to track the transitions of the player's emotions. Better transitions between music could also be planned ahead, through interpolation.

B. Interactive Coordination

A more radical approach is to substitute the sequencer-like platform (currently, Ableton Live) by a more general platform for interactive and collaborative control of musical components (being generated by our current Transformer-based architecture). We are thinking of the Skini platform [39]. It allows defining some kind of "orchestral blueprint" (actually, some cartography of possible paths) for activating various musical components of a piece of music. It separates the macro-level coordination from the actual micro-level components, as for architectural/coordination languages in software architectures [40] or distributed systems. Paths may be fixed or open with various choices, to be decided according to the interaction with the public (various active listeners). Fig. 2 shows an example of visual orchestral blueprint (musical flow) in Skini.

The control expression in the Skini platform is based on the integration of the synchronous reactive programming language Esterel [41] in JavaScript (on the Web). The advantage over a sequencer (which has a semi-rigid temporal structure) is the expressive power (Turing complete) of a language like Esterel

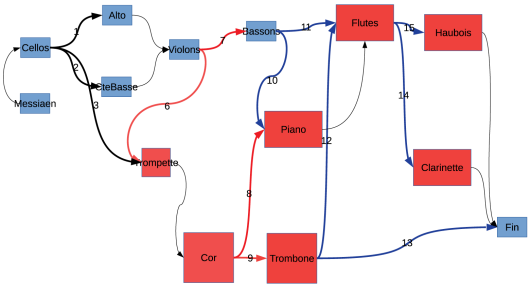

Fig. 2. Example of orchestral flow in Skini (Opus1 Piece by Bertrand Petit). Plain arrows represent fixed paths and bold arrows represent alternative paths which may be decided by the public. Each music/sound component (in blue) may be activated an unlimited number of times, except for "reservoirs" (in red) which are set to have some maximum number of activations

(which, for example, is used to control Airbus planes), to program any type of coordination of real-time musical events, depending on various in-game events. Additionally, Esterel has formal semantics and property verification tools, thus offering possibilities of formally verifying properties, such as the termination or non-simultaneous activation of two arbitrary musical components. The Skini platform's capability for collaborative interactive orchestration (e.g., for simultaneous control interactions by several actors) offers us the right level of management of various interactions and controls coming from the game engine and from the different players. The Skini platform (whose architecture is shown in Fig. 3) has already been tested recently, in a first scenario with a game platform (Unreal Engine 4), to control the scheduling and musical orchestration depending on the situation of the player within the game [42].

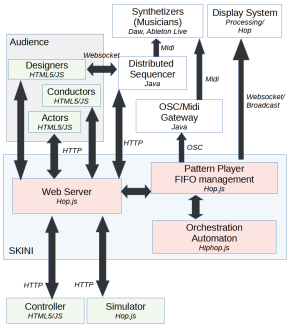

Fig. 3. Skini architecture

VI. Conclusion

In this paper, we have presented an architecture, based on deep learning (more specifically, the Transformer architecture), for generating music for video games, personalized to the user musical preference. It uses the technique of layering, with the activation of layers controlled by an emotion model, in order to adapt it to the game play. Our current architecture is a proof of concept, although it is complete and functional. It has been tested with an emulated game model and with

music generated from a corpus of ambient music. We are currently working on the design of a next version architecture and its coupling with the coordination level based on the Skini architecture. The objective is to decouple the generation and the adaptation of musical components from the way to coordinate and orchestrate them, in order to refine the control of music adaptation according to the game play, independently of the music personalization. We hope that the proposal, although preliminary, as well as the discussion and the prospects presented in this paper, may humbly contribute to a better understanding of the issues and possible directions for next generation game music generation.

REFERENCES

[1] T. Sanders and P. Cairns, "Time perception, immersion and music in videogames," in *Proceedings of the 24th BCS Interaction Specialist Group Conference*, ser. BCS '10. Swindon, U.K.: BCS Learning & Development Ltd., 2010, pp. 160–167.

[2] A. Hufschmitt, S. Cardon, and E. Jacopin, "Dynamic manipulation of player performance with music tempo in Tetris," in *26th International Conference on Intelligent User Interfaces*, ser. IUI '21. College Station, TX, USA: ACM, 2021, pp. 290–296. [Online]. Available: https://doi.org/10.1145/3397481.3450684

[3] G. Ramalho, "Communication during a debate/session," Workshop on AI for (Music and Games) Co-Creation (WAIC 2021), Nov. 2021.

[4] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep Learning Techniques for Music Generation*, ser. Computational Synthesis and Creative Systems. Springer, 2019.

[5] C. Plut and P. Pasquier, "Generative music in video games: State of the art, challenges, and prospects," *Entertainment Computing*, vol. 33, p. 100337, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1875952119300795

[6] K. Stuart, "'Mozart would have made video game music': composer Eímear Noone on a winning art form," *The Guardian*, Oct. 2019. [Online]. Available: https://www.theguardian.com/games/2019/oct/22/mozart-video-game-music-composer-eimear-noone

[7] Hyperbits, "Layering music: 20 ways to layer sounds," Hyperbits, Last access on 16/06/2022, Blog. [Online]. Available: https://hyperbits.com/layering-sounds/

[8] P. Hutchings and J. McCormack, "Adaptive music composition for games," *IEEE Transactions on Games*, vol. 12, no. 3, pp. 270–280, 2020.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," Dec. 2017, arXiv:1706.03762. [Online]. Available: https://arxiv.org/abs/1706.03762

[10] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. S. C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating music with long-term structure," Dec. 2018, arXiv:1809.04281. [Online]. Available: https://arxiv.org/abs/1809.04281

[11] J. A. Russell, "A circumplex model of affect," *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[12] K. Collins, *From Pac-Man to Pop Music Interactive Audio in Games and New Media*. Ashgate Publishing Ltd, 2011.

[13] Vlambeer, "Luftrausers," Devolver Digital, 2014, Game.

[14] G. Nierhaus, *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer, 2009.

[15] L. A. Hiller and L. M. Isaacson, *Experimental Music: Composition with an Electronic Computer*. McGraw-Hill, 1959.

[16] A. Papadopoulos, F. Pachet, and P. Roy, "Generating non-plagiaristic Markov sequences with max order sampling," in *Creativity and Universality in Language*, ser. Lecture Notes in Morphogenesis, M. Degli Esposti, E. G. Altmann, and F. Pachet, Eds. Springer, 2016, pp. 85–103.

[17] P. Weir, "The sound of 'no man's sky'," 2017, Talk. [Online]. Available: https://www.gdcvault.com/play/1024067/The-Sound-of-No-Man

[18] P. Hutchings and J. McCormack, "Using autonomous agents to improvise music compositions in real-time," in *Computational Intelligence in Music, Sound, Art and Design – 6th International Conference, EvoMUSART 2017, Amsterdam, The Netherlands, April 19–21, 2017, Proceedings*, ser. LNCS, J. Correia, V. Ciesielski, and A. Liapis, Eds., vol. 10198. Springer, 2017, pp. 114–127.

[19] S. W. Wilson, "Classifier fitness based on accuracy," *Evolutionary Computation*, vol. 3, no. 2, pp. 149–175, 1995.

[20] P. N. Juslin, "What does music express? basic emotions and beyond," *Frontiers in Psychology*, vol. 4, p. Article 596, 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2013.00596

[21] A. M. Collins and E. F. Loftus, "A spreading-activation theory of semantic processing," *Psychological Review*, vol. 82, no. 6, pp. 407–428, 1975.

[22] Solarus Team, "The Legend of Zelda: Mystery of Solarus," Solarus, 2011, Game.

[23] Blizzard Team, "StarCraft II: Wings of Liberty," Blizzard Entertainment, 2010, Game.

[24] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," Dec. 2016, arXiv:1609.03499. [Online]. Available: https://arxiv.org/abs/1609.03499

[25] F. Pachet and P. Roy, "Markov constraints: Steerable generation of Markov sequences," *Constraints*, vol. 16, no. 2, pp. 148–172, 2011.

[26] F. Pachet, "Musical virtuosity and creativity," in *Computers and Creativity*, J. McCormack and M. d'Inverno, Eds. Springer, 2012, pp. 115–146.

[27] A. Papadopoulos, P. Roy, and F. Pachet, "Assisted lead sheet composition using FlowComposer," in *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, ser. Programming and Software Engineering, M. Rueher, Ed. Springer, 2016, pp. 769–785.

[28] J.-P. Briot, "From artificial neural networks to deep learning for music generation – History, concepts and trends," *Neural Computing and Applications (NCAA)*, no. 33, pp. 39–65, Jan. 2021, Special issue Neural networks in Art, sound and Design, J. Romero and P. Machado, Eds.

[29] G. Hadjeres and F. Nielsen, "Interactive music generation with positional constraints using Anticipation-RNN," Sep. 2017, arXiv:1709.06404. [Online]. Available: https://arxiv.org/abs/1709.06404

[30] Magenta, "Make music and art using machine learning," web Site. [Online]. Available: https://magenta.tensorflow.org/

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[32] M. Phi, "Illustrated guide to Transformers – step by step explanation," Towards Data Science, Apr. 2020, Blog. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

[33] D. Jeffries, "The musician in the machine," Magenta blog, Aug. 2020. [Online]. Available: https://magenta.tensorflow.org/musician-in-the-machine

[34] ——, "Ambient songs for creativity and calm," Spotify, 2022, Playlist. [Online]. Available: https://open.spotify.com/playlist/6qaujvXpcysfuyFMtp7Ljn?\-si=79e3b076defb4952

[35] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," Jun. 2018, arXiv:1710.11153. [Online]. Available: https://arxiv.org/abs/1710.11153

[36] Ableton team, "Ableton Live," Ableton, Last access on 16/06/2022, Music creation and performance software. [Online]. Available: https://www.ableton.com/en/live/

[37] Pachyderm team, "Pachyderm," Github, Last access on 16/06/2022, Code documentation. [Online]. Available: https://github.com/pachyderm/pachyderm/blob/master/README.md

[38] C. Plut and P. Pasquier, "Music matters: An empirical study on the effects of adaptive music on experienced and perceived player affect," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8.

[39] B. Petit and M. Serrano, "Skini: Reactive programming for interactive structured music," *The Art, Science, and Engineering of Programming*, vol. 5, no. 1, p. Article 2, Jun. 2020.

[40] M. Shaw and D. Garlan, *Software architecture – Perspectives on an emerging discipline*. Prentice Hall, 1996.

[41] G. Berry and G. Gonthier, "The Esterel synchronous programming language: Design, semantics, implementation," *Science of Computer Programming*, vol. 19, no. 2, pp. 87–152, 1992.

[42] B. Petit, "Skini et jeu vidéo," Jan. 2021, Blog. [Online]. Available: https://youtu.be/wDoY20ewiWY