Blind Creation: Emerging Music Through Implicit Collaboration

Mário Escarce Junior[1*], Georgia Rossmann Martins[1*], Leandro Soriano Marcolino[2], Yuri Tavares dos Passos[3]

[1] Phersu Interactive, Belo Horizonte, Brazil
[2] School of Computing and Communications, Lancaster University, Lancaster, United Kingdom
[3] Centro de Ciências Exatas e Tecnológicas, Universidade Federal do Recôncavo da Bahia, Brazil
{mario,georgia}@phersu.com.br, l.marcolino@lancaster.ac.uk, yuri.passos@ufrb.edu.br

Abstract. Normally agents cooperate when they have a joint goal or are able to get a higher payoff by doing so. We present a new perspective, where an agent cooperates with another without an explicit intention. We study this perspective in the context of Art Games, by introducing a novel algorithm where a human agent cooperates with a video game system in generating music in an emergent fashion, without needing awareness that he/she is doing so. We present a theoretical analysis of our system, and preliminary real-life experiments performed with human subjects.

1 Introduction

Cooperative agents are able to accomplish hard tasks through their joint work. However, most systems assume that agents explicitly collaborate, by having a joint goal, an utility function that foster collaboration, or even pre-specified coordination rules. In many situations, however, we may have a system where the actions of an agent unintentionally help another. In particular, we may be able to use the actions of an agent to produce works of art in an emergent fashion, without requiring artistic knowledge from the agent, nor an explicit intention to create an artistic piece.

Recent works view the creative process as a collaboration between a human and an AI system [4]. Pachet et al. (2013) [19] present a system where a human musician plays a music sample, and an AI system, after learning the basic music pattern, joins the musician in producing music. Hence, both human and system "jam" together, creating a unique music that neither would construct alone. Moreira et al. (2013) [18] present a system where a set of agents react to human musicians, and human and agents cooperate in producing a live music performance. Note, however, that in all these works the user has to explicitly collaborate with the AI system in the music generation process, even requiring musical background in order for the system to work well.

In this work we present a new algorithm where the actions of an user are used to dynamically emerge a musical piece. This system may be used in the context of Art

[* * *] Mário Escarce Junior and Georgia Rossmann Martins are both first authors of this paper.

TEAMAS'17 is a workshop without formal proceedings. An improved version of this paper later appeared as "Emerging Sounds Through Implicit Cooperation: A Novel Model for Dynamic Music Generation", in AIIDE'17.

Games, a new genre on the game industry that view games as an artistic experience rather than just entertainment. Our algorithm places (invisible) musical cells on the floor of a virtual scenario, which are arranged in a way that fosters the musical production. We present a theoretical analysis of our algorithm, where we show that: (i) our algorithm correctly generates grids that foster the generation of arpeggios, but maintaining the diversity of notes; (ii) we can generate complete grids for an odd number of notes; (iii) a human agent walking in our grids has a higher likelihood of generating music than random walks or randomly drawing notes. Additionally, we develop an art game with our algorithm, and evaluate our approach with real human players. We show that real humans, without realizing the effect of their actions, effectively generate a large number of arpeggios, and classify the product of the system as "music".

2 Related Work

This work is related to the study of cooperation and collaboration in multi-agent systems and mechanism design, besides the study of AI for video games and music generation.

Cooperation is an important topic in multi-agent systems. Cohen and Levesque (1991) [3] presented the Joint Intentions theory, where agents work to accomplish objectives in a joint mental state. Grosz and Kraus (1996) [7] introduced SharedPlans, where actions are composed of a hierarchy of subactions, and agents have individual plans for performing subactions. STEAM [25] combines these ideas in an implemented framework, where agents make a hierarchy of joint intentions. It is also possible to coordinate a team through task allocation, for instance using the contract net protocol [23]. Large groups of agents may also present a global, organized behavior by following local coordination rules [15]. Additionally, in Cooperative Game Theory [2], agents form groups when it allows them to achieve a higher payoff. Our implicit cooperation approach for music generation, however, is essentially different than these previous works. Our work also relates to mechanism design [10], a model in game theory that focuses on designing the rules of a game in order to achieve desired objectives. Traditionally, mechanism design assumes rational agents, while our approach does not need rationality assumptions (in fact, our implicit cooperation is not based on game theory).

Artificial intelligence is used extensively in (video) games. The main focus is on creating strong agents, that are able to win or achieve a high score. It is notorious the recent success of the UCT Monte Carlo algorithm to play board games [6], and the development of machine learning algorithms that are able to perform well in a range of Atari games [1]. In the industry, however, creating strong agents is not as crucial. One of their main objectives is to create non-player characters (NPCs) that make the game enjoyable, often by using scripts and simple techniques [21]. Designers also explicitly program the agents to make mistakes [14]. They still hold the belief, however, that the major role of AI is for creating realistic (or human-like) agent behavior [22].

AI techniques have also been used to automatically create levels of games. For example, Sorenson et al. (2011) [24] combine genetic algorithms and constraint satisfaction for automatically generating "enjoyable" levels. Similarly, Zook and Riedl (2011) [27] present an approach where the difficulty level of the game adjusts dynamically to the current player, in order to make the game enjoyable. In Art Games, however, the objective is in creating a new expressive experience, and not necessarily in making a game "fun".

Concerning music generation, many works present systems that are able to create music without human intervention, by applying machine learning techniques over a music corpus [5]. Others, however, see the creative process as a collaboration between a human and an AI system [4, 19, 18], which aligns with our view. These systems, however, require the human to explicitly join the music generation process, and the human must be an expert in music.

3 Art Games

Art Games consist of a relatively new strand on the game industry. Its projects are normally produced with a low budget and focus on breaking expectations and paradigms, on being provocative, and on creating unique experiences that may be motivated by artistic movements and interventions. The term *Art Game* was used for the first time in 2002 [9], and it can be understood as a game whose structure is destined to produce different reactions in the audience. Similarly to entertainment-focused games, Art Games normally have audio and video output and interactive interfaces, but in a non-conventional way. Often their visual is aesthetically surreal, and not necessarily realistic. The way the player interacts with the game (i.e., the game mechanics) is generally unusual. Moreover, instead of a fixed, linear script, they may borrow, for example, characteristics from contemporary literature, and present stories that are non-linear and often open to a variety of interpretations.

Comprehended within the Art Games genre, there are sub-genres focused on specific artistic segments, such as music, as we can observe on the *Music Video Games*. In Music Video Games, the game mechanics are oriented around the interaction with very elementary musical elements, such as rhythm, which makes them easily associable with many traditional puzzle games that uses rhythmic structures to overcome obstacles (for example, in games such as Vib-Ribbon (Sony Interactive Entertainment, 1999) and Patapon (SIE Japan Studio, 2007)). The creative expressiveness of the user in the current projects of this genre is normally very restricted, since the only action allowed for the player is usually to press determined buttons at predetermined moments.

Within this context, we propose a system for Art Games that allows the user's creative participation on an interactive musical system, even when she/he is engaged in other aspects of the system, such as exploring the environment or collecting artifacts.

4 Implicit Cooperation

In this paper we present a novel model of implicit cooperation, applied in the context of emergent music generation. We will first introduce the implicit cooperation problem, and then we will explain our implicit cooperation algorithm for the generation of music through the user interaction with a system.

Implicit cooperation consists of a multi-agent system where agents collaborate without the *intention* of doing so. That is, while an agent is pursuing its own objectives, its actions end up aiding another agent. In the context of human-computer interaction, we design a system which will accomplish its objectives with the help of a user, but without requiring from him/her an explicit intention of collaborating with the system.

That is, given an agent ϕ, which receives a reward r_a for each action a. Let's assume that ϕ wants to maximize its total reward (for instance, explore the world as much as possible, or collect items in an environment). Given now a system S, with an objective O (for instance, generate music with a certain characteristic). The implicit cooperation problem is: how can we induce agent ϕ to accomplish objective O? In this paper we present a solution for emergent music generation.

4.1 Emergent Music Generation

We present a system where the movement of an agent produces music in an emergent fashion. First, we provide a definition of "music" in the context of our work.

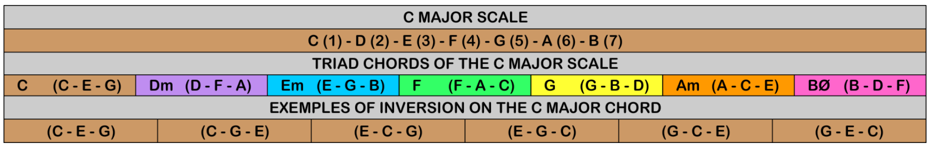
Music: As any art form, there are many possible definitions for music. Our work aligns to a particular one, attributed to the modernist composer Edgard Varèse [17], that music is nothing more than "organized sounds". We first introduce the *scale*, a musical concept that is important to the game mechanics we will present later on. A musical scale is a set of musical notes arranged in sequence from the lowest note to the highest note. For example, from a major C scale, we have the sequence CDEFGAB until we reach the C once again, one octave above. In this work we will not consider accidents (which increase or decrease a note in half a tone).

Another key musical theory element that served as a guideline for our system is: *repetition*. Repetition is a strong factor for musicalization, because it "breaks" a song into pieces and seams them together forming new patterns in a way to preserve an initial structure, making it easier for our brains, an avid "devourer of patterns" [12] to easily assimilate it and recognize it as music. According to Elizabeth Hellmuth [16], if we are asked whether a particular piece is music or not, a remarkably large part of the answer appears to be: 'I know it when I hear it again.' She also stated that repetition serves as a "handprint" of human intent, and a phrase that might have sounded arbitrary at first may sound reasonable the second time it is heard.

The other concept we will introduce is **chord**. Chords are any harmonic set of three or more notes that are heard resonating simultaneously [11]. The most frequent type of chords, which will be widely approached in this work, are the triads, that consists of three distinct notes: the fundamental, which gives the name to the chord; the third, which is the note next to the one following the fundamental; and the fifth, which is the note next to the one following the third [11]. Hence, a triad is a sequence of thirds, where the next note in the scale is always skipped. There is also a different type of chord called **power**

chord, commonly executed with the effect of distortion on electric guitars [26]. In this arrangement, only two notes are played: the fundamental and the fifth. For example, CEG and CG are examples of a triad and a power chord, respectively.

It is also possible to form chords when playing only one note at a time. These are called ***arpeggios***, which is the successive execution of the notes of a chord (in any order) [20]. In our music generation system, we will have only one note being played at a time. Therefore, we will focus on the presence of arpeggios rather than chords. Figure 1 shows an example of possible arpeggios of triads, and all possible orders for the CEG case.

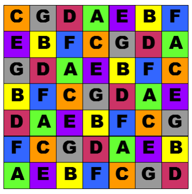


C MAJOR SCALE						
C (1) - D (2) - E (3) - F (4) - G (5) - A (6) - B (7)						
TRIAD CHORDS OF THE C MAJOR SCALE						
C (C - E - G)	Dm (D - F - A)	Em (E - G - B)	F (F - A - C)	G (G - B - D)	Am (A - C - E)	BØ (B - D - F)
EXEMPLES OF INVERSION ON THE C MAJOR CHORD						
(C - E - G)	(C - G - E)	(E - C - G)	(E - G - C)	(G - C - E)	(G - E - C)	

Fig. 1: The C Major Scale, all possible triads of thirds, and an example of all possible inversions for the triad CEG.

Emergent Generation: We consider our agent as a character in a scenario, controlled by a human player. This agent will pursue some objective in this scenario, for instance, collect items or exploration. The actual objective of the player will depend on the system designed using our technique, and does not affect our approach.

We place musical cells on the floor of the scenario. That is, we divide the environment in a grid, where each cell corresponds to a piano key. When the agent steps in a cell, the corresponding key plays. The grid may be invisible to the agent, and it may or may not be aware of this construction. Additionally, we consider that the agent is able to jump on the same place, and that would re-play the same note.



C	G	D	A	E	B	F
E	B	F	C	G	D	A
G	D	A	E	B	F	C
B	F	C	G	D	A	E
D	A	E	B	F	C	G
F	C	G	D	A	E	B
A	E	B	F	C	G	D

Fig. 2: 7x7 building block.

When placing the grid, we use a "building block", which is concatenated in all directions in order to cover the full scenario. This can also be seen as if the "building block" is a torus, that is: upon going right in the last column, the agent will reach the first column; upon going down in the last row, the agent will reach the first row. We show in Table 1 (a) one 3x3 block, and in Table 1 (b) how it would cover a 6x6 scenario.

We generate the blocks in a way that when the agents moves towards the south, it follows a sequence of thirds, and thus creates arpeggios. Similarly, if the agent moves towards the east, it follows a sequence of fifths, also creating arpeggios. For example, in the case of 7 keys, we can use the block shown in Figure 2 (where the colors are used to help visualize different keys). These blocks can be generated as described below.

Let $\mathbf{M} = \{m_1, ..., m_n\}$ be a set of notes, and B an $n \times n$ matrix. We can generate our proposed block by the Algorithm 1. That is, we start from the upper left corner of the matrix, and fill in each cell of the first row in a progression of fifths (i.e., skip the next 3

A	B	C
C	A	B
B	C	A

(a) 3x3 Block

A	B	C	A	B	C
C	A	B	C	A	B
B	C	A	B	C	A
A	B	C	A	B	C
C	A	B	C	A	B
B	C	A	B	C	A

(b) 6x6 Scenario

Table 1: Example of a 3x3 block covering a 6x6 scenario.

elements of the set). After finishing the first row, we fill all columns in a progression of thirds (i.e., skip the next element of the set).

Algorithm 1: Block generation algorithm.

```
$B[1,1] := 1$ ;
for $c := 1 \dots n-1$ do
    $B[1,c+1] := mod(B[1,c]+4, n)$ ;
end
for $c := 1 \dots n$ do
    for $l := 1 \dots n-1$ do
        $B[l+1,c] := mod(B[l,c]+2, n)$ ;
    end
end
```

Therefore, Figure 2 shows the case where $\mathbf{M} = \{C, D, E, F, G, A, B\}$. Note that in this example we start with C, following the usual musical scale, but different starting notes could be used. Also, when moving north the agent will play a decreasing sequence of thirds, and likewise when moving west a decreasing sequence of fifths. As a consequence, for $|\mathbf{M}| = 7$ we also have that: (i) When moving northeast or south, the agent plays a sequence of thirds; (ii) When moving north or southwest, the agent plays a sequence of sixths; (iii) When moving west, the agent plays a sequence of fourths; (iv) When moving east, the agent plays a sequence of fifths; (v) When moving southeast, the agent moves in a sequence of sevenths; (vi) When moving northwest, the agent moves one tone up. This shows that even though we are emphasizing the generation of thirds and fifths, the agent can still generate a great variety of notes from its current position, increasing the diversity of the musical production (in fact, for $|\mathbf{M}| = 7$, we can generate any possible note from a given position).

Additionally, in our analysis we will also consider sets of notes of size different than 7. That could represent, for instance, notes of the next octave; or even a non-traditional division of a given frequency range in n different notes.

Analysis: We will start by analyzing the correctness of Algorithm 1. It is clear that there exists a bijective function that maps the set $\mathbf{M}$ to $\mathbb{Z}_n$. Also, the "third" of a note m_i is equivalent to the note $m_{(i+2) \mod n}$. In a general way, a note m_i changes in a sequence of k-th to $m_{k+n-1 \mod n}$. Therefore, we can consider the cyclic group $(\mathbb{Z}_n, \oplus)$, with $\oplus$ representing addition modulo n, as an isomorphism to set $\mathbf{M}$ under operation of changing in a sequence of k-th.

The following theorem (from [13]) will be useful to prove Algorithm 1 correctness:

Theorem 1. *Let $(G, *)$ be a cyclic group, $|G| = n$, and $a^k = \underbrace{a * a * \cdots * a}_{k\ times}$. If $a \in G$ is a generator of G and k is relatively prime to n, then a^k is also a generator of G.*

Hence, considering the group $(\mathbb{Z}_n, \oplus)$, 1 is its generator. Also, we have for every integer $k > 0$ that $1^k = k \mod n$. So, every $0 < k < n$ relatively prime to n is also a generator.

The following observation states that generating a progression of thirds and fifths in south and east direction respectively allows the agent to move as enumerated above. Consider below that an integer $k > n$ is the same as $k \mod n$ and for any $a \in \mathbb{Z}_n$, its inverse is $-a = n \ominus a = n - a \mod n$. Let $b = B[i, j]$.

Observation 1 *If $B[i, j+1] = b \oplus 4 \wedge B[i+1, j] = b \oplus 2$, then: $B[i-1, j-1] = b \oplus -6$, $B[i-1, j] = b \oplus -2$, $B[i-1, j+1] = b \oplus 2$, $B[i, j-1] = b \oplus -4$, $B[i+1, j-1] = b \oplus -2$, $B[i+1, j+1] = b \oplus 6$*

As it is valid for every integer i, j, $B[i, j-1] = b \oplus -4$ and $B[i-1, j] = b \oplus -2$ is trivially true. Thus we have that: $B[i-1, j-1] = B[i-1, j] \oplus -4 = b \oplus -2 \oplus -4 = b \oplus -6$; $B[i-1, j+1] = B[i-1, j] \oplus 4 = b \oplus -2 \oplus 4 = b \oplus 2$; $B[i+1, j-1] = B[i+1, j] \oplus -4 = b \oplus 2 \oplus -4 = b \oplus -2$; $B[i+1, j+1] = B[i+1, j] \oplus 4 = B[i+1, j] \oplus 2 \oplus 4 = B[i+1, j] \oplus 6$.

Table 2 shows Observation 1 applied to moves in a set $|\mathbf{M}| = 7$. Note that positive relations ($B[i-1, j+1]$,$B[i+1, j+1]$,$B[i+1, j]$,$B[i, j+1]$) will remain as in Table 2 for any set size n, while negative relations will change according to the calculation of the inverse $n - a \mod n$. Also, we will use the following lemma:

$\oplus 1$ B[i-1,j-1]	$\oplus 5$ B[i-1,j]	$\oplus 2$ B[i-1,j+1]
$\oplus 3$ B[i ,j-1]	$\oplus 0$ B[i ,j]	$\oplus 4$ B[i ,j+1]
$\oplus 5$ B[i+1,j-1]	$\oplus 2$ B[i+1,j]	$\oplus 6$ B[i+1,j+1]

Table 2: Neighborhood of a cell as stated in Observation 1 for $|\mathbf{M}| = 7$.

Lemma 1. *If for every integer $i \in \{1, \ldots, n-1\}$ and $j \in \{1, \ldots, n\}$, $B[i+1, j] = B[i, j] \oplus 2$ and for all $j \in \{1, \ldots, n-1\} B[1, j+1] = B[1, j] \oplus 4$, then: $\forall i \in \{1, \ldots, n\} : \forall j \in \{1, \ldots, n-1\} : B[i, j+1] = B[i, j] \oplus 4$.*

Proof. We use induction on i. Base: As hypothesis is given in terms of $i = 1$, we begin with $i = 2$. Thus, we have for every $j \leq n-1$, $B[i, j+1] = B[2, j+1] = B[1, j+1] \oplus 2 = B[1, j] \oplus 4 \oplus 2 = B[2, j] \ominus 2 \oplus 4 \oplus 2 = B[2, j] \oplus 4 = B[i, j] \oplus 4$.

Induction: As induction hypothesis, consider that $\forall i \in \{1, \ldots, n-1\} : \forall j \in \{1, \ldots, n-1\} : B[i, j+1] = B[i, j] \oplus 4$. So, for $i = n$, we have: $B[n, j+1] = B[n-1, j+1] \oplus 2 = B[n-1, j] \oplus 4 \oplus 2 = B[n, j] \ominus 2 \oplus 4 \oplus 2 = B[n, j] \oplus 4$. $\square$

Now we can show the correctness of Algorithm 1:

Theorem 2. *Algorithm 1 generates blocks so that the agent movement plays notes in the proposed way.*

Proof. By Observation 1, we only need to prove that Algorithm 1 generates blocks such that $B[i, j+1] = B[i, j] \oplus 4$ and $B[i+1, j] = B[i, j] \oplus 2$, for every integers i, j. At end of line 4 we have the following postcondition: $B[1, 1] = 1 \wedge \forall j \in \{1, \ldots, n-1\} : B[1, j+1] = B[1, j] \oplus 4$. We need to show that second for loop has the following postcondition: $\forall j \in \{1, \ldots, n\} : \forall i \in \{1, \ldots, n-1\} : B[i+1, j] = B[i, j] \oplus 2$.

This is done by showing a postcondition for the innermost for loop, and then the postcondition above. At innermost for (lines 6-8), we have the following precondition: $1 \leq c \leq n+1 \ \wedge \forall i \in \{1, \ldots, n-1\} : \forall j \in \{1, \ldots, c-1\} : B[i+1, j] = B[i, j] \oplus 2$, and state the following loop invariant in innermost for: $1 \leq c \leq n \wedge 1 \leq l \leq n \ \wedge \forall i \in \{1, \ldots, n-1\} : \forall j \in \{1, \ldots, c-1\} : B[i+1, j] = B[i, j] \oplus 2 \ \wedge \forall i \in \{1, \ldots, l-1\} : B[i+1, c] = B[i, c] \oplus 2$. **Initialization**: Until comparison $l \leq n-1$ at line 6, this is trivially true; **Maintenance**: At line 7, $B[i+1, c]$ becomes $B[i, c] \oplus 2$. Then, for every line i from 1 to l, $B[i+1, c] = B[i, c] \oplus 2$. After increment of l, loop invariant is maintained; **Termination**: All lines $i < n$ in column c obeys $B[i+1, c] = B[i, c] \oplus 2$. Thus, we have as postcondition of innermost for: $1 \leq c \leq n \ \wedge \forall i \in \{1, \ldots, n-1\} : \forall j \in \{1, \ldots, c\} : B[i+1, j] = B[i, j] \oplus 2$.

Now, for outermost loop (lines 5-9), we state the following loop invariant: $1 \leq c \leq n+1 \ \wedge \forall i \in \{1, \ldots, n-1\} : \forall j \in \{1, \ldots, c-1\} : B[i+1, j] = B[i, j] \oplus 2$. **Initialization**: After initialization of c, loop invariant is vacuously true; **Maintenance**: Loop invariant is precondition of innermost for, hence, before increment, for every line $i < n$ in column c, $B[i+1, c] = B[i, c]$. After increment, invariant is maintained; **Termination**: When c becomes $n+1$, loop invariant becomes: $\forall j \in \{1, \ldots, n\} : \forall i \in \{1, \ldots, n-1\} : B[i+1, j] = B[i, j] \oplus 2$.

This proposition together with postcondition of first for (line 4), give us: $B[1, 1] = 1 \wedge \forall j \in \{1, \ldots, n\} : \forall i \in \{1, \ldots, n-1\} : B[i+1, j] = B[i, j] \oplus 2 \ \wedge \forall j \in \{1, \ldots, n-1\} : B[1, j+1] = B[1, j] \oplus 4$. Hence, by Lemma 1, we have the result. $\square$

In the following Corollary, we show that the blocks will always by cyclic, i.e., will allow the agent to navigate as exemplified in Table 1.

Corollary 1. *Algorithm 1 generates cyclic grids for any set* $\mathbf{M}$*, and B[1,1] set initially with any* $m \in \mathbf{M}$.

Proof. Algorithm 1 generate the same pattern of notes for every cell with $i, j > n$, because for every $i, j : B[i+n, j] = B[i+n-1, j] \oplus 4 = B[i+n-2, j] \oplus 4 \oplus 4 = ... = B[i, j] \oplus \bigoplus_{k=1}^{n} 4 = B[i, j]$, and $B[i, j+n] = B[i, j+n-1] \oplus 2 = B[i, j+n-2] \oplus 2 \oplus 2 = ... = B[i, j] \oplus \bigoplus_{k=1}^{n} 2 = B[i, j]$. □

Besides generating thirds and fifths, we also need a system that can generate all possible notes, increasing the richness of the musics produced. Hence, we call a block “complete” when it has all elements of $\mathbf{M}$. In the next proposition, we show the conditions for our algorithm to generate complete blocks:

Proposition 1. *Algorithm 1 generates complete blocks for any set* $\mathbf{M}$ *with* $|\mathbf{M}|$ *not divisible by 2 or 4 and B[1,1] set initially with any* $m \in \mathbf{M}$.

Proof. By Theorem 1, we have that every $0 < k < n$ relatively prime to n is a generator of the group $(\mathbb{Z}_n, \oplus)$. Hence, for $|\mathbf{M}|$ not divisible by 2 or 4, we have that both 2 and 4 will be generators. Therefore, all row and columns in B will have all elements in $\mathbf{M}$. □

Let’s assume now random walks in our proposed blocks. We will consider two different kinds of random walks: (i) From a given cell, uniform probability to move to any neighboring cell; (ii) Greater probability of moving in straight and lateral directions (i.e., “human”-like movement). We will focus our analysis now in blocks where $|\mathbf{M}| = 7$.

In the analysis below, we define music as a repetition of a sequence of notes containing a sequence of thirds or fifths, with other notes between repetitions. In other words:

Definition 1. *Let* $N, M \in \mathbb{N}$, *and* $M \geq N$. *A sequence of notes* $\{a_i\}_{i \in \{1,...,M\}} \in \mathbb{Z}_n$ *is a* music, *if there is a sequence of notes* $A = \langle a_1, a_2, \ldots, a_N \rangle$, *such that* $\forall i > 1 : a_{i+1} - a_i \in \{2, -2, 4, -4\}$, *and* $\{a_i\}_{i \in \{1,...,M\}} = \langle A, B_1, A, B_2, \ldots, A, B_K \rangle$, *for a given* $K \in \mathbb{N}$*; and* B_i *are any sequence of notes of any size, even size zero.*

Proposition 2. *Random walks in blocks generated by Algorithm 1 have a higher probability of generating music than randomly selecting notes.*

Proof. Clearly, sequences of type $\{a_i\}_{i \in \{1,...,M\}} = \langle A, B_1, A, B_2, \ldots, A, B_K \rangle$, will be generated with higher probability as the probability of generating a sequence $A = \langle a_1, a_2, \ldots, a_N \rangle$, gets higher. Hence, we focus on studying the probability of generating a sequence A. Given a sequence A of size N, the first note can be any from $\mathbf{M}$, so there are seven possibles outcomes with $1/7$ probability. From our definition of music, for $i > 1$, the i-th note must be any of four possibles notes among seven from $\mathbf{M}$. Thus, the probability for generating music from this sequence randomly is $P_{r.p.}(A) = 7\frac{1}{7}\prod_{i=2}^{N}\frac{4}{7} = (\frac{4}{7})^{N-1}$, assuming uniform distribution for drawing notes.

Algorithm 1 generates 8 neighbors and the agent can repeat the same note when jumping in the same cell. Hence, random walks in the neighborhood of a cell at every iteration has 9 possible notes and 2 repeated notes for a third above, 2 repeated notes for a third below, and one cell for a fifth above, and another cell for fifth below (see number of cells for +2, +5, +4, and +3 respectively in Table 2). Assuming uniform probability to move to any of these nine blocks, and that the first note is chosen randomly, the random walk probability of a sequence $A = \{a_1, \ldots, a_N\}$ is $P_{r.w.}(A) = 7p(a_1)\prod_{i=2}^{N} p(a_i|a_{i-1}) = 7\frac{1}{7}\prod_{i=2}^{N} p(a_i|a_{i-1}) = (\frac{6}{9})^{N-1}$, since, for $i > 1$: $p(a_i|a_{i-1}) = \frac{6}{9}$, if $a_i \ominus a_{i-1} \in \{2, -2, 4, -4\}$; $\frac{1}{9}$, otherwise. Hence, whatever note chosen initially, a random walk has probability $\frac{6}{9}$ at each step for generating music, because there are six directions that contribute for generating a sequence A: north, east, west, south, southwest and northeast (Table 2). Against $\frac{4}{7}$ probability of randomly drawing notes, it is more probable for random walking in our proposed blocks to generate music. □

Additionally, we will assume that when humans are playing a game, it is more likely that they move in horizontal and vertical directions (north, south, east, west) than diagonals. For instance, there are no diagonals keys in computer keyboards, which would make these movements less likely. Therefore, we have that:

Proposition 3. *Humans have a higher probability of generating music than random walks, when moving in blocks generated by Algorithm 1.*

Proof. By the assumption above, let's consider that human agents move according to the following probability: $p(move) = p+\epsilon, move \in \{\text{north, south, east, west}\}; p$, otherwise.

Additionally, we have that $\epsilon > 0, p > 0$, and

$$9p + 4\epsilon = 1. \tag{1}$$

Thus, a human has, at each step, a probability of $4 \times (p + \epsilon) + 2 \times p$ to generate music. As observed in Proposition 2, random walking has probability $\frac{6}{9}$. So, we must have

$$4(p + \epsilon) + 2p > \frac{6}{9}, \tag{2}$$

for human moves be more probable to generate music. The line segment of Equation 1, restricted to $\epsilon > 0$ and $p > 0$, is always inside the region determined by Equation 2. Therefore, any value of p and ϵ greater than zero that satisfy Equation 1, also satisfy Equation 2, completing the proof. □

5 Results

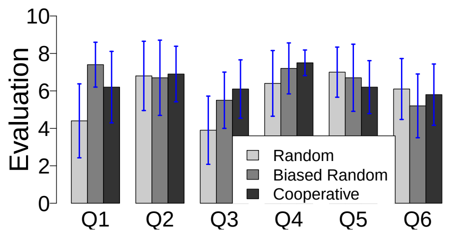
We evaluate our approach in experiments with human players, and we analyze the sounds produced. For comparison, we analyze three different systems: (i) *Random:* Every time the agent steps in a cell, a note drawn uniformly randomly from $\mathbf{M}$ is played; (ii) *Biased Random:* Similar to *Random*, but notes that are the third or the fifth of the note that was played previously are drawn with a probability of 70% (equally distributed), while all other notes are drawn with probability 30% (equally distributed); (iii) *Cooperative:* Follows our implicit cooperation scheme described in the previous section.

Hence, in *Random* notes are drawn completely arbitrarily; while *Biased Random* still draws notes randomly, but following the basic principle from music theory that thirds and fifths should appear with higher likelihood, forming arpeggios. We implemented our system in a game, where a character (controlled by the user), is able to walk freely in an environment and collect objects. These objects are created with the purpose of motivating the users to explore the environment. We uniformly randomly select 3 cells that are currently visible to contain objects. Once new cells become visible (due to the user movement), we repeat the same procedure. We display a score based on how many items were collected, to represent the system as a "video game" to the users.

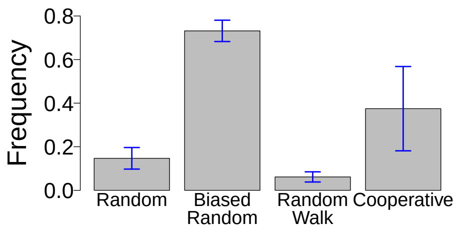
We had a pool of 10 users, and each one played all 3 systems. We randomized the order that each user played each system, in order to avoid ordering issues. Additionally, the users did not know how our system works, nor which one of the 3 systems they were currently playing (the 3 variations were presented to them as X, Y and Z). Each variation was played for 180s, and after that they had to fill in a form about their experience. A video showing the systems is available at `https://youtu.be/EtmtwldRCQs`. We queried the users the following questions: (i) On a scale of 0 to 10, where 0 means "completely arbitrary" and 10 means "very interesting", how do you classify the audio of the system?; (ii) On a scale of 0 to 10, where 0 means "no relation at all" and 10 means "very related", how do you classify the relation between your actions and the audio of the system?; (iii) If you believe there is a relation, on a scale of 0 to 10, where 0 means "very rarely" and 10 means "very frequently", how do you classify the occurrence of a major motivation to "compose" a song while exploring the virtual environment?; (iv) On a scale of 0 to 10, where 0 means "definitely not" and 10 means "absolutely", do you classify the sound output of the system as "music"?; (v) On a scale of 0 to 10, where 0 means "very disturbing" and 10 means "very pleasant", how do you classify the audio experience provided by the system?; (vi) On a scale of 0 to 10, where 0 means "not engaging at all" and 10 means "very immersive", how do you classify your experience with the system as a whole?

We can see the result in Figure 3 (a). Humans could perceive that *Random* produced more arbitrary sounds, while the sounds produced by *Biased Random* and *Cooperative* were considered more interesting (Q1). We also notice that users were not able to distinguish the importance of their actions in the audio generation process across the three systems (Q2). Additionally, when queried to assume that there is a relation, they considered feeling a stronger motivation to generate music in *Cooperative*, even though they did not perceive that their actions had a greater effect in *Cooperative* (Q3). We also notice that the audio of *Cooperative* had the greatest tendency to be classified as "music", with *Biased Random* close behind. *Cooperative* also had the lowest variance in this aspect, indicating that users were more likely to agree in classifying the system as producing "music" than in the other systems (Q4). Interestingly, although users tended to agree more that *Cooperative* generates music, they also tended to perceive it as more "disturbing" than in the other systems (Q5). Finally, in terms of feeling engaged with the system, both *Random* and *Cooperative* had similar results, with *Biased Random* right behind (Q6).

It is interesting to note that users were not able to perceive that their actions had a greater effect in the music generation process in the *Cooperative* system, but they were more



(a) Survey comparing the different systems. Error bars show the 90% confidence interval.



(b) Frequency of triads of thirds. Error bars show standard deviation.

Fig. 3: Results of the experiment with real users.

likely to classify the audio of *Cooperative* as "music", and also felt a greater motivation to produce a song in *Cooperative*. It is also interesting that users perceived the experience as more "disturbing". We do not see that as a negative result, as art does not necessarily have to be pleasant; in fact providing a disturbing experience is also one of the main objectives of contemporary arts [8]. In order to better confirm these results, however, it is necessary to run experiments with a larger pool of human subjects.

We analyzed the audio produced. In Figure 3 (b) we show the frequency of occurrences of triads of thirds, in either increasing or decreasing order, across 10 executions. *Random Walk* refers to walking in our blocks with uniform probability to any direction (including jumping in the same cell), while *Cooperative* is the data with 10 real human users. Hence, as we can see, the presence of a human agent is essential in our system for the formation of arpeggios (which increase the sound quality), even though the user is not aware of how our system works, and is not actively trying to generate those structures. Additionally, even though *Biased Random* has a higher frequency of arpeggios, it did not have a higher tendency to be classified as music as our proposed system. In terms of power chords, we find a frequency of $24.3\%(\pm 8.9\%)$ in the real executions of *Cooperative*.

6 Conclusion

We proposed a system where a human agent collaborates in emergent music generation. However, the agent collaborates as a "side-effect" of its behavior, and does not need to be actively involved, and is not required to be a music expert. We prove the correctness of our algorithm, and study the probability of generating music, showing that it is greater with the presence of a human agent. Our experimental results also indicate a larger frequency of arpeggios when a human uses our system, which indicates musical quality. Additionally, experiments with 10 human players show that users were not aware of their impact (in comparison with randomly drawing notes), but were more likely to define the product of the system as "music" when using our approach. It is still necessary, however, to runs experiments with a larger pool of human subjects, in order to better confirm our conclusions; and to verify our assumption that humans tend to move less in diagonals.

References

1. Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M.: The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research 47, 253–279 (2013)
2. Branzei, R., Dimitrov, D., Tijs, S.: Models in Cooperative Game Theory. Springer Berlin Heidelberg (2008)
3. Cohen, P.R., Levesque, H.J.: Confirmation and joint action. In: Proceedings of the International Joint Conference on Artificial Intelligence. IJCAI (1991)
4. d'Inverno, M., McCormack, J.: Heroic versus Collaborative AI for the arts. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence. IJCAI (2015)
5. Eigenfeldt, A., Pasquier, P.: Realtime generation of harmonic progressions using constrained markov selection. In: Proceedings of the First International Conference on Computational Creativity. ICCC (2010)
6. Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with patterns in Monte-Carlo Go. Tech. rep., Institut National de Recherche en Informatique et en Automatique (2006)
7. Grosz, B., Kraus, S.: Collaborative plans for complex group actions. Artificial Intelligence 86, 269–358 (1996)
8. Henley, D.: Art of disturbation: Provocation and censorship in art education. Art Education 50(4), 39–45 (1997)
9. Holmes, T.: Art games and breakout: New media meets the american arcade. In: Computer Games and Digital Cultures Conference (2002)
10. Hurwicz, L., Reiter, S.: Designing Economic Mechanisms. Cambridge University Press (2006)
11. Karolyi, O.: Introducing Music. Penguin Books; Reissue edition (1965)
12. Koster, R.: A Theory of Fun for Game Design. Paraglyph Press (2004)
13. Ledermann, W.: Introduction to the theory of finite groups. Oliver and Boyd (1949)
14. Lidén, L.: Artificial stupidity: The art of intentional mistakes. In: Rabin, S. (ed.) AI Game Programming Wisdom, vol. 2, pp. 41–48. Charles River Media (2004)
15. Marcolino, L.S., dos Passos, Y.T., de Souza, Á.A.F., dos Santos Rodrigues, A., Chaimowicz, L.: Avoiding target congestion on the navigation of robotic swarms. Autonomous Robots pp. 1–24 (2016), http://dx.doi.org/10.1007/s10514-016-9577-x
16. Margulis, E.H.: On repeat : how music plays the mind. New York, NY: Oxford University Press (2014)
17. McAnally, J.K.: Edwin Franko Goldman, Richard Franko Goldman, and the Goldman Band: Professionals and Educators. The Bulletin of Historical Research in Music Education 17(1), 19–58 (1995)
18. Moreira, J., Roy, P., Pachet, F.: Virtualband: Interacting with styslistically consistent agents. In: Proceedings of the 14th International Society for Music Information Retrieval Conference (2013)
19. Pachet, F., Roy, P., Moreira, J., d'Inverno, M.: Reflexive loopers for solo musical improvisation. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2013)
20. Policastro, M.A.: Understanding How to Build Guitar Chords and Arpeggios. Mel Bay (1999)
21. Robertson, G., Watson, I.: A review of real-time strategy game AI. AI Magazine 35(4) (2014)
22. Scott, B.: The illusion of intelligence. In: Rabin, S. (ed.) AI Game Programming Wisdom, vol. 1, pp. 16–20. Charles River Media (2002)
23. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers C-29(12) (December 1980)
24. Sorenson, N., Pasquier, P., DiPaola, S.: A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games 3(3) (2011)

25. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
26. Walser, R.: Running with the devil: Power, gender, and madness in heavy metal music. Middletown, Connecticut: Wesleyan University Press, p. 9 (1993)
27. Zook, A.E., Riedl, M.O.: A temporal data-driven player model for dynamic difficulty adjustment. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference. AIIDE (2011)