Robinflock: a Polyphonic Algorithmic Composer for Interactive Scenarios with Children

Raul Masu
interaction Lab,
University of Trento
raul.masu@unitn.it

Andrea Conci
interaction Lab,
University of Trento
Andrea.Conci.1@ unitn.it

Cristina Core
interaction Lab,
University of Trento
cristina.core@ unitn.it

Antonella de Angeli
interaction Lab,
University of Trento
antonella.deangeli@unitn.it

Fabio Morreale
Centre for Digital Music, EECS
Queen Mary University of London
f.morreale@ qmul.ac.uk

ABSTRACT

The purpose of this paper is to present Robinflock, an algorithmic system for automatic polyphonic music generation to be used with interactive systems targeted at children. The system allows real time interaction with the generated music. In particular, a number of parameters can be independently manipulated for each melody line. The design of the algorithm was informed by the specific needs of the targeted scenario. We discuss how the specific needs of the polyphony with children influenced the development choices. Robinflock was tested in a field study in a local kindergarten involving 27 children over a period of seven months.

1. INTRODUCTION

Polyphonic music is widespread in western music tradition. It consists of at least two melodic lines that are simultaneously performed and that produce coherent harmonies when overlap. Polyphonic music originated with vocal music, grown during the Renaissance [22], and has been fundamental for classical composers (e.g. Bach, Haydn, and Mozart), for modern composers (e.g. Schoenberg and Stockhausen), and for music studies [31].

Performing polyphonic music is particularly demanding as it requires musicians to master score reading and the ability to play with other performers. The difficulties of performing polyphonic music are even more demanding for young children due to their limited musical experience. Despite these difficulties, children seem to have an innate predilection for polyphony [33].

We argue that an algorithmic solution can increase children confidence with polyphonic music. In particular, an algorithmic composition system that generates polyphonic music in real time can be applied in interactive situations to familiarise children with polyphony. We propose to delegate part of the complexities of performing polyphonic music to the computer, enabling children to directly manipulate different lines by interacting with a number of parameters.

The contribution of this paper is the proposition of a real time algorithmic composer of polyphonic music. Indeed, to the best of our knowledge, related studies in algorithmically generated polyphonic music never address interactive purposes. As a consequence, the development of an algorithm that generates polyphonic music in real time was not strictly necessary. Previous algorithmic systems for polyphonic music generation exist; however, they have been developed from a computational linguistic perspective, aiming at generating polyphonic music that avoid composition and counterpoint errors. The objective of these studies was to imitate specific styles [8] or to apply compositional rules derived from music theory [2, 10, 32]. These algorithms successfully generated good quality counterpoint compositions, but their application domain was limited to off-line non-interactive music.

The novelty of the Robinflock lies in the possibility to independently manipulate each musical line on several musical parameters in real time. Robinflock was developed in the context of Child Orchestra [9], a research project aimed at exploring new solutions to use technology in context of music making for children. In this paper, we describe the technical implementation of the algorithm underling how the technical choices are grounded on specific requirements: design solutions were taken to meet a number of requirements related to the specific interactive scenario of polyphonic music for children. The architecture is based on Robin, an algorithm developed by some of the author that generates piano music for interactive purposes [24, 26].

Robinflock was adopted in the Child Orchestra field study, which involved 27 kindergarten children over a period of seven months [9]. The result of this field study suggested that the algorithm is particularly effective to help children to familiarise with polyphonic music.

The remainder of this paper is organized as follows. Section 2 presents related work in algorithmic composition and in music for children. Section 3 describes in details the architecture of Robinflock. Section 4 introduces the field work in which the algorithm was used and presents the results. We conclude this paper with discussions and considerations on future work.

2. RELATED WORK

The work presented in this paper is mainly based on related literature in algorithmic composition. We also present the main child-related musical theories on which we grounded the choices at the basis of the interactive system.

2.1 Algorithmic composition

Algorithmic composition has been widely explored for both research and artistic purposes since the beginning of computer music [17] (see [11, 28] for comprehensive surveys). Given the scope of this paper, we here mainly focus on real time algorithmic systems and systems that generate polyphonic music.

2.1.1 Algorithmically generated polyphonic music

Researchers in algorithmic composition explored polyphony and counterpoint, i.e. the set of rules to compose polyphonic music, mainly with a generative linguistic perspective, mostly with the objective of exploring the potential of computers to imitate specific styles.

David Cope's pioneering work in *Experiment with Music Intelligent* (EMI) approached counterpoint to compute different voices in a specific music style, for instance in a Bach-style Chorale. EMI works by analysing, deconstructing, and then recombining elements of existing music. EMI obtained great results in emulating the style of different composers [8].

Other studies explored counterpoint species (see [19] for a definition of species.) For instance, Aguilera developed an algorithm that generates first specie counterpoint using probabilistic logic [2]. Other systems explored the possibilities to compute more complex polyphonic structures adopting fifth specie counterpoint. For instance, Polito, Daida, and Bersano-Begey [32] developed a genetic algorithm that can compose fifth specie counterpoint based on an existing cantus firmus. Their system adopted a fitness function based on rules proposed by Fux for the counterpoint species. Similarly, Herremans and Soorensen [12] developed Optimuse, a neighbourhood search that generates fifth species counterpoint fragments. Another approach was proposed by Martín-Caro [23] who developed a system to compute two-voices fifth species counterpoint using a first-order Markov chain.

Other systems were developed specifically to algorithmically compose four-part counterpoint music. Among these systems, Phon-Amnuaisu developed a genetic algorithm [30] based on a subset of the four-part rules as a fitness function and Donnelly [10] presented an algorithm based on melodic, harmonic, and rhythmic rules.

All the presented systems achieved a good quality imitation of renaissance and baroque counterpoint. But, as interactivity was behind the scope of these researches, none of them allowed a manipulation of the different line in real time.

2.1.2 Real time interactive algorithmic systems

Real time algorithms have been largely explored for performative purposes, especially to create dialogues with human musicians. Most of these systems adopted a multi-agent architecture. For instance, Rowe's *Cypher* [34] is a real time algorithmic music system designed to improvise with human performers. The system is composed by two main agents, a listener and a player and coordinates a number of small modules such as chord analyser, beat tracking, phrase grouping, and other compositional modules that generate the new musical output. A similar approach was adopted in Lewis' *Voyager*. Indeed, in Lewis' system, the control on the music is shared among 16 "players" [21]. Another real time algorithmic system based on a multi-agent architecture is the *Free Improvisation Simulation* by Nicholas Collins [7].

The above described systems have been mainly developed with performative purposes. Interactive algorithmic composition systems have also been developed with different objectives. The Roboser, for instance, is an algorithmic composition system developed to use real-world behaving as input source to an algorithmic composition system [40]. Another example is Music Blox, which explores the use of generative algorithm in the context of interactivity [12].

With respect to the role of a specific style in interactive algorithms, Aspromallis and Gold recently presented a study on a method for interactive real time music generation that preserves traditional musical genres [3].

All these systems generate a quite large variety of musical genre for interactivity, but the adoption of polyphonic music in this contest is currently overlooked.

2.2 Music and children

In this section, we introduce relevant aspect of music pedagogy and some musical interactive systems for children.

2.2.1 Music theories for children

Since the beginning of the last century, a number of theories have been proposed to discuss musical activities for children [12, 14, 18, 37].

Music Learning Theory by Gordon, one of the most relevant theories for young children, underlines the importance of learning music by ear [14]. Learning by ear implies the development of the listening abilities, which are fundamental also for spatio-temporal reasoning [16]. Listening abilities can be enhanced by physical interaction and active participation of the children, as active participation engages children's attention and memory. Physical interaction and active participation increase children's attention as they are required to respond to the music by changing their body movements [36]. In addition, body movements could help to memorise rhythms [18]. The awareness of rhythmic patterns in music is promoted as it helps the development the control of body language and perceptual skills [14, 18]. A general consensus also suggests that to maximize musical experience that target children should be both complex and musically coherent [14].

2.2.2 Interactive music systems for children

In this paragraph, we introduce musical tools and interactive systems aimed at providing musical experiences to a young population. Some systems were developed to allow children to manipulate musical tracks through their movements. An example of body interaction for children aged between 7 and 10 years is offered by Antle and colleagues

[4]. Their system, which was designed with the help of four music experts, proposed a metaphor based on body movements. The movement of the children were captured by a tracking system and mapped into a percussive sound output, which varied according to the children's movements. Specifically, *tempo* was mapped to the speed of the children, *volume* to the intensity of the activity, and *pitch* to the proximity among children. The system did not control harmonic and melodic features, and consequently did not require a complex algorithmic system to deal with such elements. Another example is the *Continuator*, that is particularly relevant for the scope of this paper as it involves algorithmic elements [29, 1]. The *Continuator*, takes as input the music performed on the keyboard and plays back coherent excerpts that mirror performer's play. The system facilitated piano improvisation by automatically generating musical replies. The system resulted to be effective in promoting flow state for children [1], but does not have any aim in acculturating children in any particular musical technicality.

3. SYSTEM ARCHITECTURE

The algorithmic composer was developed following a number of musical needs that emerged during a preliminary design phase. The design phase, whose details are discussed in a related publication [34], involved experts in the domain of music theory, music teaching, and pedagogy, who participated in four focus groups and one workshop. We present here the main musical needs that emerged is the design phase.

The generated music should be varied, coherent, and compliant with compositional rules, thus aiming to provide children with a valid stimulus for learning by ear [14].
Moreover, the system is required to allow multiple children to interact with it at the same time. Thus, each child needs to have control over a specific voice of the music. In particular, low-level parameters should be directly accessible to actively stimulate the children and foster participation. The parameters individuated during the design phase were volume, speed, and articulation.
These interactive and musical requirements can be met by using an algorithmic solution insofar as it allows to generate and manipulate the different lines in real time. This would provide each child with the control over his own melodic line, thus allowing children to modify the music without reciprocal interference. Furthermore, as opposed to a solution based on triggering pre-recorded melodies, algorithmic composition guarantees continuous access and control to structural parameters. Finally, an algorithmic solution can guarantee a virtually infinite number of compositions but, at the same time, restrain the musical outputs to correct melodic and harmonic rules.
As highlighted in the related work section, most of the systems that algorithmically generate polyphonic music are not designed for interactive purposes [2, 10, 12, 23, 30, 32]. We therefore decided to base the architecture of the algorithm on Robin, an algorithm previously developed by some of the authors [24, 26], which follows a rule-based approach to manipulate a number of musical parameters in real-time (i.e. speed, articulation, melody direction, timbre, volume, consonance, and mode). Robin

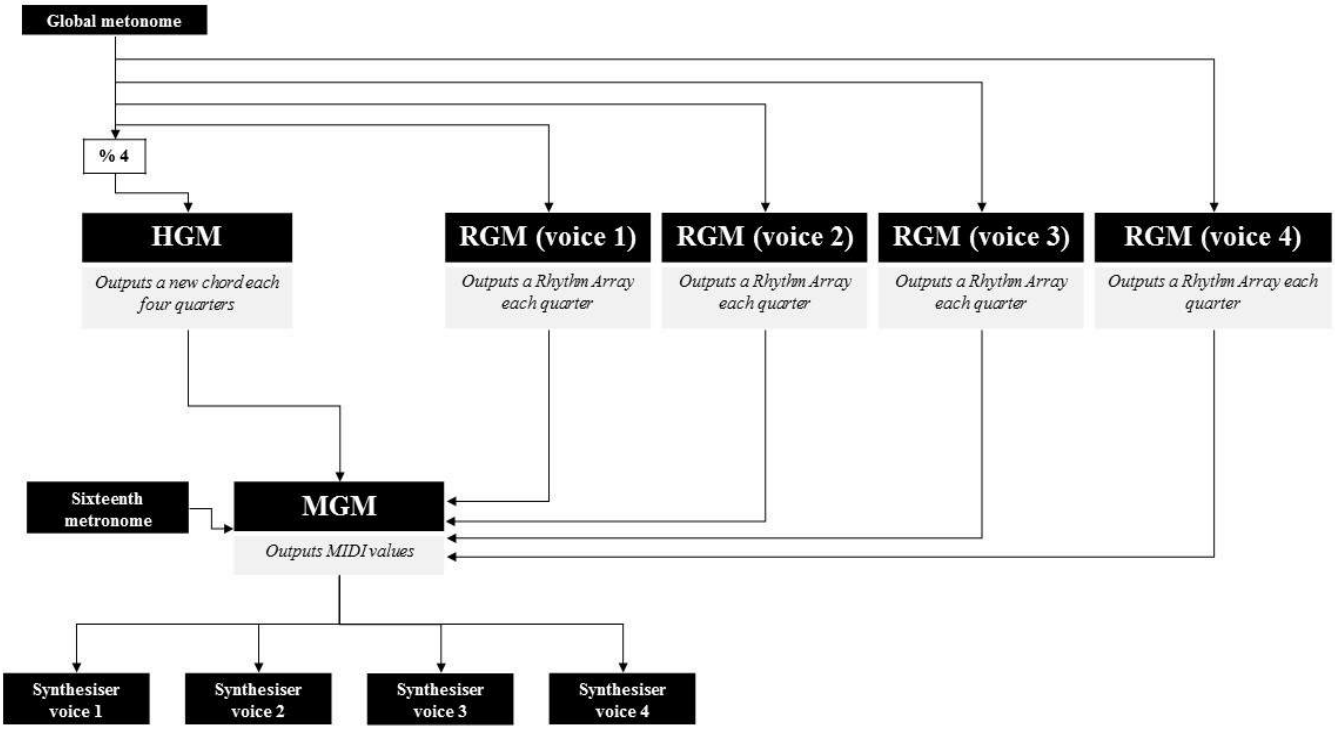

Figure 1. Scheme of the architecture of the algorithm.

controlled a single voice with accompaniment generated by using some piano clichés of classical music, and was adopted in several interactive contexts [25, 27]. However, as each child needs to control a different voice, the algorithm had to be extended to support multiple voices. To this end, we implemented an algorithm that allows independent manipulation on each line of polyphonic music. Robinflock was developed with a rule based approach.

Similarly to most of the interactive systems previously described [7, 21, 34], the structure of Robinflock is modular. The algorithm is composed of three modules that separately deal with: i) *harmony, rhythm, and melody*. The Harmony Generation Module (HGM) follows traditional harmonic rules [31, 35] implemented with a first order Markov chain. The Rhythm Generation Module (RGM) is composed of four modules that compute the rhythm of each line. The rhythm is computed each quarter while the harmony is computed each four quarter. A global metronome synchronises the two modules. The Melodies Generation Module (MGM) fills all the four rhythms with notes that are coherent to the current chord to avoid counterpoint errors like parallel fifths and octaves [17]. The generated music is performed with four FM synthesisers, one for each line. The overall architecture of Robinflock can be seen in Figure 1.

Robinflock is developed in Max-MSP, and each module is implemented with JavaScript, using the Max-JS object, with the only exception of the rhythm module, which is composed of four JS objects, one for each line.
To ensure time synchronisation of the four voices, a metronome is set to the minimum metric unit (the sixteenth) and each voice is encoded using arrays of sixties. All the four arrays are generated by the MGM, which combines the four rhythm arrays generated by the RGMs with the current harmony. A metronome scans the four arrays at the same time: each cell of the array (sixteenth note) contains a MIDI pitch value in case a new note has to be performed or a -1 flag if a new note is not required. In the latter case, the previous note continues to play. These MIDI values are sent to the four synthesisers, that are also implemented in Max-MSP.

3.1 Harmony Generation Module

Traditionally in tonal/modal music, harmony is organized on the basis of chord progressions and cadences. One common approach to manage harmony is that of Generative Grammar, which can compute good quality chord progressions but it is based on time-span reduction of the musical structure, and requires a knowledge *a priori* on the overall structure [38]. Following the design requirements, we needed a high level of flexibility in real time. Therefore, we computed harmony as a stream of chords that are not a priori generated. Each chord is defined as a scale degree and computed as a state of a Markov chain. Since the historical works of Xenakis, Markov chains have been adopted in algorithmic composition [39] to manage temporal successions of musical events. Concerning harmony, the transition probabilities between successive chords have already been modelled as a Markov process [33]. Chords transition data can be extracted by the analysis of existing music [8], surveying of music theory [33], or following personal composing aesthetic [39].
To guarantee a high degree of music adaptability, chord and degrees' correlation does not depend on previous states of the system: a first-order Markov process determines the harmonic progression as a continuous stream of chords. We implemented the matrix following a rule-based approach deriving the transition probabilities from harmony manuals (Table 1) [31, 35].
As the music was required to offer a variety of stimuli, Robinflock was developed to compute music in all the major and minor keys, and in all the Gregorian modalities.
To cope with this requirement, the system computes the harmony progression applying the scale degree to the desired tonality or modality. For example, a first degree in C major key corresponds to the C-E-G chord while the same degree in G minor key corresponds to the G-Bb-D chord.

	I	**II**	**III**	**IV**	**V**	**VI**	**VII**
I	0	0,05	0,07	0,35	0,35	0,08	0,1
II	0,05	0	0,05	0,15	0,65	0,2	0
III	0	0,07	0	0,2	0,8	0,65	0
IV	0,15	0,15	0,05	0	0,6	0,05	0
V	0,64	0,05	0,05	0,13	0	0,13	0
VI	0,06	0,35	0,12	0,12	0,35	0	0
VII	1	0	0	0	0	0	0

Table 1. The Markov matrix in the Harmony Module

The harmonic rhythm of Robinflock is similar to that of Robin: each chord corresponds to a bar, and every eight chords/bars the system is forced to cadence to a tonic, dominant or subdominant (I, IV, V). The introduction of cadences allows to coherent music harmony, while maintaining real time flexibility.

3.2 Rhythm Generation Module

The Rhythm Generation Module (RGM) manages the rhythm of each musical line autonomously, thereby it is composed of four different instances of the same agent.
During the design process, speed was identified as one of the musical parameter that should be manipulated autonomously on each voice. To keep coherence among the four voices, we decided to express speed as note density. Changing the value of the BPM on each voice would indeed have resulted in unwanted results. The rhythm density ranges from a continuum of sixteenths notes up to a four-quarters note, with a simple metric division (no tuples are adopted).
To reduce the latency, thus guaranteeing a real time response, the rhythm is computed at each quarter (four times each bar). Nevertheless, for low note density situations, it could be necessary to have two, three, and four quarters notes. Consequently, the length of the note has to be modified while the note is playing. The length of the note is thereby not computed at the beginning of the notes but defined at the end, allowing continuous access to the note density parameter.

To this end, the rhythm was encoded using note ON-OFF messages. Given that each line is rhythmically computed autonomously we use the note ON message also to terminate the previous note. The ON-OFF messages are stored in an array - the Rhythm Array - of four Boolean values. Each position of the array corresponds to a sixteenth notes: 1 means new note; and 0 means that the previous note continues to play. For example, the array [1,1,1,1] corresponds to four sixteenths note and the array [1,0,0,0] corresponds to one quarter note (Figure 2 and Figure 3).

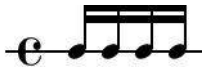

Figure 2. Rhythmic pattern corresponding to the array [1,1,1,1].

Figure 3. Rhythmic pattern corresponding to the array [1,0,0,0].

The density can range between an entire 4/4 bar of sixteenths, and a minimum of four quarter notes. The corresponding array for one entire bar are [1,1,1,1] [1,1,1,1] [1,1,1,1] [1,1,1,1] for the sixteenths (maximum density), and [1,0,0,0] [0,0,0,0] [0,0,0,0] [0,0,0,0] for the four quarter note (minimum density).
The continuum that ranges between the two extremes is discretised in sets of possible rhythms. Each set refers to a basic note length: two quarter note, one quarter note, eighth note and sixteens note. The sets contain regular and irregular patterns, for instance the sixteenths note, contains [1,0,1,1] but also [1,1,0,1] and quarter notes set contains [1,0,0,0] but also [0,0,1,0]. According to the selected density a pattern is chosen from the corresponding set.
To avoid illegal syncope, the RGM saves information on the previous quarter and which quarter of the bar is currently being computed. For instance, if the density value suddenly jumps from maximum to minimum the system would compute the succession [1,1,1,1] [0,0,0,0] that produces illegal rhythm. In this case the system bypasses the rhythm density and computes: [1,1,1,1] [1,0,0,0].

3.3 Melodies generation module (MGM)

At each quarter, the Melody Generation Module (MGM) receives the four Rhythm Arrays and the current chord from RGMs and HGM, and computes the four melodies arrays, one for each voice. The MGM outputs four streams of MIDI notes.

The computation of the notes of the arrays occurs in two steps.

1. The downbeat notes of the four voices (first position in the position in the Rhythm Array) are filled with notes of the chord, as in first species counterpoint. The first voice to be computed is the bass: it can be the fundamental or the third note of the chord. This note cannot be the fifth of the chord to avoid the second disposition of the chord, which was not used in polyphonic music [19]. Then, the other voices are computed in the following order: tenor, alto, and soprano. A chord note is assigned to each voice. If the current Rhythm Array corresponds to the first quarter of the measure, and consequently to a new chord, the algorithm checks that it does not produce a parallel octave or a fifth interval with the previous chord in a two-steps recursive procedure. In the first step, the system selects a note of the chord; in the second step, the system checked that this note does not produce octave or fifth with the previous quarter. If the selected note produces illegal parallel octave or fifth, the algorithm goes back to step one, choosing a different note. For the second, third, and fourth quarter of the measure, the chord is the same, and the melodies reposition chord notes. As a consequence, it is not necessary to pay attention to the illegal octaves [31]. Having the harmony stable for four quarters was not common in historic counterpoint. However, we adopted this solution to guarantee the flexibility of the note length while keeping the harmony coherent.
2. The other notes of the four voices are computed. The algorithm enters the second step only if the density of the voice is higher than a quarter. Two different main cases are identified: i) *syncope*: as the chord is the same, the algorithm computes the note as a consonant arpeggio; ii) *non-syncope*: the second eight is computed with a chord note, and the sixteenth upbeat notes are computed as passage notes.

Given the real-time constraints, no prediction on successive notes can be made. Thus, in the current version of the system, some elegant ornamentations of the fifth species counterpoint (e.g. dissonant suspensions, neighbour note, and double passing tones) are not implemented.

3.4 Synthesiser

The synthesiser is realized using simple FM synthesis with a different carrier modular ratio for the nearby voices, (bass FM ratio 2, tenor FM ratio 3, alto FM ratio 2, soprano FM ratio 3). The choice of using the FM and avoiding samples were suggested by the expert in music pedagogy during the design process. Using a real instrument timber could indeed cause the child to be positively or negatively emotionally influenced by a specific instrument, and this could decrease his attention over the specific exercise (e.g. the sound of a piano for the daughter of a pianist).
The synthesiser controls both the volume and the articulation. The articulation is manipulated by changing the ADSR (attack, decay, sustain, release) parameters, in the range that continues between *02 02 0 0* for the staccato and *200 100 1 100* for the legato. These values follow the standard ADSR (attack, decay, sustain, release) codification of the Max-MSP object. Attack, decay, and release are expressed in milliseconds. The sustain is a multiplicative factor used to calculate the volume during the sustain of a note, its value is normalized between 0 and 1.

4. ROBINFLOCK IN THE CONTEXT OF CHILD ORCHESTRA

This section shows a field application of Robinflock. We present the practical adoption of Robinflock in a kindergarten and report the main observations collected in this context. Full details on the field study that involved Robinflock can be found in a dedicated paper [9].

4.1 Activities

Robinflock was used over a period of seven months in a local kindergarten. A total of 27 children (14 females) aged 3 and 4 took part in the study. They were divided into two groups: 15 children of 3 years old (4 males) and 12 children of 4 years old (9 males). Both groups followed the same cycle of 12 lessons, which were facilitated by a music teacher with a specific expertise in music pedagogy for pre-schoolers. One kindergarten teacher was also present at each lesson to supervise the activities.
For this field study two different use of the system have been adopted. In the first half of the lessons, Robinflock was controlled by the music teacher via a mobile application. In the second half of the lessons, the system was directly operated by the children by means of their movements, which were tracked via motion tracking (details on the interfaces below). The observations were conducted by three of the authors.

4.2 Interfaces

Two different interfaces were developed to control Robinflock: a web app, which allows the teacher to remotely manipulate the music using a smartphone, and a motion tracker, which allows the children to directly manipulate the music.

4.2.1 Web app

The web app was used by the teacher in the first phase of the field study to help children familiarise with the music played by Robinflock and to propose a number of exercises that were specifically centred on polyphony.
The implementation of the app was based on web application technologies (node.js and javascript), which allow compatibility with smartphones and tablets. The capability of interacting wirelessly through a smartphone allowed the teacher to freely move in the room and to offer support to children. The application was implemented using sliders and buttons typical of mobile UI.

4.2.2 Motion tracking

Following the suggestions of the related studies that stressed the importance of movement to stimulate music learning [14, 18, 36], we implemented an interface that involved body movements to control the music. We adopted a solution based on accelerometers and gyroscope, which were embedded on a small (3x4 cm) and lightweight Bluetooth board (Texas instrument CC2650) powered by a small coin battery. The sensors were hidden inside a number of colourful cardboard objects aimed at attracting children's attention such as a crown and a magical wand.

The mapping between the movement of the children and the music was discussed with the experts and adapted according to the observation of the system. Initially, we planned to map the speed of melodies with the speed of the children, and the articulation with the distance of children's barycentre to the floor. During the field work, we realised that the manipulation of more than one parameter would have been too demanding for the children. As a consequence, we maintained only speed.

4.3 Observation

The experience in the kindergarten was evaluated integrating observations, interviews, and more than 300 drawings sketched by children at the end of each session. We summarise here the main considerations. Full detailed on the methodology adopted for the analysis can be found in [9].
The first important observation is that children identified and reacted properly to the three parameters. We observed how children reacted to changes of musical parameters using specific exercises designed along with the music teacher. We asked children to change their behaviour, for instance jump outside or inside a carpet, following the changes of the music. Volume, was the most difficult parameter for some of the children, as they struggled performing the exercise. By contrast, differences in articulation and speed were immediately perceived.
Robinflock was also particularly successful in helping children to familiarise with polyphonic music. This fact was assessed by observing how children reacted to the changes of specific musical lines. For the polyphony we also asked children to change their behaviour following the changes of the music, in this case different child were asked to follow the changes of different voices.
Combining the observation of children behaviour with the discussion with the music teacher allowed us to understand children that children had difficulties to manipulate more than one parameter. We also observed that a four lines polyphony was too complex to be manipulated by children. For this reason, we reduced the polyphony to two voices.
Despite these limitations, the observations revealed that Robinflock succeeded in encouraging children to actively participate in music making. Comments collected from the children confirmed this finding, as they commented on the experience stating that "I did the music" and "we played the music".
To summarize, Robinflock proved helpful to familiarise children with polyphonic musical structures and peculiarities.

5. DISCUSSION

In this paper. we presented an algorithmic system that generates polyphonic music in real time developed for interactive scenarios with children. With respect to other attempts to algorithmically compose polyphonic music, we introduced the possibility to manipulate in real time the generated music. Due to the real-time necessity, we needed to come to terms finding a trade-off between accuracy of the counterpoint and the responsiveness of the system. With respect to other systems, the accuracy is indeed re-

duced because of the evolution of music cannot be anticipated. Despite of this, this interactive system offered several benefits.

We propose that algorithmic composers can be adopted to help children to familiarise with polyphonic music: delegating part of the complexity of composing and performing polyphonic music to a computer allowed children to experience the control of this complex musical structures. In particular, Robinflock proved useful in helping children to recognise different lines in polyphonic music.

Reflecting on our experience with Robinflock, we propose that the development of algorithmic system for interactive scenarios should be based on the specific needs of the specific scenario, considering for instance, the target users, the environment. These needs were necessary to define the characteristic of the music and the parameters to be manipulated. For instance, Robinflock was influenced by the necessity of having a variety of music stimuli and by the necessity of manipulating three musical parameters (volume, speed, and articulation) independently for each voice.

The necessity of offering a variety of music stimuli is reflected in the HGM, which computes music in all the major and minor keys and in all the Gregorian to widen the music variety.

With respect to the three parameters, we shown that volume and articulation can be managed by a synthesiser, whereas speed has to be manipulated by an algorithmic composition system.

The necessity of manipulating the speed in real time influenced the decision to express speed as the density of the notes in the Rhythm Array. This choice allows different voices to have different behaviours. For instance, bass and soprano voices can be very slow, performing four quarter length notes, while the alto and tenor are performing sixteens notes. The manipulation of speed is probably the most significant contribution of this algorithmic system.

The interactive necessity introduces the need to determine the length of the note while the note is performing as opposed to computing it at the beginning of the note. This specific requirement leaded us to develop the ON-OFF Boolean note encoding that influenced all the RGM structure, and, to the best of our knowledge is a novelty in algorithmic composition of polyphonic music.

6. FUTURE WORKS

A number of solutions described in this paper are grounded on the specific needs of our target users. Future implementations of the system could aim at targeting different ages, for instance an adult population. This study would test the system in different interactive contexts and study how to adapt musical features and the interactive parameters to fulfil different needs.

From a musical point of view, the main limitation of the current implementation of Robinflock is the absence of elegant ornamentations of the fifth species counterpoint. Future version of the algorithm could include more complete counterpoint rules and include polished ornamentations such as dissonant suspensions, neighbour note, and double passing tones.

Acknowledgments

This paper was supported by Inf@nzia DIGI.Tales 3.6, an Italian Ministry of Education project aiming to renewing and updating preschool (3-6 years old) education programmes with ICT. We are grateful to the artworks designer Adriano Siesser, and the music teacher MariaPia Molinari for her guidance throughout the study as well as to the teachers and the children of the Kindergarden in the school of Cavedine. This research was also partially supported by EPSRC under grant EP/N005112/1.

7. REFERENCES

[1] A. R. Addessi, and F. Pachet, "Experiments with a musical machine: musical style replication in 3 to 5 year old children." in J. British Journal of Music Education 22.01, 2005, pp. 21-46.

[2] G. Aguilera, J. L. Galán, R. Madrid, A. M. Martínez, Y. Padilla, and P. Rodríguez, "Automated generation of contrapuntal musical compositions using probabilistic logic in derive." in J. Mathematics and Computers in Simulation 80.6, 2010, pp. 1200-1211.

[3] C. Aspromallis, and N. E. Gold, "Form-Aware, Real-Time Adaptive Music Generation for Interactive Experiences Data." in Proc. Int. Conf. Sound and music computing (SMC2016), Hamburg, 2016, pp 33-40.

[4] A. N. Antle, M. Droumeva, and G. Corness, "Playing with the sound maker: do embodied metaphors help children learn?." in Proc. Int. Conf. Interaction design and children, ACM, 2008, pp. 178-185.

[5] J. D. Bolter, and D. Gromala, "Transparency and Reflectivity: Digital Art and the Aesthetics of Interface Design." in J. Aesthetic computing, 2006, pp 369.

[6] W. Chai, B. Vercoe, “Folk music Classification using Hidden Markov Models” in Proc. Int. Conf. Proc. of ICAI 01, 2011.

[7] N. M. Collins, Towards autonomous agents for live computer music: Realtime machine listening and interactive music systems. Diss. University of Cambridge, 2006.

[8] D. Cope, Virtual music: computer synthesis of musical style. MIT press, 2004.

[9] C. Core, A. Conci, A. De Angeli, R. Masu, F. Morreale, "Designing a Musical Playground in the Kindergarten" to appear in Proc. Int. Conf. British HCI, 2017.

[10] P. Donnelly, and J. Sheppard, "Evolving four-part harmony using genetic algorithms." in Proc. Int. Conf. European Conference on the Applications of Evolutionary Computation, Springer Berlin Heidelberg, 2011, pp. 273-282.

[11] J. D. Fernández, and F. Vico, "AI methods in algorithmic composition: A comprehensive survey." in J. Journal of Artificial Intelligence Research 48, 2013, pp 513-582.

[12] Gartland-Jones, Andrew, "MusicBlox: A real-time algorithmic composition system incorporating a distributed interactive genetic algorithm." In Workshops on Applications of Evolutionary Computation, Springer Berlin Heidelberg, 2003, pp. 490-501.

[13] J. Glover, and S. Young, Music in the early years. Routledge, 2002.
[14] E. Gordon, A music learning theory for newborn and young children. Gia Publications, 2003.
[15] D. Herremans, and K. Sörensen, "Composing first species counterpoint with a variable neighbourhood search algorithm." in J. Journal of Mathematics and the Arts 6.4, 2012, pp. 169-189.
[16] L. Hetland, "Listening to music enhances spatial-temporal reasoning: Evidence for the" Mozart Effect"." in J. Journal of Aesthetic Education 34.3/4, 2000, pp. 105-148.
[17] Jr. Hiller, A.Lejaren, and L. M. Isaacson, "Musical composition with a high speed digital computer." Audio Engineering Society Convention 9, Audio Engineering Society, 1957.
[18] E. Jaques-Dalcroze, Rhythm, music and education. Read Books Ltd, 2014.
[19] K. Jeppesen, Counterpoint: the polyphonic vocal style of the sixteenth century. Courier Corporation, 2013.
[20] S. Kalénine, and F. Bonthoux, "Object manipulability affects children's and adults' conceptual processing." in J. Psychonomic bulletin & review 15.3, 2008, pp. 667-672.
[21] G. E. Lewis, "Interacting with latter-day musical automata." in J. Contemporary Music Review 18.3, 1999, pp. 99-112.
[22] A. Mann, The study of fugue. Courier Corporation, 1987.
[23] V. P. Martín-Caro, and D. Conklin, "Statistical Generation of two-voices florid counterpoint" in Proc. Int. Conf. Sound and music computing 2016, pp. 380-387.
[24] F. Morreale, A. De Angeli. Collaborating with an Autonomous Agent to Generate Affective Music in J. Computers in Entertainment, ACM, 2016.
[25] F. Morreale, A. De Angeli, R. Masu, P. Rota, and N. Conci, "Collaborative creativity: The music room." In J. Personal and Ubiquitous Computing, 18(5), Springer, 2014, pp.1187-1199.
[26] F. Morreale, R. Masu, and A. De Angeli. "Robin: an algorithmic composer for interactive scenarios Proc. Int. Conf. Sound and music computing (SMC2013) Stockholm, 2013 pp. 207-212.
[27] F. Morreale, R. Masu, A. De Angeli, and P. Rota, "The music room" in Proc. Int. Conf. CHI'13 Extended Abstracts on Human Factors in Computing Systems. pp. 3099-3102, ACM.
[28] G. Nierhaus, Algorithmic Composition: Paradigms of Automated Music Generation. Springer, 2009.
[29] F. Pachet, "The continuator: Musical interaction with style." in J. Journal of New Music Research 32.3, 2003, pp. 333-341.
[30] S. Phon-Amnuaisuk, and G. Wiggins, "The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system." in Proc. Int. Conf. AISB'99 Symposium on Musical Creativity, London: AISB, 1999, pp. 28-34.
[31] W. Piston, Harmony. Norton, 1948.
[32] J. J. Polito, M. Daida, and T. F. Bersano-Begey, "Musica ex machina: Composing 16th-century counterpoint with genetic programming and symbiosis." in Proc. Int. Conf. International Conference on Evolutionary Programming, Springer Berlin Heidelberg, 1997, pp. 113-123.
[33] D. Pond, "A composer's study of young children's innate musicality." in J. Bulletin of the Council for Research in Music Education,1981, pp. 1-12.
[34] R. Rowe, "Machine listening and composing with cypher." in J. Computer Music Journal 16.1, 1992, pp. 43-63.
[35] A. Schoenberg, Theory of harmony, Univ. of California Press, 1978.
[36] W. L. Sims, "The effect of high versus low teacher affect and passive versus active student activity during music listening on preschool children's attention, piece preference, time spent listening, and piece recognition." in J. Journal of Research in Music Education 34.3,1986, pp. 173-191.
[37] J. M. Thresher, "The contributions of Carl Orff to elementary music education." in J. Music Educators Journal 50.3, 1964, pp. 43-48.
[38] R. Jackendoff, A generative theory of tonal music. MIT press, 1985.
[39] I. Xenakis, Formalized music: thought and mathematics in composition, Pendragon Press, 1992.
[40] K. Wassermann, M. Blanchard, U. Bernardet, J. Manzolli, and P. F. Verschure. "Roboser-An Autonomous Interactive Musical Composition System." Proc. Int. Conf. ICMC, Berlin, 2000.