Agent-Based Guitar Performance Simulation

Leandro L. Costalonga, Rosa M. Vicari and Evandro M. Miletto

Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
Phone: +55 (51) 3316-6168
PO.Box 15.064 – 91.501-970
Porto Alegre – RS - Brazil.
{llcostalonga, rosa, miletto}@inf.ufrgs.br

Received 14 January 2008; accepted 21 July 2008

Abstract

The goal of this paper is to describe a system-aided performance and composition tool that aims to expand guitarist capacities by providing innovative ways in which the user can interact with the system. In order to achieve that, we decided to use an agent-based approach, independently modeling the active elements involved in a guitar performance as autonomous agents - named Left-Hand, Right-Hand, and Speaker (the guitar itself). These agents are able to communicate to each other in order to make some musical decisions, specially related to the chord's shape choice. The musical elements (harmony and rhythm) are independently defined respectively by the Left-Hand and Right-Hand agents. The most relevant aspects of this work, however, are the algorithms and strategies to process both harmonic and rhythmic data. Finally, we perform an evaluation of the system and discuss the results of the implemented techniques.

Keywords: music performance, multiagent, guitar computational model

1. INTRODUCTION

One of the difficulties in musical system simulation is modeling both the human performer and the musical instrument itself [5], [6]. Due to this fact, the simulation (as natural as possible) of the rhythm sounds of folk guitar performances, solo or several of them mixed in a synthesizer, is our challenge.

This paper will focus on the modeling of these elements, the performer and the guitar, as artificial agents. The result of the interaction of such agents is a music performance itself. We consider the music e/or the guitar performance by the system as the execution of an entry, i.e., a score given in some symbolic notation equivalent to common music notation, played by the system

Over the last few years, agent-based simulation has been used to solve problems in different fields, including music [8],[9],[11],[14],[15]. One of the advantages of such approach is the possibility of simulating situations beyond the musician's natural physical restrictions[1]; e.g. multiple limbers with high-accuracy and speed.

In our case it allowed simulate situations as for example, creating more than two virtual hands for the system musical performance, and also the new interesting sound possibilities that could be done only by the computing systems. Thinking this way, it was possible overlap rhythm patterns to create new ones, as well as include melodies, harmonies in different ways and numbers. It is like a unique guitar performed by many players.

Considering our case, the use of this approach allows us to simulate situations like the use of more than two virtual hands for the system musical performance. Also, it allows the exploration of some new interesting sound possibilities that can be done only by computing systems. By thinking this way, it is possible to overlap rhythm patterns to create new ones, as well as include melodies, harmonies in different ways. It is like a unique guitar performed by many players.

To achieve our goal, simplifications of the cognitive and biomechanical processes involved in a folk guitar performance were made. Any decision taken by the guitarist involving the articulation of his left-hand was programmed into the Left-Hand (LH) agent. The same principle was applied to the right-hand and the guitar itself, represented by the Right-Hand(RH) agent, and the Speaker agent respectively.

This paper is organized as follows: the next section presents a brief introduction to general characteristics of the guitar and the guitarist. Section 3 describes the agent-approach to simulate guitar performances, followed by the agent's descriptions in the sections 4, 5 and 6. Sections 7 and 8 are related to technical information regarding the system implementation and usage. Results, future works and concluding remarks are addressed in sections 9, 10 and 11, respectively.

2. THE GUITAR AND GUITARIST'S CONSTRAINTS

The interaction between a guitarist and his guitar is unique under the performance point of view. Any alteration in this relationship has a direct influence on the outcomes of the performance [1]. A guitarist does not play two different guitars in the same way; as well as a guitar is not played equally by two different performers. Therefore, to model a guitar performance both the guitarist and the guitar must be modeled.

Although performers can be right or left-hander, this work considers a right-hander as default.

A guitar can be classified by its acoustics, aesthetics, playability and fitting. The last two classifications deal with the comfort and matching of the guitar to the performer. Many are the attributes involved in the guitar's playability, however, in the context of this work we are going to be limited to the most commonly found among fretted instruments. They are:

Number of strings: Usually, a regular guitar has 6 strings. Other string-fretted instruments have different settings, but normally this number stays between 4 and 12;

Tuning: The standard guitar tuning is E(low) A D G B E(high), from the 6th to the 1st string. Other instruments may have different tunings, and even the guitar may have its tuning changed to play specific styles of music;

Number of frets: Frets indicate fractions of the length of a string and consequently the note. The number of frets varies according to the style and model of the guitars; usually the number stays between 12 and 18 clear frets;

Some guitarist's attributes are commonly expressed in terms of the guitar characteristics. Usually guitars are built for right-handed people with a scale length around 25 inches. The scale length is a very important attribute related to the guitar's playability since it determines the fret's positioning. Higher the scale length value, higher the distance between frets and, as a result, bigger must be the guitarist hands. Some of the guitarist's attributes considered in this work are:

Number of left-hand fingers: The classical technique requires the use 4 fingers to execute the chords. Some guitarists also use the thumb in the upper strings, which means, they can use all the 5 fingers. Deformations or debilities may reduce the number of available fingers available.

Number of right-hand fingers: Once again, the classical technique requires the use of 4 digits of the right hand to pluck the string. However some right hand techniques such as the ones found in Spanish Flamenco may require more fingers or different use for the fingers (i.e. tapping).

Finger Stretching: A chord shape that requires a large stretching of the left-hand fingers might be difficult to be performed by beginners. This stretching normally increases with practice. Usually, a finger stretching of 4 frets is desirable for guitarists.

3. AGENTS APPROACH PROPOSAL

The proposed multiagent system is composed of three types of agents. Each of the agents represents an element involved in a guitar performance: (i) the guitarist's left hand, responsible for the harmony of the music; (ii) the guitarist's right hand, responsible for the rhythm; (iii) the guitar itself, responsible for producing the sound.

Figure 1 shows the proposed agent's society and its communication schema.

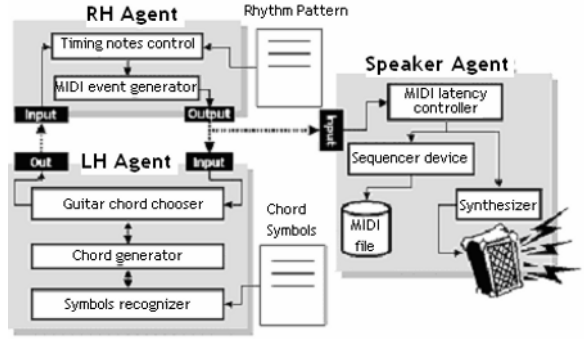


Figure 1: Architecture of the proposed multiagent system

The separation of the musical elements (harmony, melody, and rhythm) is a common practice in musical software development because it simplifies the understanding of the software [12]. In the context of this work, the separation occurred based on the role of the agents as further explained.

4. THE LEFT-HAND AGENT (LH)

In a guitar performance, the guitarist left-hand fingers shorten the strings in the precise length necessarily to produce the desired note. This is done by pressing the string against the frets located on the fretboard. The map with the place for each finger over the fretboard is called chord shape.

Choosing an appropriate chord shape is one of the tasks given to the LH Agent. To do so, first the LH Agent needs to translate a textual chord notation (characters) into a musical chord (notes). After, the LH needs to calculate the chord shapes for that particular chord. These tasks will be detailed in next subsection.

4.1. CHORD TEXTUAL NOTATION RECOGNITION

In popular guitar, guitarists usually write their songs using just the chord's textual notation or an execution notation such as a tablature ciphers. Due to this fact the first problem to be addressed by the LH Agent is the recognition of a string of characters representing musical chords;

The textual notation used to describe chords is not completely standardized. The same set of notes can be written in different ways. For example, the notes: C, E, G, and B can belong to either a C7M or an Em/C chord. Moreover, C7M chord can also be written as Cmaj7.

To endow the LH Agent with the ability to validate a string of characters as a musical chord, a finite automata represented by a 5-tuple D1 = (Σ, Q, δ, q0, F), was implemented, where:

a) Σ is the input symbols alphabet, Σ = note $\cup$ alt $\cup$ sus $\cup$ var
   nota = {A, B, C, D, E, F, G}
   alt = {#, b}
   var = {º, m, 5}
   susN = {sus2, sus4, sus9, sus11}
b) Q is the possible states set, Q = {S1, S2, S3, S4, S5, S6, S7}
c) δ is the transition's partial function $\delta : Q \times \Sigma \rightarrow Q$
d) q0 is the initial state, q0 = S1
e) F is the final state set, F = {S3, S4, S5, S6, S7}

Figure 2 illustrates D1, which is just a partial automata used to demonstrate the main idea. The final states S3, S4, S5, S6, and S7 conduct the output of D1 to the respective sub-automaton responsible for validating the whole word.

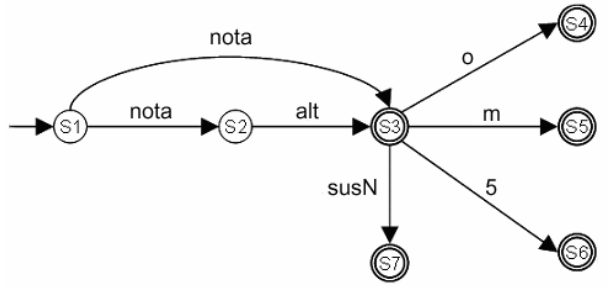


Figure 2: Initial automata.

The alphabet in use (as seen in Table 1.) came from the Brazilian "Bossa Nova" known by its complex harmonic structures, hence should cover any simpler music style. Nevertheless, it could be customized or even replaced to attend any musical background.

Table 1: Brazilian Chord Notation

Symbol	Description	Symbol	Description
A	Note A	dim	Diminished
B	Note B	2	Major 2nd
C	Note C	b2	Minor 2nd
D	Note D	4	Perfect 4th
E	Note E	#4	Augmented 4th
F	Note F	5	Perfect 5th
G	Note G	#5	Augmented 5th
#	Sharp	b5	Minor 5th
b	Flat	6	Major 6th
add	Interval addition	7	Minor 7th
(	Begin of a note alteration	7m	Major 7th
)	End of a note alteration	9	Ninth
/	Chord inversion	b9	Minor 9th
^	Interval junction	11	Major 11th
M	Major	#11	Augmented 11th
m	Minor	13	Major 13th
sus	suspension		

Although the notation can be customized, there are some fixed rules to be considered:

a) Parentheses (or any symbol for note alteration) are used when there is an alteration (# or b) in the basic intervals of the chord or when there is an interval that does not compose the basic structure of the chord (9, 11, and 13).

b) Also, it was established that only tetrachords could be considered diminished (1+b3+b5+b7). Equivalent triad must be 1+b3+b5, i.e. Cm(b5).

c) The "add" symbol is used when there is an additional interval that is not the next natural interval (related to the last one). E.g. C = 1+3+5; the next natural interval in this case is the 7th. If the 9th is desired but not the 7th, then the "add" must be used as shown: C(add 9) = 1+3+5+9.

The "^" non-musical symbol was created due to technical limitations of the automata once it considers the blank space as the end of the text word. If we have to validate a chord like C7M(9 11) we must replace the blank space by the "interval junction symbol".

Posterior to the syntax analysis, the system will run a semantic analysis to determine the type of chord and consequently its notes. For example, a minor (m) chord must have 1 + b3 + 5 intervals. Not only the notes are attached to the chord but also the interval they represent.

Types of chords are dependent on the number of notes in it. Triads (3 notes) can be major, minor, suspended, and altered. Tetrads (4 notes) extend the triads types in: diminished, 7^{th} major, 7^{th} minor; Chords with more than 4 notes are tetrads with additional notes.

Either triads or tetrads can be inverted. Inverted chords have the lower/bass note different from the root of the chord. The chord C/G, for example, has an inversion; Even though the G note belongs to the basic structure of the C Major (C + E + G), it was not meant to be the lower/bass note(C is the root). So, to set G note as the lower one, an inversion was written. The same does not happen to the Am/F# chord because the F# note is not part of the basic structure of the Am chord, so it must be added to the set of notes even if the 6th interval is not written in the chord's name.

4.2. THE CHORD SHAPES

Traditional musical notation was not designed to be used with a particular musical instrument. This generality makes the notation flexible but also incomplete for some instruments. The same happens to the harmonic notation used in this research; the chord notation is used to write down only the harmonic part of the music and cover all the harmonic instruments. Each instrumentalist should apply this information within its own context. In the guitar case, the guitarist should retrieve from his memory (or calculate) the chord shapes for that exacting chord and choose the most appropriated one.

The LH Agent does not have a memory for chord shapes. Instead, it calculates all the chord-shapes in real-time. This approach was thought considering the several variable guitar-performer combinations (different tuning, string, finger-stretching etc.) that have an impact on the chord shapes.

The chord-shapes calculation is a time-demanding process and for that reason it was necessary split it in two stages. In the first stage a set of simple chord shapes are calculated. Simple chord shapes are those with only one instance of each note, suggesting that the guitar may have some free strings that could be used to repeat some notes. The repetitions take place in the 2nd stage, later explained.

Duplications, doubling, triplication, and inversions are the operations that run over the simple chord shapes generated in the first stage. It considers instrument restrictions (tuning, number of strings, and frets and user's profile (number of available fingers and fingers' stretching). According to the musical style and guitarist preferences, the LH Agent can be configured not to calculate a chord shape with, for example, the 3rd duplicated interval.

During the 2nd stage the LH can be set to consider (or not) the following options:

Fundamental note doubling: Repeats the fundamental note.

Fundamental note duplication: Repeats the fundamental note in the same octave.

Fundamental note triplication: Allows the repetition of fundamental note up to 3 times.

3rd interval doubling: Repeats the note related to the 3rd interval (major or minor).

Perfect 5th doubling: Repeats the note related to the 5th interval.

Perfect 5th duplication: Repeats the note related to the 5th interval in the same octave.

Perfect 5^{th} suppression: Suppresses the 5^{th} interval in order to add a high priority interval. This option is only considered when it is not possible to mount the chord shape with all the intervals defined in the chord symbols, so the 5^{th} is omitted.

Octave consideration (diatonic scale): Differs, for instance, a 2nd interval from the 9^{th}, although they are the same note they are in different octaves. This is constantly ignored in guitar chords.

Fingering suggestion: Allows the system to suggest the fingering to the chord.

Bar chords: Allows the system calculate or not the bar chords (fretting).

Inversion calculus: Allows the system to calculate inversions in chord shape even though it is not written in the notation.

The processing time, measured in number of computational operations, increases significantly for each of the above options added in the calculus as given by Equation 1, where **n** is the number of parameters involved.

$$f(0) = 1$$

$$f(n) = n * f(n-1) + 1, \forall \{n \in \mathrm{N}^* | n \geq 1\} \quad (1)$$

For instance, if the system has to consider the "Perfect 5th suspension" and "Diatonic scale" then **n** = 2 and the number of computational cost is given by f(2) = 5. However, if third option "Bar Chords" is brought to the equation then f(3) = 16, exponentially increasing the computational costs involved.

We consider for this work that duplications of notes occur in the same octave, while doubling is in a different octave. Therefore, duplication is a case of doubling.

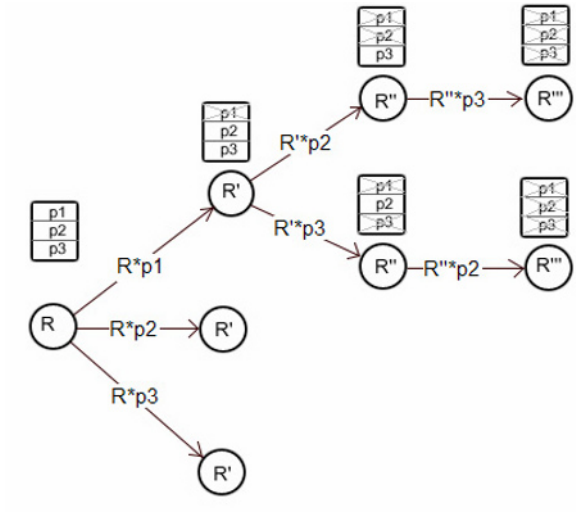


Figure 3 illustrates the 2nd stage of the process running over a simple chord shape. In the example given, 3 parameters represented by p1, p2, and p3 are tested. Observe that the simple chord shape "R" is tested for all 3 parameters in the first level. The result is then tested for the other two remaining parameters in the second level and so on.

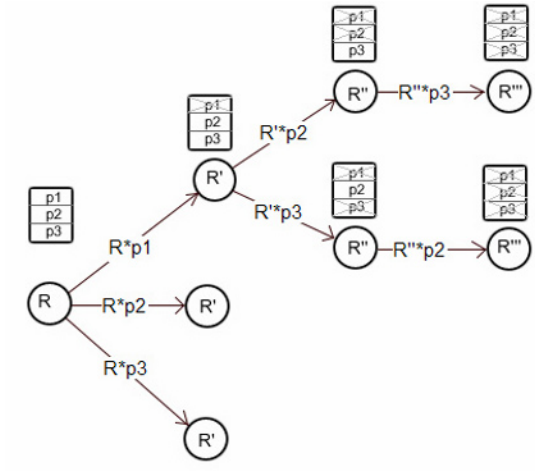


Figure 3: Processing made over a simple chord shape.

Still in Figure 3, suppose R as a C major with a simple chord shape given by 53(5th string, 3rd fret - C); 42(4th string, 2nd fret - E); 13(1st string, 3rd fret - G). Considering a standard 6-string guitar, we still have space to double notes in the 6th, 3rd, 2nd strings. The parameters that later will be turned into notes are given by "p". So, p1 could be the "Fundamental note duplication" process and p2 a "Perfect 5th duplication". In this scenario, a possible value for R' could be 53(C); 42(E); 21 (C); 13(G) – fundamental note duplicated. Once again R' still has room in the 6th and 3rd strings, so the algorithm keeps searching for p2 and, in this example, R" assumes the value 53(C); 42(E); 30(G); 21(C); 13(G) – Perfect 5th duplicated. Repeated chord shapes are discarded.

Once all the chord shapes are generated, they are sorted by: finger stretching (decreasing), number of required fingers (increasing), and fret average (increasing). The top entry is considered the easiest one but not necessarily the most appropriated or the one that sounds better.

4.3. CHOOSING A CHORD SHAPE

The choice of the chord shape is mostly determined by the musical style (strongly based on rhythm) but the instrument characteristics and performer preferences are also considered in the choice. In our approach, all the generated chord shapes are in accordance with the guitarist and instrument constraints, so the choice of the chord shape relies on two main factors: transitional effort (travel cost) and musical style (rhythmic pattern), the later is explained in next section.

The configuration of the hand during a chord in performance is a result of the contraction of instinct and extrinsic muscle of the hand and forearm. The brain's control over those muscles is improved with training;

additionally, the muscle and tissues involved in this skilled task adjust themselves to gain efficiency. This complex system is unlikely to be formalized in a single cost function, as observed by Wing et.al [13] but it has been used with success in some researches aiming to optimize guitar fingering [10].

Findings from memory-for-movement tasks show that people are poor at remembering movements but are good at remembering positions. However, instead of storing all possible positions, the human brain stores just a few postures and derives new postures from them. The posture that is going to be used is decided by the travel cost, in other words, the estimate cost of moving from the stored posture to the goal posture [13].

Based on this fact, we used a similarity function to compare the previous chord shape with the candidates chord shapes of the next chord in the sequence. The more similar the chord shape is with the previous one, lower is the effort to move from one to the other. The similarity function returns a value between 0 and 1, where 1 means the same chord. To find the most similar chord shape, all simple chord shapes (generated in the 1st stage) are compared to the previous complete chord shape. Only the most similar simple chord shape goes through the 2nd stage.

The similarity functions used in this work are given by equations system bellow:

$$A = \left[a_{i,j} \right] n \times m : a_{i,j} = fSimP\,(PosA\,, PosB\,) \tag{2}$$

$$Pos = (String, Fret), NewC = [Pos], OldC = [Pos] \tag{3}$$

$$fSimP(PosA, PosB) = 1 - \frac{1}{e} \times fDis(PosA, PosB), \quad \forall PosA, PosB : PosA(string) = PosB(string) \tag{4}$$

$$fSimP(PosA\ , PosB) = \frac{1}{2} - \frac{1}{e} \times fDis(PosA, PosB), \quad \forall PosA, PosB : PosA(string) \neq PosB(string) \tag{5}$$

$$fDis(PosA, PosB) = \left| PosA(fret) - PosB(fret) \right|, \quad \forall (PosA(fret) > 0) \wedge (Pos_j(fret) > 0) \tag{6}$$

$$fDis(PosA, PosB) = \left| \frac{\sum_{k=0}^{n-1} NewC[k][fret]}{n - qtOSN} - \frac{\sum_{k=0}^{m-1} OldC[k][fret]}{m - qtOSM} \right|, \quad \exists (PosA(fret) = 0) \vee (Pos_j(fret) = 0) \tag{7}$$

Equation 2 represents a matrix of *m x n* elements where *m* = number positions in the chord shape and *n* = number of positions in the next chord shape. Similarities of each position are given by equations (4) and (5), where *e* = fingers stretching value. Equation (7) is used with open chords, where *qtOS* represents the number of open strings used in the chord shapes.

To exemplify, suppose an Am chord shape composed by the positions 50-42-32-21-10 which is going to be followed by a G chord. The first step is to find all the simple chords shapes for G. Some of them are: 63- 40- 20; 63- 40- 34; 63- 52- 40; 510- 49- 110; 63- 34- 23; etc. Every simple chord shape of G will be compared with the complete chord shape of Am, generating a matrix as shown in Table 2.

Table 2: Am to G Transition

	-	50	42	32	21	10
63	-	.31				
40		.31	.62	.32		
20				.31	.62	.31

For every position of G (e.g. 63), a value is calculated in relation to the positions of Am in the same string (equation 2), one above and one below (equation 3). For example, the position 40 (4th string open) of the chord G will compared with the positions of the 5th and 3rd string of Am chords, respectively 50 and 32.

The chord similarity is the average of the higher values of each row of the matrix. If the higher values are located in the same column then, once again, the higher value is chosen and 2nd higher of the looser row is selected. This will prevent the use of the same finger to execute two of the positions. In this example, with no columns tie, the similarity is (0.31 + 0.62 + 0.62)/3 = 0.52.

The most similar chord shape is then filled with repeated notes in strings that are not been used, creating new chord shapes. If one of these new complete chords' shapes satisfies the rhythmic pattern then it is selected, otherwise the next most similar chord shape is processed, and so forth. In the example given, the complete chord shape for the G following the Am (50-42-32-21-10) was the 63; 40; 30; 20; 13; both with the same polyphony.

Two observations must be made at this point:

a) In string-fretted instruments there are movable chord chords. These chord shapes parallel chord the fret-board maintaining the same configuration of the hand. This "hand motion" is not considered by the similarity function in this research.

b) As one could infer, the first chord of the music has a very important role in chord shape choices of the whole composition because it can take the selection of the chord shapes to a very high region of the instrument, which is not often used for lead guitar. Thus to sound more natural the system chooses the chord shape that has the lower fret average, that is, the closest one to the guitar head.

5. THE RIGHT-HAND AGENT (RH)

It was previously mentioned that one of the factors used in the chord shape choice is the musical style. One of the most noticeable characteristics of a musical style is its rhythm. Surprisingly, the link between the chord shape and the guitar rhythmic pattern is often ignored by Instrument Performance Systems. The goal of the Right-Hand (RH) Agent is to assist the LH Agent in his decisions, making sure that the selected chord shape is appropriated to the rhythmic pattern.

The guitar rhythmic pattern (GRP) is a set of repetitive actions performed by the guitarist's right-hand aiming to displace the guitar's string, thus create a vibration which will result in the note's amplitude. These plucking actions involve direction, intensity, timing, angle of attack, plucking point, etc.

Two types of GPR are more commonly found: arpeggios and strums. Strums are characterized by fast and directional movements that hit several strings almost simultaneously. Either fingers or plectrums can be used. It is fast, loud, and very rhythmic driven, Arpeggios usually produce a more soft sound with the strings being pulled with more precision. Finger style is more common but a plectrum can also be used for sharper tones.

The RH Agent differ strums from arpeggios based on the Musical Events (ME) that composes the GRP. If the number of ME's starting in a same beat is greater than the number of right-hand fingers then a strum is assumed. Due its fast nature it is not possible to jump a string that belongs to the strumming block; as a result, the strings of the chord shape must be contiguous. Figure 4 shows an example of GRP (arpeggio with 4 voices).

The polyphony of the GPR will determine the "richness" of the sound, commonly described as a "full chord" sound. This number can go from as little as 1 up to the instrument's polyphony. Higher the polyphony greater the sound produced. Due to the lack of precision of the strums, the pattern of the movements tend to be more important than any other attribute so, if the GPR is recognized as a Strum, the polyphony of the GPR will be automatically set by the RH Agent.

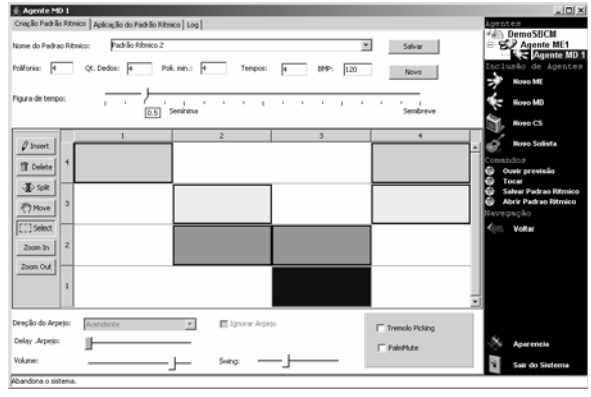


Figure 4: RH Agent's interface

In arpeggios, the number of simultaneous notes in the same beat does not exceed the number of right hand fingers meaning that is possible to pull the strings either individually or simultaneously. The strings are not necessarily contiguous and the polyphony is manually configured in the GPR.

The GPR parameters manipulated by the RH Agent are:

Attack Direction: Upper or down arpeggio. It is related to the hands movement and not to the pitch of the notes. In the down arpeggio lower notes are reinforced and in the upper the higher ones.

Arpeggio rate: How slow the system will play the arpeggio. In other words, the time gap between each note of the arpeggio. It varies based on the current time-figure note.

Swing: Time variation to more or less than 50% of the time-figure note. The goal of this parameter is to humanize the sound, playing the notes with a little delay or precipitation.

Volume: Notes amplitude. It is used to accent the beats.

Polyphony: Number of simultaneous notes that the chord shape should have (voices). A fixed number or a range can be set.

Beats: The duration of the rhythm pattern is given by the number of beats multiplied by the time-figure note value.

Time-figure note: The duration of each beat. The default is 1 (quarter note); the higher it is the slower is the execution.

Number of fingers: number of right hand fingers available for the execution of the rhythm pattern. Used by the Agent to infer the GPR type; the default value is 4;

Arpeggio ignore: When an arpeggio is identified by the system, the swing parameter is disabled since the

attack time will be calculated based on the arpeggio delay and it does not consider the swing value. This option could be used to enable manual swing setting.

6. THE SPEAKER AGENT

The interaction between LH and RH Agent determines the physical actions necessary to play the music but not the music itself. The idea is similar to playing an electronic keyboard without plugging it in to a power socket - correct movements (actions) but no sound. This happens because, although the LH and RH Agents can deal with musical elements, they are not able to translate them into sound. The role to render these performance actions into sound is from the Speaker Agent.

The Speaker Agent (SP) synthesizes and mixes the notes generated by the interaction of LH and RH Agent. It implements all the controls and parameters involved in the sound generation and music reproduction like: mute, solo, timbre\envelop, volume, play, pause, stop etc.

The main task of this Agent could be wrongly assumed to be less important than the others. In fact, the importance of this agent grows with the complexity of the society that could be turned into musical chaos if not well managed.

The Speaker Agent stores all the musical materials created by the society in a queue, sort them by execution time, and render them into sound using the local resources of the client machine, differently from the other agents that could run remotely.

Figure 5 shows the list of agents that belong to a certain compositional and how the sound attributes can be changed using the Speaker Agent.

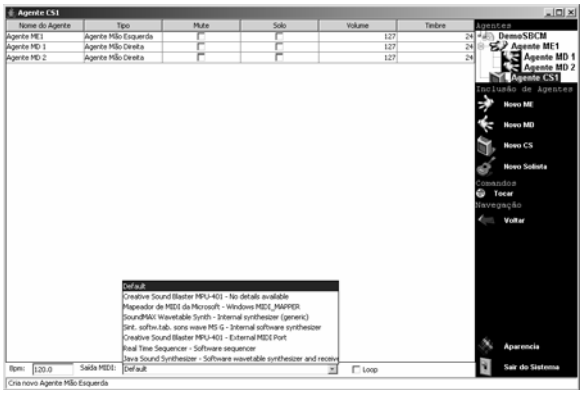
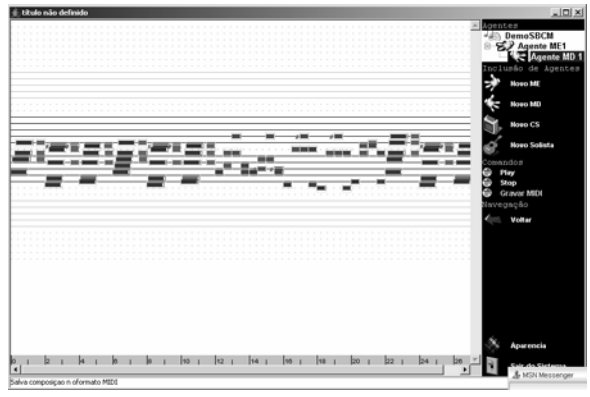


Figure 5: Agents list in the Speaker Agent

The quality of the generated sound is as good as the resources local available in the client machine. External synthesizers and samplers can also be used (bottom of Figure 5).



6: Piano roll musical notation.

The Speaker Agent shows the composition under several perspectives: list of agents (Figure 5), piano roll musical notation (Figure 6) and even the agent's decisions (Figure 7).

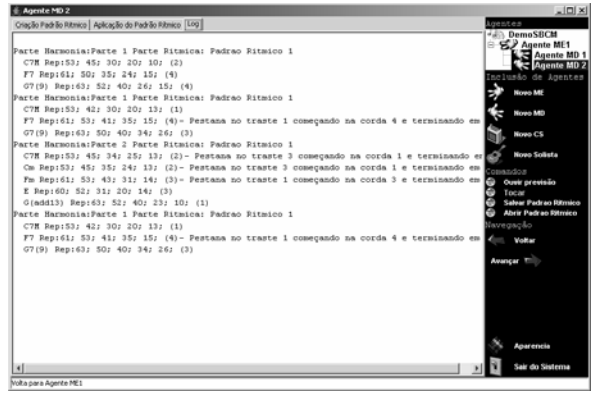


Figure 7: Chord shapes choices.

7. TECHNICAL NOTES

The system was implemented using Java JSDK 2.0. The jMusic API [7] was used to implement the control over external MIDI devices, since the Java was not able to communicate with external devices in the version used in this implementation (1.4).

The communication between agents was implemented using RMI – Remote Method Invocation. The messages format encapsulates MIDI messages and extends as follow:

Message Format: (<sender>, <receiver>, <MIDI message>, <execution time>, <id>)

Although the system was developed using Java Technology and no restrictions concerning its portability and use in the Web was expected, some problems related to execution were observed in Mac OS and Linux platforms.

8. SYSTEM USAGE

Agents are created into the context Compositions. A composition must have at least one of each type of agent. The maximum number of agents is limited by the computer's processing capabilities.

The procedure to set up the most basic composition is:

1. Create a composition;
2. Create a LH Agent;
3. Create harmonic line for the LH Agent;
4. Create RH Agent and link it to the LH Agent;
5. Create/Load the Rhythmic Patterns(GPR) to the RH Agent;
6. Link the GPR to the harmonic parts previously defined in the LH Agent;
7. Create a Speaker Agent;

After all the steps the music should be ready to be played.

8.1. THE HARMONIC LINE INPUT

Chords can be grouped into structures called *parts*. Usually, the *parts* repeat along the music. The rhythm patterns defined in RH Agent are linked to these *parts*. Thus, the part carries both harmonic and rhythmic information. Figure 8 shows how the parts can be organized along the composition.

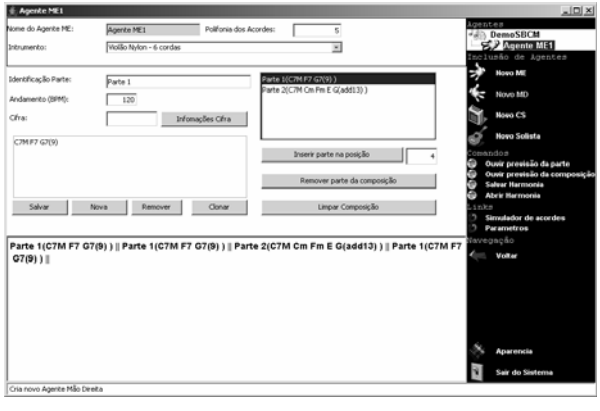


Figure 8: Harmony line input.

Every chord inputted is validated against the chord notation in use and a feed-back is given. For example, if the invalid chord "C#%" is presented to the LH Agent, a feedback message will showed reporting that the chord "C#%" is not valid according to the current chord notation and the incorrect symbol is "%". Yet, the user can query the LH Agent about a chord, obtaining the notes and intervals that compose it.

8.2. THE RHYTHMIC LINE INPUT

The connection between the RH and LH Agents is established at the moment of RH creation. The RH must always be linked to a LH Agent. In practical terms, each RH agent hooked to an LH agent has an exclusive track in the midi channel reserved for the LH Agent.

The RH Agent has an unlimited memory for GPRs that could be used in different parts of the harmonic line. Hence, in fact, multiple RH agents are only necessary in polyrhythmic compositions.

The GPR can be draw (Figure 4) from scratch, loaded and saved. Similarly to LH Agents, the RH Agent is setup through the Speaker Agent interface running into the client machine.

9. RESULTS

Thirty musicians, half of them guitarists, participated in the test of the system. They were asked to produce musical material in any way they wanted. Questionnaire and interviews were used for feedback.

The main problem reported by the testers was related to the difficulty in playing a previously known musical piece in the same way they would do it without the use of the system. This observation was expected since the system is autonomous in its decisions and the user has no control over it. To overcome this issue and make the system more user-friendly, a special notation was designed to input fixed chord-shapes instead of chord's names.

Composition-wise, non-guitarists found the system more useful and reliable than the guitarists. Forty six (46%) percent of the non-guitarists said that the system sounded like a human, but only 26% of the guitarist would have taken the same decisions as the system. In summary, 76% of the total group said the system played idiomatically.

To exemplify the outcome of the system, we submitted part of the harmony of "Girl from Ipanema" (Vinicius de Moraes e Tom Jobim) with a "Bossa Nova" GRP and the result was G7M(9) - 63; 50; 44; 34; A7 - 65; 57; 45; 36; Am7 - 65; 57; 45; 35; D7(b9) - 55; 44; 35; 24; G7M - 63; 44; 34; 23; D7(add13) - 40; 34; 21; 12;

Although the chord's shapes are correct, they are not commonly used by guitarists mostly because the system does not consider the overall sound quality of the chord shape, just its playability and difficulty.

Musical esthetic is difficult to measure even among human musicians due its subjectiveness. Thus, even though our approach could generate correct

musical performances, it may not be the one that sounds best. The musical material generated in the experiments can be found at http://cmr.soc.plymouth.ac.uk/members/llcostalonga/music/midi/.

10. FUTURE WORK

Future work that focuses on the agent's deliberation process improvement is twofold. The first research direction is concerned with the extension of the approach here presented to cover commitment strategies and deliberation using affective aspects.

Our agents will present different behavior according to affective inputs. For example, the approval/rejection of the pubic, the agent internal affective state etc. The idea is to use affectivity to generate performances not only correct but also allied with the emotion the composer wants to transmit to the audience. For more details about the algorithm that we are developing and testing see [3].

The second direction goes towards the development of a Java API with the musical structures and algorithms developed in this research context. The first version of the API is hosted by SourceForge.net under Academic Free License. Free download and detailed information is available at http://sourceforge.net/projects/octopusmusic/ .

11. CONCLUDING REMARKS

This paper presented a multiagent system capable of generating guitar performance simulation.

A brief description of a guitar performance and the elements involved in this task was introduced. The elements identified as relevant were then modeled in three types of agents: Left-Hand Agent (harmony), Right-Hand Agent (Rhythm), and Speaker Agent (Sound generation and GUI interface).

Algorithms and strategies to interpret the harmonic notation and generate chords' shape were detailed as part of the LH Agent roles. Transitions between chord shapes were implemented using a similarity function simulating the hands' travel cost.

The RH Agent was designed to help the LH Agent in his decision regarding the most suitable chord shape to a certain guitar rhythmic pattern/musical style.

Technical information related the system implementation and usage was offered in the sections 6 and 7. To summarize, the evaluation of the system showing its usability in the compositional field was presented followed by the future works, which aims to improve the quality of the generating performances by taking into account the affectivity as part of the decision process.

REFERENCES

[1] Costalonga, L. and Miranda, E., Equipping Artificial Guitar Players with Biomechanical Constrains: A Case Study of Precision and Speed, *in Proc. International Computer Music Conference*, Belfast, 2008 (to appear)

[2] Costalonga, L.; Viccari, R. M.; Multiagent System for Guitar Rhythm Simulation. *In Proceedings of the International Conference on Computing, Communications and Control Technologies*, Austin, Texas, USA, 2004.

[3] Fagundes, M. S, Vicari, R. M. and Coelho, H. (2007) Deliberation Process in a BDI Model with Bayesian Networks. *In: 10th Pacific RIM International Workshop on Multi-Agents (PRIMA 2007)*. Lecture Notes in Artificial Intelligence. 2007.

[4] Gabrielsoon, A., Music Performance. The Psychology of Music. *In. D. Deutsh, ed. The Psychology of Music,* 2nd ed. New York: Academic Press, 1997.

[5] Heijink, H. and Meulenbroek, R.G. On the complexity of classical guitar playing: functional adaptations to task constraints, J Mot Behav, 34 (4) 339--351, 2002.

[6] Honing, H. Computational Modeling of Music Cognition: A Case Study on Model Selection, Univ. of California Press, 2004.

[7] jMusic- Computer music composition in Java . Available at http://jmusic.ci.qut.edu.au/. Accessed on Jan 2008.

[8] Miranda, E. R., An Artificial Intelligence Approach to Sound Design. *Computer Music Journal* 19(2), 1995, pp. 59-74.

[9] Miranda, E. R. Emergent Sound Repertories in Virtual Societies. *Computer Music Journal*, 26(2), Cambridge, Massachussetts: MIT Press, 2002, pp. 77-90.

[10] Radicioni, D. and V. Lombardo "Guitar Fingering for Music Performance." *Proc. International Computer Music Conference*: 527—530, 2005.

[11] Wessel, D. & Wright, M. Problems and Prospects for Intimate Musical Control of Computers. *Computer Music Journal*, 26(3), Cambridge, Massachussetts: MIT Press, 2002, pp. 11-22.

[12] West, R., Howell, P., Cross, I.. Musical Structure and Knowledge Representation. *In P. Howell, R. West, & I. Cross (Eds.), Representing Musical Structure (pp. 1-30).* London: Academic Press, 1991.

[13] Wing, A. M., P. Haggard, et al. Hand and brain: the neurophysiology and psychology of hand movements, Academic Press San Diego, 1996.

[14] Wulfhorst, R.; Nakayama, L.; Vicari, R. M. A Multiagent Approach for Musical Interactive Systems. *In Proceedings of the Second International joint Conference on Autonomous Agents and Multiagent Systems*, Melbourne. 2003

[15] Zhang, Q. and Miranda, E. R. Evolving Expressive Music Performance through Interaction of Artificial Agent Performers, *In Proceedings of ECAL 2007 Workshop on Music and Artificial Life*, 2007.