Constructing Distributed Fretted Instruments for the Web

Prof. Jônatas Manzolli, BMath, BMus, MSc, PhD
Insterdisciplinary Nucleus for Sound Studies - NICS
Music Department, Arts Faculty , UNICAMP
BRAZIL.
e-mail:jonatas@nics.unicamp.br

Fernando Lindner Ramos, BCompEng
Insterdisciplinary Nucleus for Sound Studies - NICS
Institute of Computing , UNICAMP
BRAZIL.
e-mail:fernando@nics.unicamp.br

Frederick Carrilho, BMus
Insterdisciplinary Nucleus for Sound Studies - NICS
Music Department, Arts Faculty , UNICAMP
BRAZIL.
e-mail:fredcarilho@nics.unicamp.br

Abstract

In this article, we present a new computer based music instrument for distributed performance on the Web. It was named "Cordas Virtuais" and we took advantage of the recent Java2 implementation to create a general model for fretted-string instruments using class abstractions. There is a heavy usage of JAVA Objected Oriented inheritance to encapsulate gestures derived from fretted string instruments using parameters such as string tuning, group of strings, hand gestures, rhythmic patterns, fingering and alike, that carry each a set of attributes and properties. We call this kind of new musical application as Distributed Musical Instrument (DMI). As an applet, Cordas runs in any browser supporting the current Java Virtual Machine (JVM) across the Web. We describe the concept of PlayStyle that was created to define styles of fingering on the strings. In the implementation we have two class-groups: the left hand and the right hand gestures. The left-hand classes control pitch changes or chords, and a right hand control the rhythm, dynamics, micro-rhythms and rhythmic patterns. A Co-ordination Matrix controls real time changes on left-hand movements. This matrix generates chord orbits that are equivalent to the traditional chord cadences. Finally, to show the potential of Cordas, we presented four musical examples in which a set of fretted instruments varying from the Classical Guitar to the Chinese Pipa were simulated.

Keywords: ***MIDI, Java2, interactive music, fretted instruments, real time***

1. Introduction

Recently, much research has been done to explore the musical potential of the Internet: Burk [1] developed a new Client/Server architecture for multi-user musical performance, (see www. transjam.com). Helmuth [2] discussed several host configurations to allow music performance on the Internet. Complementary, Hwang [3] discussed the concept of "*Virtual Musical Environment*" (VME) where musicians act in a multi-modal feedback process using the Internet. Several new tools to explore the musical potential of the Internet have been developed. Plenty of examples of these systems can be found in (see www.transjam.com/ or http://music.calarts.edu/~tre/JavaMusic.html) the Java Music Projects

Beyond using the Internet as a new musical media, the research presented here is in line with previous works in which "Interactive Music System" (IMS) has been studied. Rowe [4] defined interactive music systems as *those whose behaviour changes in response to music input*. We studied a way of expanding the notion of IMS using generative methodologies. It was started, few years ago, with the definition of Sound Functors [5], a mathematical definition for musical construction and organisation. We explored Evolutionary Computation to created the compositional environment named VOX POPULI [6]. Later, we explore the use of a robot as an interactive actor in a composition system, this was called ROBOSER [7]. Eventually, we also explore the use of new hardware as interface for music performance as presented in [8]. Recently, we participate of the creation of *"ADA: the Inteligent Space"* that used interactive soundscapes to communicate behavioural changes to the audience. This was presented at the EXPO.02 the Swiss National Exhibition for 554.000 visitors. (see www.adaexposition.ch).

This paper presents a research that uses the Internet to expand the concept of IMS. Here we explore the notion of musical interaction creating a new environment that reproduces gestures of a musician when he/she plays a fretted instrument. In our system, class abstractions of a musical gesture are used as input. It can be also used to implement family of fretted instruments which we call *Virtual Strings Instruments* [9]. As an example, it is not difficult to construct a 20-string instrument, each one associated to a different MIDI program and performed by a 7-finger hand or, eventually, to simulate an ancient fretted instrument. As it is discussed later, these features are possible graces in heritage of the class abstraction used in our approach. The recent *Java2* provides interfaces and classes for I/O, sequencing, and synthesis of MIDI data. We used this API to implement fretted instrument characteristics like number of strings and frets, string tunings and the right hand rhythmic actions. *Cordas Virtuais'* classes are easily exchanged among the Web using and musicians can improvise, performing each one a personal string instrument.

In the next sections we present our research in details. We describe the project implementation concepts such as JAVA class abstraction, GUI and play style in the first section. It follows the definition of a simple mathematical model based on a co-ordination matrix to control the left-hand movements. Finally, we discussed the implementation of a set of fretted string instrument as Internet *applet* and this graphical instrument can be used to verify the musical potential of our system.

2. Project Implementation

Our aim was to study how JAVA class implementation can be used in order to create new kinds of sonic interactions. As present in [10], we have been worked with the concept of "*Distributed Music Instruments*" (DMI) to allow musicians to play together and improvise on the Web. In line with [1], we developed a Client/Server architecture where the JAVA based MIDI server imports several MIDI events over the Internet from clients and this allow musicians to Jam on the Web. In this way, this tested interactive architecture allows human and machine to co-operate in distributed musical performance situations.

Additionally, for modeling the musical gesture, we adopted the point of view of a right-hand instrumentalist. The left hand controls the fret choice while the right hand attacks the strings. We used the *tablature* notation to describe the left-hand actions. It allowed a spatial index of a music sequence providing a precise indication of the fret and the string to be played. The right hand movements and rhythm were described by what we called *Play Style*. This is a class abstraction related to the instrument temporal control. Micro-rhythmic structures such as those found on the *Spanish Rasgueado Style* were implemented with success, see **www.nics.unicamp.br/cordasvirtuais/**. The main classes were divided in *Instrument* and *Performer* packages. *Instrument* contains the Synthesizer, instrument name (MIDI program), number of strings (individual or grouped strings). *Performer* contains representations of the *Tablature* and the *Play Style,* see [10] for details about the classes implementation.

2.1 Strings, Frets and Graphic Interface

Cordas Virtuais is based on a Toolbox developed to enable multi-task MIDI stream control, see Costa & Manzolli [11]. Independent sequences of MIDI data are generated and managed by a *Note Collector* simultaneously.

The *Play Style* interface for micro-rhythmic manipulation (see *Fig. 1*) consists on a interaction area for each string, where the horizontal axis determines the perceptual duration (the moment where the finger touches the string), and the vertical axis represents the attack intensity (finger's velocity)..

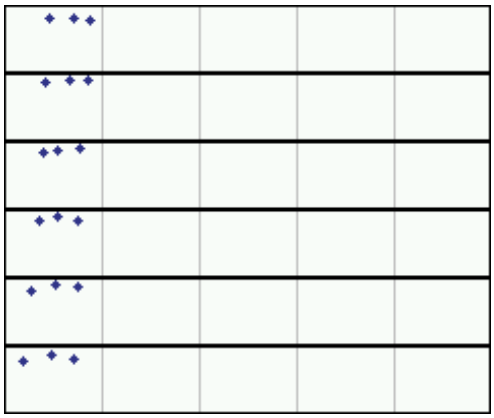

Fig. 1 - *Play Style Interface, vertical lines*

3. Generating Chords Orbits

3.1 Co-ordination Matrix

Def.1: We defined a *Co-ordination Matrix*, denoted as $C_{n\times n}$, a square matrix that is used to control the co-ordination between the left an right hand, n= number of strings group and its entries are defined in the set $\{0,1\}$.

As an example we present below the matrix $C_{6\times 6}$, it was applied in our experiment to generate sequence of chords that can be played in a classical guitar

$$C_{6\times 6} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} \\ C_{51} & C_{52} & C_{53} & C_{45} & C_{55} & C_{56} \\ C_{61} & C_{62} & C_{63} & C_{46} & C_{56} & C_{66} \end{bmatrix} \quad \textbf{(1)}$$

where $C_{ij} \in \{0,1\}, 1 \le i, j \le n$, with $n = number\ of\ string\ \ group = 6$, each group has one string.

Def.2: We define the *Tablature Vector* denoted as $\vec{x}$, an array containing the fret number played on the i[th] string denoted as $x_i = fret\ number$ $(1 \le i \le n$ with $n = number\ of\ strings)$.

This definition is equivalent to the traditional *tablature* of fretted stringed instruments. Since we are using the classical guitar as references, the order of the frets is: **the highest pitched string (E_4) is related to x_1, and the lowest pitched string (E_2) is related to x_6.**

$$\vec{x} = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6] \qquad \textbf{(2)}$$

with entries $x_i \in \{-1, 0, 1, \ldots, m\}$, where $m = number\ of\ frets$ and it depends upon the instrument we are using, i.e. for classical guitar *m=19*. The entry "-1" is used to mean "*the string is not played*" and it is assigned by the user. The entry "0" is used to mean an *"opened string"*.

Def.3: Given an initial *Tablature Vector* $\vec{x}_0$ and a *Co-ordination Matrix* $C_{6\times 6}$ we call $\vec{x}_{K+1}$ the next *Tablature Vector* in a sequence $\{\vec{x}_0, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$. This sequence is called here as *Chord Orbit* or *Cadence,* and it is generated by the matrix product and a *mod* operation, as defined below:

$$\vec{x}_{K+1} = C \cdot \vec{x}_K \qquad \textbf{(3)}$$

where $\vec{x}_K$ is the k^{nd} *Tablature Vector* and $C_{6\times 6}$ is the *Co-ordination Matrix*.

In order to confine the entries of the $\vec{x}_{k+1}$ in the set $\{0,1,\ldots,m\}$ we apply at it *k-step* the following operation:

$$\vec{x}_{K+1} = [x_1 \bmod m,\ x_2 \bmod m,\ldots,\ x_n \bmod m] \quad \textbf{(4)}$$

where *mod* operation gives the Z_m correspondent for each entry.

As an example of a cadence defined for a Classical Guiter and generated by the *Co-ordination Matrix*, we introduce an initial *Tablature Vector* related to a $\mathbf{G^7}$ chord $\vec{x}_0 = [1\,0\,0\,0\,2\,3]^T$. The *Co-ordination Matrix* that generates the next chord **C/G**, $\vec{x}_1 = [0\,1\,0\,2\,3\,3]^T$ is presented below:

$$\vec{x}_1 = C \cdot \vec{x}_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 3 \\ 3 \end{bmatrix} \qquad \textbf{(5)}$$

Def. 4: Given a *Tablature Vector* $\vec{x} = [x_1\ x_2\ x_3\ \ldots\ x_n]$ where $n = number\ of\ strings$, the MIDI note number table defined in $[0\ldots 127]$ and a string tuning note number T_i with $1 \le i \le n$ for each string, the note number y_i played by *Cordas Virtuais* is defined as:

$$y_i = T_i + x_i,\ 1 \le i \le n \quad \textbf{(6)}$$

where $x_i \in \{0,1,\ldots,m\}$ and it is generated by equation **(3)** or the entry -1 is assigned by the user.

Using **Tab 1.0**, it is possible to calculate the MIDI note number played by the system. Given the *Tablature* presented in **(4)**, the result is:

$$\mathbf{G^7} = [1\,0\,0\,0\,2\,3]^T + [64\ 59\ 55\ 50\ 45\ 40]^T = [65\ 59\ 55\ 50\ 47\ 43]^T \quad \textbf{(7)}$$

$$\mathbf{C/G} = [0\,1\,0\,2\,3\,3]^T + [64\ 59\ 55\ 50\ 45\ 40]^T = [64\ 60\ 55\ 52\ 48\ 43]^T \quad \textbf{(8)}$$

Classical Guitar Tuning Table						
string	1st	2nd	3rd	4th	5th	6th
pitch	E3	B2	G2	D2	A1	E1
MIDI Number	64	59	55	50	45	40

Tab 1.0 *Tuning system of the classical guitar. The first row is the string order, the second row contains the tuning for each string and the third row is the MIDI note number. These entries are equivalent to* $T_i, with\ 1 \le i \le 6$**.**

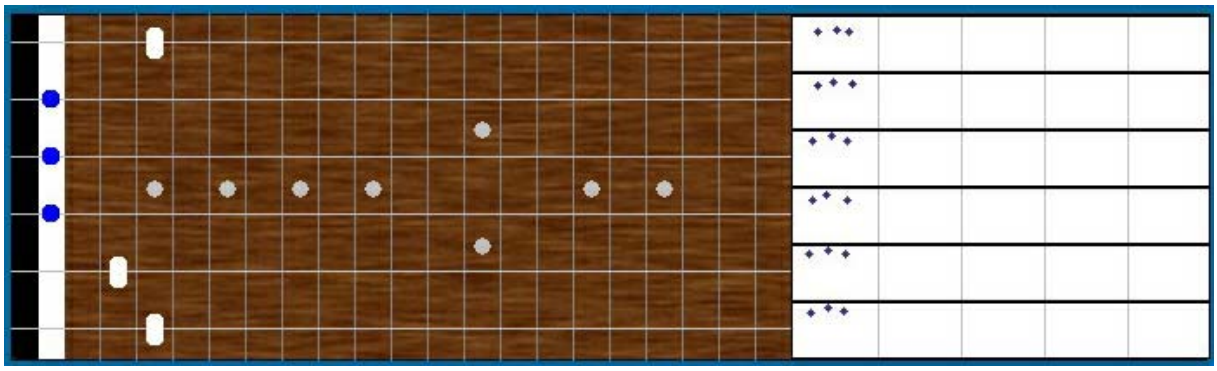

Fig. 3 – Cordas GUI Interface for a *Classical Guitar. In the left side, the white dots indicate pressed fret positions. |In the right side, dynamics, micro-rhythms and rhythmic patterns are described by blue dot positions.*

4. Instrument Case Studies

As mentioned above, the Classical Guitar was the first instrument studied, but the *Cordas Virtuais*'s objected oriented structure allowed an easy creation of new instrument instances. This section presents three applets implemented using Cordas, they are the Chinese *Pipa*, Andino *Charango* and the African *Kora*. The follow information are needed: a) Tuning Table and MIDI program number as seen on **Tab 1.0**, b) Number of string groups and c) Number of strings per group. The implementation of the *Kora* applet required to use to playstyle interfaces, since there is no fret use in this instrument and both hands are used to stroke the strings (like a harp). So, it was decided to use two *PlayStyle* pads, one in the right side and other in the left side (see

4.1 Pipa

Fig. 4 – The Chinese Pipa with four strings

It is a four-stringed Guitar like instrument, one of the oldest Chinese musical instruments created in China more than 2000 years ago. The instrument later was developed from its orginal two strings design into a form of four strings and twelve frets, plucked with fingernails and known as pipa or qin-pipa. Pipa was a general term referring to those plucked-string instruments played in hand-held positions with the outward fingering technique called "pi" and the inward one called "pa".

Pipa Tuning Table			
1st string	2nd string	3rd string	4th string
A4	D3	E3	A3
81	76	74	69

Tab 2.0 - *Tuning system of the Chinese Pipa.*

4.2 Charango

It is the main stringed instrument used in Andean music, The resonance box is usually made from an Armadillo shell or carved wood imitating the shell. It is used in almost traditional music of Peru and Bolivia.

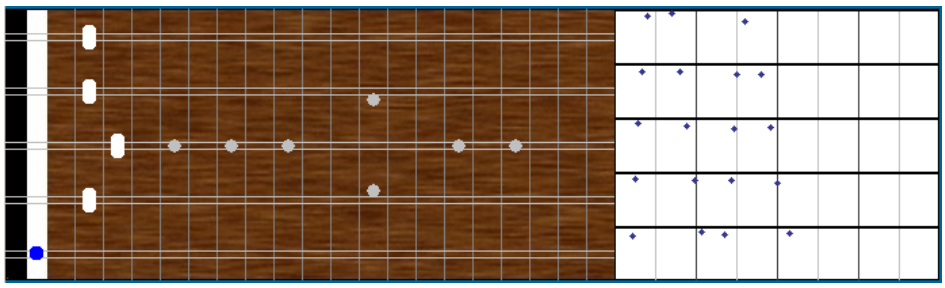

Fig. 5 – In the left**,** the Andean Charango with 5 group of double strings. Above, the GUI of Charango applet where one can see the double strings on the left side.

Charango Tuning Table				
1st double string	2nd double string	3rd octave string	4th double string	5th double string
E5,E5	A4, A4	E4, E5	C4,C4	G3, G3
88,88	81,81	76,88	72,72	67,67

Tab 3.0 – Tuning system of the Andean Charango

4.2 Kora

Fig. 6 – The African Kora

The Kora is arguably the most complex chordophone of Africa. It is played in the westernmost part of Africa in Mali, Gambia, Burkina Faso, Guinea, Sierra Leone, and Senegal. The calabash is covered with a cowhide that is stretched over the open side of the half calabash and then left in the sun to dry tight and hold the handposts in place. A Traditional Kora has 21 strings and it is not a fretted instrument, actually Kora is played with two hands simultaneously. Despite, it is not an instrument we decided to implement a new applet using two Play Style pads, one for each hand. The left pad has 10 strings and the right has 11.

Kora Left Hand Tuning Table										
B4	G4	E4	C4	A3	F3	D3	B2	A2	G2	C2
83	79	76	72	69	65	62	59	57	55	48

Tab 4.0 – The tuning system of the Kora´s Left Hand

Kora Right Hand Tuning Table									
E5	D5	C5	A4	F4	D4	B3	G3	E3	C3
88	86	84	81	77	74	71	67	64	60

Tab 5.0 - The tuning system of the Kora´s Right Hand

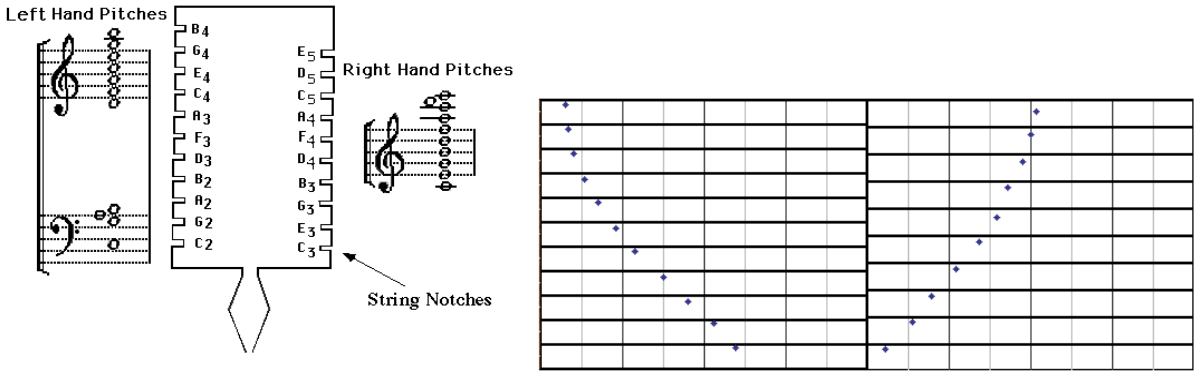

Fig. 3 – *(left) Kora Tunning system, (right)* Cordas Virtuais *Interface for Kora*

5. Discussion and Feather Developments

We presented examples of four fretted string applets: the Classical guitar, the Chinese pipa, the Andean Charango and the African Kora. These applets can be found in official Cordas Home Page (www.nics.unicamp.br/cordasvirtuais). The instrument case studies presented above showed the potential of the software and its flexibility resulting of the usage of class abstraction and inheritance. The mathematical model used to define the Co-ordination Matrix, is simple enough to be explored a dynamic system. The patterns on the Matrix can be controlled by several generative algorithms. Particularly, we intend to apply evolutionary computation as we have been studied in previous works [6]. One of the most interesting features of *Cordas* is that users can be seen as musician performing a jam section each one using a different instrument applet. We have tested this situation using a Client/Server architecture we developed [10,11], the result was really an interesting sound on the MIDI server. It produced a complex interweaving of melodic lines such as a Internet chat situation in which a group users write messages in different languages and the screen presents a multi-linguistic discourse.

The next steps of this research will be:

1) To study mathematical transformation on the co-ordination matrix in order to identify and control patterns as well generative structures;
2) To create an application version of Cordas improving several musical and control parameters parameters of the GUI, such as Load, Save PlayStyle, Save MIDI File, Record Pattern;
3) To integrate Cordas with other compositional environment we have developed at NICS, such as Roboser, VoxPopuli, etc.

Cordas will be applied to provide the exchange of fretted instrument features and performance characteristics distributed on the web. A database of *tablature* and *Play Style* will be constructed in near future.

6. References

[1] BURK, PHILIP L. 2000. Jammin'on the Web - a new Client/Server Architecture for Multi-User Musical Performance.Visual. In Proceedings of ICMC 2000, Berlin, Ed. The International Computer Music Association, 117-120.

[2] HELMUTH, M., 2000. Sound Exchange and Performance on Internet2. In Proceedings of ICMC 2000, Berlin, Ed. The International Computer Music Association, 121-124.

[3] HWANG, J., KIM, G. J., 2000. Design and Analysis of Virtual Musical Environments (VME). In Proceedings of ICMC 2000, Berlin, Ed. The International Computer Music Association, 125-128.

[4] ROWE, R., 1993. Interactive Musical Systems. MIT Press: Cambridge Massachussets.

[5] MANZOLLI, J. & MAIA, JR. A., 1998. "Sound Functors Applications". In proceedings of the V Brazilian Symposium on Computer Music, XVII Congress of the SBC, UFMG, Belo Horizonte, Brazil, pg: 115-120.

[6] MORONI, A., MANZOLLI, J., VON ZUBEN, F. & GUDWIN, R., 2000, "Vox Populi: An Interactive Evolutionary Syste for Algorithmic Music Composition", San Francisco, USA: Leonardo Music Journal - MIT Press, Vol. 10.

[7] WASSERMANN, K. C., BLANCHARD, M., BERNARDET, U., MANZOLLI, J., AND VERSCHURE, P. F. M. J. 2000. Roboser - An Autonomous Interactive Musical Composition System. In Proceedings of ICMC 2000, Berlin, Ed. The International Computer Music Association, 531-534.

[8] MORONI, A., MAMMANA, A. & MANZOLLI, J., 2001. "InstrumentALL: a Virtual Instrument". Proceedings of the 4th International Conference on Generative Art, Politecnico di Milano University, Milan, Italy.

[9] COSTA, M., MANZOLLI J. & SHARONI, D., 2002. "A Distributed Interactive Composition Tool", SISGGRAPH, San Antonio, USA, pg: 298

[10] MANZOLLI, J., COSTA M., RAMOS F., FORNARI, J. & SHARONI, D., 2002. "Solutions for Distributed Musical Instruments on the Web", Procedings of the Inagural Conference on the Principles and Practice of Programming in Java, Trinity College, Dublin. pg: 77-82

[11] COSTA, M. O., AND MANZOLLI, J., 2001. Toolbox para Aplicações Musicais na Internet. In Proceedings of SBC&M 2001, Recife, Brazil, 125-128.