U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Interfacing Jazz: A Study in Computer-Mediated Jazz Music Creation and Performance

Rui Miguel Silva Sampaio Dias

Programa Doutoral em Media Digitais

Produção de Conteúdos Audiovisuais e Interativos

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Digital Media – Audiovisual and Interactive Content Creation - of the University of Porto

Advisor: Dr. Carlos Guedes, Associate Arts Professor of Music, New York University Abu Dhabi
Co-advisor: Dr. Bruce Pennycook, Professor of Music, Department of Theory and Composition at the Butler School of Music of the University of Texas Austin

October 2018

Interfacing Jazz: A Study in Computer-Mediated Jazz Music Creation and Performance

Keywords: automatic music generation; computer-mediation; new interfaces for musical expression; jazz;

Acknowledgements

I first would like to thank my advisors, Professors Carlos Guedes and Bruce Pennycook. It was an immense privilege to work with such inspiring and insightful mentors. This work was born out of the very exciting 'Kinetic' project, led by Carlos, with a partnership with the Sound and Music Computer Group at INESC-TEC, led by Fabien Gouyon, and UT Austin, led by Bruce. The momentum gained from those two dense years of weekly meetings, brainstormings and international meeting point of people and ideas still endures and created bonds that will never disappear. Carlos, Fabien, Gilberto Bernardes, George Sioros, Samuel van Ransbeek, André Baltazar, Matthew Davies, Diogo Cocharro, Eduardo Magalhães, Mário Santos, Telmo Marques and Clara Morão. To all of them my appreciation for enduring those endless hours of (lousy computer kind of) blues demos.

During this research, I had the opportunity to make two research internships, at the University of Texas at Austin (USA), and at the New York University Abu Dhabi (UAE). The internship at UT was supported by an FCT scholarship, offered as part of the first prize of the 2011 ZON Multimedia Award, which we received with the application *GimmeDaBlues*. The internship at Abu Dhabi was an invitation from Carlos, supported by NYU AD. I want to thank Cristina Ioan and João Menezes for all the support, friendship and help around the campus.

My activity as professor and program director of the Electronic Music and Musical Production program at the School or Applied Arts (ESART) made it a difficult endeavor to reach the end of this dissertation. I want to thank my colleague Professors José Filomeno Martins Raimundo, José Francisco Pinho and Fernando Raposo for the continuous trust and support, and Luís Marques for all the help. I also would like to thank those of my students who for many times made me feel like a colleague of arms on the same side of the battle to get work done. Part of what motivates me to move on is the idea that at some point I may be at least half as inspiring to them as some teachers were to me.

I want to thank my dear friends Filipe Lopes, Rui Penha, Gustavo Costa, Patrícia Caveiro, Dimitris Andrikopoulos, António Sousa Dias, José Alberto Gomes, Pedro Motta Silva, Pedro Araújo, Joana Domingues, Maria Helena Vieira and Fernanda Queirós, for the friendship and encouragement. I want also to thank Marisa Silva for the kind and flawless support with all the formal procedures at FEUP.

This doctorate was partially supported by the Portuguese Foundation for Science and Technology (FCT) under the PROTEC 2 program for professors of the Polytechnic Institutes; “MyJazzBand” was one of the winning projects of the “Noite Branca 2016” competition in Braga, September 2016, funded by the Bracara Augusta Foundation. The multi-touch table used was kindly lended by the University of Porto. I want to thank Professors Rui Rodrigues and António Coelho for the support.

My greatest gratitude goes to my family. My wife Isabel, my super-kids Nuno and Sofia, my brother and sisters and my parents, José and Teresa, for all the love and support. This work is dedicated to all of them.

Abstract

This dissertation focuses on the study and development of computer-mediated interfaces and algorithms for music performance and creation. It is mainly centered on traditional Jazz music accompaniment and explores the meta-control over musical events to potentiate the rich experience of playing jazz by musicians and non-musicians alike, both individually and collectively. It aims to complement existing research on automatic generation of jazz music and new interfaces for musical expression, by presenting a group of specially designed algorithms and control interfaces that implement intelligent, musically informed processes to automatically produce sophisticated and stylistically correct musical events. These algorithms and control interfaces are designed to have a simplified and intuitive input from the user, and to coherently manage group playing by establishing an integrated control over global common parameters.

Using these algorithms, two proposals for different applications are presented, in order to illustrate the benefits and potential of this meta-control approach to extend existing paradigms for musical applications, as well as to create new ones. These proposals focus on two main perspectives where computer-mediated music can benefit by using this approach, namely in musical performance and creation, both of which can also be observed from an educational perspective. A core framework, implemented in the Max programming environment, integrates all the functionalities of the instrument algorithms and control strategies, as well as global control, synchronization and communication between all the components. This platform acts as a base, from which different applications can be created.

For this dissertation, two main application concepts were developed. The first, *PocketBand*, has a single-user, one-man-band approach, where a single interface allows a single user to play up to three instruments. This prototype application, for a multi-touch tablet, was the test bed for several experiments with the user interface and playability issues that helped define and improve the mediated interface concept and the instrument algorithms. The second prototype aims the creation of a collective experience. It is a multi-user installation for a multi-touch table, called *MyJazzBand*, that allows up to four users to play together as members of a virtual jazz band.

Both applications allow the users to experience and effectively participate as jazz band musicians, whether they are musically trained or not. The applications can be used for educational purposes, whether as a real-time accompaniment system for any jazz instrument practitioner or singer, as a source of information for harmonic procedures, or as a practical tool for creating quick arrangement drafts or music lesson contents. I will also demonstrate that this approach reflects a growing trend on commercial music software that has already begun to explore and implement mediated interfaces and intelligent music algorithms.

Resumo

O objetivo central desta dissertação é o estudo e desenvolvimento de algoritmos e interfaces mediados por computador para performance e criação musical. É sobretudo centrado em acompanhamentos em Jazz clássico e explora um meta-controlo dos parâmetros musicais como forma de potenciar a experiência de tocar Jazz por músicos e não-músicos, quer individual quer coletivamente.

Pretende contribuir para a pesquisa existente nas áreas de geração automática de música e de interfaces para expressão musical, apresentando um conjunto de algoritmos e interfaces de controlo especialmente criados para esta dissertação. Estes algoritmos e interfaces implementam processos inteligentes e musicalmente informados, para gerar eventos musicais sofisticados e corretos musical estilisticamente, de forma automática, a partir de um input simplificado e intuitivo do utilizador, e de forma coerente gerir a experiência de grupo, estabelecendo um controlo integrado sobre os parâmetros globais.

A partir destes algoritmos, são apresentadas propostas para diferentes aplicações dos conceitos e técnicas, de forma a ilustrar os benefícios e potencial da utilização de um meta-controlo como extensão dos paradigmas existentes para aplicações musicais, assim como potenciar a criação de novos. Estas aplicações abordam principalmente três áreas onde a música mediada por computador pode trazer grandes benefícios, nomeadamente a performance, a criação e a educação.

Uma aplicação, *PocketBand*, implementada no ambiente de programação Max, permite a um grupo de utilizadores tocarem em grupo como uma banda de jazz, quer sejam ou não treinados musicalmente, cada um utilizando um teclado de computador

ou um dispositivo iOS multitoque. O segundo protótipo visa a utilização em contextos coletivos e participativos. Trata-se de uma instalação para vários utilizadores, para ecrã multitoque, intitulada *MyJazzBand*, que permite até quatro utilizadores tocarem juntos como membros de uma banda de jazz virtual.

Ambas as aplicações permitem que os utilizadores experienciem e participem de forma eficaz como músicos de jazz, quer sejam ou não músicos profissionais. As aplicações podem ser utilizadas para fins educativos, seja como um sistema de acompanhamento automático em tempo real para qualquer instrumentista ou cantor, seja como uma fonte de informação para procedimentos harmónicos, ou como uma ferramenta prática para criar esboços ou conteúdos para aulas.

Irei também demonstrar que esta abordagem reflete uma tendência crescente entre as empresas de software musical comercial, que já começaram a explorar a mediação por computador e algoritmos musicais inteligentes.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. A guiding hand

Since its early stages, the computer has seen an unimaginable evolution, and its influence on the course of human history is unmatched by any other human invention or endeavor. In around sixty years, its power has grown so much that even the simpler computerized device today is far more powerful than the most powerful mainframe computer in the 1950's. But along with its power, the importance of the computer is even more relevant observing its role. Starting basically as a device for complex calculus, the computer has now much wider uses and permeated modern living, reaching almost all kinds of areas and applications, and creating many new ones.

As it is more and more embedded in modern life, the computer is becoming invisible. In most of its applications, it acts discretely behind the scenes, combining any number of different technologies and algorithms to provide the results to the end user. In a modern car, for example, as the driver hits the breaks, many different computations act to manage all the necessary systems to effectively render a much more effective and safe braking response. In the same way, a great number of state of the art technology, infrastructures and algorithms are triggered to provide the user with a meaningful reply to an apparently simple question like "where is the nearest gas station", to a virtual assistant in a mobile device that we carry in our pockets. In such applications, the computer acts as a mediator between the user and the background systems, to provide the intended results.

This dissertation focuses mainly on this idea of the computer as a mediator, to potentiate the creation and development of systems of automatic music generation and user interfaces for musical performance. As such, computer mediation in this dissertation refers to the existence of a middle layer between the user input and the resulting musical events. This layer comprises a set of procedures and data that constitute a corpus of musical knowledge that converts user input into significant, relevant musical output. Unlike a traditional acoustic musical instrument, where every sound produced is the result of a direct action of the player, a mediated musical system introduces process for data manipulation upon the user action, that will output a computed result, based on rules or constraints. The result is thus indirect, in the sense that there is a non-linear correspondence between the input control and the output.

1.2. Objectives

This dissertation addresses the use of computer mediation and algorithm development for music creation and performance. The main goal is to add to existing research on automatic music generation by exploring the use of high-level control strategies in the development of expressive music interfaces and platforms for collective music performance.

The work is centered on traditional jazz music, using a corpus of standard concepts and techniques from jazz theory and practice, and focuses mainly on jazz accompaniment techniques. A set of specially developed instrument algorithms were designed and implemented from the ground up to form a coherent platform for multi-user collaborative performances as well as individual human-computer performance or music production in a software sequencer.

My previous experiences in the study and development of interactive music systems and digital musical interfaces, together with the information gathered from existing literature during the initial period of this research, led me to concentrate on the potential of computer mediation, focusing on the following main questions:

1) How can computer mediation be used to devise new musical interfaces that potentiate the creation of rich musical results using simple and intuitive input methods?
2) How to implement instrument algorithms that mimic the behavior and output of human jazz musicians by truly generative procedures?

Development of question 1:

While the use of the computer in music has a long history, its use is in many cases that of a passive tool to execute linear tasks, no matter how complex, in response to user commands. As the computer integrates some type of cognitive stage to interpret input data to elaborate a dynamic response, it becomes a mediator with an active role, having a direct influence on the quality of the results as well as the type of uses that it can promote. The use of the computer as a musical interface is very often based on a virtual representation of the same interface and instrument paradigms of the physical instruments. This, however, is a limited approach to the potential of the digital instrument. As the input control interface and the intended sound properties are not interdependent, the same sound engine can have any number of different control interfaces. Because of this, the overall possibilities, quality and usability of the digital instrument will be dependent on both these components. As such, the digital instrument can potentially be created free from any physical bounds, and the interface can be planned to be more suitable to the type of usage for which it is created.

By implementing an “intelligent” layer between the user input and the sound engine, the computer can act as a mediator, and enhance the input, to maximize the output. But what type of factors and criteria will best contribute to this layer? What type of action will be the best to provide an intuitive and satisfying result to the user, while producing convincing and stylistically correct musical results?

In this dissertation, I will present my approach on the development of computer mediated instruments. This approach relies on the idea that this mediation has not only the potential to improve the quality of the interaction between the users and the digital instrument, but also has the potential to create new musical experiences that can be more effective and inclusive.

The experience of group playing is one of the most exciting and gratifying in musical practice, independently of the style or area. Extrapolating the concepts devised for the previous questions, in this dissertation I will explore the potential of the mediated interface for the creation of multi-user systems, in order to create new collective musical experiences that will allow musicians and non-musicians alike to experience the thrilling sensation of playing an instrument in a jazz band.

Such a system should integrate not only the different individual components, but should also account for the global aspects that are implicit in a musical band, which comprehend the live synchronization and communication between the several participants.

<u>Development of question 2:</u>

As will be demonstrated in Chapter 4, most of the accompaniment systems developed so far are based on the recombination of pre-recorded patterns or phrases, whether MIDI or audio. Although they can sound quite sophisticated, recombination algorithms only provide optimum results when the intended states match the same exact conditions as the ones contemplated in the stored patterns, but not so convincing results when they don't. Most of the times this will be the case, and the stitching of the phrases tend to be noticeably artificial. Also, due to the limited number of phrases that can be used at any given moment, the results tend to be too repetitive.

Just as a good improvisation isn't entirely built by the combination of patterns, the computer should also be able to build its own content on the fly, making decisions that are based on a corpus of musical knowledge, and influenced by external input by the other musicians in the band during performance. An algorithmic approach to this problem should potentially be more effective, by calculating and generating the musical events, whether rhythm, notes and/or chords in real time.

1.3. Motivation

The basis for this study was the work developed as a researcher at the Sound and Music Computing group at INESC-Porto, in the project "Kinetic: Gestural controller-driven, adaptive, and dynamic music composition systems"[1], between November 2009 and December 2011. This project, led by Professors Carlos Guedes (IPP/UP), Tomás

---

[1] "Kinetic controller, driven, adaptive and dynamic music composition systems" funded by the ERDF through the Program COMPETE, by the Portuguese Foundation for Science and Technology (FCT), Project ref. FCOMP-01- 0124-FEDER-011414, UTAustin/CD/0052/2008.

Henriques (UNL) and Bruce Pennycook (UT Austin), aimed the study of automatic music generation algorithms and gestural control strategies for musical expression.

The work developed in this project was seminal for the present dissertation. As a researcher, my main contribution was in the development of computer algorithms for the automatic generation of jazz music, used for the development of an iOS app called *GimmeDaBlues* (Dias, Marques, Sioros, & Guedes, 2012). This application in some ways reflected some of the core ideas of the whole project, namely the commitment to have musically valid output, together with a natural and intuitive control over the algorithmic processes. This resulted in an apparent simplicity that was common to all the projects here developed. Because interaction, gestural control and natural user interfaces were some of the recurrent topics across most of the discussions around the projects being developed in the group, this led me to also direct more of the project into an interactive, real-time controlled system.

With the *momentum* and experience gained in the Kinetic project, many ideas emerged to further develop the concepts and algorithms related to the computer generation of jazz music. The main concept that I believe emerged from the research was that of the concept of an “intelligent” middle-layer that allowed the transformation of simple user input into sometimes rather complex musical output. Extrapolating this to a broader perspective, I believe this concept can play a very significant role in the development of new paradigms for computer-mediated musical interfaces and computer music generation algorithms that can be applied in virtually every music application, whether for music creation, performance or education.

Having had the opportunity to play *GimmeDaBlues* on stage with live jazz musicians, I realized that it can be surprisingly engaging and effective in transmitting the sensation of playing in a band. Even more, it was extremely motivating to see how people with little or no musical training at all reacted to the experience of playing without having to think about notes or chords, and how it would be interesting to explore this in a collective experience.

1.4. Organization of the text

The contents in this dissertation were planned to clearly present the context, key concepts and results of the work developed during the research and implementation stages. Some sections present text that is a direct transcript, or adapted from the original scientific papers elaborated during the research. The text is organized in eight chapters, comprising an introduction to the main concerns, goals and motivations that led to the development of this research subject. Two main parts where the background and work are presented, and a final chapter, dedicated to a discussion of the work and conclusions.

Part I, comprising chapters 2, 3 and 4, will present the subjects and areas of study that more closely relate to the present dissertation, in order to provide a conceptual, historical and technical background.

Chapter 2: "The Computer as a Performing Tool", is a reflection on the musical interface, the characterization of the digital instrument and the concept of the meta-instrument. This chapter also provides a contextual introduction to computer music background, in both an historical perspective on algorithmic composition, interactive music, and the modeling of musical style. Chapter 3 is a contextualization of jazz music practice and idiosyncrasies, focusing on the main procedures and techniques that define traditional jazz, namely the ones that were addressed and implemented in the prototypes developed for this dissertation. Chapter 4 describes some relevant examples from computer music research in the field of automatic generation of jazz music. Section 4.5 is dedicated to *GimmeDaBlues*, an iOS application that was the starting point for this study.

Part II – Towards a Computer-Mediated Musical Experience - presents the paradigms and prototypes that were developed during this research.

Chapter 5 describes in detail the algorithms that were created and developed for the implementation of the described instruments. Chapter 6 presents the main components and concepts that were developed and form the core of this dissertation. Chapter 7 introduces two application prototypes that were created from the main components and instrument algorithms. The conclusion, in chapter 8, presents a reflection and synthesis of the work described in the dissertation, and includes some proposals for the further development of the concepts and applications.

The Appendices include the style-sheet format examples used in *GimmeDaBlues* and Pocket Band.

PART I - BACKGROUND AND CONTEXT

Chapter 2. The computer as a performing tool

This chapter provides an insight on related areas of research that somehow were directly relevant to the subjects of this dissertation, not intended as a comprehensive guide to these areas, but with the purpose of serving as an introductory reference, in order to contextualize and situate the topics that will be addressed later. This chapter will also present an overview over some of the existing software and musical interfaces that use some type of mediation process between the user input and the output, showing that there is a growing number of uses for this kind of approach.

2.1. Algorithmic Music

The relation between music and formalization processes is inherent to the nature of music itself both as a physical and as a cognitive phenomenon, as can be observed by the acoustic and psychoacoustic sciences. The term "algorithmic composition" designates the adoption of a formal, algorithmic process to obtain and/or develop music material.

An algorithm is "a set of rules or a sequence of operations designed to accomplish some task or solve a problem" (Simoni & Dannenberg, 2013, p.4). By definition, a classical algorithm must have five important characteristics (adapted from Loy, 2006, p.288):

1) Finiteness: the algorithm must have a finite number of steps;

2) Definiteness: each step must be clearly defined;

3) Input: the input must be valid;

4) Output: the algorithm must produce at least one result;

5) Effectiveness: the algorithm must produce the same output for the same input. It cannot be ambiguous and cannot depend upon unknown factors.

In many cases, an algorithm may include some type of indeterministic stage(s) in order to generate variable results from the same input data. This has several uses in algorithmic composition using stochastic or generative procedures, where the use of probabilistic algorithms allows the generation of new musical events within user defined parameters and ranges. This is the case of many applications of algorithmic music procedures where variable, unpredictable results is desired or where it is necessary to produce output indefinitely (Nierhaus, 2009, p.2), as may be the case of a program to continuously generate new music for an installation, for example.

Examples of music formalization can be found many centuries before the twentieth century and the invention of the computer. Pre-computer formal composition processes in music history include Guido d'Arezzo's solmization syllables, a system of association of the scale notes to syllables and to parts of the hand, in what is called the *Guidonian hand* (Loy, 2006, p.286), isorhythmic motets by Guillaume de Machaut and Phillipe de Vitry, contrapuntal procedures in the baroque fugue including J.S. Bach's use of the letters of his name to compose the subject of the last fugue of *The Art of the Fugue*, musical dice games, including Mozart's (Cope, 2001, p.159), and tone row transformations in serial composition (Nierhaus, 2009), amongst many others.

2.1.1 The computer as a composition assisting tool

Although algorithmic processes do not necessarily imply the use of computers, the computer-driven capabilities in the development of formal and mathematical processes are overwhelmingly greater and faster, and thus the designation usually refers to a computer-aided process. The implementation of computer algorithms for music composition began with Lejaren Hiller and Leonard Isaacson in 1955-56 in the composition of the “Illiac Suite”, a suite for string quartet with four movements, called *Experiments*, where each movement is the result of the application of one or more distinct processes (see section 2.1.2 for more detail). As the Illiac computer wasn’t able to output sound or musical notation, the resulting musical events in alphanumeric data format were then transcribed by hand to standard music notation in a conventional score to be performed by acoustic instruments (Manning, 2004). This approach to the use of the computer for offline music process calculations was also the beginning of Computer-Assisted Composition (CAC). In this approach, the use of the computer as a compositional tool became widely used, as it allowed for high-speed calculations that could otherwise be impossible or very time-consuming.

Composer Iannis Xenakis used mathematical probability functions to obtain the music materials for his pieces. For the piece *Metastaseis*, for orchestra, premiered in 1955, based on stochastic models, all the calculations were realized by hand (Roads, 1996, p.831). From 1962, Xenakis composed the “ST pieces” - ST/48-1, ST10/1, ST10/2 and ST4/1 (Harley, 2004, p.25), using an IBM 7090 computer, using the Fortran programming language to implement an algorithm that describes stochastic processes with probability functions with various constraints applied according to compositional parameters like instrumentation, range or dynamics. He created the *Stochastic Music Program* (SMP), adapting stochastic algorithms originally developed by scientists to describe the behavior of particles in gases, in which the composer introduces the global attributes of the score and runs the program (Roads, 1996, p.836). The numerical output of the program was then transcribed to conventional musical notation. The concepts and processes he used were described in his book “Formalized Music” (Xenakis, 1992) and include probability laws, stochastics, Markovian chains, game theory, group theory, set theory, Boolean algebra, and Gaussian distributions as formalizations (Cope, 2001, p.175).

The composer Gottfried Michael Koenig was interested in using the computer as a way to extend serial techniques with random-based operations. He saw the serial composition technique as “a special case of aleatoric compositional technique” (Roads, 1996, p.840), and “realized that [in a series] what comes next is not necessarily the cause of that which follows. And I realized I could replace a series with random numbers.” (Koenig, in Chadabe, 1997, p.281). Starting in 1964, Koenig implemented a series of procedures in his program *Project 1* (*PR1*) ranging from completely random (indeterministic) to completely deterministic (repetitive) (Roads, 1996, p.839). With this program, Koenig could generate sections, or “structures”, for which he could define parameters ‘Instrument’, ‘Pitch’, ‘Octave’, ‘Loudness’ and ‘Entry delay’, along with the selection of the six “degrees of unpredictability” (Chadabe, 1997, p.281). As in Hiller and Xenakis programs, Project 1 would calculate the results and output as numerical data, that would be transcribed for conventional music notation by the composer. Koenig saw the process of transcribing the printout into a musical score as an important interpretative task for the composer (Roads, 1996, p.844). From 1968 he started developing *Project 2* (*PR2*). While Project 1 was a personal tool for his own compositions, Project 2 is more directed towards a general use by other composers at the Institute (Manning, 2004, p. 203). As such, it incorporated more flexible palette of statistical procedures in the form of subroutines like ‘ALEA’, ‘SERIES’ or ‘TENDENCY’, which users can patch together in different combinations (Ames, 1987).

Some programs may also generate the output directly as a musical score, or allow exporting the data in formats that can be opened in a sequencer or dedicated musical notation software. In this approach to the use of the computer, there is a clear separation between the composition process and the performance. Also, there can be any arbitrary adjustments to the results obtained with the computer until the final score that will be performed. The use of the computer calculate the musical parameters can range from small parts to entire sections and even entire pieces. One of the innovations that were crucial to the development of music composition software was the MIDI protocol (https://www.midi.org). Introduced in 1983, as it became a standard for hardware, and so it happened for music software. Because it is a symbolic language that represents the musical events but doesn't contain the actual sound itself, much like the traditional music notation, MIDI was a very efficient choice, computation-wise, and very easy for musicians to understand and use.

By the end of the 1980's computers had become accessible and the integration of multimedia (graphics and sound) possibilities made them much more accessible to a multitude of users. While in the United States, computer music research focused more on sound synthesis and processing, in Europe the composers and researchers were focusing more on computer-aided composition (Puckette, 2006). At the IRCAM, in Paris, computer engineers, working closely with composers, developed composition software that allowed the composers, even non-programmers, to create their own algorithms for musical data manipulation. Around 1990, Mikael Laurson created the first version of *Patchwork*. Patchwork was programmed in the LISP programming language, and was basically a graphical interface for a group of functions and algorithms developed at the IRCAM in the previous years, like the program *Esquisse*, from 1988

(Chadabe, 1997, p.207). Patchwork had some very important qualities. Namely, it was much easier for most composers to program in a graphical programming environment than in a text-based language. It was modular and expansible, meaning that composers or programmers could implement their own routines, and create their own libraries. It allowed composers to see the calculation results directly as traditional notation on the screen. These and other features made patchwork a successful program amongst avant-garde composers and researchers, and defined a standard that is still in use today. Composers like Kaija Saariaho, Magnus Lindberg, Tristan Murail and Gèrard Grisey, amongst many others, worked extensively with CAC software, and contributed to the software's development and the creation of new libraries. Patchwork was replaced as the leading CAC software by the software Open Music (Agon, Assayag, Laurson, & Rueda, n.d.). In 2002, Laurson, with Mika Kuuskankare, launched the software PWGL (Laurson & Kuuskankare, 2002), a direct successor of Patchwork, including the PatchWork libraries and integrating more sophisticated notation tools with Kuuskankare's ENP - Expressive Notation Package.

Paul Berg, at the Institute of Sonology in The Hague, developed the LISP-based software AC Toolbox (Berg, n.d.). This software is a direct descendant of the PR1 and PR2 programs by Koenig and implements his random-based algorithms. Unlike Patchwork or Open Music, AC Toolbox is not a graphical language. It is a code-based programming language, but the several functions and resources are presented in dedicated windows with the appropriate fields for the user to fill in with the desired values. The output is generated as MIDI data that the user can hear or visualize inside AC Toolbox or export as standard MIDI files. Amongst other features, the program includes also dedicated functionalities to directly create CSound score files.

Other software for computer assisted composition include AthenaCL (Ariza, 2005), and Bach and Cage libraries for Max/MSP (Agostini & Ghisi, 2015).

2.1.2 Modeling of Musical Style

A specific area of computer music research deals with the development of programs and algorithms to mimic a given musical style. These programs implement existing features and rules that characterize a specific music style, in order to generate new music in that style.

> "Virtual Music represents a broad category of machine-created composition which attempts to replicate the style but not the actual notes of existing music" (Cope, 2004, p.3).

Hiller's "Illiac Suite", mentioned in section 2.1.1, can also be considered the first example of computer music modeling. For *Experiment One*, Hiller defined a set of rules to create an algorithm for melodic generation in the style of the Strict Counterpoint of the sixteenth century, as formalized and codified by J. J. Fux in *Gradus ad Parnassum*, originally published in 1725 (Mann & Edmunds, 1965). These rules are applied in the generation of monody, two-part and four-part counterpoint. *Experiment Two* deals only with four-part first-species counterpoint. However, the counterpoint rules are applied to random white-note music successively, so the music result starts with total random, and gradually gets restricted to the counterpoint rules. *Experiment Three* deals with several distinct processes to generate music with the chromatic scale, including twelve-tone rows. With *Experiment Four*, Hiller explores the use of probabilistic methods, by implementing Markov chains of different orders. With these experiments, Hiller and

Isaacson proved that the computer could model any formal procedure (Roads, 1996, p.834).

As the computer became more accessible and widespread, more researchers and composers were able to develop their own approaches to music modeling that emulate a given style or even a specific composer. William Schottstaedt's "Counterpoint solver" (Schottstaedt, 1989), like Hiller, created a system that used the counterpoint species rules by Fux to generate correct melodic counterpoint with up to six parts, in all of the five species. Kernal Ebcioglu wrote CHORAL, a program that can generate four-part chorales in the style of J. S. Bach (Ebcioglu, 1990). Ulf Berggren created a system for the generation of sonata movements by recombining excerpts of Mozart sonatas (Berggren, 1995).

Experiments on Music Intelligence (EMI), by composer and researcher David Cope, is a broad and thorough music modeling system that does not focus a particular composer or style but instead can virtually generate music in any style. Although the system was created as a personal tool for assisting in Cope's own composition process in 1984, it soon developed as an open system that could be used by other composers or that could generate music in any given style (Cope, 1987). EMI is based on natural language processing and generative grammars introduced by (Chomsky, 1957). It applies an Augmented Transition Network (ATN) parser to create logical musical phrases in a similar way to language. Using customizable dictionaries and a set of user-defined rules, the system can be programmed to produce music in different styles. Cope later developed Emily Howell, a computer program that uses EMI as a source and receives external feedback from a user or audience, to influence the decisions of the generation process (Cope, 2005). Emmy (short for Emily Howell) has generated countless music pieces in the style of several composers like Bach, Beethoven and Chopin. In his website, Cope has a collection of 5000 chorales in the style of J. S. Bach, available for download as MIDI files (Cope, n.d.).

The “Continuator” (Pachet, 2002), by the French researcher and musician François Pachet, is a dynamic system based on Markov models, which listens to the input of a human performer, and generates esthetically related music as an improviser would, responding in real-time to the human input, and developing the musical materials.

2.2. Interactive Music

“Interactive music” is a broad term for a compositional approach for music that includes elements that will be altered interactively during live performance, usually by the use of a computer system. Todd Winkler defines interactive music as “a music composition or improvisation where software interprets a live performance to affect music generated or modified by computers” (Winkler, 2001). Robert Rowe defines interactive computer music systems as “those whose behavior changes in response to musical input. Such responsiveness allows these systems to participate in live performances, of both notated and improvised music” (Rowe, 1993, p.1). David Cope states that “Interactive composition occurs when humans and computers collaborate during composition” (in Rowe 93).

Interactive systems have also been widely used to create and explore completely new musical formats, using technology as a way to interconnect with other areas. *Heartbeat* (1976) by Peter Beyls used electro-cardiogram signals from a performer’s body (in Chadabe 1997, p.305). In “Food Opera” (Houge, 2014) the music is generated according to the type of ingredients and pace of the customers dinning in a restaurant.

Since the early collaborations between John Cage, David Tudor and Gordon Mumma with the dancer and choreographer Merce Cunningham, interaction has brought new possibilities for contemporary dance. A dance performance with live improvisation is, in fact, a good example of an interactive “system”. The dancer can follow the music, but simultaneously, the dancers influence the musicians. Examples of musical systems specifically developed for interactive dance include Isadora software by Mark Coniglio (http://troikatronix.com) and M-Objects, by Carlos Guedes (Guedes, 2005a, 2005b, 2006).

2.2.1 Analogue interactive music systems

In 1981, composer Joel Chadabe coined the term *interactive composing* to describe “a performance process wherein a performer shares control of the music by interacting with a musical instrument” (Chadabe, 1997, p.293). Back in 1966, Chadabe had created the CEMS (Coordinated Electronic Music Studio) – an automated synthesizer system built by Robert Moog, which comprised a set of five interlinked modular Moog systems with an extensive number of sound generators, processors, and sequencers. Using this system, Chadabe was able to create music in which composition, performance and improvisation intertwine, as does the roles between the human and the system.

> “Because I was sharing control of the music with the sequencers, I was only partially controlling the music, and the music, consequently, contained surprising as well as predictable elements. The surprising elements made me react. The predictable elements made me feel that I was exerting some control. It was like conversing with a clever friend who was never boring but always responsive. It was, in effect, convers-

> ing with a musical instrument that seemed to have its own interesting personality." (Chadabe, 1997, p.287)

In this approach, a composition is defined as a combination of processes that include some level of indeterminacy, thus the final result will be potentially different for each performance, and the system itself can be seen like a sort of instrument, built or programmed for each piece. The composer/performer creates the system and plays the instrument in real time with a high-level control over the available processes. Chadabe referred to this process as *design-then-do* (Chadabe, 1984), in which the *design* stage is the specification of the system, defining the processes and settings, including the definition of the nature of the performer's interactions, and the *do* stage, the performance of the piece playing the designed system, to which Chadabe compared to "sailing a boat on a windy day and through stormy seas".

This idea of composing music and playing "instruments" that include a certain degree of indeterminacy was key to these and future developments in electronic and computer music. Around the same time, composer Salvatore Martirano and engineer Sergio Franco, at the University of Illinois, built the *SalMar Construction* (Chadabe, 1997, p.288; Roads, 1996, p.828), planned for stage performance. "Control was an illusion. But I was in the loop. I was trading swaps with the logic. I enabled paths. Or better, I steered. It was like driving a bus."(Chadabe, 1997, p.291). Raymond Scott, composer and inventor of the *Electronium* electronic instrument, referred "The Electronium is not played, it is guided" (Roads, 1996, p.828). Composer Iannis Xenakis wrote "With the aid of electronic computers, the composer becomes a sort of pilot: pressing buttons, introducing coordinates, and supervising the controls of a cosmic vessel sailing in the space of sound, across sonic constellations and galaxies that could formerly be glimpsed only in a distant dream." (Xenakis, 1992, p.144).

2.2.2 Computer-based systems

With the use of the computer, the possibilities for interactive music systems developed exponentially. In 1967 at Bell Labs, Max Mathews and F. Richard Moore started the development of GROOVE (Generated Realtime Operations on Voltage-Controlled Equipment), a hybrid system that combined a digital computer with a modular analogue synthesizer. This system was planned for real time operation, and featured a conducting program that enabled a person to control the tempo, dynamic level and balance of a computer ensemble that had knowledge of a predetermined musical score (Winkler, 2001, p.13).

In 1968, Peter Zinoviev and his associates from the Electronic Music Studios of London presented one of the first computer-based interactive improvisation systems at the Institute for Contemporary Art, in London (Roads, 1996, p.685).

The use of computers was also explored for group performance with the first appearance in 1976 of the "League of Automatic Music Composers", at the Exploratorium in San Francisco (Chadabe, 1997, p295). "The League" was formed by composers Jim Horton, John Bischoff and Rich Gold, all former students at the Mills College, to explore the possibilities of networked computers for musical performance. After many concerts between 1976 and 1983, the group stopped, mainly due to health reasons of Horton. In 1985, Bischoff and Perkis started "The Hub", together with composers Chris Brown, Scot Gresham-Lancaster, Phil Stone and Mark Trayle. Each musician had its own custom-built hardware and software, and were interconnected through a central microcomputer – the Hub itself (Cope, 2001, p.171) built by Perkis and Bischoff in 1984.

"The League" and "The Hub" were pioneers in the use of computers in networked and collective environments, opening up a world of new possibilities for computer music composition and performance. In such an environment, "each piece defines ways of interacting" (Brown in Chadabe, 1997, p.297). As Bischoff remembers, "Each musician interacts with a computer that in turn interacts with other computers to create an ensemble performance" (Roads, 1996, p.685). "The emphasis was on the network being a shared instrument" (in Chadabe, 1997, p.297).

In the mid 1980's and 1990's, music software saw a huge development, not just in music research centers and universities, but also by private software companies that begun to develop tools for the home and studio musician. The growing accessibility of the personal computer allowed several composers and researchers to explore the possibilities of computers in interactive music, and address some of the challenges associated with real time communication between human musicians and computers.

Score-following

One of the lines of research that emerged in this period was "score-following", in which the computer follows a human performer, by comparing in real time the musician's live performance, with a pre-stored score. Ideally, this technique allows the creation of automatic accompaniment systems that can adapt to the player during live performance, as opposed to the player having to follow a fixed music track. Such systems can be used for example in applications for music learning, where the instrument or singing student can practice alone, while being dynamically accompanied by the computer. The system can also analyze the student's performance and provide statistical data such as the number of wrong notes or tempo deviations. Score-following has also been used in interactive music pieces, where the computer automatically triggers events like audio files or effects parameter changes, as the player advances in the piece.

Three main problems in score-following are: how to schedule musical events in real time, how to get a computer to follow a score, and how to get a computer to recognize musical input (Winkler, 2001, p.15). Working independently, composers/researchers Barry Vercoe, working at the IRCAM in Paris, and Roger Dannenberg at Carnegie Mellon University, both addressed these problems and presented their findings at the International Computer Music Conference in 1984. Vercoe presented the concept of a "synthetic performer" (Vercoe, 1984), a computer model that could "replace any member in a group so that the remaining live members cannot tell the difference". By using sensors on a conventional flute, he was able to retrieve the musical events as symbolic data that would then be used to track the position in the score in order to synchronize the reproduction of a stored part. Dannenberg described a general algorithm for searching and matching the performance input data from a MIDI keyboard with the expected input, and the concept of a "virtual time", using two computers to implement a real-time accompaniment system (Dannenberg, 1984). In the following implementation, Dannenberg used a tracking system with a trumpet. In 1991, composer Bruce Pennycook developed a tracking system for clarinet, to use in the piece "Praescio IV" (Roads, 1996, p.682).

Score-following has since then been continuously developed. At IRCAM, several developments of the initial synthetic performer by Vercoe have been introduced by the

IRTM - IRCAM Real-Time Musical Interactions team, recently changed to ISMM – IRCAM Sound Music Movement team, who developed the current state-of-the-art score-following software *Antescofo* (Cont, 2008).

An important part of the algorithms and prototypes developed for this dissertation is also dedicated to automatic accompaniments, as they produce drums and bass lines algorithmically. However, while a score-following system follows a player in order to adapt the tempo of a previously programmed score, the systems I will present are based on the improvisation practice of jazz music, and as such the notes are generated in real time, by trying to emulate the way a musician thinks while improvising. In addition, although there is a data collector that analyzes the events that the players produce (the Listener), it is restricted to the midi events played on the virtual, touch-based instrument interfaces. As such, it is currently not capable of tracking an audio signal and analyzing the notes of an external acoustic instrument, although it can receive OSC or MIDI messages from external audio analysis software. It would certainly be very interesting in future research to explore this feature in order to allow the system to accompany live musicians and respond accordingly.

Real time computer music software

Other applications were developed that explore the computer as a real time tool. Unlike the above-mentioned software, these applications have to deal with musical time scheduling in order to generate musical events in real time, which allow the composer to hear the results immediately as he changes the algorithm’s parameters. While this approach can be related to the 1967 GROOVE system, by Max Mathews and F. Richard Moore at the Bell Labs, this type of software programs appeared by the mideighties, with the advent of the personal computer, and the MIDI protocol, to send midi events to external hardware synthesizers.

The 1985 software *Music Mouse* (http://musicmouse.com), by composer and programmer Laurie Spiegel, was one of the first examples of software intended for improvisation and more general use (Dean, 2003, p.62). The player could improvise by moving the mouse with one hand, while making parameter selections with the other on the qwerty computer keyboard. The software would then output MIDI messages that could play any MIDI compatible synthesizer (Spiegel, n.d.).

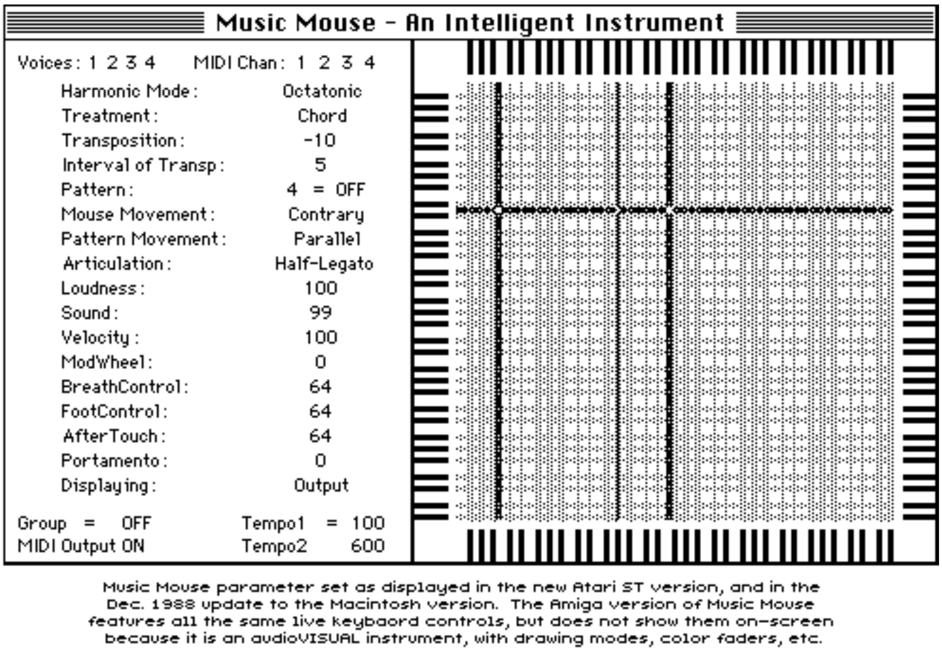


Fig. 1 *Music Mouse*, by Laurie Spiegel. Version for the Macintosh computer.

One of the more interesting aspects of *Music Mouse* was the embedded musical knowledge about the relations between scales, modes and chords that were taken care of automatically by the computer. It worked by applying stylistic constraints using this musical knowledge, which would conform the mouse position and movements to adequate notes of the harmony, according to the parameter selections made with the keyboard. The available parameters included:

- Type of harmony, selecting between: Equal tempered scale; Octatonic mode; Middle Eastern scale (non-microtonal approximation); Diatonic scale; Pentatonic scale; and Cycle of fourths.
- Transposition;
- Addition of Melodic Patterning;
- Voicing;
- Loudness, Muting, Articulation;
- Tempo;
- Rhythmic Treatments;
- Internal sound selection;
- MIDI sound selection and control;

The TAB key allows pitch quantization defeat, for microtonal frequency space.

Using some combinations of parameters, *Music Mouse* could also be used as a generative system. However, the generated music is meant for human interaction, and as such, it isn't able to autonomously control the overall compositional form or the parameter settings.

Other software programs using similar paradigms appeared, like the MIDI Grid (Hunt & Kirk, 2003), MousMuso ("MousMuso," n.d.) or FlexiMusic ("FlexiMusic," n.d.). *In 1986,* the company *Intelligent Music*, founded by composer Joel Chadabe, released the *M* and *Jam Factory* software packages, by Joel Chadabe, David Zicarelly, John Offenharts and Anton Widoff. Both programs deal with the automatic generation of musical events, implementing random functions (Cope, 2001, p.168) that can be controlled in real-time. They were based on Chadabe's concept of "interactive composing", and allow the user to configure his/her own set of parameters and control them in real-time. The graphic interface is planned in order to have all the sections of the program always visible and accessible, so that the user can always modify the composing "machine" during performance, blurring the distinction between the act of making the machine and using it (Zicarelli, 1987).

In 1986, Robert Rowe, working in his piece *Hall of Mirrors*, for bass clarinet and computer, noticed that while the clarinet player was reacting to what the computer was doing, the computer lacked the listening skills to react to the clarinet. In 1987 he started working on *Cypher*, (Chadabe, 1997, p.314), an interactive computer music system (Rowe, 1993, p.39) that would be capable of listening and analyzing musical input, in order to understand what was happening musically. Cypher consisted in two main elements: an analysis section – the *Listener* - and a composition section - the *Player*. The Listener was the central element and could characterize performance input by analyzing a data stream of MIDI messages in real time, attending to parameters like register, dynamics, vertical density, horizontal density, and articulation. This information was then sent to the player, which generated musical response using several algorithmic styles.

Autobusk (Barlow, n.d.), by composer and programmer Clarence Barlow, is a program that deals with real-time probabilistic generation of musical events as MIDI messages. In the author's own words, "Autobusk itself took 272 days to write, spread between 18 August 1986 and 30 October 2000" (Barlow, 2001).

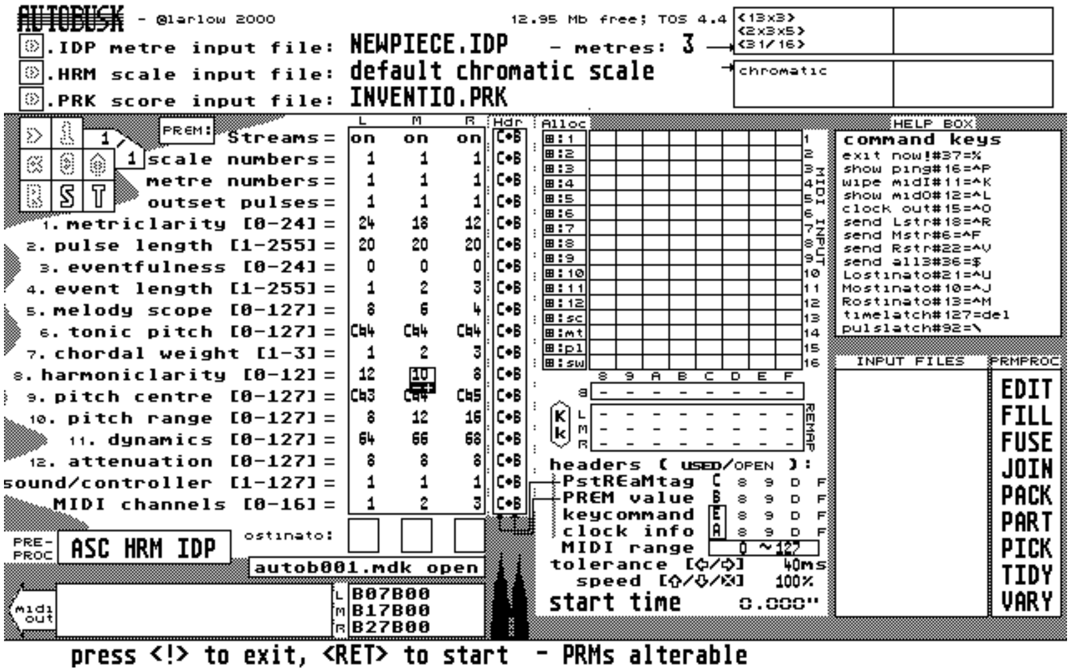


Fig. 2 Autobusk software, by composer Clarence Barlow.

Autobusk implements several algorithms developed by Barlow concerning the quantification of musical parameters, namely pitch and harmonic relations, and rhythm. Three streams of music generators named Left, Middle and Right are available, running in parallel, and comprise twelve main parameters that can be altered in real-time. Using these parameters, the program can generate a wide scope of musical textures, phrases and rhythms, whether in real-time, or to be stored in a file. Extensive options for real-time control of the parameters by MIDI input are available, as well as options for post manipulation of pre-recorded files. The thorough insight on the musical parameters and their use in a computer program are quite interesting, as well as the user interface, in which Barlow was able to condense a great number of complex parameters and file manipulation options in a single window. The theoretical background of the music algorithms implemented in Autobusk are described in (Barlow, 2008, 2012).

Musical Programming Languages for real-time midi and audio

A crucial addition to computer music was the development of musical programming languages dedicated to real-time performance. In the middle 1980's and during the 1990's, as computer and audio processing hardware became available, specific tools were developed that allow the use of the computer for real-time manipulation of performance data and sound processing.

Max, created by Miller Puckette around 1986, is a graphic programming environment for developing real-time musical applications, initially written for the Apple MacIntosh computer (Puckette, 1991). It was initially created as a real-time control system to control the 4X signal processing station at the IRCAM (Favreau et al., 1986; Puckette, 1986), and soon after the ISPW (IRCAM Signal Processing Workstation) for which Puckette created the FTS (Faster Than Sound) functions. It was developed with a graphical interface called the "Patcher" (Puckette, 1988), and ported to the NeXT computer. In 1990 it was adapted by David Zicarelli and distributed commercially by Opcode systems, and later by his own company Cycling'74 (https://cycling74.com). Max allows the creation and use of custom functions and sub-routines and provides a wide set of formats that the user can choose from, according to his/her needs. A large number of external libraries exist that vastly extend Max's already large set of native functions.

The FTS functions were the basis for the real-time signal processing operations in Pure data (https://puredata.info), a free programming environment by Puckette, very similar to Max, and later the base for the real-time signal processing library for Max, which Zicarelly named MSP (Max Signal Processing).

Max has been widely used by the community of computer musicians and developers and probably to this day the programming language with the larger community of users. Of the innumerous projects and libraries developed for Max, some are dedicated to algorithmic composition. The *Real Time Composition (RTC) library* (Essl, n.d.-b), by composer and programmer Karlheinz Essl, is a well-known set of sub-routines that facilitate the creation of algorithmic music generation programs that can be used in real-time. RTC library originated from the software created by Essl for his 1992 piece *Lexicon-Sonate* (Essl, n.d.-a) for automated piano, which is actually a computer program that generates the piece in real-time for each performance.

The *Bach library* is a real-time computer-aided composition and musical notation in Max (Agostini & Ghisi, 2015). It was based on the paradigms of Open Music and PWGL, but, as a library inside the Max, it is much more apt to be used in real-time. The Cage library extends Bach with a set of functions and interfaces for contemporary music procedures.

Other music languages for real-time operation include RTcmix (Garton, Brad; Topper, 1997), and adaptation of CMix by Paul Lansky in 1986, a direct descendant of the Music-N series (Lansky, 1990), for use in real-time. SuperCollider (McCartney, 1996) is a powerful text-based music programming language for real-time sound synthesis and processing.

Live-coding

Real-time musical programming gained an interesting expression in a performance and compositional approach called Live Coding. In live coding concerts, programming, composing and performing happen simultaneously. The programmer’s screen is usually displayed to the audience, has he/she creates and triggers the sound producing code lines. SuperCollider, Max and Pd were initially the main programs used for live coding, and are still widely used. Other languages appeared that are specially oriented for this purpose, including ChucK (Wang & Cook, 2003), Impromptu (Sorensen, 2005), IXI Lang (T. Magnusson, 2011).

Web-based music programming

Recently, web-based programming languages like HTML 5, Javascript and P5js, include powerful tools for audio and music computing, and are quickly becoming a valid platform for the development of new musical applications. Web-based applications are not platform-dependent and run directly in the browser.

2.2.3 Classifications of Interactive Music Systems

Several researchers have addressed the characterization and classification of real time and interactive music systems, attending to different aspects.

In 1985, Charles Dodge and Thomas Jerse distinguished between five modes of real-time computer music (Dodge & Jerse, 1985, p.402):

- Electronic-organ mode;
- Music-minus-one mode;
- Player-piano mode;
- Conductor mode;
- Synthetic performer mode;

Electronic-organ mode refers to the use of the computer in a very similar way to how a regular electronic organ or synthesizer would be used, with the added benefits of a wider range of sound possibilities, and features like non-standard tunings. In the *music-minus-one mode*, preset music materials are prepared in advance and recorded on disk, and retrieved during the performance, alongside with live performer(s). In *player-piano mode*, the computer runs some type of pre-recorded or pre-programmed music materials, but this time without live performers. As Dodge and Jerse refer, although this mode can sometimes be very similar to “playing a recording at the concert, it can also be different. For example, unlike a recording, the score could easily be changed from one performance to the next, either by determinate or random means” (Dodge & Jerse, 1985, p.403). The *conductor mode* is similar to the *player-piano mode*, but here the performer influences the evolution of the performance, by altering parameters of sound or processes that can alter the whole score. The *synthetic-performer mode* refers to the term used by Barry Vercoe, where the computer is controlled by the “gestures of the live performer interpreting a piece of music”.

In his book “Interactive Music Systems”, Robert Rowe proposes a classification of interactive systems (Rowe, 1993, p.6), distinguishing them according to three *dimensions*, of which each interactive system can have any combination:

1) Score-driven vs. performance-driven;
2) Transformative, generative or sequenced;
3) Instrument vs. player paradigms;

Regarding the implementation of an interactive system, Rowe distinguishes three main constitutive parts, namely *sensing*, *processing* and *response*.

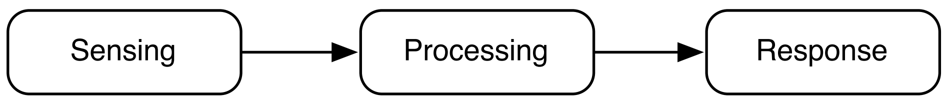


Fig. 3 Robert Rowe's three stage model of and interactive musical system.

(Image based on (Drummond, 2009))

Todd Winkler (Winkler, 2001) proposes a five-stage model (see Fig. 4), expanding that proposed by Rowe. The *sensing* stage in Rowe's model can be limitative because it suggests that every processing occurs after the sensing stage. This is not entirely true, as some types of sensing techniques involve their own processing. As such, Winkler's description adds further detail, by expanding Rowe's processing stage in three different ones: *Computer Listening*, *Interpretation*, and *Computer Composition*. The first and last stages corresponding to Rowe's Sensing and Response stages are named in Winkler's model as *Human Input* and *Sound Generation*, respectively.

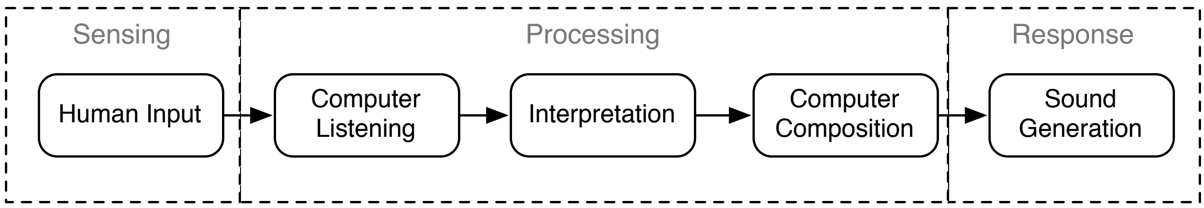


Fig. 4 Drummond (2009): comparison of Winkler's five-stage system model and Rowe's three-stage model.

These models, however, are conceptually circumscribed to a human-computer interaction paradigm, and limited by the available technological resources in the 1980's. The described models imply the compositional and performance approach in which a computer system interacts with a live musician on stage, by receiving a live input of the instrument's sound signal or control events, and somehow responding in real-time by processing the sound or triggering some type of pre-defined musical events, to which the live performer can also react to.

Other interaction possibilities have been created, however, that would be difficult to describe in these terms. In the project *Electronic Sound Creatures* by Felix Hess, (in Chadabe, 1997, p.306), the sound creatures interact with each other. In this case, there is no human input to analyze, like all the previous classifications suggest. Still, they represent the basic paradigms that were in the origin of many future developments, and provide a categorization that can be very helpful to contextualize other projects and descriptions of this type of systems.

2.3. Music Interfaces: between an instrument and a controller

> Is this [Ocarina[2]] a toy or an instrument? Maybe it's both. But for me I think the more important question is: is it expressive? (Wang, 2014)

The idea of a music interface in a broad sense can go way back to acoustic and mechanical devices, long before electric and digital technologies appeared. Considering a music interface as any kind of device to control some process of producing sound, an acoustic instrument can be seen as a music interface itself, in the sense that the type of action that the instrument requires from the player for the sound to be produced, will somehow influence and define the resulting sound. The differences between the type of sound and musical possibilities obtained from a viola or a guitar, for example, are not only related to their construction characteristics like size, form, number of strings, materials, etc., but also greatly due to the fact that in the viola the strings are rubbed with a bow while in the guitar they are plucked with the fingers or fingernails, and to the position in which the instrument is played. But while in acoustic instruments

[2] Ocarina is an application for mobile devices (Smule, 2013).

the design of the instrument is dictated by the nature of the sound generating process (Bongers, 2000), in the electronic and digital media, any parameter can be controlled with any type of input, without necessarily altering the sound results, or even usability features.

The last ten to fifteen years have been incredibly proliferous in the creation of new music interfaces. From extra knobs or sliders in keyboard MIDI controllers to unthinkable *sci-fi-ish* wireless gestural recognition, virtual controllers in multi-touch surfaces or long-distance network group performance, we assisted not only to a growth in *controllerism geekiness* but also to the birth of completely new possibilities and new ways to think about music creation and performance. A controller is by definition not an instrument, as it does not produce its own sound. However, a controller may be an element of a system that may constitute an instrument. As a controller is an input device, it defines the mode of interaction that the player will adopt. As such, the controller can be an important part in the instrument setup, and even play an important role in defining the identity of an instrument setup. The distinction between an instrument and a controller is not always clear. The main reason for this might be related to the definition of *instrument* itself.

2.3.1 Defining instrument

> "Man's music has always been acoustically limited by the instruments on which he plays. These are mechanisms which have physical restrictions. We have made sound and music directly from numbers, surmounting conventional limitations of instruments. Thus the musical universe is now circumscribed only by man's perceptions and creativity." (Mathews, Pierce, & Guttman, 1962)

Historically, the notion of musical instrument has always been tightly connected to its physical properties, whether acoustic, ergonomic and aesthetical. A musical instrument is defined by its sound - producing and propagation - properties, as well as by its performative qualities, i.e. the way it is played (which in turn is mostly determined by its sound properties) and its capabilities and limitations. The instrument's technique, repertoire and learning methodologies are developed around these properties. However, the notion of instrument has to account for the purpose, or intention, without which it can be ambiguous. Ambiguity in the notion of instrument becomes clear when any object not intended as an instrument is used as such, or when a musical instrument is used for some other purpose.

Observing the definitions of musical instrument in some reference encyclopedias may help to identify the issue. In Britannica online (n.d.), the search returns "any device for producing a musical sound". This definition, although eloquent, is quite vague. The same query in Oxford Music online (n.d.) returns "Objects or devices for producing mus. sound by mechanical energy or electrical impulses". The reference to 'objects' solves the previous problem. This seems useful because it includes many objects that can be used to make music that would difficultly be categorized as a device. Oxford also adds a reference to the sound production source, whether "by mechanical energy or electrical impulses". This doesn't seem necessary, as a musical instrument will be a musical instrument independently of its type of sound source. It seems to be there as a way to include the electric or electronic music instruments, but it probably should need a new update to include digital signals. In Grove dictionary, "'Musical instrument' (…) is less easy to apply on a worldwide scale because the notion of music itself in such a wide context escapes definition." (Grove 2008, in (Kvifte, 2011) (Kvifte, 2008)). In this case, the concept of instrument is linked to the concept of music itself. As such, it inherits the subjective notions of music in different cultures, and as such, evades a sole universal definition. In an older definition, Einrich Hornbostel states "...For purposes of research everything must count as a musical instrument with which sound can be produced intentionally" (Hornbostel, 1933). Although this might be too broad a definition, it includes a very interesting reference to intention. Intention implies a purpose. In fact, purpose is essential to this discussion. The notion of 'musical sound' itself includes purpose. Whether a sound can be considered musical or not is dependent on the intention with which it has been played, or even dependent on the intention with which it has been heard (perceived, understood). Not all sounds that can be considered musical are created intentionally. Ludvig Bielawski (1979, in Kvifte, 2011) describes the instrument as a *transformer*, transforming bodily gestures in physical time and space into musical gestures in musical time and space. In this perspective, the focus is on the action of playing the instrument and the direct action-response phenomenon. The concept of the hyperinstrument (Machover, 1992) extends this idea by using electronic sensors to retrieve input data from the gestures of the player, to augment the instrument's expressive capabilities. In this case, the instrument becomes simultaneously a controller, that can be used to trigger and control any kind of sounds or midi events.

From these definitions, it becomes evident that this is not a simple matter, and it is interesting to note the differences between the definitions. Including electronic and digital resources into the definition of instrument becomes even harder. Traditional acoustic instruments have a clear structure and role in an action-response cycle between the player and the instrument. This is not necessarily true in the electronic and digital instruments, as there might not be a direct link between the input action and the output sound.

2.3.2 Instrument formats

Based on the afore mentioned authors and existing literature, in order to distinguish between different instrument types, I will describe an instrument model, according to its three basic structural constituents, namely: Input model, Sound source and its sound Amplification and Diffusion method (Fig. 5). The input model is the way the instrument is played, or what type of interface model it uses.

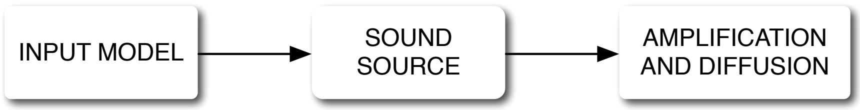


Fig. 5 Instrument model.

Observing these characteristics, the instruments can be Acoustic, Electric or electronic, Digital, or any combination of the above (e.g., electro-acoustic, electric-digital, acoustic-electric-digital).

In acoustic instruments, all the elements of the instrument are based on mechanical forces actuating to produce sound, and physical properties to modulate and amplify the sound. The amplification is taken care by a resonator, which can be for example the wooden box of a guitar body, or the body of the tube in a flute. The properties of the resonator used, like the material, thickness, size and configuration, influence both how much sound power the instrument has, and how it propagates from it. Joel Chadabe describes an acoustic instrument as a “system of three components: a controller (to call it by an electronic instrumental term), a sound generator, and a link that connects the controller to the sound generator" (Chadabe, 2002).

Electro-acoustic instruments are electrically or electronically augmented instruments that produce sound by conventional means, with an acoustic sound source, which is then amplified and usually modified by electrical means. This category includes the electric guitar and the electric piano.

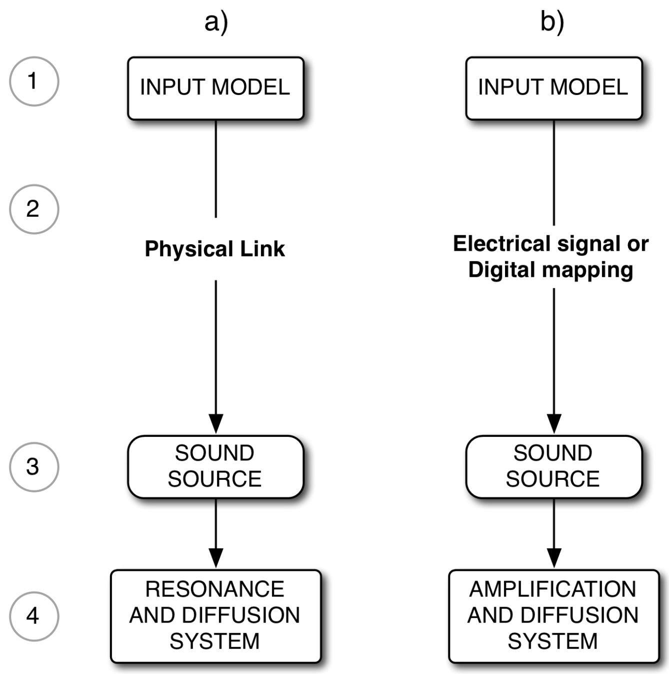


Fig. 6 Compared structure of acoustic and electronic and digital instruments.

In Electric and electronic instruments, the sound is completely produced by some type of electric or electronic circuits. These circuits produce sound by generating electrical signals with a behavior that is analogue to the behavior of acoustic sound waves. This is the case with the pioneering instruments created during the first half of the twentieth century, like the *Theremin*, the *Ondes Martenot*, or the *Hammond*, and with the many analogue synthesizers from the 1960's onwards.

Digital instruments are software or software and hardware systems that establish some type of performative entity. *Music Mouse* (section 0) is an early example of a software application that was created to be used as an instrument.

Decoupling

An important distinction of the electric, electronic and digital instruments, compared to the acoustic instruments, is that the sound-producing mechanism and control surface are decoupled (Chadabe, 2002). While in acoustic instruments, the resulting sound is a direct consequence of the action of the player, with electric, electronic and digital instruments, there are no physical restrictions to enforce this relationship. As such, there is no longer a fixed and direct correspondence between the interface and sound production mechanism (Bown, Eldridge, & McCormack, 2009). These instruments have a mapping stage (Fig. 6), in order to create the link between the player input and the intended parameters that can be of any level of complexity.

Haptic Feedback

Because of the decoupling of the sound source from the controller, the resulting sound can be completely external to the input interface and with no physical contact with the player, as is the case in many amplification systems. In fact, the two may not even be in the same physical space at all. This separation can result in a loss of feedback (Drummond, 2009), which happen naturally with acoustic instruments.

2.3.3 Computer-mediated Meta-Instruments

> The decoupling between a control interface and the process of sound production in electronic instruments, as compared to acoustic instruments, which are inherently tightly coupled (…) allows musicians to work compositionally with the design of instruments, with complex behaviours mediating control and sound output. (Bown et al., 2009)

The concept of a computer-mediated meta-instrument that I will describe here serves the purpose of this dissertation, by focusing on the idea of a performance system that uses some type of computer mediation algorithm to optimize the input of the user, to produce some type of enhanced musical results. This was a key concept when developing all the prototypes during this research, which guided the development of the instrument algorithms. By mediated, I refer to a system which has some type of algorithmic procedure that exerts any kind of dynamic process to trigger one of more events that may change, according to some criteria. This can be seen as an intermediate cognitive stage between the input and the output.

This idea relates to the notion of a reactive versus interactive system described by Bongers (2000), and to the idea of the "interactive instrument" described by Chadabe (2002), but focuses more on performance and less on the compositional potential. By compositional I mean that while using the interactive instrument, the roles of the performer and the composer are correlated, as the performer influences, even if indirectly, much of the musical contents and resulting events, defining the composition itself. His own CEMS analogue system (Chadabe, 1997, p.286) or M software (Chadabe, 1997, p.207) are clear examples where the instrument and player "shares control of the music with algorithms as virtual co-performers such that the instrument generates unpredictable information to which the performer reacts, the performer generates control information to which the instrument reacts, and the performer and instrument seem to engage in a conversation" (Chadabe, 2002). Still in this article, Chadabe describes the degree of indeterminacy of an instrument as an axis, ranging from the totally deterministic, on the left, to the totally indeterministic, on the right.

For this dissertation, the musical context is circumscribed to traditional tonal jazz. As such, the interface prototypes and algorithms were developed in order to adapt the user input to produce musical events that fit the ongoing musical context, but not to alter the musical composition itself, such as a jazz musician can improvise over a given tune, without changing the composition structure itself. In addition, the concept of interactive instrument implies a bi-directional response, where the player and the instrument are "mutually influential" (ibid.). In the case of the prototypes in this dissertation, this feature is not a central idea.

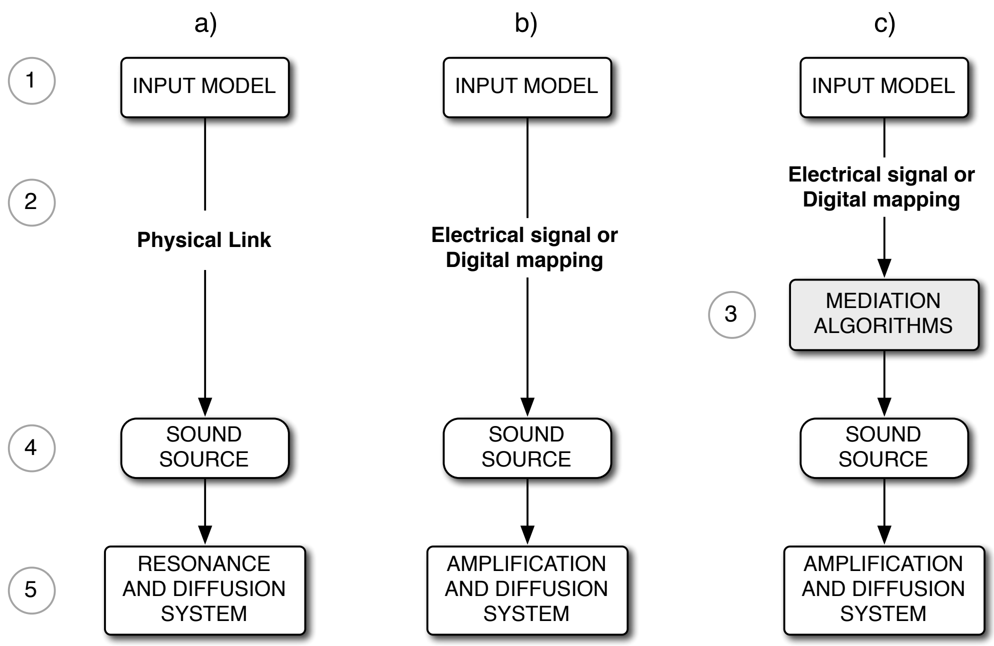


Fig. 7 Global structure of three different musical instrument model. a) Acoustic instrument; b) Electrical or Digital instrument; c) Digital Meta-Instrument.

The idea of a digital meta-instrument involves concepts and techniques gathered mainly from algorithmic composition, interactive music and interface design. The structure, when compared to the acoustic and electronic instruments, includes an extra stage (see Fig. 7), where the mediation algorithms receive the input data from the user, and run some type of process to influence the results, according to the intended results and algorithm definition. This layer is the cognitive stage that incorporates some form of musical knowledge that will be essential for the quality of the musical results.

In order to exemplify the type of mediation that was used for my projects, described in part two of this dissertation, I will use the description by Bielawsky of the instrument as a transformer from physical to musical gesture (in Kvifte, 2011) and the decoupling of the electronic instrument (Chadabe, 2002).

Bielawsky describes the mapping dimensions of the physical gesture to musical gesture in an acoustic instrument as:

- Time >> Duration;
- Space >> Pitch, Timbre;
- Dynamic >> Pitch, Loudness;

As with electronic and digital instruments, the input gesture and output are decoupled, computer mediation can, for example, use the time dimension to quantize the input timing to synchronized rhythmic events. In *GimmeDaBlues (section 4.5)*, as well as in the prototypes *PocketBand* and *MyJazzBand* (sections 7.1 and 7.2), the space dimension was used to generate different chord qualities, on the piano interface, as well as dynamics, according to the touch position on the virtual keys.

2.3.4 Towards a definition

From the ideas presented above and considering the many different formats that are part of a universe of music interfaces, I will present my own considerations to reach my own definition of instrument.

Focusing on features that might help to define an instrument - or at least a characterization of instrument that is satisfactory enough for the discussion in this dissertation – it may be possible to identify some key properties that seem to be pervasive to all the possible formats, namely:

1) Sonic identity:

   The resulting sound must be permanent, in the sense that it must not change drastically, so that it does not lose a clear sonic identity. It must be coherent in the sense that the resulting sound is only one sound and not many;

2) Interfacing identity:

   The control input strategy that defines the playing technique must not change. It must be coherent, in the sense that it must be clearly defined and reduced to the necessary minimum;

3) Predictability:

   It must have a permanent and coherent relation between the playing technique and the sonic result. The relation between the input and the output must be predictable enough so that the player can use it intentionally;

4) Evolutiveness:

   The instrument requires a playing technique that may be developed with practice.

For the purpose of this dissertation, the notion of instrument is a system that comprises any combination of acoustic, electrical, electronic or digital means, which form an expressive identity in which some type of input results in some type of sonic output. If there is a mediated, indirect relation between the input and the output events, the system constitutes a Meta-Instrument.

2.4. Computer mediation in commercial music software

While most of the advances in computer music research still lie in the academic world, some examples of its usage on commercial software can be found. Most importantly, there seems to be an overall tendency by the software companies to search for new functionalities to include in their music production software. As the several sequencers by different companies have grown to be very complete, but also to be very similar to each other, exploring algorithmic composition and automatic music generation models might be an interesting way to go.

This section will look into some of those commercial software applications that include some type of musical computation or non-standard features, to provide new and more effective ways to use the potential of computer mediation in order to help the user to achieve better and faster results.

2.4.1 Band-in-a-Box

Band-in-a-Box (Gannon, 1991) is a music application that automatically generates music in a given style, upon a given chord structure. Introduced in 1990 for PC and Atari ST, it has been unbeatably the most successful and almost one-of-a-kind application for the automatic creation of music accompaniments and even solos. Given a chord progression, it automatically generates music in one of the available musical styles that cover a wide range, like blues and Bossa Nova, for example. The chord progressions can be obtained from a library of available songs or introduced manually by the user, by typing the chords directly in the corresponding bars, using standard chord symbols. The user can define the song sections, number of bars, and the music style. The available instruments include piano, bass, drums, guitar, strings and horns. The "real tracks" are a special kind of tracks that add real life studio recordings by professional musicians.

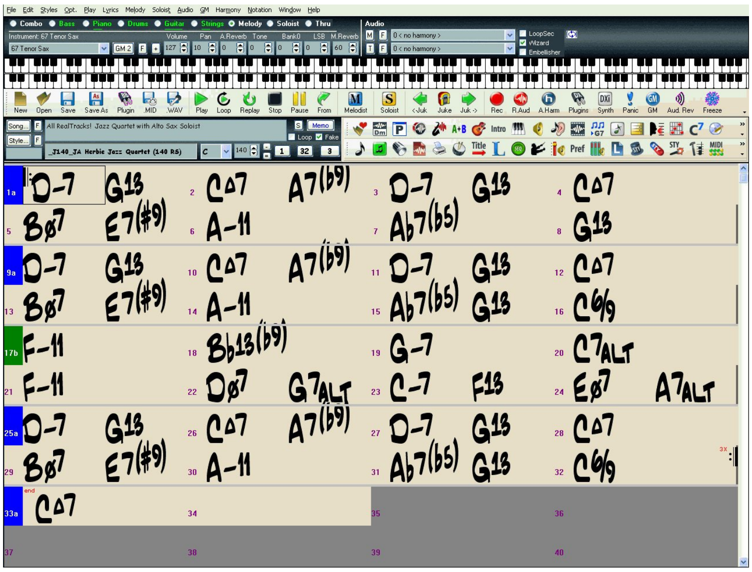


Fig. 8 Band-in-a-Box

2.4.2 Apple Logic Pro and Garage Band

Apple Logic Pro is a full-feature digital audio workstation (DAW). Version 10, released in July 2013, introduced a new set of features that go beyond the normal features of a sequencer track and virtual instrument. The "Drummer" is a special type of track that provides a virtual session player, built from recordings of real life drummers. The core feature of this special track is a large corpus of indexed pre-recorded patterns that can be accessed very easily using a bi-dimensional controller with the complexity level on the horizontal axis, ranging from “simple” to “complex”, and the strength or intensity in the vertical axis, from “Soft” to “Loud”. By moving this controller, the user navigates through the pattern space and builds the drums automatically. The movements can also be automated by recording or manually editing the slider movements and can later be altered like any other control parameter.

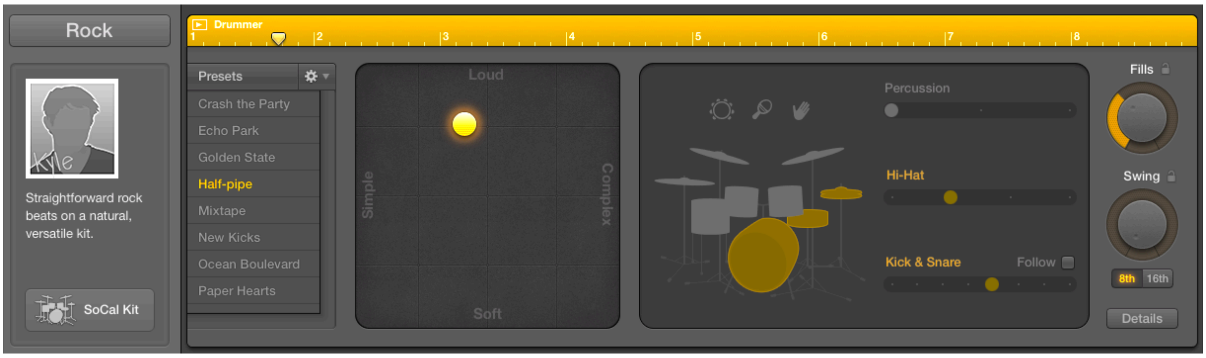


Fig. 9 Logic Pro Drummer track controls.

Besides the complexity map, the drummer also features controls over the “Fills” level and the “Swing” level. The first controls how often the virtual drummer will produce rhythmic variations in relevant turning points in the song structure. The “Swing” control will add the swing effect, very common in jazz and blues, by slightly delaying the second 8^{th} note in every beat.

Fig. 10 Logic Pro X drummer track “Details” view.

In the “Details” view, the user can further refine the playing style of the generated patterns. The “Feel” parameter is a fine adjustment of how the rhythm feels more laid back or more energetic, without changing the actual tempo of the song. “Ghost Notes” are secondary hits by the drummer, that although they don’t define the pattern, they can add a lot of feeling and groove to it. The control sets how much they will be heard, from “Quiet” to “Loud”. The “Hi-Hat” adjustment sets the proportion between how closed or open it will be.

These controls provide a high-level control over the drums, very different from the usual note-by-note sequencing paradigm.

2.4.3 Cubase Chord Track

In version 7, Cubase, by Steinberg, introduced the “chord track”. This is a special type of track, which is not meant to hold musical notes, but instead, chord sequence data. This chord sequence will be available for all the other tracks to serve as a base for their own harmonic content.

Fig. 11 Cubase chord track.

At its most basic functionality, the chord track is an easy way to create chords by simply selecting the chord root and type from a drop-down list. This track doesn't have any sound on its own, but, by linking an instrument track to be controlled by this track, the chords can be heard. The most interesting use is, however, when other midi tracks with their own content are set to follow the chords in the chord track. In this case, the midi notes in the tracks get automatically changed to fit the current chord triggered in the chord track.

The chord selection window has another feature, the chord assistant (Fig. 12), which is interesting on its own. The chord assistant helps the user select a chord to use by analyzing the previous and the next chord, if any. It then presents a color-coded list of suggestions for possible chords, ordered by each one's distance to the original ones as a color-graded scale, so that the last chords on the list, in red, create a greater contrast than the ones on the top, in dark green.

Fig. 12 Cubase Chord Assistant.

2.4.4 Multi-touch interfaces

Multi-touch devices have evolved and disseminated incredibly fast, due mainly to the wide range of uses in several areas, but with the most impact in mobile communications, where the advantages of the versatility and simplicity of use of the virtual screen interfaces practically took completely over the existing hardware counterparts. The virtual interface eliminated or significantly reduced the space and hardware interface limitations of the mobile telephones thus allowing them to become much more than telephones and smartphones with basic and mostly unusable extra features. The iPhone, introduced by Apple in June 2007 (“Apple product timeline,” n.d.) redefined the paradigm of the mobile phone as a new line of mobile mini-computers, with features that were really usable. In fact, this new line of devices was so effective, that the concept was reaffirmed with the introduction of the iPod Touch in September 2007, and the iPad, in April 2010.

Along with the devices, the key to this success was the opening of their operating system (iOS) to third party developers, thus creating a very attractive platform both for users and developers, deploying their software through the App Store. Since its introduction in 2008, Apple's iOS App Store has seen an amazing growth on the number and quality of applications of all types, exceeding every prediction. Music apps were no exception, and very soon showed up, using the iOS technical possibilities as well as exploring new possibilities for multi-touch control and mobility.

Different types of music apps can be found. Here I will enumerate some of the more relevant ones, related to music playing and music production, namely:

- Virtual Instruments;
- Music Production applications;
- Music Controllers;

Other types of music related apps exist, like music players, browsers, song databases and recognition algorithms, music notation, and more. These are, however, not so relevant for the current dissertation and as such will not be included here.

Virtual Instruments

Virtual instrument applications use the device as a musical instrument, by providing some kind of control inputs that convert the user's physical actions into musical events. A distinction can be made between a) virtual instruments that simulate existing hardware musical instruments, whether acoustic or electronic; b) virtual instruments that simulate existing music software; and c) virtual instruments that present original interfaces and modes of interaction.

Applications in the a) and b) categories try to emulate as close as possible the experience of playing the corresponding original instrument or software, frequently using photorealistic graphical interfaces and the devices features like the accelerometer or the microphone. In Ocarina (Smule, 2013), the user can blow into the microphone in order to simulate blowing into a real ocarina, while pressing the virtual holes on the touchscreen for note selection. Guitarist (MooCowMusic, 2013), Virtual Guitar (NETTuno s. r. l., 2013) and Real Piano (Cookie Apps Inc., 2013) are examples of acoustic instrument simulation, while Korg MS-20 (Korg inc., 2013) and Reactable Mobile (ReacTable SystemsSL, 2013), for example, are very well known examples of simulations of existing hardware.

The real-life simulation approach, however, is not necessarily the best way to go regarding the use of the screen and multi-touch capabilities. In fact, these virtual instruments can be quite difficult and even ineffective when getting down to actually using them, whether because of limitations of the screen space, response latency or simply because of the lack of the tactile sensation and physical feedback of the real instrument. Apps like Bloom (Opal Limited, 2013), SoundPrism (Audanika, 2013), Nodebeat (Affinity Blue, 2013), Thumbjam (Sonosaurus LLC, 2013) or Orphion (Trump, 2013), for example, have a different approach by presenting original instruments and interfaces which can be more interesting and effective to play and to explore new musical possibilities. Others, like the impressive Animoog (Moog Music Inc., 2013), combine both approaches to allow very expressive performance while retaining some of the visual and organization paradigms.

Music Production

Music production apps include sequencers, audio recording and editing tools, and virtual studios, which can have much functionality and can put together several types of instruments and/or effect processors. Most of these apps have also a simulation approach, by porting the experience of existing computer software to the mobile device. Apps like MusicStudio (Gross, 2013) and the very impressive Garage Band (Apple Inc., 2013) are very good examples of this, while also using some new features exploring for example the multi-touch properties of the iOS devices. Others, like Tabletop (Retronyms, 2013) or Rhythm Studio (Pulse Code Inc., 2013) provide the user the experience of having a modular set of devices in a virtual tabletop.

Music Controllers

This type of applications allows the use of the iOS device as a controller for external software or hardware. Some apps like for example AC-7 (Saitara Software, 2013) and ProRemote (Far Out Labs, 2013) are DAW (Digital Audio Workstation) controllers, which work like remote wireless controllers for a computer running the corresponding DAW software. Some well-known DAWs have nowadays their own release of remote controllers for iOS, including MOTU's DP Control (Motu inc., 2013) and Steinberg's Cubase iC (Steinberg, 2013).

Some more open apps like Touch OSC (Fischer, 2013), MrMr (10base-t interactive, 2013) or Fantastick (Pink Twins, 2013) allow the user to build his own layout and program the OSC or MIDI messages for every item. In fact, these can be used for any other non-music software that accepts configurable external control messages, like real-time graphics or video applications.

2.5. Conclusion

The topics and examples addressed in this chapter provided a base for some important aspects that were considered in the development of the concepts and solutions devised for the algorithms and software created for this dissertation. In this case, the selected topics inform about the main considerations in the definition of the output format of the software, which are both conceptual and technological.

The concept of musical interface is key to the proposed model of a computer-mediated performance system that I will describe in the second part of this document. The main concepts addressed in this chapter include a reflection about musical interfaces, the definition of instrument and different instrument formats, in order to reach my own definition of instrument, that, hopefully, is simultaneously broad and encompassing, but also explicit and specific enough to serve the objective and purpose of this dissertation.

The notion of "meta-instrument" is one of the core concepts of this dissertation. The control of musical processes at a meta-level has been around since very early in computer music history and the computation power and the technological resources available today renders possible an endless number of possibilities for the creation of musical systems empowered by sophisticated algorithms, and with any combination of input and output formats. Using specifically developed mapping and computation algorithms, the meta-instrument can provide a reference for the development of new formats of musical interfaces and software.

The provided examples of commercial software are important to emphasize how mainstream software companies are gradually approaching and integrating features obtained with computer mediation.

The provided sample list of musical applications illustrates the new range of possibilities in the music hardware and software development business, and that the multi-touch tablets are undoubtedly a powerful and attractive platform for musical use, responding to some important features that were always desired in digital musical in the digital domain, namely power, versatility, mobility and interface flexibility and intuitiveness. However, they are still limited by the lack of a tactile sensation and physical feedback, but this may not necessarily relevant in many cases.

Chapter 3. The Jazz Music Practice

Jazz is certainly one of the most influential and iconic musical currents of the twentieth century. Its development through the first half of the twentieth century is marked by a series of important influences from unlikely social and musical backgrounds, becoming one of the century's cultural symbols of social and musical intersection and liberation. Its practice spread to every continent, and influenced different music contexts and styles, from classical and contemporary to film and dance music. In spite of its origins in the streets and illiterate backgrounds, with the intersection with classical tonal music, jazz developed into a complex and rich musical language, strongly marked by the use of improvisation, with its own syntax and vocabulary, and a rich and widespread repertoire. Jazz is currently taught in schools and universities around the world. This chapter will present an insight on the act of improvisation itself in the jazz context that forms the conceptual ground of this dissertation.

The conventional jazz performance practice consists on the realization of versions or re-interpretations of a given tune, whether original or a song from the existing repertoire. The degree to which these versions can differ from the original one is radically different than in the classical music approach. The jazz musician is usually free to alter practically every aspect of the song, as long as it remains somehow recognizable, even if very subtly[3]. These differences are not limited to the phrasing, character, energy or speed of each version. Instead, these differences can happen at any level, including the melody, harmony and rhythm. This is especially evident in the solo sections, where a

[3] Compare for example the original version of "Summertime" by George Gershwin in "Porgy and Bess", with John Coltrane's version in the 1961 album "My Favourite Things".

soloist improvises freely over the song structure, and can develop the solo differently for each performance. From a compositional perspective, these standards of the traditional repertoire are almost invariably tonal, due mainly to the western music influence and the practice of the Broadway musicals songs.

Recent work is being developed to create a database of machine-readable lead sheets for jazz standards, which will hopefully compile all the different versions and serve as a comprehensive and flexible way to access digitally to this archive (Pachet, Suzda, & Martin, 2013).

Besides playing and arranging previously composed pieces, a great number of jazz musicians tend to also write their own compositions. This is another aspect where jazz music practice differs greatly from the classical music practice, where there is a strongly established separation between the role of the interpreter or player and the composer.

3.1. Improvisation

In this section, I will provide an insight into the act of improvising, both from a general perspective and in a more specific use in jazz music. My purpose is not to explain or provide any new contributions to the analysis of the process of improvisation, but rather to use existing research to identify and illustrate some aspects of the act of improvisation which I consider relevant to validate the idea that computer mediation - as presented in this dissertation - does not eliminate the essence of the improvisational act.

Of the several characteristics of jazz music, the most prominent one is very likely the extensive and inherent use of improvisation. Apart from a few exceptions that can include written jazz scores to be played by non-jazz musicians that lack improvisational skills, or some forms of group or big band arrangements that may be totally written, improvisation in jazz music is basically pervasive in the performance and composition, and can be observed in different levels. It can range from subtle variations of the original melody to complete free-form improvisation without a base structure.

In western classical music tradition, since the 19^{th} century, a large gap has been formed between music creation and music performance. In the 18^{th} century all the composers were performers, and virtually all performers composed (Levin, 2009, p143). The instituted western musical education system emphasized this gap by focusing on the technical and historical aspects of the interpretation and reproduction of the established repertoire, while leaving almost completely behind the musical awareness, knowledge and practice needed to create and improvise music. Going through the established European music education schools, the classical musicians have highly developed skills in music reproduction, but little or no training at all at inventing it (ibid.).

Going the opposite direction of the typical western classical musician, the common jazz musician learns to improvise from the very beginning of his/her training, and develops his/her own improvisation skills and personal style throughout his/her career.

In the traditional performance of jazz standards, the improvisational skills of the musician have its peak during the solos. Solos are a central and defining element in jazz music performance. The attention and relevance of the solo(s) and soloist(s) in the performance of a jazz piece usually surpasses the importance of the song itself, and can be much longer than the melody, which is typically played once or twice in the beginning and end of the performance. The melody acts as a kind of frame to the central sections, where the solo or solos happen, while the chord sequence provides a harmonic ground base over which the improvisation is developed.

3.1.1 The nature of improvisation

According to (Campbell, 2009, p.122), the likely root of improvisation is the Latin “improvises”, meaning “unforeseen”, which argues for improvisation as a process that is not premeditated. Music that is improvised is never fully predictable. The author also mentions that “the momentary phenomenon of improvisational expressions is not easily (if ever able to be) duplicated, and the performer may not be able to verbalize what comes tumbling out”. Nevertheless, however unpredictable an improvisation may be, it is a fundamentally different process then that of indeterminacy.

> “It seems to me that there is a fundamental difference between aleatoric and improvisational music. Improvisation is concerned with the realization in real time of defined artistic goals. Aleatory, by its very nature, does not recognize the existence of goals. (…) Improvisation, at its highest, seeks meaning through spontaneity.” (Hellerman, 1971)

Several authors (Bailey, 1993; P. N. Johnson-Laird, 1991; Philip N. Johnson-Laird, 2002) have studied the nature of improvisation as a creative process, in its cognitive, artistic or technical properties. One of these properties relates to the degree to which improvisation relies on pre-apprehended musical materials, rather than being completely original. Patricia Shehan Campbel refers that “the act of improvisation (…) requires conscious as well as unconscious selection from a reservoir of musical sound expressions that have been acquired over time (Campbell, 2009, p.121) Lukas Foss mentioned that in improvisation, one plays what one already knows” (Foss 1968, in Cope, 2001, p.77). According to (Philip N. Johnson-Laird, 2002), "creations cannot be constructed out of nothing". Hence, all creation is based on existing elements, and an improvisation can be seen as novel, in the same way as in natural language a new utterance can be seen as novel.

This seems to be coherent with the common practice of the learning process of a jazz musician, in which a large part of the exercises are based on the repetition and variation of model phrases or motifs and harmonic sequences. An example of a very common exercise is to play a melodic sequence using a single phrase transposed to every key through the entire circle of fifths (see Fig. 13).

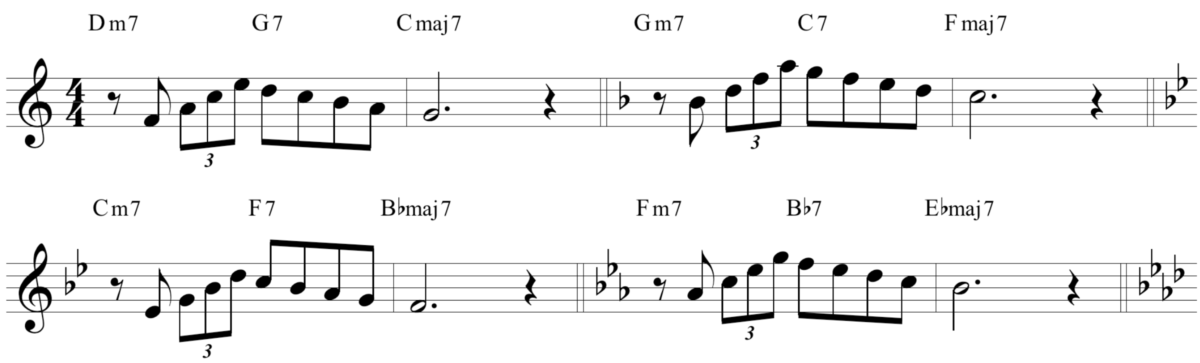


Fig. 13 Example melodic pattern in the circle of fifths progression (first four keys starting in C major).

This practice enables the musician to develop the technique and mastering of the instrument, to become proficient with the articulation of phrases in the different harmonic spaces, and to develop a reservoir of musical materials that become part of the musician's own vocabulary. However, it would be very simplistic and reductive to think of the whole process of improvisation as a mere re-combining of these pre-existing phrases.

> ”(...) musicians string together a sequence of motifs/licks as they used to be called modified to meet the constraints of the chord sequence. (...) Yet, the motif theory cannot be the whole story.” (Philip N. Johnson-Laird, 2002)

Psychologist Philip Johnson-Laird refutes the idea. “a common misconception about improvisation is that it depends on acquiring a repertoire of motifs (…) which are strung together one after the other to form an improvisation…” (P. N. Johnson-Laird, 1991). In this article, Johnson-Laird presented an extensive insight based of the premise that the theory of improvisation should be describable as a computational model, and argues that improvisation cannot be thought as just a repetition of pre-existing phrases. During improvisation, the performer combines his previously learned knowledge and technique, with aspects unique to each performance. While it is easy to find the use of these motifs in a solo by a beginner improviser, it is not necessarily the case with more experienced ones. A skilled jazz musician is able to create original melodies during improvisation, and, even if using motifs and motivic repetitions, he/she is able to integrate them subtly and organically, so that it becomes part of a fluid new melodic contour.

3.1.2 Real time and non-real-time aspects in improvisation

Here, an important distinction can be noted, between the aspects in improvisation that are used from a corpus of the knowledge previously learned by the musician, from those that happen in real-time, during the performance. This is particularly relevant for this dissertation, as the concepts and algorithmic models and interfaces developed rely on this distinction to provide a usable model in which the theoretical knowledge about harmonic rules and constraints that take many years to master, is taken care by the computer, while most of the decisions that are taken during improvisation are left available to the user.

Possibly the most empowering aspect of improvisation is the action of deciding WHEN to play. The improviser is free to “attack” a new note or phrase at any moment in the sequence and metrical grid. Miles Davis described the most important part of his solos to be the empty space between notes, the “air” that he placed between one note and the next. Knowing precisely when to hit the next note, and allowing the listener time to anticipate it (Levitin, 2006).

Simultaneously, the improviser decides WHAT to play. This choice is of course constrained by the harmonic contents, and requires the background theoretical knowledge and practice, but there is a range of possibilities that arise and are decided during performance. The decision to play a given note at a given time is influenced by the notes, phrases and silences that happened before, and will happen next, and may be influenced by external events, like something played by another band member, or even by the reactions from the audience (Blacking, 1973).

Along with the momentary decisions and actions, an improviser deals with considerations on the linear continuity of the solo in different time-scales, managing and directing the solo not just in short-scale but also in large-scale continuities (Hodson, 2007, p.10). The large-scale development of a given improvised solo can reveal many of the features present in written (non real time) composition, like thematic variation, continuous build in density and tension, the use of motivic repetition and variation in pivotal moments, and contrasting sections. In fact, music theorists have applied and adapted Schenkerian analysis to study the presence of different time scales in jazz improvisation (Larson, 1987, 1998, 2009, Martin, 2011a, 2011b).

3.1.3 Group Interaction

While improvised solos can be analyzed by its independent features, which can be useful for pragmatic reasons, it's important to remember that an improvised solo is not an isolated process, but instead it is "supported by, responded to, and responsive of the parts being played by the other musicians in the group" (Hodson, 2007, p.1).

> "At its best, a jazz performance can be a communal affair in that the musicians are mutually supportive of one another (…) The community is further expanded to include the audience who, in a sense, become "community members" and participants…" (Stephans, 2013, p.78)

Group practice in jazz music is particularly interesting. Due to its improvisational character, there can be big differences between performances of the same piece, making each performance unique. Jazz musicians develop a very strong sense of group coordination and communication, as they learn to be constantly aware of what the other members are doing. This requires a strong listening practice and ear training. During a solo, for example, the other musicians can be accompanying, contributing to the solo, or simply waiting. This is particularly relevant in that it provides a very rich context for the creation of group interactions that contribute to the overall result and direction of the piece. Whether in pre-arranged sections or during the solos, the musicians can be constantly communicating and exchanging all sorts of licks and musical suggestions. Basically, all of the members of the ensemble are improvising simultaneously (Hodson, 2007, p.7).

3.2. Texture and Roles in traditional Jazz

The traditional practice in jazz led to some common practices and "formulas" for group playing. This section will provide a brief description of some relevant aspects and differences on the roles and idiomatic features of some representative music elements and instruments that were especially relevant for the development of the software specially created for this research. The selection comprehends a Solo instrument, Piano, Bass and Drums. As mentioned before, these represent one of the most common jazz group formations, and can present a clear picture of the role assignments to create a conventional jazz texture.

Some of these aspects will be further discussed in Part 2, in the technical description of the instrument algorithms and user interfaces that were developed.

3.2.1 Form

The form or structure of a traditional jazz standard is usually a song-like structure, comprising a small number of structural sections (usually three or four) and very often added sections like intros, interludes, special endings, shout choruses, and verses, which are generally not part of the solo form (Levine, 1995). The global macro-form throughout the performance, however, consists almost invariably in the three part "head arrangement" (Hodson, 2007, p.75), with the initial exposition of the theme, followed by improvised solos, and re-exposition with the theme again. Table 1 shows the form and harmonic grid of the tune "A Night in Tunisia", by Dizzie Gillespie. The main form is an A-A-B-A structure, with an *Introduction* section of four bars, plus an *Inter-*

lude. The final four bars of the interlude are a *Solo break*, in which one of the soloists plays alone until the reentering of the other instruments, in the *A* section.

A Night in Tunisia			
			Dizzy Gillespie
INTRO			
Eb 7	D -	Eb 7	D -
SECTION A			
Eb 7	D -	Eb 7	D -
Eb 7	D -	E -7b5 A 7b5	D -
Eb 7	D -	Eb 7	D -
Eb 7	D -	E -7b5 A 7b5	D -
SECTION B			
A -7b5	D 7b9	G -	G -7 C 7
G -7b5	C 7b9	F 6	E -7b5 A 7b5
SECTION A’			
Eb 7	D -	Eb 7	D -
Eb 7	D -	E -7b5 A 7b5	D -
INTERLUDE			
E -7b5		Eb #11	
D -		G 7#11	
G -(maj7)	G -7	Gb 7#9	
F maj7		E -7b5	A 7b9

Table 1 “A Night in Tunisia” by Dizzie Gillespie - Sections and harmonic grid. (as it appears in “The Real Book” compilation).

This form originates from the traditional sonata form, with the [Exposition – Development - Re-exposition] sections. The tune melody is played at the beginning and end of the performance, with any number of solo improvisations as the development section. As said, the form can also have an *Intro* section before the first exposition, and some type of *Ending*. These will usually only be played once in the entire performance.

The solo sections are in many respects, central in jazz practice. As more than any other musical genre in western music, jazz is strongly based on the improvisation skills of the performers. In fact, the tune is basically a provider of a harmonic and melodic base, over which the performers improvise. These sections can vary considerably whether in its contents, duration or instrument sequence. Traditionally, some or all of the players in the group take their turn in soloing over the tune's chord sequence. Each run of the chord sequence is called a *chorus*, and each solo lasts for one or more *choruses*.

3.2.2 Instrument roles

The instrument roles can be varied and observed under distinct criteria, according to their contribution to the overall result. By their idiomatic characteristics like pitch range, polyphony, dynamic range, agility and articulation possibilities, each instrument can have its distinct role in the group, complementing each other to form a balanced and coherent result. These roles can be basically situated between two main poles - the melody and the accompaniment - where the accompaniment can be divided into rhythm and harmony (Fig. 14).

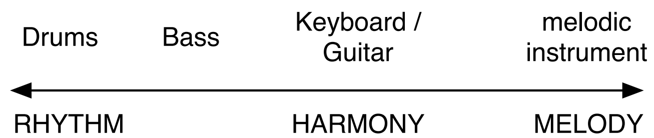


Fig. 14 Instrument roles according to a rhythm to melody axis.

The melodic instruments, like the trumpet or saxophone are sometimes referred to the *front line*, while the instruments playing the accompaniment - rhythmic and harmonic instruments - are commonly referred to as the rhythm section (Hodson, 2007, p.7, 25).

3.2.3 Rhythm Section

In a typical jazz band, whether a small group or a big band, the melody, harmony and rhythm, can be heard distinctively. Taking as an example a jazz quartet, a very common formation having a melodic instrument, like a saxophone, an harmonic instrument such as piano, a bass and a drum set, we can very clearly hear the assignments of the melody to the saxophone, the harmony to the piano and bass, and the rhythm to the drums. Because the piano and bass also contribute to the overall rhythm, they are included in what is usually referred to as the *rhythm section*. The rhythm section acts essentially as a harmonic and rhythmical base, while the soloist improvises freely.

This is, however, a simplistic view of the roles assigned to the instruments, as in practice they can be much more intertwined and the elements melody, harmony and rhythm disseminated in the different instruments. The piano, as the guitar, for example, can double as a melodic instrument and as a rhythmic instrument, and even the drummer can supply some melodic contributions by using the melody's rhythm or even pitches, using the tuning on the tom-toms or some other percussion element. Also, some examples can be found in the repertoire, in which the roles are intentionally contradicted, like in Miles Davis's "So What", in which the melody is played by the bass, or his version of "Nefertiti" by Wayne Shorter, where the horns repeat the melody over and over, while the rhythm section improvises collectively (Levine, 1995, p. 395, 391). However, this study focuses mainly on the study of the traditional forms of jazz, and as such, the conventional uses of the instruments in a typical jazz set.

The role of the drums in Jazz is particularly interesting. In most other music styles, the drums main role is to establish a solid and steady rhythmic base with a constant behavior that sets the pulse speed and provides a clear metrical sense. In jazz however, the drums appear to have an almost opposite behavior. Typically it contributes to the sense of pulse and metric by playing the *ride pattern*, and the *backbeat* (Hodson, 2007, p.30). The ride pattern is a two-beat rhythmic pattern of a quarter note and two eight-notes (see Fig. 15) played on the ride cymbal. These two eight notes are generally "swanged" (ibid.), i.e., the second eight is slightly delayed so that it becomes almost an eight-triplet. The backbeat is played on the hi-hat cymbals and is one of the most characteristic elements in jazz music. It corresponds to the accentuation of the 2nd and 4th beats (in a four-beat bar).

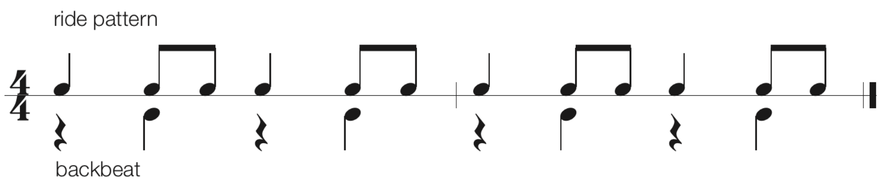


Fig. 15 The ride pattern and the backbeat.

While he plays these elements, however, the drummer is constantly creating variations, and improvising other elements on the snare and bass drum, changing the rhythmic patterns in such a way that he may not even repeat the same pattern a single time during an entire tune. Jazz drums have much more freedom, in the sense that they can do much more than just repeating the same pattern repeatedly and slightly varying during the breaks. The task of keeping a steady beat is actually mostly left to the bass player, especially while using the walking bass technique during solos.

3.2.4 Walking Bass

A particular technique was developed in jazz bass playing called “walking bass”. The walking bass is a very common playing procedure used in jazz music, in which the bass “walks” through the chord and scale notes in a regular pulse. This is especially used when accompanying a solo, firmly setting the base pulse like a metronome, and simultaneously exposing the underlying harmony of the song. This procedure has its roots deep in the first decades of the twentieth century, and was developed through the years with bass players like Jimmy Blanton, Ray Brown, Ron Carter and Charlie Mingus, amongst many others. A very good insight on the evolution of the bass role and many of the key innovators in jazz history can be found in The Jazz Bass Book, Technique and Tradition, by (Goldsby, 2002).

The basic idea of a walking bass line is to go from one chord to the next, linking them by filling the middle beats with notes of the chord or scale, typically describing a smooth melodic line or pattern. Far from being completely passive, however, the roll and behavior of the walking bass can dramatically change from almost neutral smooth lines to very abrupt register changes and energized rhythms, dynamically contributing to the overall group energy and musical result. The continuous flow of the regular walking bass notes, together with its harmonic and melodic content, form one of the most charismatic and important elements in traditional jazz playing.

The learning of this technique usually implies the learning of melodic licks, for each of the commonly used chord progressions. These licks are small melodic phrases that are particularly efficient and musical, and are usually related both to the available notes of the current chord and underlying scale and to physical placement of the notes and fingers on the bass. Bass learning methods like the ones by (Brown, 1999; Carter, 1998; B. Magnusson, 1999; Mooney, 2011) lead the student through a series of these licks, covering most common harmonic progressions, in order to provide the student with the basic formulae that can then be adapted to any song.

As the walking bass player develops his technique and gains more experience, however, he will be able to use these phrases more articulately. More importantly, he will be able to move away from them, intuitively creating more natural sounding lines and integrating several other elements like ornaments, theme-specific elements and group feedback.

3.2.5 Solo vs. Comping

An important distinction for this study is between the solo and the accompaniment, also called *Comping* (Hodson, 2007, p.33). While the soloist has a predominant role, a good accompaniment can contribute immensely to the overall result, both by directly enriching the music texture and development, and by closely following and even pushing the soloist.

As we'll see in the next chapter, most research studies on computer-generated jazz music lie on the automatic generation of solos. In the software developed in this research, although some aspects can be used for soloing, the main focus is on *comping*, both automatically and interactively by the user.

Chapter 4. Jazz-related Computer Music Research

Attempts to formalize and implement jazz music into computer programs have been around for over thirty years. Because it is based on improvisation, jazz is especially appealing to be targeted from a computer music generation and artificial intelligence perspective. Simultaneously, traditional jazz is particularly well suited to be decoded into algorithmic processes due to its formal nature and clear set of procedures in the conventional performance practice. This makes it relatively easy to define as a set of rules that can be implemented in the form of computer algorithms. This is, however, no small task if we consider the countless number of factors that go into human improvisation. Both the formal and the empirical processes implied are the result of a long learning and appropriation process by the musician throughout his/her development, in which an important part of it is based on cultural inheritance and empirical knowledge.

In this chapter I will address some research topics from existing literature, that specifically focus the computation of jazz music, which I consider relevant for the development of this area as well as for my own research.

4.1. Automatic generation of jazz solos

The automatic generation of solos is by far the most recurrent subject in jazz-related computer music research. Several authors over the years have addressed this problem using a multitude of different approaches and techniques, from stochastic models to evolutionary algorithms.

David Levitt (1981) was one of the first authors to describe a computer program for the generation of jazz solos. This program generates a single voice of improvisation, given a song melody and harmony. The program is divided in two main parts. The first part is dedicated to the analysis of a given input tune's melody and harmonic sequence. The harmonic sequence data will provide an important base for the analysis of the melody and to infer modes for the solo generation. The melody is subdivided into two-bar segments, and each segment is analyzed and ranked according to criteria of simple harmonic, intervallic and rhythmic measures. Using these criteria, the program chooses the "most interesting" phrase, which will be used for the generation stage. The second part of the program concerns the generation of new solos, creating new phrases by generating variations of the melody using a recursive algorithm. Starting with the original "most interesting" melodic segment obtained in the analysis stage, the program creates a variation of the segment by altering its features. The resulting variation is then subject to the same process, going through the analysis stage and the generation stage. If the generated phrase going through the analysis process turns out to be not so interesting, the program will select a new phrase from the original melody.

Levitt's program was a thorough research of the generation of solos. Creating variations on the previous phrase is a clever way to assure that the solos will have a strong coherence. This idea of constructing a solo from variations of pre-existent phrases was also presented in the model by Ulrich (1977). His computation model creates melodic phrases by creating variations of the input melody of a given song. Starting with an analysis of the tonal center and tonal functions of the chords, the system finds scales that fit the harmony, and creates the variations by adapting fragments of the original melody to those scales.

Other systems for the generation of jazz solos include (Chen, 1992), Cybernetic Composer (Ames & Domino, 1992), (Pennycook, Stammen, & Reynolds, 1993) and Flavours Band (Fry, C. in (Schwanauer & Levitt, 1993).

One of the most successful and enduring systems for the generation of jazz solos was developed by John Al Biles using genetic algorithms (GA). *GenJam* (short for Genetic Jammer) is a real-time system that is able to generate quite convincing solos over a given harmonic progression, and play along with a human soloist, earing and responding with coherently related melodic phrases, during a stage performance. It was implemented in a Macintosh/Think-C environment using the CMU MIDI Toolkit (Dannenberg, 1993) and the accompaniment, usually with piano, bass and drums, are MIDI files previously created with the software Band-in-a-Box (see section 2.4.1). Described initially in (John Al Biles, 1994), GenJam went through continuous developments (J.a. Biles, 1999; John Al Biles, 1998, 2001, 2002a, 2002b, 2003, 2013; John Al Biles, Anderson, & Loggi, 1996; John Al Biles & Eign, 1995), and is still used in live concerts as a virtual soloist in Biles' "virtual quintet", improvising along with Biles himself on the trumpet[4].

Each song is especially prepared to use with GenJam, and described in a file with the necessary data, namely the tempo, style and chord progression. After loading a song file, the program parses the corresponding information and maps the necessary data. A section of the program receives the chord progression data and assigns a scale for each half-measure. These scales are not necessarily a complete scale, but a custom scale planned in order to optimize the use later on by the genetic algorithm. The scale notes are then mapped to fourteen note-event indexes, starting on the low-

[4] example performance in: https://www.youtube.com/watch?v=rFBhwQUZGxg

est note and spread to about two octaves, corresponding to a confortable and useful range for the virtual solo.

The corpus of notes that will compose the solo is acquired from two different reservoirs: *measures* and *phrases*. A *measure* is a one-bar melodic fragment (four note events in a 4/4 measure), and a *phrase* is a concatenation of four *measures*. Each *measure* is represented as a string of numbers, and each number corresponds to a note event in the pre-calculated scales.

Measure number	Measure score	Note events
38	-4	7 8 7 7 15 15 15 0

Table 2 One of the example measures provided by John Biles.

Each measure has a number and a score. The note event numbers range from 0 to 15, where the 0 (zero) represents a rest, and the 15 represents a “hold” event. Numbers 1 to 14 correspond to one note event. The example measure in Table 2 is number 38, which as a negative score of -4. The eight note events correspond to the eight eigth-notes of a 4/4 measure. While the first four events are new notes, the 4^{th} to 7^{th} events are hold events, which means the last note will be sustained. The final zero is a rest, so the previously held note will stop sounding.

In the “phrase” population, a similar format is used to represent the number associated with each measure.

Phrase number	Phrase score	Measures
23	-12	57 57 11 38

Table 3 Example Phrase

The example in Table 3 shows a phrase numbered 38 with a negative score of -12. The phrase is constituted by four measures, with index numbers 57, 57, 11 and 38.

In its initial version, presented by Biles (John Al Biles, 1994), GenJam had three modes of operation: *Learning*, *Breeding* and *Demo*. The Genetic Algorithm is applied to both the *measures* and *phrases* populations. After an initial randomly generated population of 64 measures and 48 phrases, the system has to go through the learning mode in order to access and evaluate the contents. This is done by the concept of the *Fitness function* of the genetic algorithm, which will selectively filter out the measures and phrases with lower scores and use the higher scored ones for the generation of new ones. The Fitness function in this case is done manually by the user. In the Learning mode, the system plays back the existing population of phrases, and the user scores them using a control device, by pressing a button named 'g' for good or a button named 'b' for bad, during or immediately after the measure is played. Every press will assign the corresponding score that is accumulated to the previous value. So, if the user ears the measure in Table 2 with a score value of -4, and presses the 'g' button, the value will be incremented positively to -3. The scoring affects both the measure and the phrase population, so, for example, if the phrase containing the measure would be the one in the example in Table 3, the phrase would change its score to -11. If so desired, the user can emphasize a measure or phrase by pressing any number of times for the same evaluation. In order to avoid an exaggerated convergence to a few very similar high scored measures and phrases, a limited score range of -30 to 30 is enforced.

In the Breeding mode, the existing populations and corresponding scores are fed into the genetic algorithm, which generates new materials by applying the Crossover and Mutation operations. For the crossover operation, the measure population is divided groups of four randomly selected measures called *families*. For each family, the GA discards the measures with the lower score and uses the two with the higher score for the single-point crossover operation, which crosses the first part of a measure chromosome with the second part of another. The two resulting offspring will replace the discarded measures in the family. By doing this, the measure population is rapidly renovated and evolved according to the fitness values. Because the consecutive application of this procedure would tend to breed populations that converge to very closely related measures and phrases, that could become repetitive and boring, the Mutation stage is used to create some variation and surprise. Mutation is applied to one of the offspring, while the other is left intact. In order to potentiate better resulting offspring, the mutation operation is not neutral but instead it is biased with six types of operations that have a musically significant result. The operations are: 1) Reverse; 2) Rotate right; 3) Invert; 4) Sort notes ascending; 5) Sort notes descending and 6) Transpose notes.

For the same reasons, the Mutation stage for the phrases uses six operations types: 1) Reverse; 2) Rotate right; 3) Genetic repair; 4) Super Phrase; 5) Lick thinner and 6) Orphan phrase. The mutations are planned to allow both small changes to total surprise, with the Super Phrase operation, in which all the member measures of the phrase are substituted by measures that had a higher score.

GenJam’s Genetic Algorithm can be described in six steps (see Table 4), as described in (Miranda & Biles, 2007, p.146).

- *Repeat*
    - *Select 4 individuals at random to form a family (tournament selectiion);*

- *Select 2 family members with the greatest fitness to be parents;*
- *Perform crossover on the 2 parents to generate 2 chidren;*
- *Mutate the resulting 2 children until they are unique in the population;*
- *Assign 0 as fitness for both children;*
- *Replace the two non-parent family members with the new children;*

- *Until half the population has been replaced with new children;*

Table 4 The Genetic algorithm in GenJam.

In 1995 edition of the International Computer Music Conference, Biles presented *GenJam Populi*, an experiment with a live audience in which the members of the audience act as “human fitness functions” (John Al Biles & Eign, 1995). The goal of the experiment was to explore the parallelization of the evaluation of the phrases as a way to speed up the fitness feedback, and turn it more reliable. Biles reports some interesting observations about the audience participation in this and other similar sessions. The participants, who act as mentors in training the IGA, seem to feel more comfortable if they possess “at least one of three attributes: 1) they like jazz, 2) they are musicians, or 3) they are active in the computer field. People who lack all the three attributes tend to be intimidated (…) this seems to be due to their inability to form a mental model of what a mentor is supposed to do. They literally lack the knowledge on music in general and jazz in particular to be able to form opinions about GenJam’s improvisations, and they often feel intimidated at having to render an opinion on something they don’t really understand”. (J.a. Biles, 1999).

Continuing the development of GenJam, in 1998 Biles presented a new version, with significant new features, namely the note data acquisition from a real performer by using an audio-to-MIDI converter, which detects the pitch of a live acoustic instrument during performance and outputs the pitch data to the computer as MIDI messages. The pitch data is then inserted as the measure population, and the genetic algorithm's mutation stage with the same measure and phrase operations. The mutation stage occurs in the last thirty milliseconds of the phrase, during which GenJam stops listening to the soloist input.

Other curious observation in (J.a. Biles, 1999) concerning the reactions when the audience mentorship happens during the live performance of GenJam interacting with a live soloist, is that “from a performance standpoint, the biggest problem is that the interaction is only at an aural level and is not visual. Audiences at jazz performances expect to see the interaction between musicians, not just hear it”.

The projects developed for this dissertation, presented in Part II, have some similarities with the solutions found in GenJam. Namely, the chord/scale mapping and the use of the data files for each tune. The same mapping strategy is used to algorithmically assign scale (and chord) notes to the instruments, and the same concerns regarding some potential harmonic problems, by avoiding some scale notes, as will be explained in detail in the corresponding sections.

Also from Biles' experience, it is interesting to note the observations about the reactions of the participants of the audience mentorships. In *GimmeDaBlues* and *Pocket Band*, the users are the players, but the results, even with the mediation strategies implemented, still vary according to how familiarized the user is with jazz music.

In a 2008 article, Kjell Bäckman and Palle Dahlstedt refer a limitation of GenJam to produce solos that build up from a low intensity level to a climax and rounding it off at the end (Bäckman & Dahlstedt, 2008). In this article, they describe a genetic-based system to produce solos that deal with the overall coherence of an improvisation during a solo. The system uses an interesting technique based on a “rubber band” principle to generate an overall contour for the entire solo. This principle deals with the energy distribution to produce a contour that defines the tension level throughout the solo, and calculates the note pitches, as well as note lengths and volumes. The resulting stream is then divided into small phrases (named *Delta phrases*) and passed through a series of transformative operations, organized in a hierarchical tree, to create more variation and richer phrase results. The raw melody material and the operator tree constitute the material of each genome, to undergo the process of genetic evolution, including crossover, mutation and a fitness stage, where the selection is done empirically by the authors, by listening the results and using their own background experience as jazz musicians.

4.2. Harmony

Harmony in jazz derives directly from the European classical tonal music tradition. Composers like George Gershwin and Cole Porter, were classically trained musicians that adopted the rhythms and popular character of the jazz and blues roots and developed with more sophisticated harmonic procedures. By doing this, they helped defining traditional jazz.

As usually happens in classical tonal music, harmonic progressions in jazz are based on the highly hierarchical organization of the tonal system, in which the chords follow some type of logical path between cadences, like some sort of asteroids in gravitational fields, around the tonic. The constant movement of attraction/repulsion between the degrees in relation to the tonic constitutes the base ground for the development of the piece in time, and the piece is organized by phrases that end with a cadence. Tonal cadences are a specific group of chord progressions that became standard in tonal music composition, and that mark the end of a section or phrase. Usually a tonal piece can be observed at a macro level by the final chord in each cadence, as this marks the arrival to important moments in the piece, together with the first chord after the internal cadences. This type of analysis was greatly developed by Heinrich Schenker (Forte & Gilbert, 1982).

An important consideration about an underlying chord sequence "is that it is not improvised. It is composed" (P. N. Johnson-Laird, 1991). Whether the starting point is the harmonic sequence or the melody, the selection of the chords is done following a logic tonal construction, that conducts the overall form of the music, and supports the melody. The approach to the algorithmic generation of chord sequences mostly applies to the creation of variations of a given base harmonic grid.

Mark Steedman (Steedman, 1984) describes a generative grammar for the harmonic transformation of a standard twelve bar blues chord sequence. Using the harmonic procedures described in (Coker, 1964) as the main reference, Steedman defines a set of six substitution rules, with which one can obtain several blues variations from the same base structure. The possible substitutions are illustrated in a relational diagram with different complexity levels of chord substitutions.

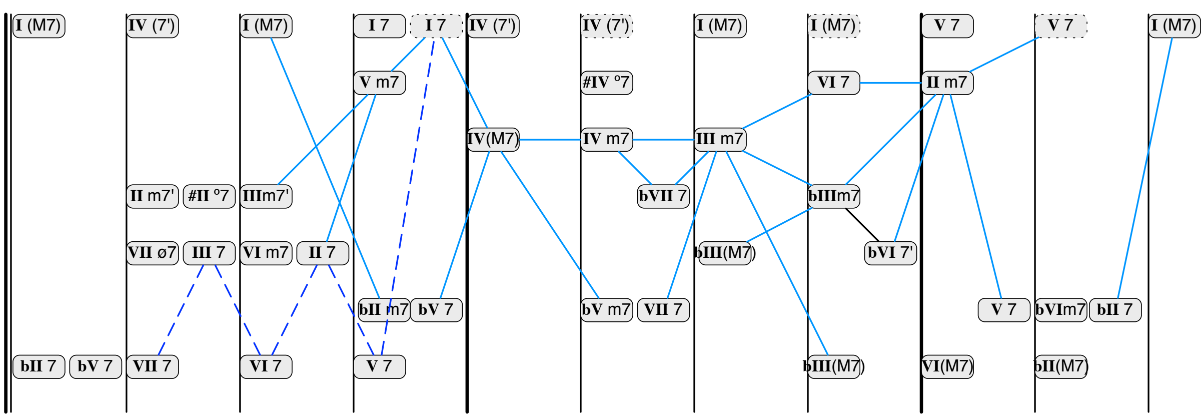


Fig. 16 Diagram of the chords substitution rules described by Steedman, 1984.

Although Steedman's article doesn't define a computational algorithm, it clearly formalizes the procedures for chord transformation that jazz musicians currently use. Mark Chemilier (Chemillier, 2004a, 2004b) implemented Steedman's grammar into a real-time computer program, using three of the substitution rules. This system, developed at the IRCAM, Paris, with Carlos Agon and Gerard Assayag, used Open Music and Max/MSP to calculate new chord sequences from a given chord structure by applying Steedman's rules.

In his broad article, (P. N. Johnson-Laird, 1991; Philip N. Johnson-Laird, 2002) devises a computational theory for chord sequence generation, based on Steedman and Cork's notion of "harmonic building blocks" (Cork, 1988). Steedman's grammar for chord substitution left open the question whether the substitution rules would still apply to the initial chord sequence as well. As Johnson-Laird demonstrates, the substitution rules attend to all possible progressions within the tonal framework, except for a sudden change of tonic, for which he defines a new rule. By reducing the basic building blocks of tonal chord sequences to three, Johnson-Laird creates a grammar for the generation of simple chord sequences. Applying the substitution rules and working backwards according to the circle of fifths, a great number of sequences can be produced, including the well-known "Rhythm changes"[5].

Extending on this method, researcher and musician François Pachet suggests an algorithm that introduces the idea of surprise in harmonic procedures (Pachet, 1999).

[5] "Rhythm Changes" is the chord progression of "I Got Rhythm", by George Gershwin, that became widely popular as an improvisational framework and used in several tunes by other composers, like in Duke Ellington's "Cotton Tail".

Pachet bases this idea in the observation and analysis of existing practical examples from the jazz repertoire and improvisation practice.

As will be described in Part 2, PocketBand allows the creation of different routes for harmonic variation, by defining alternate chords in the style-sheet templates. In a future development, the Sequencer can implement an algorithm to create harmonic variations automatically, derived from the same harmonic progression and substitution rules described by Steedman and Pachet.

4.3. Bass

Due to the non-repetitive nature of the walking bass technique, its use in computer software is actually quite limited. Commonly, walking bass lines in use are pre-recorded or manually written for the entire song length, whether as audio recordings or MIDI events. This practice has its roots in play-along recordings, like the widely known *Jamie Aebersold*'s[6] long list of score + CD Play-A-Long albums, with comping tracks recorded by real jazz musicians, allowing the practitioner to play-along with the recording. Computer software facilitates this method by easily allowing the independent mixing for each track, as well as change the tempo and transpose an audio or MIDI track, even on iOS devices with apps like Smudge Apps *Band* (Smudge Apps, 2017) and *iReal Pro* (http://irealpro.com) with multi-track recordings and mixer.

More advanced software use pre-recorded small phrases for each chord-type and/or chord progressions, which are then transposed and chained together according to some more or less intelligent algorithm. This seems to be the case with software like

[6] A list of Jamie Aebersold's albums can be found at http://www.jazzbooks.com

the extensive *Band-in-a-box* (see section 2.4.1), and iReal Pro (section 2.4.4), on mobile and desktop versions. This kind of implementation can use audio or MIDI clips. While audio clips keep all of the little nuances, sound and groove of the original player, the MIDI clips allow more flexibility for editing notes, instrument, and even tempo and phrase elements.

These implementations based on the use of pre-recorded phrases, whether audio or MIDI, have, however, some limitations:

- If the number of pre-programmed phrases is small, the output will easily sound repetitive;
- The larger the number of pre-programmed phrases, the larger the chances of melodic inconsistencies and non-musical results;
- In order to obtain smooth transitions between chords, the pre-programmed phrases have to be very neutral, resulting in a very neutral sounding bass line;
- It is not easy to handle less conventional harmonic progressions.

The walking bass algorithm I will describe in section 5.2 was based on these observations, and tries to avoid most of these problems by introducing a phrase generation routine based on the automatic calculation of phrases, that "stitch" the chord transitions according to some control parameters. These phrases and control parameters delineate contour profiles that are based on the way a jazz bassist thinks when improvising a walking bass line, "walking" from one chord to the next, instead of just using pre-defined patterns to fit the current chord.

4.4. Performance-oriented systems

Some of the existing research focuses on performance-oriented real-time systems. These type of systems aim to simulate not only the melodic, harmonic or rhythmic events, but also some of the characteristics of jazz performance, strongly based on improvisation. The system developed for this dissertation is mainly a performance system, and as such, this section is dedicated to some examples from existing research that I believe are related to my own work.

Some authors have addressed the creation of automatic accompaniment systems. Most of these systems use some type of algorithmic approach in order to respond to the input of a soloist. The system described in (Hidaka, Goto, & Muraoka, 1995) listens to the MIDI input of a keyboard or MIDI guitar and produces bass and drums lines automatically. This system is particularly focused on the anticipation of the soloist's intentions during the solo. For that, they defined five intention parameters. Namely: 1) excitement, 2) tension, 3) emphasis on chord, 4) chord substitution and 5) theme reprise. During the solo, the system is continually observing the live input for these parameters, to create a profile of the player and understand the player's intentions, in order to respond accordingly. In what seems to be a continuation of this project, a system named, VirJa Session (Virtual Jazz Session System) is presented in (Goto, 1996), that develops the concept of a system for human and computer players interacting and improvising. This system uses a network of computer workstations to handle the computation of the different elements and processes, and includes a video analysis system to use the analysis of the human's gestures to better understand and respond automatically. It also includes an interactive graphical representation of the virtual bassist and drummer. The presented tests in both articles were realized with a human pianist and a virtual bassist and drummer. Each of these intervenients correspond to an independent workstation, and the network and real and virtual cameras handle are the eyes of the players. Using the video analysis and MIDI data, the system uses the five parameters described above to detect the intentions of the players and generate a response and interplay between them.

Both articles however, mention very little detail on the musical output of these systems. There is a clear emphasis on the study and implementation of the collaborative system between humans and computers and on the analysis of the human player, but not much information on the quality of the generated materials, and on the influence on the human players.

The output is an automatic bassist mentions the use of a pre-existing database of patterns, that are then combined on every beat to generate the musical performance.

A somewhat related project named IVM - Interactive Virtual Musicians (Rowe & Singer, 1997), was created around the same time in New York by Robert Rowe and Eric Singer, where a multi-modal data acquisition system was implemented to analyze the input of live human musicians, and the interplay with virtual musicians that improvise with the human players. The multi-modal system includes video recognition algorithms, but also electronic sensors and speech-recognition algorithms to respond to voice commands. The musical output is based on Rowe's Cypher (Rowe, 1993), and can improvise constrained random lines and patterns based on chord changes and can also generate Markov-based improvisations.

In the "Walking Machine" (Nilsson, 2008) Nilsson uses a gamepad in a live performance, to control a system that generates a rhythm section, with bass, cymbal and drums. The player controls the overall behavior of one instrument at a time, while the generator algorithms, based on stochastic models, calculate the events at the note level. Unlike the previous systems, the walking machine's aim is not to emulate a traditional jazz band, but to explore in a free-jazz context.

4.5. *GimmeDaBlues*: a case study

GimmeDaBlues is a music application for iOS devices (iPhone, iPad, iPod) that allows users to play music in blues/jazz styles with a virtual jazz quartet with trumpet, piano, bass and drums, in real-time. While the trumpet and piano are played directly by the user, the bass and drums are generated automatically, responding to the user's input activity. The instrument selection aimed to represent a traditional standard jazz group formation and also to address four instruments that have very different characteristics and roles. For each one, a specific algorithm was developed, focusing on idiomatic features and role in the group, but also in the playability and ease of use that were key features in this project.

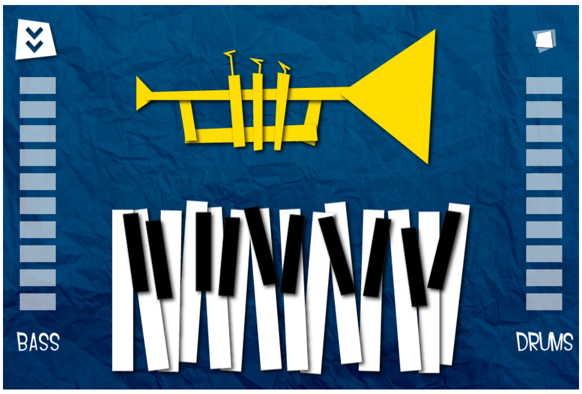


Fig. 17 *GimmeDaBlues*: Main window.

As the work presented in this thesis was based on the work developed in the creation of *GimmeDaBlues*, a description of this project is in order, in that it will clearly illustrate several of the main ideas, problems, challenges and solutions presented in the subsequent developments, however, more detailed information can be found in (Dias, Marques, Sioros, & Guedes, 2011; Dias et al., 2012).

4.5.1 Concept

GimmeDaBlues was the result of a set of initially separate research subjects on automatic generation of rhythm, harmony and solos, together with a strong background on interactive music systems and new interfaces for musical expression.

The base ideas were to approach the development of music algorithms for real time control from a performance perspective, and to maintain the control of the application as intuitive as possible. Two main aspects were thus particularly important to address. Namely, the algorithms had to be designed for live control, so that the user intervention actively influences the computed results during performance. Secondly, the control interface design strategies have to account for the physicality of playing the real instruments, even if simplified. The way to address these aspects was through the use of a meta-control over the main formal elements of the musical contents. The computer algorithm acts as a middle layer between the user's input and the musical output, taking care of the necessary calculations and synchronization with the musical form and different instruments.

The initial versions that integrated algorithmic generation with live control used a Nintendo Wii Remote controller and mainly focused on the rhythm. As the harmonic and melodic instruments were approached, the responsive multi-touch properties of the iPhone seemed to be a better solution, both from the application design and from the user interface points of view. The use of the iPhone's accelerometer was initially explored but proved to have a very low accuracy and time resolution when compared to the *Wiimote*. As such, the use of the device inclination was put aside very early[7].

4.5.2 Global Structure

The structure of the application can be separated in four main parts, namely the sequencer, the harmonic structure, the user interface and the instruments.

The sequencer leads the runtime operation of the application, running an ongoing clock that reads through the harmonic contents in real time and triggers the musical events. All the necessary data is sent to the instrument algorithms dynamically. The Piano and the Solo instruments require user input to trigger the events, while the Bass and Drums run automatically.

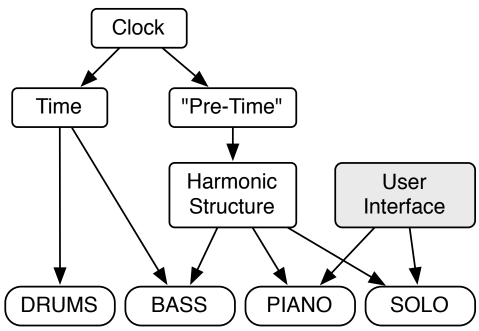


Fig. 18 *GimmeDaBlues*: global operation flowchart

[7] The inertia measurement accuracy and speed in recent iOS devices have improved significantly. Starting with the iPhone 5S, they now include a dedicated chip working exclusively for the readings of the inertial motion data.

In the above diagram, the “Clock”, “Time” and “Pre-Time” blocks are part of the Sequencer. As it will be explained below, the “Time” and “Pre-Time” correspond to two slightly separated timelines, created to allow the use of anticipated musical events.

The sequencer comprises an internal clock and a reader for the harmonic structure. It creates a timeline that will be played live and distribute a synchronized beat to all the instruments, as well as the current harmony to the trumpet, keyboard and bass.

From the main clock, two distinct synchronization signals are created: the Time and the “Pre-Time”. The Time is the regular tempo that will be perceived during performance. It drives the bass and drums algorithms. The “Pre-Time” is a slightly anticipated timeline that was created to address the technique of anticipating the beat, very commonly used by jazz musicians. The “Pre-Time” will read the chord structure, anticipating it by a little less than an eight-note, and sending the corresponding data to the solo and keyboard modules. Using this, if the user plays an anticipated note just before the next bar, it will correctly correspond to a note or chord of the harmony of the next bar.

4.5.3 Harmony

The chord sequence (or *chord changes*) and scales that constitutes the harmonic contents of the currently selected song is defined in the style sheet template. While the sequencer is running, it reads through the harmonic data and sends the current chord and scale information to the instrument algorithms that use harmonic data (all but the drums). Each instrument then uses this data differently according to it’s own needs, as will be explained below.

The harmony description comprises the chord sequence of the song, plus scale information, that will be used in the solo instrument. The chords are described in the style sheet as close as possible to the standard chord notation in jazz music, with a fundamental note and chord type (Example: “C 7”). Although there are different forms to symbolize the same chord type (Levine, 1995, p.ix), any form can be used, as long as both the chord type definition and the chord sequence use the same one. The distribution of the chord information is run in the “pre-time” pipeline in order to allow for chord anticipation.

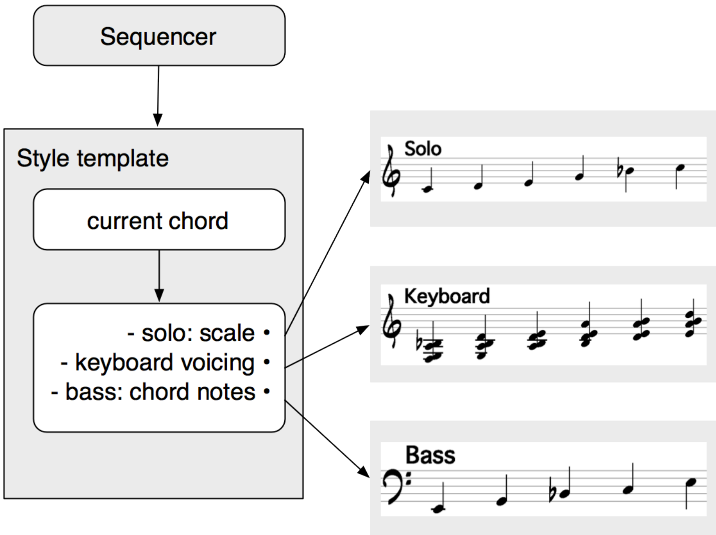


Fig. 19 Harmony parsing to the instruments.

4.5.4 Interface

As mentioned before, the interface was planned having in mind the idea of simplicity and ease of use, so that the user doesn’t have to read instructions or navigate through several buttons and menus in order to start playing. To accomplish this, all the essential performance controls are laid out in the same screen, in the main window.

The main window is divided in two main areas in the center, plus the volume controls for the bass and drums on the left and right edges of the screen (Fig. 17). The main areas on the center are the instrument areas. The upper half is the solo instrument while the lower half is the keyboard (chord) instrument. It is not expected for the user to play the right keys especially on such a messy keyboard. Specific keys are not important but moving from left to right on the keyboard results on chords or notes from low to high according to the zone pressed.

The instrument selection to include in the application aimed to represent a traditional standard jazz group formation and also to address four instruments that have very different characteristics and roles. For each one, a specific algorithm was developed, focusing on idiomatic features and role in the group, but also in the playability and ease of use that were key features in this project.

Along with the core interactive player, some other functionalities were included in the application. These are accessible from a drop-down menu (Fig. 20), by pressing the two-arrow button on the upper-left corner of the main window.

Fig. 20 Drop-down menu.

Pressing the Record button jumps to the main window and starts recording after the countdown. When the player stops the recording, GimmeDaBlues creates a MIDI file that can be named and stored in the Library. This file can later be exported to the computer by using iTunes app view. Once in the computer, the MIDI file can be opened in any DAW, Notation editor or any other MIDI compatible software.

The Setup menu allows the selection of instruments, style, root key and tempo.

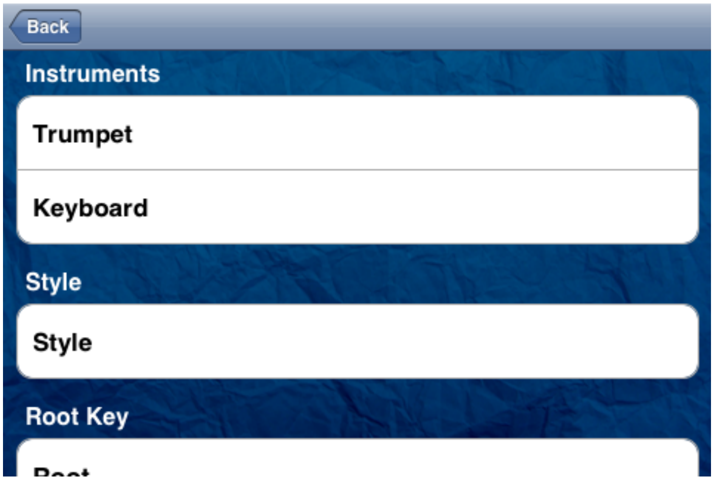


Fig. 21 The Setup menu.

4.5.5 Style definition

Because the application focused on blues, namely the common blues forms used in Jazz music, some different blues styles were selected from the traditional repertoire. *GimmeDaBlues* allows the selection between four different blues styles, namely “Classic Blues”, “Minor Blues”, “Bird Blues” and “Freeloader”, which cover a wide range of jazz tunes. Each style contains information about the formal structure, harmonic contents, chord *voicings* and scales. A style doesn’t include information about the melody, as it is not supposed to refer to one particular tune, but to a generic chord structure that can be found in several different tunes. “Freeloader” is an exception, in the sense that it refers to a particular chord structure, from the tune “Freddie the Freeloader”, by Miles Davis, which is a slight variation on the “Minor Blues” style. Also, there are current copyright issues that concern the use of melodies that would prevent their use. This, however, was not a problem because, as mentioned above, *GimmeDaBlues* never intended to address the melodic contents of the repertoire.

In order to have style definition, a style sheet template was devised that describes each style's different properties in a coherent structure. The style templates for the "Classic Blues" and "Minor Blues" are included in Appendix I. A parser then reads the templates and assigns the data to the appropriate modules of the program. The harmonic structure of the selected style as well as the tempo and key are sent to the sequencer, while the scale and voicing types are sent to the solo trumpet, the keyboard and the bass algorithms.

4.5.6 Conclusion

GimmeDaBlues was the result of many experiments that led to this format using the multi-touch capabilities and computation power of the iPhone and iPad. In turn, the solutions found for this application inspired a number of ideas for subsequent developments, which were the base ground for my own research project for this dissertation. Although there were many developments in all the contents of *GimmeDaBlues*, the basic format, structural elements and interaction approaches remained basically the same.

GimmeDaBlues was presented at the University of Porto and Coimbra (Portugal), the University of Texas at Austin (USA) and SigGraph Asia (Singapore). In January 2012 received the first prize of the Portuguese national "ZON Multimedia Prize competition 2011", in the category "Multimedia applications and contents".

PART II - TOWARDS A COMPUTER-MEDIATED MUSICAL EXPERIENCE

Chapter 5. Instrument Algorithms

This chapter will present the instrument algorithms in detail. Although they derive directly from the ones created for *GimmeDaBlues*, they are the result of several distinct versions and developments that led to completely new algorithms. The piano and bass algorithms were completely rebuilt, while the strategies for the implementation of the solo and drums parts remained basically the same for this system. Some experiments and developments were realized, and essential adaptations were needed, though, to accommodate to the new clock and sequencer system, both regarding the metrical data as well as the harmony data, in the case of the solo instrument. In the drums part, important adjustments to the recombination algorithm were made, and adaptations to the automatically manage the activity level arriving from the Listener module.

The development of the instrument algorithms was a particularly exciting part of this research, as it required a deep observation of the way a human musician's mind thinks during improvisation, as well as attend to the idiomatic characteristics of the instruments. Each of the instrument algorithms presented here was custom tailored to account for these aspects. This is especially necessary because none of them uses pre-recorded phrases, and, as such, they need different strategies according to their particular capabilities. I strongly believe that very exciting new algorithms will continue to be developed in a near future, that mimic the behavior of human players improvising, that incorporate the idiomatic features of the specific instruments;

5.1. Piano comping

The main role of the piano for this purpose is to support a harmonic base for the melody and solo parts. Other polyphonic instruments are very commonly used, namely the guitar and other keyboard instruments like the electric piano or the electric organ.

This section will focus on the main aspects taken under consideration for the development of the present voicing algorithm.

Very few strategies for devising chord voicings have been approached. (Emura, Miura, & Yanagida, 2006; Kitahara, Katsura, Katayose, & Nagata, 2008; Watanabe, Watanabe, & Emura, 2008) (see chapter 4). These studies focus on the harmonization and voicing calculation of a given melodic line, using a rule-based implementation of jazz chord theory. Although some of the problems and solutions have common aspects, the algorithm here described departs from a very distinct base concept and aim, which impose important differences, namely:

- it was created for a real-time interface where the user can freely play in any pitch register at any time. This means that the user has an important control on *sequenciality* (Kitahara et al., 2008), or horizontal voice-leading and inversion calculation between consecutive chords;
- it is planned for live jazz comping, where the melody and solo come from an external soloist, hence there is no prediction of the notes that will be played melodically;

5.1.1 Voicings

Regular chords in tonal music have at least three notes, starting from the fundamental (base) note, and adding up consecutive major or minor third intervals. The basic three note chords (triads) can be expanded by adding extensions notes, which are built following a logic of ascending thirds, respectively the 7^{th}, 9^{th}, 11^{th} or 13^{th} degrees, in any combination, and played in different octaves. So, far from being basic and straightforward, a chord can be played in many different combinations of these notes, each combination forming a configuration called a *voicing*. In Jazz, the term *voicing* refers to the way a given chord is played. The same chord can be played in innumerous different ways (Dias, Guedes, & Marques, 2014). For this study, the possibilities for voicing variety were traduced in four parameters or characteristics: 1) position, 2) register, 3) wideness and 4) note selection. The decision on what voicing to use for a certain chord at a given moment can depend on several aspects like musical style, personal style, voice leading, density, and it is usually also directly influenced by what is being played by the other musicians, mainly the soloist.

Position

The position corresponds to the order of the notes of the chord, as mentioned above, and includes the initial root position, as well as the inversions. The position is also commonly referred to as being open or closed, depending on the interval order (see for example ("Voicing (music)," n.d.). For the sake of clarity, and because at the complexity level of the concept of voicing here described this designation wouldn't be so relevant as only very few voicing are closed, this aspect is included in this study as openness (see below).

Register

The register corresponds to the octaves the chord is being played on. The same voicing can sound very differently in different octaves due to the way we perceive the frequency scale. One single note, interval or chord played in a low octave sounds much denser than in a higher register. Hence, a well-balanced voicing tends to avoid small intervals in the lower octaves and use the notes distributed as uniformly as possible.

Openness

By openness I refer to how open (wide) or closed (narrow) the voicing is. The basic configuration of a chord, ordered by major and minor third intervals, or their inversions, as long as the chord degrees are consecutive, are closed positions. However, this can lead to some ambiguity when considering chord extensions above the octave.

For example, a possible voicing for a C major chord can have an added note D (a tension), which is the 2th degree (Fig. 22a).

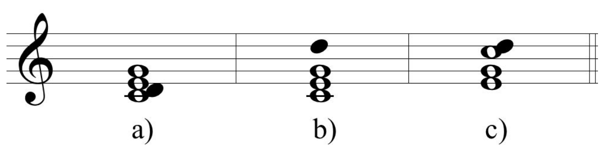


Fig. 22 Open and closed positions with chord extensions

The procedure for chord construction tells us that this D is indeed an extension of the chord, in this case, the 9th degree of the chord. This is because the way chords are built is by adding an extra third interval, and as such, the D is the chord extension right after the 7th degree, the note B. So, if we consider the root position of the chord including the 9th, and taking into account the above statement that a chord is in its closed position if the chord notes appear in its consecutive order, the chord with the D above the octave is the closed position (Fig. 22b). From a register and range perspective, however, the chord with the D inside the octave (Fig. 22a) is comparatively more closed. In addition, if we take the chord in the first inversion (Fig. 22c), the same high D (9^{th} degree) is now the lowest one possible, and as such, there's no other closer form than this one.

In order to avoid any ambiguity, I consider only the "wideness" of the voicing, meaning the distance from the lowest note to the highest, independently of the degree. In the example above, the voicing in Fig. 22a is a perfect 5^{th}, while the voicing in Fig. 22b is a major 9^{th}, and thus has a greater wideness factor.

Considering two-hand piano comping, voicing chords can spread for two or three octaves, and even more, as hands move apart.

Note selection

The selection of notes to include in a voicing is by far the most complex parameter, and inherently subjective. Much more than in any other music language, in jazz it is a common practice to make alterations to the chords, which can radically change the resulting sonority. Each jazz pianist has his/her own voicing style, and depending on the character they want to imprint on the tune, the voicing can vary significantly its density, by removing or adding notes, and its "color", by choosing note closer or further from the basic triad, as well as the way they are distributed - the intervals between them.

As mentioned before, there are several ways that a chord can be altered, regarding its note contents. The main one is the expansion of the chord adding extension notes: 7^{th}, 9^{th}, 11^{th} and 13^{th}. These extensions can appear in any form, and don't have to show sequentially. Notes can also be omitted from the base chord. For example, if the band has a bass player, it is common for some of the piano chord voicings to not include the tonic, as it will probably be played in the bass. Other notes outside of the scale can also be used to create a rough sounding chord.

Many aspects can influence the selection of the notes, including external to the vertical harmony, like passage notes or strange notes played by other band members.

5.1.2 Interface

The interface is planned for iPad/iPhone/iPod, but any other multi-touch device can be used, as long as it allows sending OSC control messages to the computer. A version to use the computer keyboard was also created. Other devices with pads or Ableton Live clip launcher buttons like Ableton's own *Push* controller can be used as well.

The software was developed in the Max programming environment (www.cycling74.com), using both regular Max objects and javascript code (also inside Max). The iOS device uses *Fantastick* (Pink Twins, 2013) an application that sends multi-touch data by Wi-Fi. Using the UDP network protocol, the data is received in Max.

The piano interface and algorithm were developed having in mind the simulation of the experience of playing jazz piano with a fair degree of sophistication in the resulting sound. Given that on a conventional keyboard, with direct control over each individual key, and the four parameters described earlier, the number of possible voicings for a given chord is far too great to attain with a simple-to-use approach and with the limitations of the physical device, the voicing algorithm developed was a compromise between complexity and usability.

Very early in the research, an assumption was made by analyzing several aspects of the jazz piano practice. The assumption that we can, to some measure, separate the decision over the notes being played from the decision of the actual action of playing, in a rhythmic, percussive sense. We believe that an important part of the performative aspect of the live improvisation experience in jazz - and most likely in every other improvisational music - can be thought of as rhythmical, in the sense that the action of deciding WHEN to play a note or chord in any harmonic or melodic instrument is not so different from the same action by the drummer or percussionist. Of course, there are several other decisions implied, but they are mostly idiomatic and related to the role assigned to each of them. The decision over WHAT notes to play is the result of the learning process about harmony and tonal music grammar, including chord progressions, scales, voicings and melodic patterns.

Simultaneously, some decisions about the notes to play can be thought of in a sort of meta-level control, independently of the actual note selection. The register and wideness parameters mentioned above, that determine WHERE the notes are being played, can be decided in order to complement the soloist, by playing in opposite registers, or conversely to match the soloist by playing in coincident registers.

This “meta-control” can be observed also by analyzing the overall melodic contours that can happen in groups of a few beats or bars. These contours delineate phrases and behaviors that can greatly contribute to the overall comping quality, and characterize some of the personal style of the player, and can be considered, analyzed and controlled independently of which exact notes are being played. In other words, the present interface and algorithm provides the WHAT, leaving the WHEN, WHERE and the WHY to the user.

5.1.3 User input

The user interface and interaction modes are very similar to the one developed for the *GimmeDaBlues* app, since it proved quite successful in providing a natural and intuitive way to experience most of the afore mentioned rhythmical feeling and meta-control over the note and chord events.

It comprises a virtual keyboard with a variable number of virtual keys (according to the device being used) that allow playing with one or two fingers, each finger corresponding to one hand of the player. With every chord change, the voicings are calculated and dynamically mapped to the appropriate keys, so that every key plays a useful and correct note or voicing. With the iPad multi-touch screen, a total of 16 keys were used (see Fig. 23). Like in a conventional piano, the lower pitches are on the left side and the higher pitches on the right side.

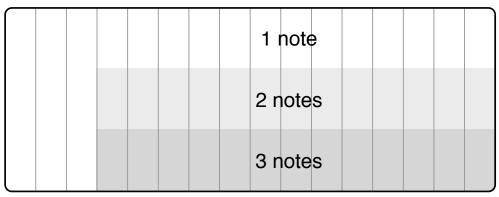


Fig. 23 Virtual keyboard on the iPad screen.

While the keyboard in *GimmeDaBlues* had two rows across the entire width, the new version presents three, starting on the fourth key. The different rows correspond to the number of notes to include in a chord. A finger in the lower (darker) row will trigger a three-note voicing, the middle row will trigger two note voicings, and the top row (represented in white) plays only one note. The first three keys on the left side are reserved respectively to the root, fifth and octave of the current chord being played, and trigger one single note. This configuration allows for a fast, intuitive and versatile playing technique, by combining two fingers in different horizontal and vertical positions. The top, single note row, allows the user to play simple melodic phrases that can complement the chords, dialogue with the soloist or other accompaniment instruments, or even be used to play the main or secondary melody, giving a whole new range of possibilities.

The selection and mapping process of the assigned notes to the keys will be explained below.

5.1.4 Voicing algorithm

The voicing algorithm consists of three main sections: the Voicing Map Calculator, the User Input Mapping, and the Player (see Fig. 24). Every chord change in the song will be read, formatted and sent in real-time by the sequencer. The Voicing Map Calculator will create a list of voicings, with one voicing for each virtual key that will be available for the Chord Player section. When the user touches a key, the key number and touch coordinates will be sent to the Player, that will read the corresponding voicing from the voicing map. The final sounding voicing will be a subset of one, two or three notes that will be different depending on the key number and vertical touch position that was used.

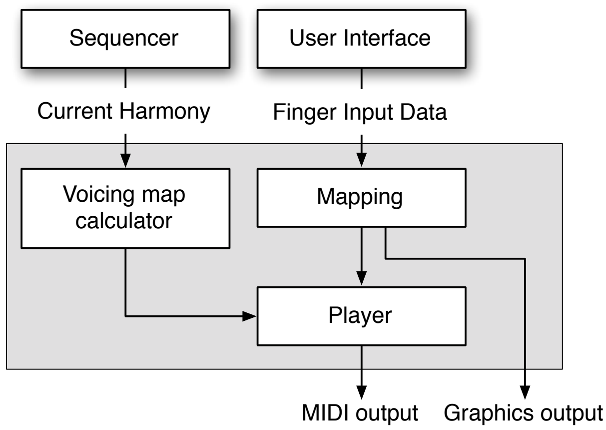


Fig. 24 Global structure of the voicing algorithm.

Chord information input

The algorithm takes as real time input the chord data of the current harmony arriving from the sequencer timeline. The real time data includes the chord name and type, as well as an associated scale. For example, "C 7 mixo" specifies the C chord, of type Dominant 7^{th}, and a scale, in this case a mixolidian mode. The chord root (C) will be used as Pitch Class, with a transposition factor, which for C is zero. The chord type correspondence is described in the style template, where the "7" will correspond to a predefined Dominant 7^{th} voicing.

Scales

The scales can be defined in the song template, and correspond to the notes that will be assigned to the virtual keys. The term scale is used here to refer to any combination of notes that will be available for the user to play, and doesn't necessarily correspond to a conventional scale. It can be a conventional major or minor scale but it can also be a mode or a subset of scale or mode degrees.

In order to have a better keyboard range and to avoid possible incompatibilities between the scale notes and chord notes, better results are obtained using non-complete subsets. The “mixo” (mixolidian mode) in the example above can for example be specified as the subset [0 2 4 7 9 10], which corresponds to the mixolidian mode without the 4^{th} degree.

Voicing format

The voicings are described as a list of ordered pitch intervals, according to the chord type specification in the harmonic structure. This list defines the notes that can be included in the voicing, but also each one’s index number, by ordering them from left to right according to their priority. A 7^{th} chord for example, can be defined as “10 4 2 9 7 0”. Pitch Class “10” (Bb) corresponds to the minor 7^{th} of the Dominant 7^{th} chord, “4” (E) to the major 3^{rd}, “2” (D) to the major 2^{nd} and so forth.

Voicing calculation

The Voicing Map Calculator receives the voicing for the current harmony and calculates all the voicings for the entire keyboard. This received voicing is not, however, the final voicing that will be triggered. Instead, this list is a sort of map or key to the construction of the final voicings that will be played. The starting point for the calculation of the voicings is the scale. The scale is mapped to the keyboard. Then, for each key a voicing is calculated by finding the defined degrees below the scale note.

Having the example mentioned before: “C 7 mixo”, with the scale “mixo” “0, 2, 4, 7, 9, 10” and the voicing “key” “10, 4, 2, 9, 7, 0”, the calculator algorithm will map the scale to the keys:

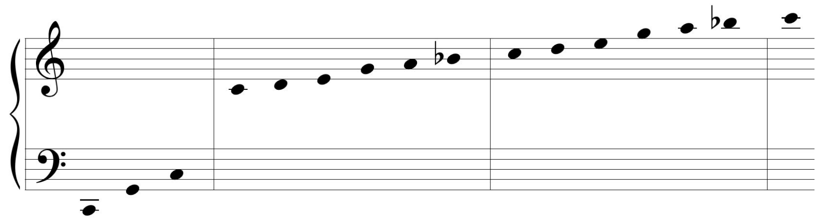


Fig. 25 Scale mapped to the virtual keys.

And then calculate the rest of the notes for each voicing, using the order defined in the key. For this example, the resulting voicings would be:

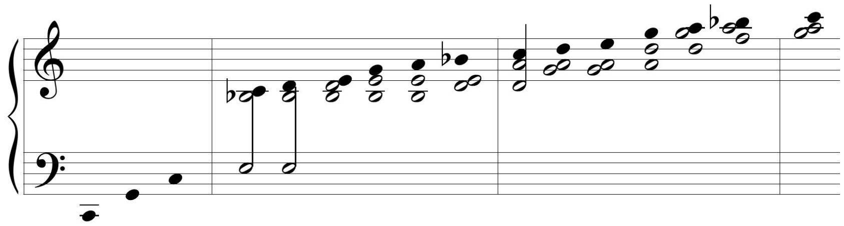


Fig. 26 Scale and voicings for each key.

The black notes are the notes of the scale. The white notes are the voicing notes automatically calculated. As mentioned before, the first three keys always have the root, fifth and the octave. From the fourth key on, the algorithm will calculate the voicing by searching for the degrees defined in the “voicing key”, by their priority.

In the fourth key, where the scale note is C3 (middle C), the voicing notes are a Bb2 and a E2. These correspond respectively to the first two degrees in the voicing key “10, 4”. The same happens in the next voicing, D3. In the sixth however, because the scale note is already an E, the algorithm will advance one index and look for a “2”, which is the next degree in the list.

The algorithm allows also changing the starting index of the lookup list, in any scale degree. In the example voicings above, a change is visible from the scale note C4 on, where the voicing notes are not the same (E, Bb) but instead are “D, A”, which correspond to the “2, 9” defined in the list. This is intended in order to avoid repetition and to create more variety.

Using two hands, the user can play rather sophisticated voicings.

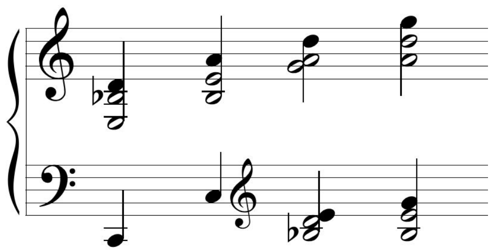


Fig. 27 Four two-hand voicing examples.

5.1.5 Player

The “Player” receives the voicing map from the voicing map calculator and manages the input data arriving from the user interface, to play the corresponding events.

Using the vertical position of the fingers, it uses one, two or all three notes of the voicing, following the same order of the “voicing key”. When playing only one note, it will always be the scale note. When playing two or three notes, the second and third are always below the scale note (see Fig. 26). This way, the highest pitch in any resulting voicing corresponds to the scale note of the pressed key. This idea comes from the fact that the highest note in a chord is the one we generally hear the most, and as such, has an enhanced importance in voice-leading, which reflects on the practice of piano playing, where the highest note in a chord is the lead tone in the melody. Using this feature, the interface proved much more intuitive and effective.

5.2. Walking Bass

The bass algorithm in *GimmeDaBlues* served the basic function of marking the beat, while playing the basic chord notes. On the first beat of the bar it always played the root note of the chord, while on the other beats any note of the chord can be played. It provided a stable base for the song, keeping the tempo, metrical structure and harmonic progression easy to understand and follow, which also allow the other instruments to be more independent and free to improvise.

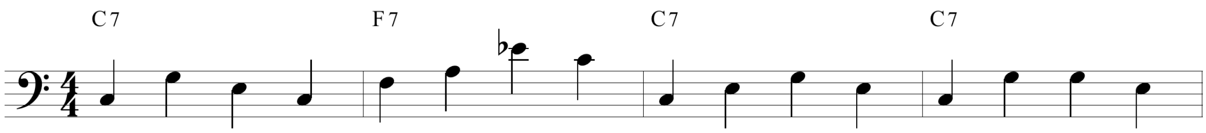


Fig. 28 Example bass line (4 bars) of a blues form produced by the *GimmeDaBlues* algorithm.

Although this is effective for presenting the harmony and pulse, it is clearly poor and simplistic, when compared with the lines of a real bass player. Some of the aspects that can contribute to enrich the walking bass lines, like using notes outside of the chord or chromatic passing notes were left out. Other important aspects relate to the rhythm. Although the base idea of the walking bass is to keep the steady pulse of the beat, a good walking bass line includes many small rhythmic variations, like ghost notes (subtle *appogiaturas*, sometimes almost imperceptible), eight-notes and triplets. Also, the bass can play longer notes, to create some variation of the otherwise very repetitive pulse.

For this dissertation, a completely new walking bass algorithm was developed from the ground up, based on the automatic generation of phrases that define shapes or contours by interpolating from one chord to the next, calculating the correct in-between notes to provide natural and musical bass lines. The text below was partially published in (Dias & Guedes, 2013).

The phrase generation algorithm consists basically in three stages: Target Note calculation, Trajectory calculation, and an event manager, the Player.

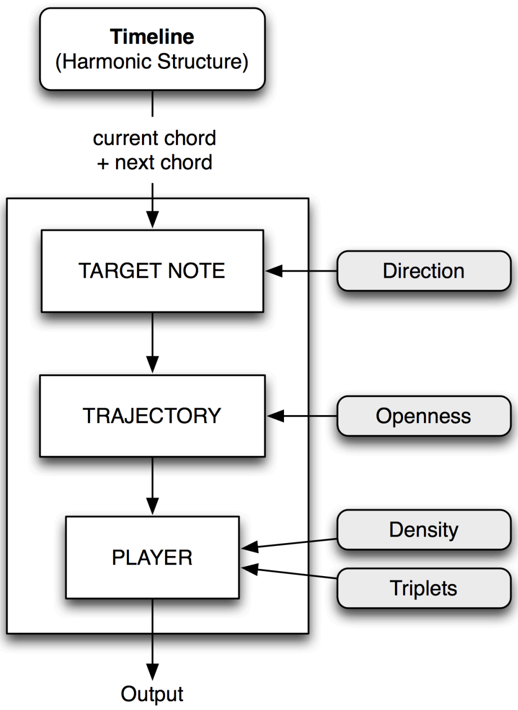


Fig. 29 Global structure of the walking bass algorithm.

5.2.1 Target note calculation

The algorithm needs to know the current chord and the next one in order to be able to calculate a phrase. The target note is the last note of the phrase to be generated. A simple probabilistic algorithm chooses which of the notes belonging to the next chord will be used e.g. root, 3rd, 5th, 7th, etc. Currently, in order to maintain a strong sense of the base harmony, a setting of 100% probabilities of choosing the root note of the chord is used. Then, according to the current note and to a direction parameter, the algorithm will find the chosen chord note in the right octave.

The direction parameter defines whether the target note will be selected up or down, relatively to the current note. There are five different settings: lowest, down, nearest, up, and highest. The down and up settings tell the algorithm to search for the nearest note in that direction, while with the lowest and highest settings, the algorithm will select the lowest and highest note in the instruments range. This parameter can be defined manually or automatically. So, for example, considering a double bass instrument defined with a range from E0 to G3 (having C3 as the middle C), if the current note is a C2, and the target note is an F, the direction parameter will define which F will be selected. The down setting would select F1, while the up setting would select F2. The lowest note setting will return F0 - the lowest F on the defined range - and the highest setting will return F3 the highest F on the defined range.

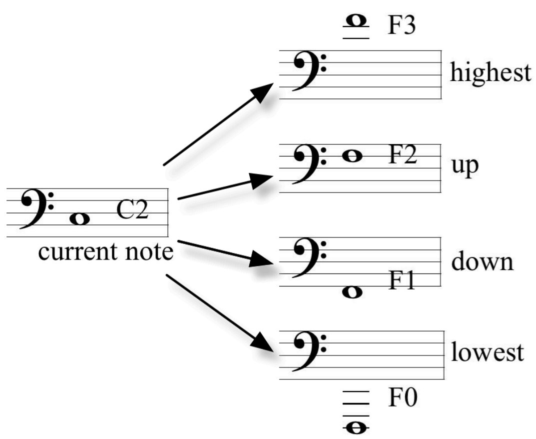


Fig. 30 target direction selection

With the ”nearest” setting selected, the algorithm will automatically go up or down, choosing the F which is nearest to the current C2, which will be F2, because its a Perfect 4th interval, while F1 would be a Perfect 5th.

5.2.2 Trajectory

The trajectory is constituted by a selection of notes that define the path that the bass line will take from the starting (current) note to the final (target) note. In a typical case for a chord duration of one bar in a 4/4 measure with the bass playing quarter notes, the complete generated bass line will have five notes, in which the fifth is the first note of the next measure. The three passing notes between the starting note to the target note are calculated recursively, going from the strongest beats of the measure to the weakest, and depending on the trajectory calculation algorithm.

An openness parameter will set how direct or indirect the path will be, influencing the selection of the middle notes in the calculated bass line. The name relates to the notion of closed and open forms of a chord. The closed position will be the more direct path to the target note, while more open path will tend to use a wider trajectory, using chord notes in an open form.

The notes to use will be drawn from chord tones, scale degrees and chromatic inflections, according to each steps beat position. The stronger beats of the bar will tend to have chord tones, while the weak beats will tend to have scale notes acting as passing notes from one chord note to the next. The last beat of the bar can also be a chromatic approximation to the target note. This is a very commonly used technique, as the chromatic passing note creates a strong attraction to the target note, emphasizing it as well as the sense of direction in the melodic phrase.

Fig. 31 shows four examples of possible trajectories of the calculation of a bass line for a C7 F7 progression, where the initial note is C2, and the target note is F2.

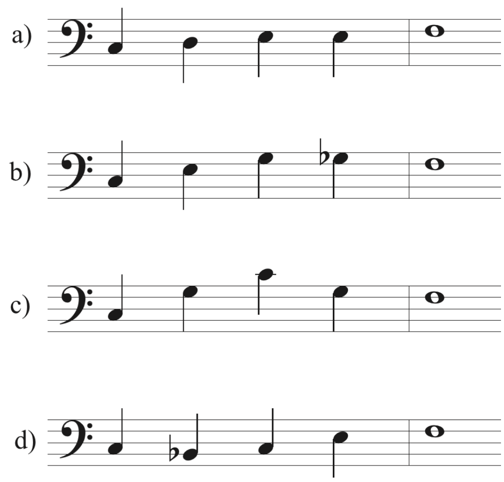


Fig. 31 Possible trajectories.

While the four examples have the same starting and target notes, the trajectories are different. In a), the trajectory follows a direct line from C to F. In examples b) and c), there is an upward - downward motion, with the higher note in the third beat, but the openness parameter is higher, producing a wider phrase. The F# in b) is a chromatic passage note from G to F. The phrase in example d) has a downward - upward motion, having the lower note – Bb – in the second beat.

5.2.3 Player

The player executes two main functions: 1 – Executes some transformations on the calculated phrases, received from the Target Note and Trajectory sections; 2 – Formats and sends the data to the output MIDI system.

Density

The phrases generated by the Target Note and Trajectory sections have all the beats of the bar. One of the functions of the Player is to manage these phrases by letting pass all the beats of the phrase, or omitting some of the notes. By acting as a filter, the Player implements a setting for the Density of the phrase. With a density setting at the maximum value, all the notes in a phrase pass to the output. By reducing the density, the output phrase will have fewer notes.

The criteria for this filter are the same of the metrical relevance that was used initially to calculate the phrase in the Trajectory section. As the phrases are generated by prioritizing the different beats of the bar, filtering according to the same criteria will guarantee that the phrases won't loose their tonal logic and directional sense.

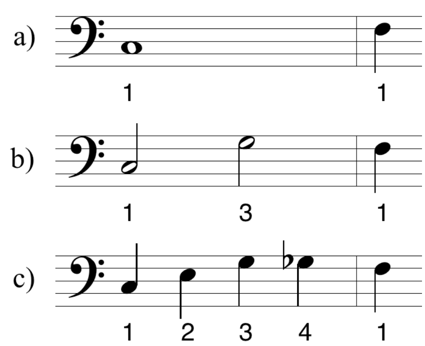


Fig. 32 A phrase with different density settings

The example phrase in Fig. 32 is displayed with three different density settings. In the lowest value, a), only the first beat is played. In b), the first and third beats are played, and in c), with the highest density value, all the notes are played.

This density parameter is fundamental to produce more or less intense walking bass lines that can correspond to different sections of a song, or a solo.

Triplets

Although the construction of the phrases are the base of the walking bass technique, there are several other aspects regarding the notes, rhythm and articulations that have an important role in a good performance. These aspects, here referred to as ornaments, are little nuances and additions to the phrases that don't change nor define the main contents of the phrases, but nevertheless can contribute considerably to the quality and the dynamic of the walking bass lines.

The current implementation allows for the use of eighth-note triplet variations (or eighth-notes with a swing feeling) that can be set probabilistically. This is one of the most common rhythmic variations in the walking bass technique, in which some of the notes are anticipated by one triplet (or swinged eighth) with the same or another pitch. The control is done by a single percentage value, setting the probability factor.

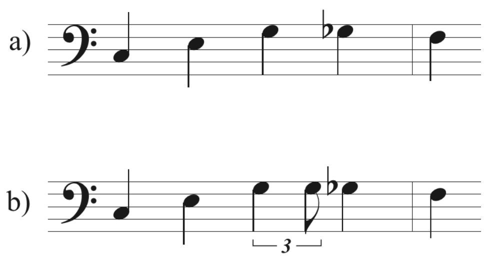


Fig. 33 Triplets.

The example above shows one of the phrases from the previous example, going from C2 to F2. In b), the use of a triplet probability factor produced an added G before the F#. This would be a very common phrase for a walking bass line by a real bassist, and the triplet ornament adds extra groove and richness to the initial phrase. In the triplet percentage scale, a value of 0% would correspond to zero triplets added, and a value of 100% corresponds to a triplet added in all the beats in a bar. The usage tests in comparison with real walking bass lines showed that a value of around 30% provided the most convincing results.

Dynamics

Another important aspect that characterizes a good sounding bass line is the dynamics for each note. Although the phrases are continuous, in traditional jazz styles there is a tendency to reinforce the weaker beats of the bar, namely by slightly accentuating the second and fourth beats. This is usually a subtle effect on the bass, but it can make a difference that we can feel, even if we don't perceive it. This accentuation can be observed more clearly on the drums, where the second and fourth beats are very clearly marked on the hi-hat or ride cymbals.

The Player takes care of this aspect by imposing a dynamic level to the notes, which can be different according to their position in the metric grid. The second and fourth beats are slightly accentuated relatively to the first and third.

5.2.4 Control

The algorithm was designed for real time, but the user control over the parameters does not affect the results directly, at a note-by-note level, but at a bar level. This means that changing the parameters influences the overall contour of the produced lines but not for an immediate and direct response for each note. Because the algorithm calculates a line for each bar, changing the parameters will influence the next bar but not the currently playing one.

The available parameters are: Direction, Openness, Density and Triplets. These four parameters provide a rather versatile control over the resulting phrases and overall behavior of the virtual bassist. For the control over these parameters, several different approaches can be used. The combination of the Direction selection and the trajectory Openness provide a contour-based definition of the walking bass line, which not only creates smooth and natural lines but also allows an effective and intuitive control in interactive real-time implementations.

A simple example of a possible manual controller for this walking bass generator would be a joystick type controller, where the vertical axis would control the direction parameter, while the horizontal would control the openness parameter (Fig. 34).

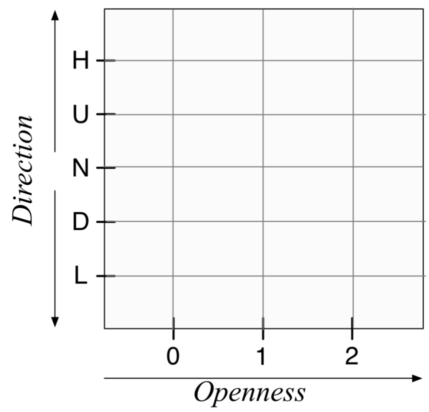


Fig. 34 Control of the walking bass algorithm.

The use of this bi-dimensional layout of the *Direction* and *Openness* parameters provide an intuitive control over the output contour of the generator's phrases.

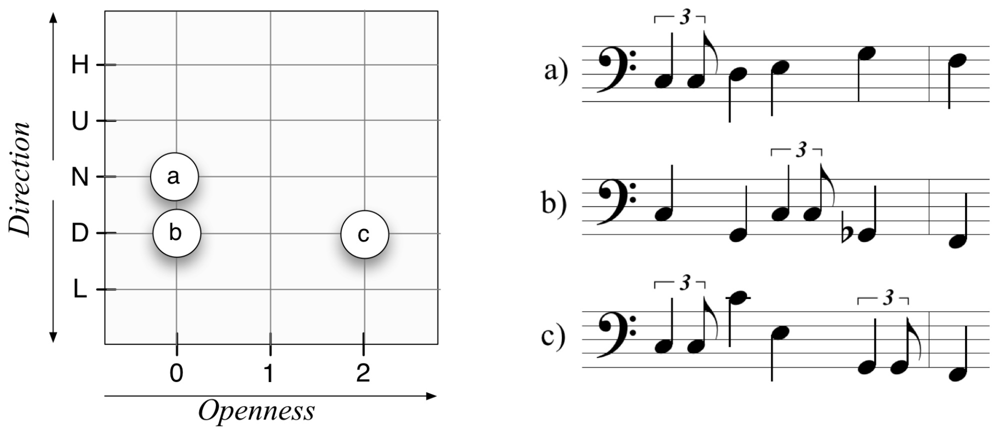


Fig. 35 Three example settings of the Direction an Openness parameters, and three example output phrases.

Fig. 35 shows three example settings of the direction and openness parameters: a) direction: nearest / openness: low; b) direction: down / openness: low; c) direction: down / openness: high. On the right, example output phrases using the corresponding settings.

These controls, along with the Density and Triplets parameters, allow a very efficient control of the algorithm, whether for a manual control or automated. In this application, an automatic mode is implemented, so that the bass can be completely autonomous, or respond automatically to external input from the user or from the other instruments. In the present implementation, the contour profile is controlled automatically, using a simple Markov chain, and the density level is influenced by the activity level arriving from the solo and keyboard instruments. As such, the virtual bassist follows the human players, by responding more actively when the players are more active.

5.3. Solo instruments

The solo instrument, commonly a monophonic instrument, like the saxophone or trumpet, is implemented as a set of dynamically changing scales that fit the currently playing harmony while the song is playing. Similarly to the piano algorithm, the solo part is not meant as an automatic generator. It is meant to be manually played by the user, and as such provide an experience, at least partially close to the actual experience of playing an instrument. The mapping algorithm of the solo part takes care of the calculation of the notes to be used at any given moment in the harmonic sequence it receives from the sequencer. The solo receives a scale tag, which will correspond to an optimized scale template. These scale templates are pre-defined to better fit the musical contexts and the interface characteristics, and can be designed arbitrarily and included in the style-sheet. These scales are built dynamically during runtime, according to a custom algorithm that calculates the correct notes for each harmony, following the definitions in the song style-sheet.

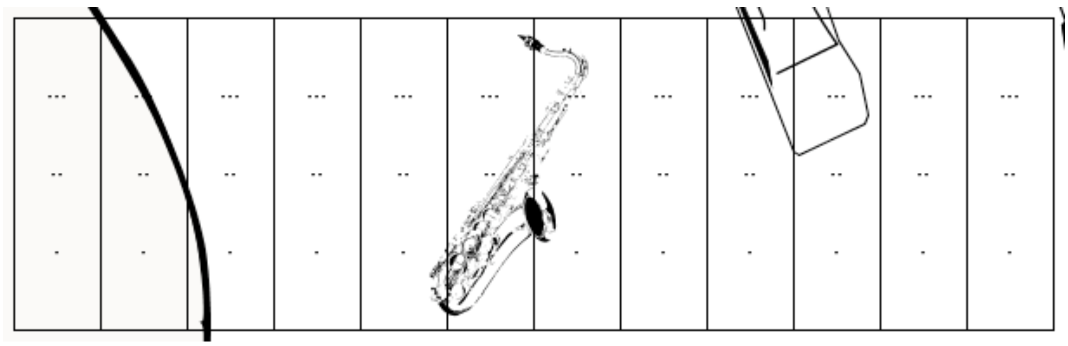


Fig. 36 Example solo instrument interface from "MyJazzBand" (tenor saxophone).

The control is made with control messages that trigger one of the mapped notes. Both PocketBand and MyJazzBand present the same approach of a two-dimensional axis, as they were planned for touch events in a touch-screen. The control message indicates the horizontal and vertical position coordinates of each touch event. From the horizontal coordinate, the program calculates which key was pressed, triggering the corresponding note index. The vertical position is used for the dynamics.

The saxophone keyboard interface from MyJazzBand, shown in Fig. 36, has 12 virtual keys. Each key will play a note, but the note changes dynamically according to the harmonic changes and the definitions in the style-sheet.

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
midi note	60	62	64	65	67	69	70	72	74	76	77	79	81	82	84

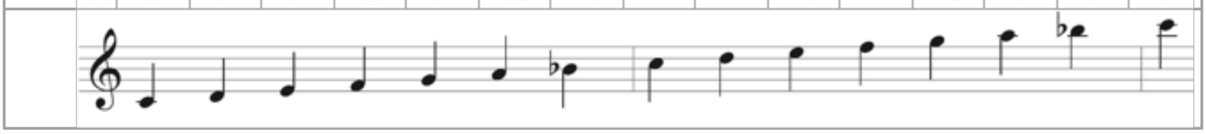



Fig. 37 Example scale produced with an indication of the mixolydian scale.

Fig. 37 shows an example of a scale mapping of a mixolydian scale in a C7 chord in the harmonic sequence. The two top rows show the key index and the midi note assigned to each one. The example tenor saxophone interface in Fig. 36 has twelve keys. In this case, the notes would range from index 0 (note C) to index 11 (note G). The range of the instrument is then taken to account, so that the octave lays within the playable range of the instrument.

In the vertical axis, we can control the dynamics of the played notes. The higher the touch in the key, the louder it sounds.

5.4. Drums

The algorithmic generation of the drums part is based on the stochastic recombination of sub-beat-level rhythmic cells from a pre-defined set of rules and patterns for each of the drum elements. By combining a small number of elementary rhythmic patterns, tailored for Blues and Jazz, an effective drum part is obtained, which, as with the bass and piano algorithms, is meant as a comping device.

The algorithm is divided in two main branches. One for the generation of the kick and snare parts, and other for the ride, hi-hat and crash cymbals. The reason for this is the inter-dependency between these elements. The kick and snare usually combine and complement each other, and rarely play at the same time (see Fig. 38).

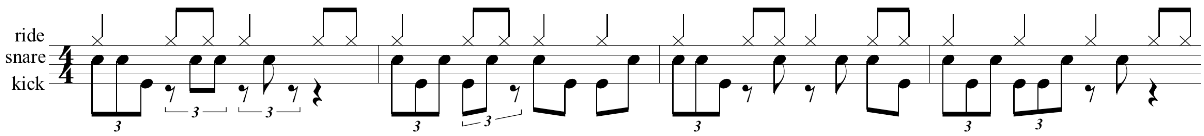
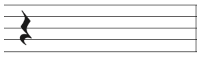
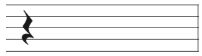
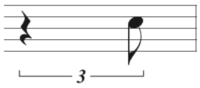
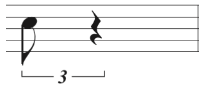
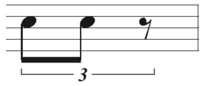


Fig. 38 Example jazz drum score (adapted from (Pickering, 1978, p.47)).

Snare/Bass drum

For the snare and bass drum, two sets of four sub-beat level patterns were defined, based on the playing technic in jazz drums. These patterns are one beat long (one quarter note), with a ternary sub-division, with each eight-note triplet represented by a binary array that can be 1 or 0, depending if it is a trigger or a rest (see Table 5). They are ordered according to the number of triggers, so the higher the index, the higher the number of triggers.

Main patterns		Secondary patterns	
0, 0 0 0;		0, 0 0 0;	
1, 0 0 1;		1, 1 0 0;	
2, 1 0 1;		2, 1 1 0;	
3, 0 1 1;		3, 1 1 1;	

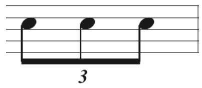



Table 5 Snare and bass drum trigger patterns.

A probabilistic function then selects the patterns, according to a density parameter. For a given density value, a certain pattern has a greater probability of being selected, followed by its adjacent neighbors, in a Gaussian-like distribution. The two sets of patterns (main and secondary) are also selected according to a simple stochastic algorithm: the main patterns have a constant 80% probability of being selected, while the secondary have a probability of 20%. A non-repetition filter was implemented, to prevent the patterns from repeating. Although it is not uncommon for a jazz drummer to occasionally repeat a pattern, it is much more as an exception than as a rule. Also, the occasional repetition of patterns is usually due to some occasional melodic phrases or due to the interchange of phrases with another player. For this purpose, the use of the non-repetition filter proved to be more effective in providing interesting results. Following the filter, yet another probabilistic stage was implemented, to manage the selection between the snare and the bass drum. As explained before, the interchanging nature of the snare and bass drum roles in jazz playing create a complementary motion between the two. For this stage, a stochastic algorithm was used. Each trigger event is routed to the snare or bass, according to a set of pre-defined probabilities, which, in this case are different for each beat of the bar.

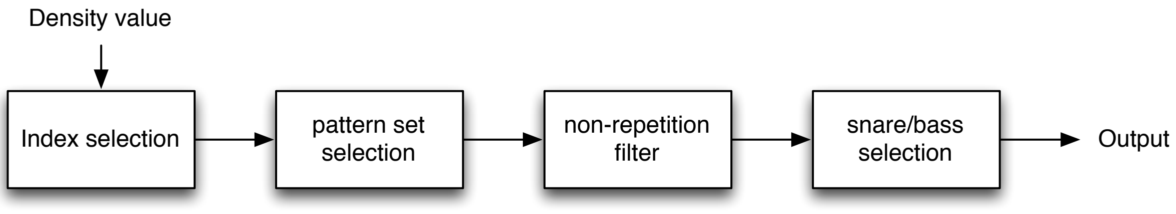


Fig. 39 Snare/bass drum pattern selection algorithm.

Cymbals

The algorithm for the cymbals has many similarities, but also some important differences. It is also based on the recombination of pre-defined patterns, but these patterns are, in this case, four beats long (one bar), instead of one beat. An initial version of the algorithm, using one-beat patterns revealed a poor quality of the results. This happens because the role of the cymbals is quite distinct from the snare/bass drum. The rhythm played on the cymbals are much more stable and repetitive. In fact, most of the time, the ride and/or hi-hat cymbals are frequently the more regular and repetitive elements in the whole band. Better results were attained using four-beat (one bar) patterns (Fig. 40).

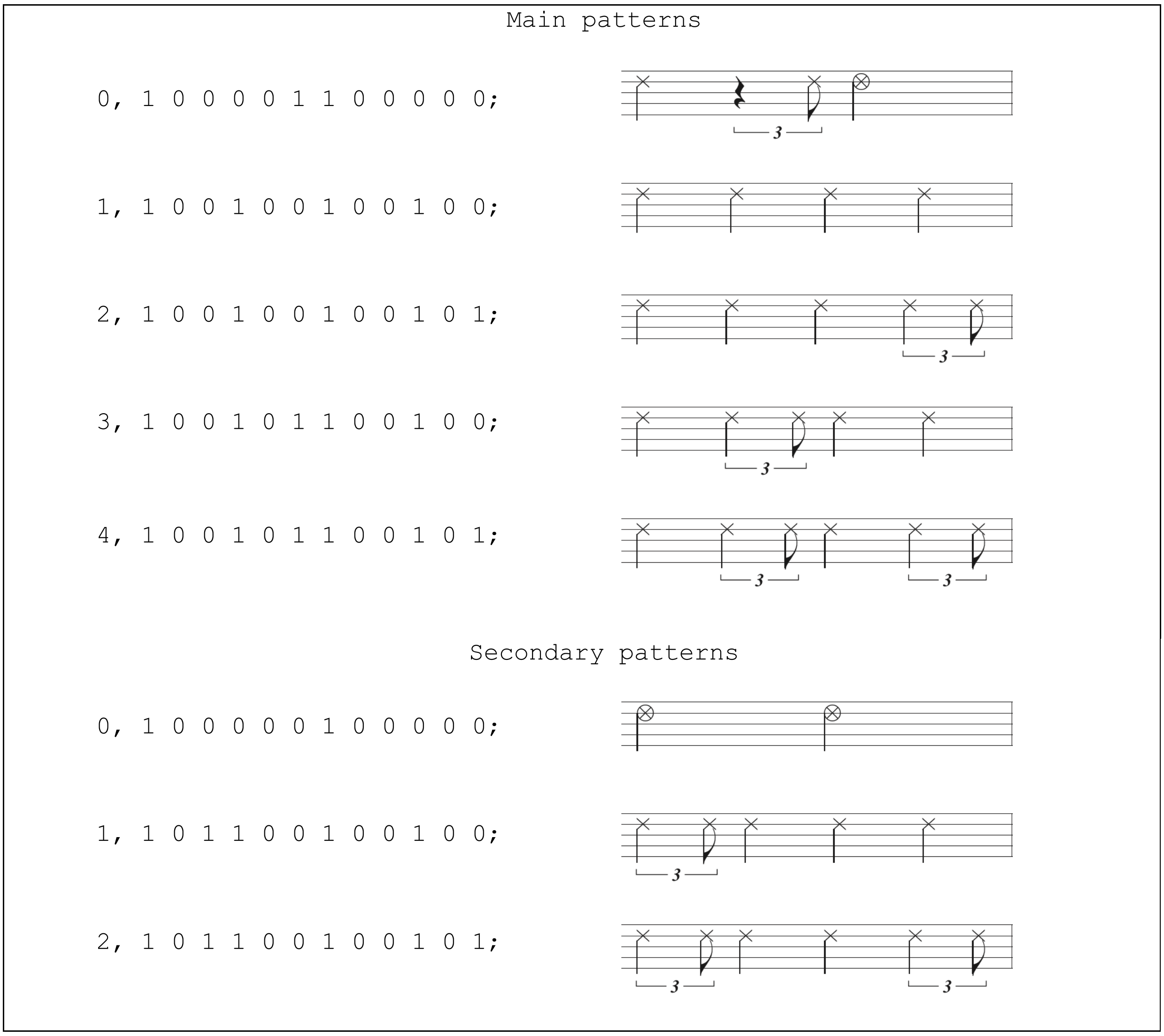


Fig. 40 Ride/Hi-Hat patterns.

The choice of the patterns reflects the regular character of these elements. The range of variations is not very wide, and some of the pulses - namely on the first and the third beat of the bar – are always triggered. Also, unlike the snare and bass drum, there is no complementarity between the ride and hi-hat. They are implemented together because their behavior is similar, but they are used separately. The output is only one of them.

When the cymbal being generated by this algorithm is the ride cymbal, a secondary hi-hat is used, but this time it is only playing one single pattern, with a trigger on the second and fourth beats of the bar.

All the solutions and values presented, were the result of several tests, based on the analysis and comparisons with existing repertoire and knowledge of jazz playing tradition. These values were found to provide a balanced result between richness, variation and stability.

Chapter 6. The mediated interface: base platform

This chapter explores the use of computer-mediated interfaces for the creation and development of music applications. The main focus is on how this mediation can potentiate and render possible the creation of new experiences in music generation and performance. As will be demonstrated, by approaching the development of music controllers using high-level, meta-control strategies, the users of such systems can have any level of knowledge of music theory and practice. In fact, they do not have to be trained musicians at all. To achieve this goal, a <u>base platform</u> was created, consisting in <u>five main components</u> that form a base structure for the development of mediated musical applications, and instrument algorithms that integrate with this structure. Using the base structure and the instrument algorithms, <u>two different prototypes</u> were developed, to better study and illustrate some of the possibilities. Both the base platform and instrument algorithms derive directly from the experience and knowledge acquired with *GimmeDaBlues*, and as such, they share a great number of features and paradigms.

6.1. Overall structure

The base platform is constituted by five main components: Sequencer, Style-sheet, Input, Mixer and Listener. These components set the basic structure of a system that reads a given song structure, defined in the style-sheet, and drives the global clock and communication aspects needed for the synchronized real-time performance of the instruments, while coordinating and directing the input data from the user(s).

The global structure in Fig. 41 shows how the components interconnect and communicate with the interface and to the instrument algorithms.

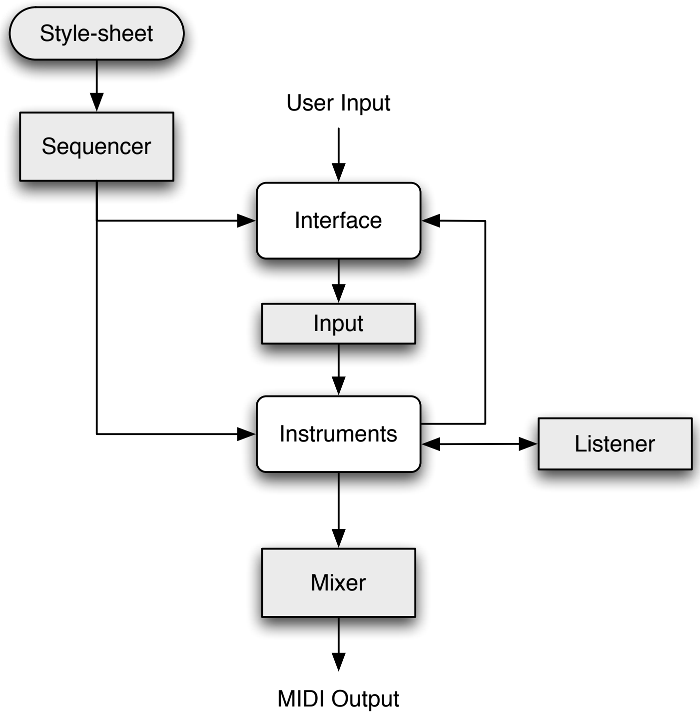


Fig. 41 The five components of the base platform.

The Style-sheet is loaded to the Sequencer, which manages all the timing events and parses the song data to the instruments and interface components. The Input module receives OSC data from the interface, and directs it to the instruments, which can have some type of feedback to the user interface. The output of the instruments is sent to the Mixer, which then routes all the data to the correct MIDI ports and channels. The Listener receives all the musical events produced by the instruments, which meters the overall activity, and can be to control and influence the instruments.

6.2. Sequencer

All the instruments are synchronized with a global clock and timeline in a global SEQUENCER module. This module handles the global transport that will lead all the instrument modules. An internal clock or metronome creates a common *tempo* of a given song, and distributes it as a pulse count marking the speed at which the song will be played and the timeline position.

The sequencer is also responsible for the selection and parsing of song and style data, and handles the distribution of the harmonic contents to all the instruments.

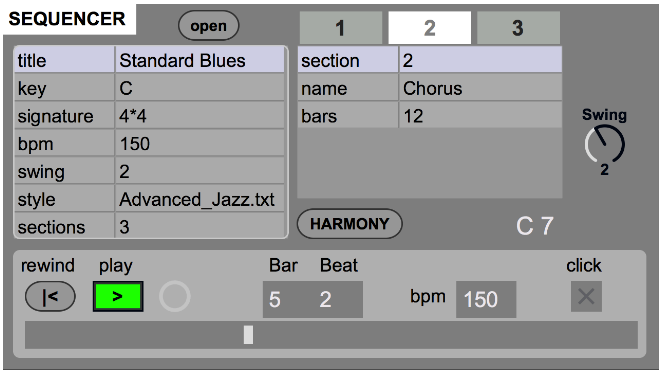


Fig. 42 The Sequencer module main window.

The Sequencer has controls for global commands and three main information areas. The open button on the top is used to open song files, the Song style-sheets custom developed for this application. Pressing the open button will open a system dialogue window to locate the song file. As soon as the song is loaded, the section selectors on the top right will change according to the number of sections in the song. In the example above, the song has three sections, so three buttons appear.

On the right, the "Swing" dial sets the *swing* parameter, with values from 0 to 5. The value of 2 in the above example will produce 8th note triplets. This parameter will be further explained below, in the "clock" section.

The "HARMONY" button opens the "Harmony window", presenting a floating view of the currently selected section's harmonic sequence.

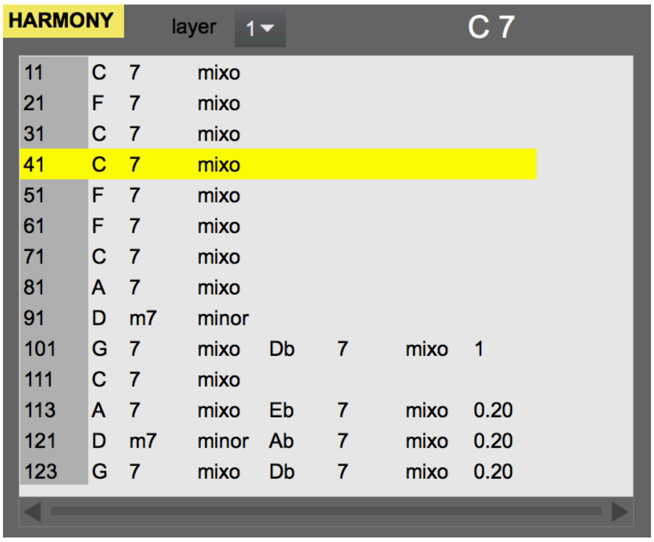


Fig. 43 Harmony window

6.2.1 Clock

The clock drives the sequencer elements and all musical events. It was implemented to provide the necessary impulses of the metrical grid, accounting for the bar and song lengths as well as the sub-beat level swing settings. For this, the clock produces a continuous stream of “ticks” or impulses which are then used to convert into rhythmic figures. At a musical level, the smallest valued rhythmic figure being used by the application is the 8th triplet. Fig. 44 illustrates the rhythmic sub-divisions of the beat generated by the sequencer.

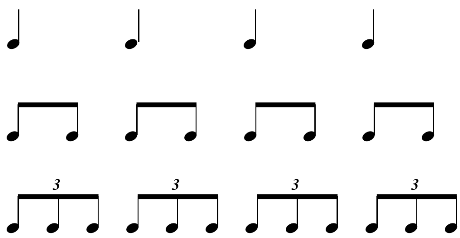


Fig. 44 Rhythmic sub-divisions of the beat

The way the triplets are played by human musicians, however, is not so strict and rigorous as the notation would suggest. This would produce a regular rhythm that would sound mechanical, artificial and dull. Instead, the rhythmical events are slightly delayed or anticipated, creating a more natural sounding “swing” feeling. *Swing* is a very common and essential procedure in traditional jazz playing. It is a type of rhythmic inflection that delays the second subdivision of a beat (second 8th note) turning it into, or close to, a triplet, producing a ternary division of the beat. The Swing field specifies the degree of rhythmic inflection using a range of values from 0 to 5, allowing for 6 levels of delay of the 8th-note. A setting of 0 plays straight eights (a binary division of the beat), a setting of 2 delays the 8th note to the third 8th note triplet (ternary division of the beat). Higher values approximate the 8th note to the end of the bar, close to a 16th note (Fig. 45).

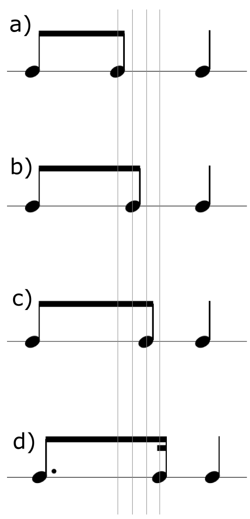


Fig. 45 different degrees in which the second 8th note can be delayed producing a “swing” feeling. a) straight 8ths b) and c) “swinged” 8ths and d) 16th note.

To achieve this, the clock runs at a speed which is twelve times faster than the tempo of the song, i.e., it produces twelve ticks per beat, to allow the necessary time resolution for all the swing levels. The ticks are distributed as an integer number, counting the sequential pulses from the first 8th note of the song to the last one. As each beat is divided into 12 sub-beats, in a typical song with 12 bars in 4/4 beats, the running counter will have a total of 576 for the entire song structure.

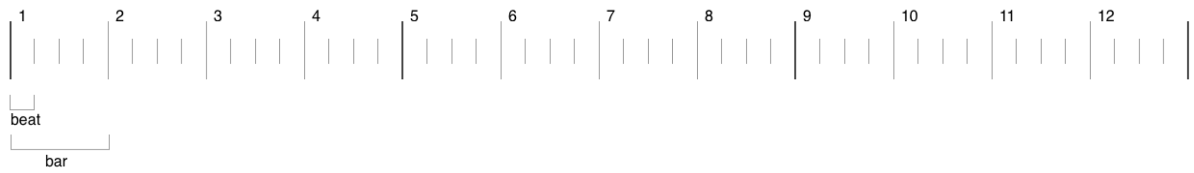


Fig. 46 Metrical grid of a 12-bar song.

6.2.2 Parser

The sequencer module integrates a parser that was developed to read and interpret the custom song style sheet and style-sheet formats created for this purpose (see below). The parser reads the song style sheet and distributes the contained data to all the corresponding modules, as well as global song data that will be common to all the instruments.

6.3. Style-Sheets

After the model of the *GimmeDaBlues* style-sheet, a new format was needed in order to take advantage of the new features in PocketBand. A distinction was made between the style data and the song data. The new "Song Style-Sheet" holds the data that is specific to each song, while the new "Style-Sheet" contains the information relative to the voicing style.

Song Definition

The global attributes are situated in the "SONG DEFINITION" section in the beginning of the style-sheet and include the following items:

```
// SONG DEFINITION
song title
key
signature
bpm
swing
style
sections
```

Table 6 Song definition area of the song style-sheet.

The name of the song is defined in the "song title" field and has to be written between brackets. The "key" field is used to specify the root key of the song, which will be necessary for transposition purposes. The "signature" defines the time signature of the song, and is written as [*beats* * *units]*. A four by four signature is written as "4 * 4". The "bpm" (beats per minute) is an integer value. "Swing" sets the *swing* level, ranging from 0 to 5, where a value of 0 will introduce no *swing* at all, and a value of 2 will produce 8th triplets (see section

The "style" field specifies a template for voicings styles. The voicing is the set of notes or chord degrees that a given chord or scale definition will include. See below for more details.

The “sections” defines the number of sections the song will include. The number of section buttons available in the main sequencer interface will correspond to the number of sections defined in this field.

Section Definition

Following the song definition, there is a “SECTION DEFINITION” area for every section of the song. The section number and name are defined respectively in the “section” and “name” fields. The name of the section can be useful to help navigate and select the sections, which can include informative names like “Intro”, “Solos”, “Ending”, or any other desired text.

The number of bars in the section is defined in the “bars” field. The bars will have the number of beats defined in the “signature” setting. If no signature is defined in this section, the parser will use the bars defined in the “Song Definition” section. The same happens for the “bpm” setting.

```
// SECTION DEFINITION
section
name
bars
// signature
// bpm
// swing
```

Table 7 Section definition area of the song style-sheet.

Harmonic contents

The section definition area also defines its harmonic contents. After the "chords" keyword, the section's chords are defined in the bar and beat where each chord will occur, followed by the chord root, chord type, scale type and duration. The example below specifies the chord sequence of a typical 12-bar blues form:

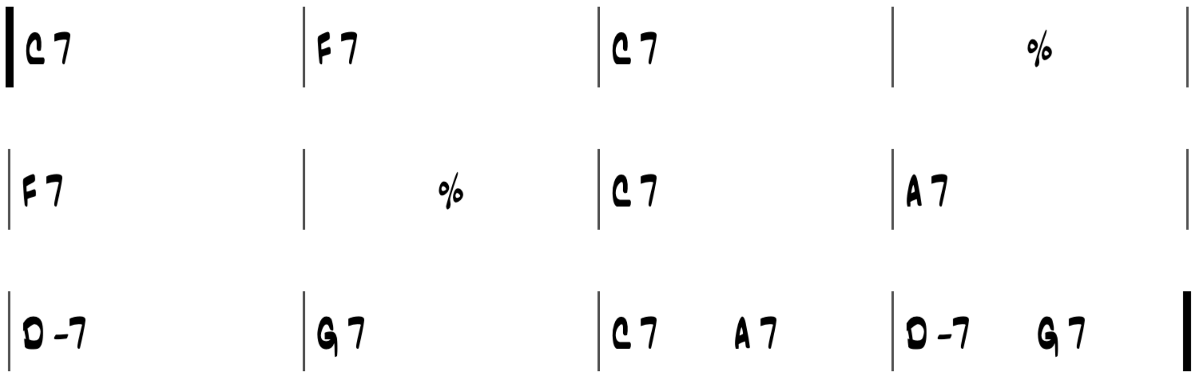


Fig. 47 A typical 12-bar blues chord progression.

The above blues form would be described in a style sheet as an indexed event list:

```
chords
1.1 C 7 mixo 1
2.1 F 7 mixo 1
3.1 C 7 mixo 1
4.1 C 7 mixo 1
5.1 F 7 mixo 1
6.1 F 7 mixo 1
7.1 C 7 mixo 1
8.1 A 7 mixo 1
9.1 D m7 minor 1
10.1 G 7 mixo 1 Db 7 mixo 1
```

```
11.1 C 7 mixo 0.2

11.3 A 7 mixo 0.2 Eb 7 mixo 0.2

12.1 D m7 minor 0.2 Ab 7 mixo 0.2

12.3 G 7 mixo 0.2 Db 7 mixo 0.2
```

Table 8 harmony definition for a 12-bar blues form in the section definition area of a song style-sheet.

The first element of each line represents the Bar and Beat at which the chord occurs. The number on the left of the "." is the bar while the number on the right is the beat. "1.1" represents the first beat of the first bar. 11.3, for example, specifies the third beat of the eleventh bar. The following two elements are the Fundamental Note of the chord and the Chord Type. Chord fundamental is represented in the common A to G naming system. The type of chord notation is planned in order to keep it as similar to traditional jazz notation as possible. Chord types are defined in the "style-sheet" (see below). The next item is the Scale Type, which is also defined in the style-sheet. The scale type is use by the solo as well as the keyboard instrument algorithms. In the example above, *mixo* refers to an adaptation of the mixolidian mode. The last element, the Duration, is the length of the chord, defined in bars and beats. A "1" states a duration of one bar, while a "0.2" for example states a duration of two beats.

Harmonic variations

A part of the standard improvisational procedures in jazz playing occurs at a harmonic level. The jazz musician learns to create harmonic variations by altering the chords of the song on the fly. Some of the possibilities or strategies for harmonic variation include chord substitution rules by Steedman, 1984 (see section 4.2). The Pocket

Band Sequencer implements a simple mechanism to allow for harmonic variations, by allowing the user to define alternative chords for any chord entry. The alternative chords are defined after the first one, and use exactly the same syntax.

```
11.1 C 7 mixo 0.2

11.3 A 7 mixo 0.2 Eb 7 mixo 0.2

12.1 D m7 minor 0.2 Ab 7 mixo 0.2

12.3 G 7 mixo 0.2 Db 7 mixo 0.2
```

Table 9 Alternative harmony settings

The chords defined in Table 9 show three lines with alternative harmonies, in lines 11.3, 12.1 and 12.3, which are defined as an extra set of chord, scale and durations. This corresponds to a second layer of harmonic possibilities that can be used to create different harmonic trajectories. The sequencer then allows the user to introduce harmonic variations by changing the layer control in the harmony window. Using different layers will produce distinct variations as harmonic paths. As with the first path, the alternate chords data will be sent to the several instruments.

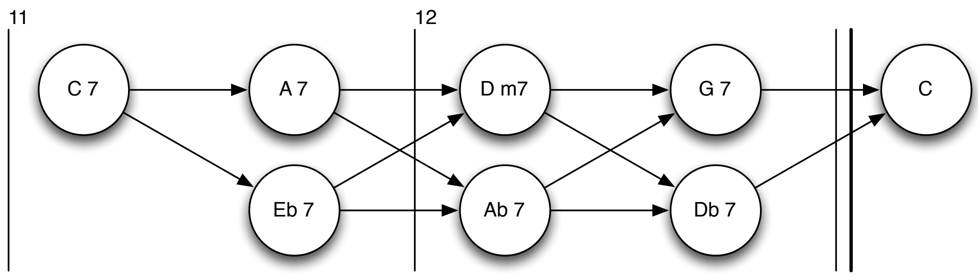


Fig. 48 Chord substitution example in the last two bars of a blues structure.

In the example above, using the chord definitions in Table 9 of the last two bars of a blues 12-bar structure, we can see the several possibilities by moving arbitrarily between the two chord layers.

A future development of this system could include a probabilistic algorithm for harmonic variation, using, for example, Chemilier's implementation of Steedman's chord substitution rules.

6.4. The Listener module

Borrowing the name and concept from Robert Rowe's Cypher program (Rowe, 1993), the LISTENER module acts as a central hub for the incoming events produced by the instruments. It receives the output data from all the instruments and allows a global view and control over the activity of the instruments.

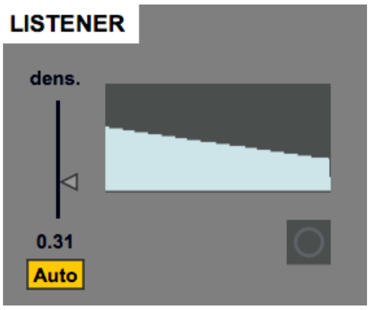


Fig. 49 The Listener module.

In its current version, the Listener's role is limited to the global activity meter and control. However, many other functions can be implemented in future versions. The current control over the activity is a simple but effective way to influence the virtual band, and namely the virtual musicians. As the users play more notes in the solo and keyboard instruments, the activity level rises. If desired, the Listener can use this level to control the bass and drums algorithms, to produce a response to the user input. As would usually happen in a band with human players, the bass and drums respond accordingly to the user activity, producing a natural and effective group result.

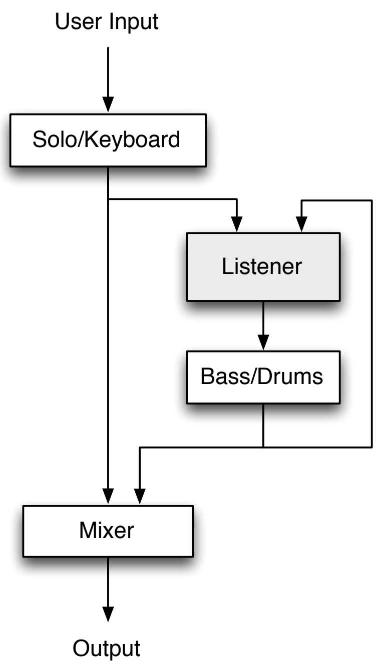


Fig. 50 Inter-connections between the Listener and the Instruments.

Future implementations can take in account several other features, like the pitch and timing of the incoming notes and use it to have more refined influence over the virtual musicians. It can also receive data from the sequencer timeline and harmonic structure, to create location aware musical responses that depend on the location in the song structure at any given moment.

6.5. Input

The input module is a bridge between the input data from the user interface and the input data to the instruments. By isolating this function, it is easier to make any adaptation to different data formats from different input devices, to a single, normalized, data format to the instruments. This module also manages the number of events allowed to each instrument, acting as a filter by limiting the amount of input events that pass thru. The solo instruments are limited to one note at a time, and only allow one input event, while the keyboard instrument allows two input events, that correspond to the two hands. An indicator in each input channel lights when data is received.

Fig. 51 Control input module.

6.6. Mixer

The MIDI event data sent from the instruments is directed to the mixer module. The mixer is a centralized hub for all the note events, and control of the MIDI output ports, channels and program change numbers for each instrument. A play/mute button and a volume slider are available for each instrument, including three solo instruments, keyboard instrument, bass and drums.

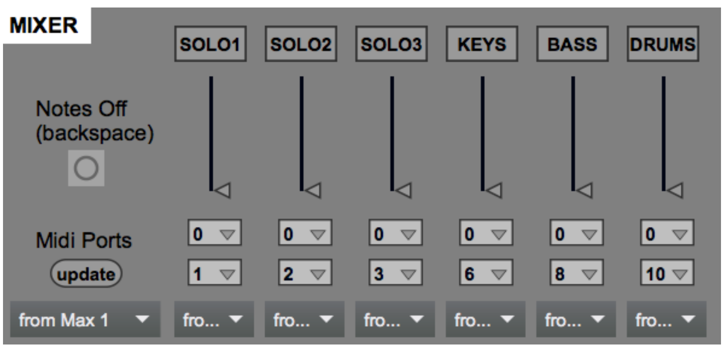


Fig. 52 The Mixer module.

6.7. Conclusion

The five presented components: Sequencer, Style-sheet, Input, Mixer and Listener, constitute a base platform from which the prototypes that will be presented in the next chapters were developed. This platform deals with the global musical aspects that will be used by the instrument algorithms, provides a common format for tempo and harmonic parsing, and takes care of the communication with input control events as well as the output to the MIDI system.

The listener module is a promising late addition to this platform. In its current form, it can monitor the events that are generated by the instruments and use this data to act as a mediator with all the instruments. This module can be developed to act as a central brain of the system, that can co-relate all the elements, to somehow mutually influence the user(s) and the generation algorithms.

Chapter 7. Application prototypes

The modular implementation of the components presented in the last section was very important to potentiate the experimentation and creation of different formats and possibilities for the computer-mediated musical applications.

During this research, several possibilities were devised for the instrument algorithms and global control, as well as for the control strategies and interfaces, and potential applications for such systems. This chapter will present two different approaches that make use of the global and instrument components.

The two prototypes that will be described were mainly planned to explore different configurations where the mediated interface and algorithms could be used to effectively create new tools and new musical possibilities. The first one, "Pocket Band", is the direct development of *GimmeDaBlues*. While several features and concepts are common, most of the elements were completely redesigned and expanded.

The second prototype, "MyJazzBand", focuses on the adaptation of the concept to a multi-user environment, exploring the collective experience of playing in a music band. It was planned as an interactive installation, where up to four users can participate as members in a virtual band, using a big format multi-touch display.

7.1. Single-user approach: "Pocket Band"

As mentioned before, "Pocket Band" is the direct development of the *GimmeDaBlues* application, sharing basically the same concept of an integrated single-user, "one-man-band"-type application, for one user to play one or two instruments, while the application generates the bass and drums automatically. Unlike *GimmeDaBlues*, however, this prototype's interface was now developed for the iPad only. As it involved a greater number of keys for each instrument, as well as a new set of transport and performance controls, the full size of the iPad screen was necessary.

Fig. 53 “Pocket Band” iPad interface prototype.

Like *GimmeDaBlues*, the application uses four different types of generative algorithms that model the behavior and idiomatic characteristics of the corresponding real instruments as used in jazz music playing. The instruments that were addressed are “Solo”, Piano, Bass and Drums, reflecting one of the most common group formations in traditional jazz. The Solo refers to a melodic instrument like the trumpet or saxophone, for example. The Piano, Bass and Drums are mainly planned for an accompanying role. They are centered on the automatic generation of musically and stylistically correct musical events used in jazz playing, while allowing the live control of some main parameters that influence the algorithm's output.

The interface comprises three main areas. The first, on the top (Fig. 54), includes transport controls for play start/stop and section selection, and displays information of the current and next chord. It also provides access to a mixer and menu. This interface is a prototype and only the main direct performance features actually work. The menu and mixer were left as placeholders in case of future implementation but didn't work in this prototype version. The settings and mixer were however available in the main application on the computer.

Fig. 54 PocketBand: transport section.

The features in the transport area reflect many improvements and differences over *GimmeDaBlues*. The creation of different sections of a song, which are described in the song style-sheets, allow for a deeper use in real performance, as well as practice. The song's sections can be tailored for any type of chord, meter and tempo changes, and the number of measures can be different for every section, so it is possible to have a more complete control of the entire performance. The progress bar below the transport buttons shows the number of measures in the current measure and indicates the current measure being played.

The current and next chords displays are intended mainly for practice purposes.

Instruments

The wider center area has the virtual keyboards for the instruments. Like *GimmeDaBlues*, the top keyboard is addressed for the solo instruments, while the one below is addressed for the keyboard instruments. On the lower part of the screen, this prototype interface also displayed an experimental control interface for the bass generator.

This graphical interface went through several revisions until the one in Fig. 53. The differences between the three keyboards illustrate one of the evolutions since *GimmeDaBlues*, in the distinction between each of the individual instruments. As the instruments have different characteristics, the interface must reflect this in order to provide an optimal control for each one.

Solo

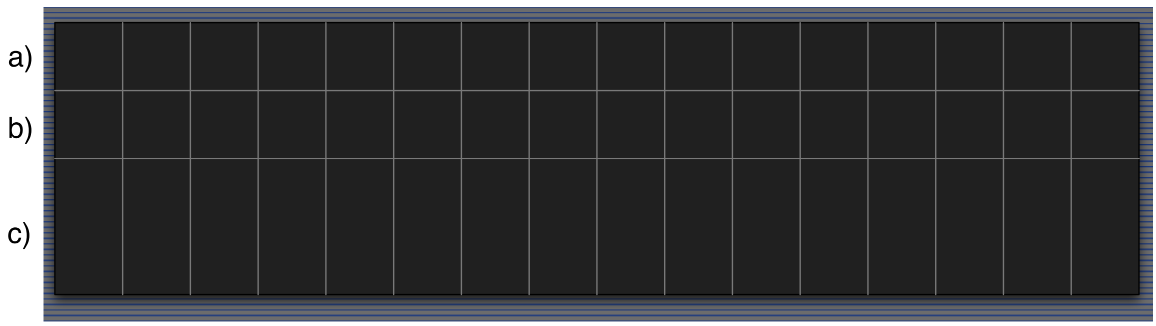


Fig. 55 Solo area.

Inside the instrument areas, the one on the top is the solo area. This area is divided in sixteen columns and three rows. Each column corresponds to a different note index of the available scale at any given moment in the harmonic sequence. The three rows represent two different functions. The lower and largest row (Fig. 55 c) is similar to the virtual keyboards in *GimmeDaBlues*, where the vertical position of the finger touches inside this row controls the dynamics, setting the note attack velocity. The first and second rows - Fig. 55 a) and b) – are slots for preset licks or ornaments. These are very short phrases that contribute to enrich the solos with small gestures that are usual in jazz improvisation, but would be difficult or impossible to obtain with only the lower row. In this case, the ornaments are appoggiaturas to the current key's note, whether upper (in the top row) or lower (in the lower row).

The development of the solo interface and algorithm had the collaboration of Mário Santos, a professional jazz saxophonist, who provided precious stylistic information about jazz playing, and specifically regarding the use of ornamental notes and short licks.

Keyboard

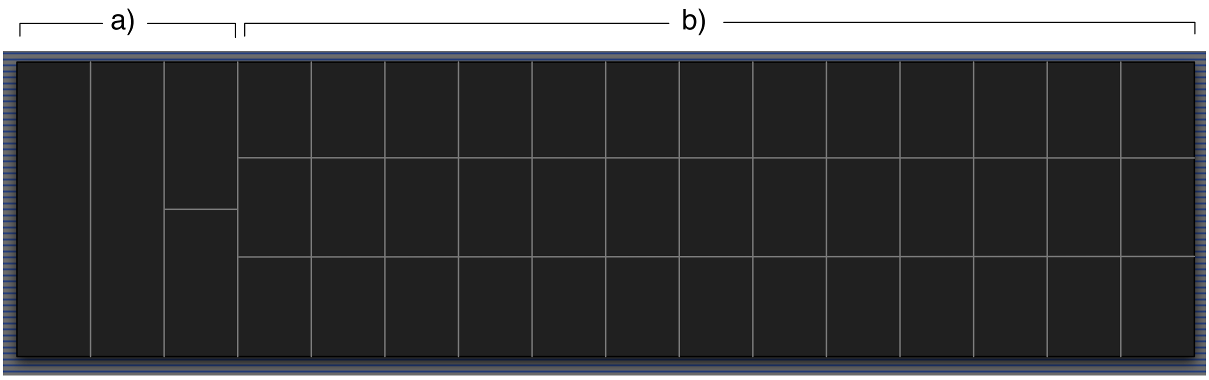


Fig. 56 PocketBand keyboard area.

The middle strip of the instrument area is planned for the keyboard instruments. Like the solo interface, the 16 columns correspond to 16 note indexes, ranging, from left to right, to lower to high pitches. Unlike the solo, however, the keyboard is capable of producing chords as well as single-notes. For this, the three rows from column 4 to 16 present three separate rows (Fig. 56-b). These rows allow the user to play single-notes or two and three-note chords. Pressing the top row plays one single-note, pressing the second row produces a two-note chord and the lower plays three note chords. The first three keys on the left are always reserved for the current chord's first degree (fundamental) in the low register (column 1), the 5th degree (column 2), and the octave above (column 3). The latter is divided in two-sections. Pressing the upper middle key plays the 5th and 8^{ve} simultaneously.

The development of the keyboard interface and algorithm had the collaboration of jazz pianist Telmo Marques, with whom we tried different approaches to the creation of the chord voicing algorithm.

Bass

Fig. 57 PocketBand: bass.

The lower part of the interface is reserved for the bass. It was one of the experiments in studying possible control interfaces for the bass algorithm. The "Auto" button will turn the bass completely automatic. In this mode, the bass algorithm (described in section 5.2) will take over, and the 16 keys will display the current note-index, but will ignore any user input. With the auto mode off, the bass has a similar behavior has the solo area. As the current song's chords change, the 16 keys will be dynamically mapped to a custom scale, that the user can play. The "Force" button will force the automatic bass generator to play every quarter note, or beat of the bar, overriding the activity level parameter, arriving from the automatic or manual activity setting.

Activity

The “Activity” slider on the middle-left part of the screen sets the global activity parameter to all the instruments. This slider acts both as a display for the automatic level of activity, and as an input for the user to control this parameter. The automatic level will be constantly updating, according to the user activity in the instruments. If the user stops playing or decreases the number of keys pressed in a regular interval, the global activity level will decrease. As the user plays more notes, the activity will rise. The level displayed or controlled with this slider is continuously being sent to all the automatic instruments, so they will respond as a human jazz band would usually do.

7.2. “MyJazzBand”: towards a collective experience

“MyJazzBand” is an interactive musical installation that allows up to four users to play in a virtual jazz band, using a custom-developed graphical interface on a multi-touch display.

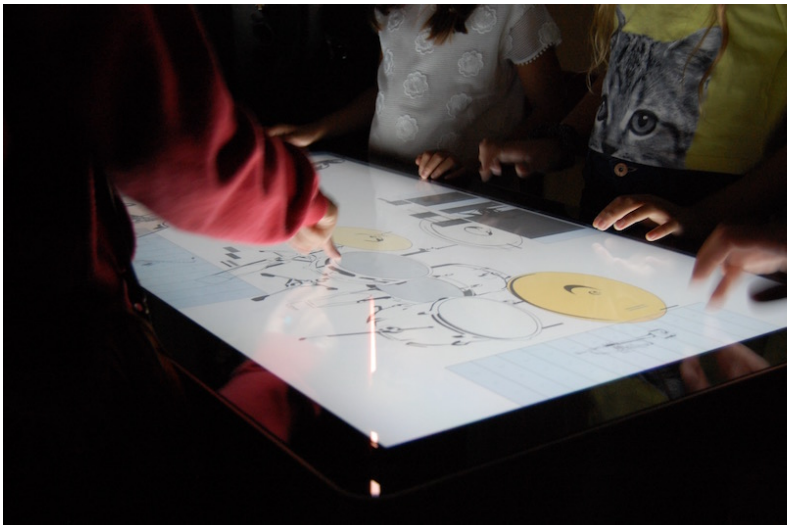


Fig. 58 MyJazzBand exhibition at Theatro-Circo, Braga, September 30th and October 1st 2016.

In this application, the base platform and algorithms are oriented towards a multi-user experience, with the aim of exploring the potential of computer mediation in collective musical contexts.

MyJazzBand uses a multi-touch screen display with custom-designed graphics as an intuitive device to acquire the multi-user input data. The graphical interface was created in Processing language, using custom-made vector graphics. The main application with all the instrument modules run in a single computer and receives input data from all the players from the multi-touch screen using the OSC protocol.

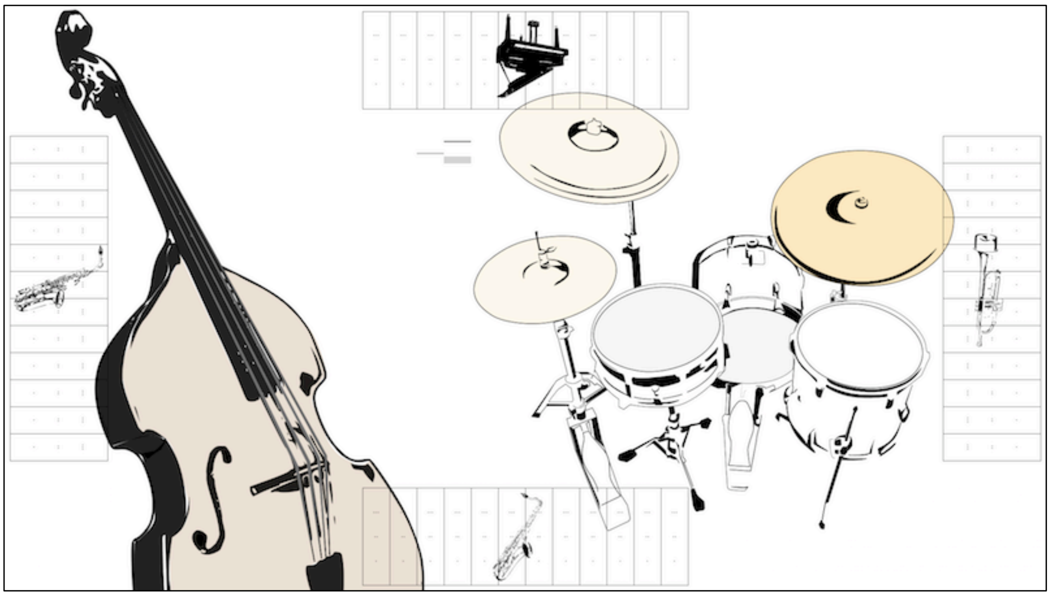


Fig. 59 “MyJazzBand” graphical interface.

The interface presents four interactive virtual keyboards, one on each side of the screen. Each keyboard has a different instrument assigned, represented graphically, namely: muted trumpet, alto saxophone, tenor saxophone and piano. The virtual keys have the approximate size of a normal piano keyboard, and each keyboard has twelve keys. Similarly to the virtual keyboards in Pocket Band, these keys don’t have static assigned notes, but instead are changing dynamically, according to the current harmony from the song at any given moment.

A visual feedback is provided by each key, changing color when pressed, and by a graphic animation of the pressed keys flowing upwards, out of the keyboard. The visual feedback add esthetical value, making it more attractive visually, but also contributes to a sense of group collaboration, as the notes flow to the middle area, where the bass and drums are, and where all the keyboards flow to as well. The bass and drums, although not playable by the users, respond graphically to the musical events they are producing, contributing to the understanding of what is happening.

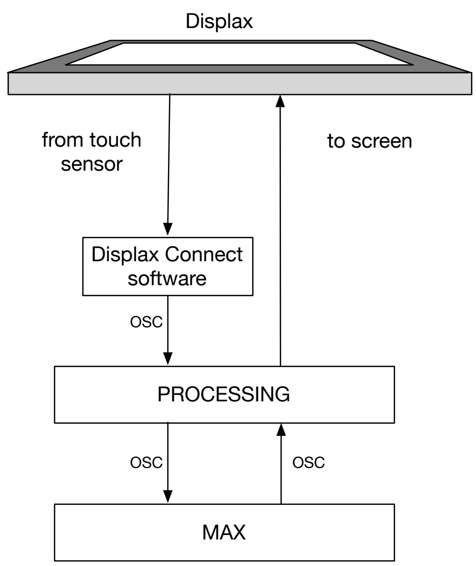


Fig. 60 Communication between the sensor, software and screen in MyJazzBand.

The multi-touch interface used was the Edigma Displax 47'' (http://www.edigma.com). Inside its case, the Displax hardware couples two different devices: a normal LCD screen, and a touch sensing transparent surface, integrated between the screen and the top glass surface. The communication is done to the computer by USB, for the touch data, and by HDMI or VGA, for the screen. The Displax also includes its own windows-based computer system, but in this case, it was bypassed as its performance was not fast enough for this application. A special wood furniture was built to support the Displax in a horizontal position, and accommodate the computer, sound system and cables, so that the only thing coming out of the system was a single power cable.

The touch data is parsed by Edigma's own software, Displax Connect, and sent using OSC/TUIO (http://www.tuio.org) format to MaxMSP. After the corresponding data is sent and processed by software, the processed results are sent to Processing by OSC messages, that then uses it to create the graphical feedback. There was a noticeable latency between the touch and the response, but it is fast enough so that it still allows a quite responsive experience. Although the Displax itself can sense up to 40 distinct simultaneous touches, this Max patch only uses five: one for each keyboard area on the trumpet and saxophones, and two for the piano.

This setup provided the perfect conditions for this experience, creating a great opportunity to create not just a laboratory prototype, but a fully functional implementation of all the concepts and techniques that were developed for this research. The response of the users was quite encouraging, as they felt quite engaged, independently of their musical background or experience. As with any other interactive installation, the audience response can vary incredibly. There were those who didn't even approach the screen, others that went away as soon as they triggered the first sound, and others that kept touching it randomly, as if they had no ears at all. Fortunately, in most cases, they were very positively engaged, and participative. From little children to older people, they all seemed quite enthusiastic about the experience. The choice of a single twelve-bar standard major blues form for these presentations was, I believe, very appropriate. It is a very familiar sounding structure, and most people seemed to be comfortable, and understand where the downbeats, the bar and song changes were. They seldom reacted accordingly, by starting or ending phrases in relevant structural moments, or even changing instruments.

It was initially presented at the "Noite Branca" art festival in Braga (September 2nd to 4th), at the D. Diogo de Sousa museum. It was presented at the "BRG Collective" show at the Theatro-Circo, in Braga (September 30th and October 1st) and at the "MMXX" event, at FEUP, Porto, for the celebration of the twentieth anniversary of the Multimedia masters program of the University of Porto (November 2nd).

7.3. Conclusion

The two prototypes presented were the result of a great number of experiments on different approaches and techniques for software development and algorithmic strategies. Related issues like control strategies and user interface ergonomics and aesthetics, were considered, to create an efficient and engaging user experience.

While PocketBand was developed as a natural continuation of *GimmeDaBlues*, it mainly was used as a test bed for all the experimentations with the algorithms and the control interface. It was also tested in live performance contexts, jamming with live musicians. MyJazzBand is a fully developed application and interactive installation that applied the concepts and techniques in a collective, multi-user context, and it was publically presented in Braga and Porto. The visible engagement and positive response of the users was quite encouraging and was a confirmation of the efficiency of the mediated interface approach.

The concepts and solutions presented in this dissertation address several issues and perspectives that I consider important for the development of computer enhanced musical experiences, and the research on new musical interfaces and automatic music generation. From this experience, I consider the following dimensions to be especially relevant:

- Accessibility

It is always very stimulating to observe technology as a medium for newfound accessibility. In this particular case, computer mediation allowed the creation of musical applications that musicians and non-musicians alike are able to use, thus enjoying the pleasure of playing music, which otherwise would not be able to play.

The main idea around which all the main solutions were built upon begun on the observation of the improvisation practice, in that we can, at least at a basic level, separate the act of improvising in two distinct parts. One part concerns the theoretical knowledge and physical practice that constitutes the long learning process that a musician goes thru, that require many hard-working years to achieve. The other refers to the intuitive action of improvising itself, comprising all the on-the-fly decisions that rely on knowledge, but in a great part to intuition, listening and reference.

The presented algorithms and integrated platform were developed to allow the user to experience the act of improvisation in a jazz band context, while leaving all the harmonic and melodic calculations to the computer. The presented prototypes exemplify the use of this platform in two different approaches.

- Creation and performance

As discussed in section 2.4, computer mediated tools are being developed, not just in academic and scientific research, but also by commercial software companies. Integrating musical knowledge turns the software into intelligent assistants that can create musically valid output with very little effort from the user. These tools can very efficient in the fast creation of musical sketches. As the algorithms and applications get more sophisticated, they can easily become a standard asset in any amateur or professional musical production workflow. Moreover, computer mediation can inspire the creation of new music and new means for musical expression.

Performance-oriented mediated applications become a new type of digital instruments or meta-instruments, that can render musical performance accessible to new users, but can also explore new types of musical expression formats and languages which are not possible in conventional instruments.

Chapter 8. Conclusion

8.1. Summary

The aim of this dissertation was the study of strategies for the development of music generation and performance systems that explore the potential of the computer as a mediator between the user input and the musical output. The addressed musical context was specifically focused in traditional Jazz and Blues, as these are strongly based on improvisation, but simultaneously with clearly defined procedures and stylistic behaviors, that are quite appropriate for an algorithmic implementation. This choice was also due to my own personal background as a jazz piano musician, which proved more important than anticipated, both from a theory standpoint and from the practical experience in jazz improvisation.

The literature review led me to the conclusion that although there are several systems for the computation of jazz music, they mostly address the automatic generation of jazz solos, and very few have addressed the accompaniment, or the idea of the mediated instrument format.

The concept pursued for this research was of an integrated system that rely on three main ideas:

1) The solo parts were planned as mediated instruments, and not as generators, which means the solos were addressed as dynamic mapping algorithms (solo and piano algorithms). Together with the specifically designed interfaces, such systems become performance meta-instruments that are imbued with musical knowledge, so that they take care of the necessary calculations and musi-

cal processes in the background, while the decisions and act of performance itself is left to the users;

2) The automatic generation algorithms focus on the comping section of a jazz band. Namely, the bass and drums algorithms were developed in order to complement the integrated system of a virtual jazz band;
3) In order to be as flexible and efficient as possible, the generation algorithms are procedural, meaning that they do not rely on the recombination of pre-recorded phrases. Although such systems exist that produce very interesting results, I believe in the long term, fully algorithmic procedural generators will become much more effective and flexible.

These ideas were based on the previous experience gained from *GimmeDaBlues*, and led to the development of custom software, in order to explore, implement, and test new ideas for the generation and control of the mediated instruments.

My proposal was an integrated framework that handles the control of global events, a set of four instrument algorithms, and two prototypes, that explore and demonstrate the use of the algorithms and the potential of the proposed concepts dealing with computer mediation.

8.2. Original contributions

In pursuit of an integrated system that can accommodate the idea of computer-mediated interfaces and music generation algorithms that focus the performance of traditional jazz music, I developed an integrated software framework, using Max. This framework deals with global performance aspects, namely tempo, synchronization, parsing of the harmonic contents, inter-communication between the modules and communication with external input control and output destinations. This framework comprises a set of global components - Sequencer, style-sheet Parser, Input, Mixer and Listener - and a set of instruments – solo, piano, bass and drums.

I presented two prototypes built upon this framework, that demonstrate two possible applications of the presented concepts. The first prototype - Pocket Band – is the direct successor of *GimmeDaBlues*, described in section 4.5. Pocket Band uses a custom-made interface for the iPad as the user interface. It was the test bed for several ideas about the overall control and performance-oriented questions. With a similar approach to the one in *GimmeDaBlues*, Pocket Band proposes a one-man band type of approach, where the user interface presents two or more instruments and some global parameters for the user to control.

The second prototype – *MyJazzBand* - is a fully developed application for a multi-touch table, where up to four people can play a solo instrument or piano, while the bass and drums are generated automatically. It was presented publicly as an interactive installation.

The development of the concepts and implementation of the prototypes led to the creations of four stylistic Jazz music computer algorithms, namely a jazz piano dynamic voicing calculator, a contour-based jazz walking bass generator, a jazz drums generator and a dynamic mapping algorithm for the solo instrument interfaces.

While addressing the use of computer mediation in the creation of musical applications, several interesting challenges emerged. The concept of instrument itself and the proposed notion of digital meta-instrument were important to the pursuit of the mediated interface as a valid means of musical expression and as new-found accessibility. The related examples from existing literature and the presented prototypes aim to explore and present some effective control strategies of such systems. Hopefully, these ideas can be useful for future discussions and research in related areas of study.

During this research, the following papers were published:

- Dias, Rui; Guedes, Carlos; Marques, Telmo (2014). “A computer-mediated Interface for Jazz Piano Comping”. SMC/ICMC 2014. Athens, Greece.
- Dias, Rui & Guedes, Carlos (2013). “A contour-based walking-bass algorithm”. In Sound and Music Computing Conference (SMC 2013). Stockholm, Sweden.
- Dias, Rui & Guedes, Carlos (2012). “GimmeDaBlues app for iOS: overview and ongoing developments”. ACM SigGraph Asia 2012, Singapore.
- Dias, Rui; Marques, Telmo; Sioros, George; Guedes, Carlos (2012). “GimmeDaBlues: An Intelligent Jazz/Blues Player And Comping Generator for iOS devices”. In proc. CMMR 2012, London.
- Dias, R.; Marques, T.; Sioros, G.; Guedes, C. (2012). “Gimme ‘Da Blues: A Jazz/Blues Player And Automatic Comping Generator For iOS Multitouch Devices”. In InForum Coimbra 2011.

And oral presentations:

- *A contour-based walking-bass algorithm*. In Sound and Music Computing Conference (SMC 2013). Stockholm, Sweden. August 2013;

- *Computer Mediated Control of Interactive Music Applications*. MAT (Media, Arts & Technologies) Workshop. INESC-TEC, Porto. April 29th 2013;
- *GimmeDaBlues app for iOS*. University of Texas at Austin. Austin, Texas. November 2011;
- *Automatic Generation of Jazz Accompaniments*. @Future Places Digital Media Doctoral Symposium. FBAUP, Porto, October 2011;
- *Gimme 'da Blues: a jazz/blues player and automatic comping generator for iOS*. InForum Coimbra. UC, Coimbra, September 2011;
- *GimmeDaBlues, iPhone application*. 1st Symposium on Automatic Music Generation. Coimbra, February 2011;

8.3. Guidelines for further study

I believe computer mediation for musical applications is still in its infancy, even more so with musical applications. Together with the telecommunication technologies that power the "internet of things", computer mediated systems are being developed in most areas of expertise and permeating our daily lives. These systems enhance the existing paradigms and create new ones probably every day. The application of these systems to musical applications is slowly but steadily being introduced in expert systems and mainstream applications.

The systems developed for this dissertation bring together a great number of resources and techniques to produce an integrated solution of a creation and performance-oriented system, that uses custom made algorithms and interfaces to propose a paradigm of a virtual jazz band and collaborative improvisation system. Although it presents a solution in itself, the developed platform and algorithms present an auspicious source for many new applications and developments, and the modular approach in the creation of the base platform and instrument algorithms will potentiate the further development of the several components.

Some of the most prominent resources that can be addressed in the prosecution of this research include:

Instrument algorithms – The options that were taken for their development of the instrument algorithms served the purpose of this dissertation, as parts of the integrated system. However, they can be developed further to encompass several other features that I did not address here. During the description of the algorithms in Chapter 5, I left some suggestions for further improvements in the corresponding sections.

Composition tool – Allowing the user to directly create his/her own chord sequences and record melodies could direct this system to a more composition-oriented application. This could be accomplished with an inline editor, which could be able to edit the global settings like tempo, number of sections, swing factor or others, and compose, by trying different chord combinations.

Automatic creation of song style templates – It would be very interesting to develop an automatic retrieval and adaptation algorithm of jazz standards from existing digital score databases like the one described by François Pachet (Pachet et al., 2013). Such system would greatly improve the usability of this platform by extending the available repertoire.

Harmonic generator - The integrated sequencer gathers information for the harmonic content from the template files, which include the optional harmonic paths to create the chord variations. An intelligent generator would ideally be able to create these variations automatically, according to general practice of jazz chord substitutions. Such system could be based on Chemillier's implementation of Steedman's chord substitution rules.

Interactive Bass and Drums – There were some preliminary studies on possible strategies to have the bass and drums algorithms playable by the user, like the solos and piano. The bottom row in the Pocket Band interface was one of such studies. This was however not a priority, as for the current purposes, it was more important to have them automatic, but surely can be an interesting development for future work.

Automatic piano accompaniment – As a part of the comping section, it would be interesting to develop an automated piano algorithm. It should be interesting to implement an automatic generator, having the present voicing calculator as a base. The new algorithm would then have to address the generation of rhythm and phrasing.

Full automation – This accompaniment system can be coupled with an existing solo generator, like GenJam (John Al Biles, 1994), in order to create a fully automated, generative musical system. The comping generator algorithms would be adapted so that it follows the output of the solo generator to truly accompany it. Reversely, the comping section can influence the solo generator.

Group communication – A very captivating aspect about collective experiences is to study how the different parties relate, communicate and contaminate each other. This platform should be a great starting point for such a study. The Listener was implemented in order to potentiate the development of a global observer that can act as a central brain that gathers performance data of the several components, and is able to act and interfere with global or specific elements (somewhat of the "Critic", in Rowe's Cypher program (see section 0). In the present version, the Listener receives data from all the instruments, but was implemented mainly to influence the virtual bassist and drummer, in order to respond to the user input activity. Further developments can analyze other features of the input control data to use data like pitch, timing, durations, or even phrasing. As a real group during improvisation is constantly exchanging small ideas, licks or harmonic modulations, a virtual system could incorporate such aspects to create more sophisticated results.

Max for Live devices - Another idea that came up during this research led to the use of this platform in a more conventional music production context. It will be interesting and very useful to explore the idea of researching the potential of the mediation algorithms in a typical non-real time music production software, the DAW environment. A direct path would consist basically in the adaptation of the Pocket Band components to music plugins, using the Max for Live platform. Since this idea came up, I decided to program all the components in Max, having in mind a possible port to the Max for Live platform, so it should be fairly easy to adapt. This would suggest an integrated range of Live devices, which are not sound processors nor virtual instruments, but virtual players. We would be able to load for example the walking bass generator to one track, and automatically it would generate a walking bass line in real-time, as the other tracks can have pre-recorded data. It would also be possible to record the generated data and edit it like any other track, if necessary. This could provide a whole new way to work with the sequencer, in a similar way as a graphic designer or 3D modeler uses a procedural texture instead of a real picture to create a certain texture.

There are some software examples of some of these features already available, mainly Band-in-a-Box (see section 2.4.1), that even allows us to select an algorithm that generates solos in the style of a given musician, like Bill Evans or Charlie Parker, or the virtual drummer track in Logic or Garage Band.

This approach differs significantly from the performance applications presented in the previous chapters, mainly in that it is more oriented towards composition than performance. This integration opens many possibilities for music creation, by introducing meta-control features in the regular music production DAW-based workflow.

REFERENCES

10base-t interactive. (2013). MrMR (version 2.5) [Mobile application software]. Retrieved from http://mrmr.noisepages.com

Affinity Blue. (2013). Nodebeat (version 2.0.5) [Mobile application software]. Retrieved from http://nodebeat.com

Agon, C., Assayag, G., Laurson, M., & Rueda, C. (n.d.). Computer Assisted Composition at Ircam : PatchWork & OpenMusic. Retrieved December 21, 2016, from http://recherche.ircam.fr/equipes/repmus/RMPapers/CMJ98/

Agostini, A., & Ghisi, D. (2015). A Max Library for Musical Notation and Computer-Aided Composition. *Computer Music Journal*, *39*(2), 11–27.

Ames, C. (1987). Automated Composition in Retrospect: 1956-1986. *Leonardo Music Journal*, *20*(2), 169–185.

Ames, C., & Domino, M. (1992). Cybernetic Composer: an overview. In *Understanding Music with AI*. MIT Press Cambridge, MA, USA.

Apple Inc. (2013). GarageBand (version 1.4) [Mobile application software]. Retrieved from http://www.apple.com/uk/apps/garageband/

Apple product timeline. (n.d.). Retrieved April 17, 2015, from https://en.wikipedia.org/wiki/Timeline_of_Apple_Inc._products

Ariza, C. (2005). *An Open Design for Computer-Aided Algorithmic Music Composition : athenaCL by Christopher Ariza*. Boca Raton, Florida: Dissertation.com.

Audanika. (2013). SoundPrism (version 2.5) [Mobile application software]. Retrieved from http://www.soundprism.com

Bäckman, K., & Dahlstedt, P. (2008). A Generative Representation for the Evolution of Jazz Solos. In *Applications of Evolutionary Computing Lecture Notes in Computer Science Volume 4974* (pp. 371–380). Springer.

Bailey, D. (1993). *Improvisation: Its Nature and Practice in Music*. New York: Da Capo Press.

Barlow, C. (n.d.). Autobusk. Retrieved June 6, 2017, from http://www.musikwissenschaft.uni-mainz.de/Autobusk/

Barlow, C. (2001). *Autobusk, a reat-Time Pitch & Rhythm Generator*. Universitat Mainz.

Barlow, C. (2008). *Von der Musiquantenlehre* (Feedback P.). Cologne, Germany: Feedback Studio Cologne.

Barlow, C. (2012). On Musiquantics. Mainz.

Berg, P. (n.d.). AC Toolbox. Retrieved December 21, 2016, from http://www.actoolbox.net

Berggren, U. (1995). *Ars Combinatoria: Algorithmic Construction of Sonata Movements by Means of Building Blocks Derived from W. A. Mozart's Piano Sonatas*. Upsalla University, Department of Music.

Biles, J. a. (1999). Life with GenJam: interacting with a musical IGA. *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, *3*(FEBRUARY 1999), 652–656. doi:10.1109/ICSMC.1999.823290

Biles, J. Al. (1994). GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the International Computer Music Conference* (p. 131).

Biles, J. Al. (1998). Interactive GenJam: Integrating real-time performance with a genetic algorithm. In *Int. Computer Music Conf.(ICMC'98)* (pp. 10–13). Retrieved from http://www.academia.edu/download/30592586/BilesICMC98.pdf

Biles, J. Al. (2001). Autonomous GenJam: eliminating the fitness bottleneck by eliminating fitness. *Proceedings of the 2001 Genetic and Evolutionary …*. Retrieved from http://www.ist.rit.edu/~jab/GECCO01/

Biles, J. Al. (2002a). GenJam: evolution of a jazz improviser, (August), 165–187. doi:http://dx.doi.org/10.1016/B978-155860673-9/50042-2

Biles, J. Al. (2002b). GenJam in Transition: from Genetic Jammer to Generative Jammer. *Proceedings of Generative Art*, 1–5. doi:10.3183/NPPRJ-1998-13-03-p225-232

Biles, J. Al. (2003). GenJam in Perspective: A Tentative Taxonomy for GA Music and Art Systems. *Leonardo*, *36*(1), 43–45. doi:10.1162/002409403321152293

Biles, J. Al. (2013). Performing with Technology: Lessons Learned from the GenJam Project. *Aiide*, (Xrijf), 14–19. Retrieved from http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/download/7422/7668

Biles, J. Al, Anderson, P. G., & Loggi, L. W. (1996). Neural Network Fitness Functions for a Musical IGA. In *International ICSC Symposium on Intelligent Industrial Automation (IIA'96) and Soft Computing (SOCO'96)*.

Biles, J. Al, & Eign, W. (1995). GenJam Populi: Traning an IGA via Audience-Mediated Performance. In *International Computer Music Conference* (pp. 347–348).

Blacking, J. (1973). *How Musical is Man*. University of Washington Press.

Bongers, B. (2000). Physical interfaces in the electronic arts. Interaction theory and interfacing techniques for real-time performance. In *Trends in Gestural Control of* (pp. 41–70). Citeseer.

Bown, O., Eldridge, A., & McCormack, J. (2009). Understanding Interaction in Contemporary Digital Music: from instruments to behavioural objects. *Organised Sound*, *14*(2), 188. doi:10.1017/S1355771809000296

Britannica online. (n.d.). In *Britannica online*. Retrieved from https://www.britannica.com/art/musical-instrument

Brown, R. (1999). *Ray Brown's Bass Method*. Hal Leonard.

Campbell, P. S. C. (2009). Learning to Improvise Music, Improvising to Learn Music. In G. Solis & B. Nettl (Eds.), *Musical Improvisation: Art, Education and Society* (pp. 119–142). The University of Illinois Press.

Carter, R. (1998). *Building Jazz Bass Lines*. Hal Leonard.

Chadabe, J. (1984). Interactive Composing: An Overview. *Computer Music Journal*, *8*(1), 22–27.

Chadabe, J. (1997). *Electric Sound*. Prentice Hall.

Chadabe, J. (2002). The limitations of mapping as a structural descriptive in electronic instruments. *Proceedings of the 2002 International Conference for New Instruments for Musical Expression (NIME-2002)*, 1–5. Retrieved from https://pdfs.semanticscholar.org/4751/10b3403a9344464ae869748be56469d88cf8.pdf

Chemillier, M. (2004a). Steedman's grammar for jazz chord sequences.

Chemillier, M. (2004b). Toward a formal study of jazz chord sequences generated by Steedman?s grammar. *Soft Computing*, *8*(9), 1–6. doi:10.1007/s00500-004-0386-3

Chen, D. (1992). *Computer Improvisation of Jazz Solos*. Rochester Institute of Technology.

Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.

Coker, J. (1964). *Improvising Jazz*. New Jersey: Prentice Hall.

Cont, a. (2008). ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music. *Proceedings of the 2008 International Computer Music Conference, Belfast, Northern Ireland*. Retrieved from http://articles.ircam.fr/textes/Cont08a/%5Cnpapers://616814c5-046b-40e4-b13b-55fce5f60979/Paper/p6954

Cookie Apps Inc. (2013). Real Piano (version 2.1.3) [Mobile application software]. Retrieved from http://cookieapps.com

Cope, D. (n.d.). David Cope. Retrieved May 22, 2017, from http://artsites.ucsc.edu/faculty/cope/5000.html

Cope, D. (1987). An Expert System for Computer-Assisted Composition. *Computer Music Journal*, *11*(4), 30–46.

Cope, D. (2001). *New Directions in Music*. Waveland Press, Inc.

Cope, D. (2004). *Virtual Music*. The MIT Press.

Cope, D. (2005). *Computer Models of Musical Creativity*. MIT Press.

Cork, C. (1988). *Harmony by LEGO Bricks□: A New Approach tothe Use of Harmony in Jazz Improvisation*. Leicester: TadleyEwing Publications.

Dannenberg, R. (1984). An On-Line Algorithm for Real-Time Accompaniment. *Proceedings of the 1984 International Computer Music Conference*, 193–198.

Dannenberg, R. (1993). *The CMU Toolkit, version 3*. Carnegie Mellon University.

Dean, R. (2003). *Hyperimprovisation: Computer-Interactive Sound Improvisation*. A-R Editions.

Dias, R., & Guedes, C. (2013). A contour-based walking bass generator. In *Proceedings of the Sound and Music Computing Conference 2013, SMC 2013, Stockholm, Sweden* (pp. 305–308).

Dias, R., Guedes, C., & Marques, T. (2014). A computer-mediated Interface for Jazz Piano Comping. In *International Computer Music Conference* (pp. 14–20).

Dias, R., Marques, T., Sioros, G., & Guedes, C. (2011). Gimme 'Da Blues: A Jazz/Blues Player And Automatic Comping Generator For iOS Multitouch Devices. *inForum Coimbra*.

Dias, R., Marques, T., Sioros, G., & Guedes, C. (2012). GimmeDa Blues□: A n Intelligent Jazz / Blues Player And Comping Generator for iOS devices. In *CMMR London 2012* (pp. 482–490).

Dodge, C., & Jerse, T. A. (1985). *Computer Music: Synthesis, Composition and Performance*. Macmillan Library Reference.

Drummond, J. (2009). Understanding Interactive Systems. *Organised Sound*, *14*(2), 124. doi:10.1017/S1355771809000235

Ebcioglu, K. (1990). An expert system for harmonizing chorales in the style of J. S. Bach. *Journal of Logic Programming*, *8*, 145–185.

Emura, N., Miura, M., & Yanagida, M. (2006). A System yielding Jazz-style arrangement for a given melody. In *Western Pacific Acoustics Conference* (pp. 1–8). Seoul. Retrieved from http://miu.i.ryukoku.ac.jp/miura/pdf_int/INT2006-06-WESPAC9-JAZZSTYLE.pdf

Essl, K. (n.d.-a). Lexicon Sonate. Retrieved June 20, 2017, from http://www.essl.at/works/Lexikon-Sonate.html#art

Essl, K. (n.d.-b). RTC-lib. Retrieved June 20, 2017, from http://www.essl.at/works/rtc.html#art

Far Out Labs. (2013). ProRemote (version 2.5.4) [Mobile application software]. Retrieved from http://www.folabs.com

Favreau, E., Fingerhut, M., Koechlin, O., Potacsek, P., Puckette, M., & Rowe, R. (1986). Software developments for the 4x real-time system. In *International Computer Music Conference* (pp. 369–373).

Fischer, R. J. (2013). TouchOSC (version 1.8.1) [Mobile application software]. Retrieved from http://hexler.net/

FlexiMusic. (n.d.). Retrieved from http://fleximusic.com/product/fleximusic-composer

Forte, A., & Gilbert, E. (1982). *Introduction to Schenkerian Analysis*. W. W. Norton & Company.

Gannon, P. (1991). Band-in-a-Box. PG Music. *Inc., Hamilton, Ontario*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Band-in-a-Box.+PG+Music,+Inc.#0

Garton, Brad; Topper, D. (1997). RTcmix - Using CMIX in Real Time. In *International Computer Music Conference*.

Goldsby, J. (2002). *The Jazz Bass Book Technique and Tradition*. Backbeat Books.

Goto, M. (1996). A Jazz Session System for Interplay among All Players. In *International Computer Music Conference*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.32.9832&rep=rep1&type=pdf

Gross, A. (2013). MusicStudio (version 2.2.1) [Mobile application software]. Retrieved from http://www.xewton.com/musicstudio/overview/

Guedes, C. (2005a). *Mapping movement to musical rhythm: A study in interactive dance*. New York University. Retrieved from http://gradworks.umi.com/31/66/3166525.html

Guedes, C. (2005b). The m-objects□: a small library for musical rhythm generation and musical tempo control from dance movement in real time. In *International Computer Music Conference*.

Guedes, C. (2006). Extracting Musically-Relevant Rhythmic Information from Dance Movement by Applying Pitch Tracking Techniques to a Video Signal. *Proceedings of the 2006 Sound and Music Computing Conference*, 25–33.

Harley, J. (2004). *Xenakis His life in Music*. (Routledge, Ed.). New York.

Hellerman, W. (1971). Questions and Answers. In T. Everett (Ed.), *The Composer Magazine* (pp. 79–92).

Hidaka, I., Goto, M., & Muraoka, Y. (1995). An automatic jazz accompaniment system reacting to solo. *In Proc. of ICMC*, (Figure 1), 167–170. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.1098

Hodson, R. (2007). *Interaction, Improvisation, and Interplay in Jazz*. *Notes* (Vol. 64). Routledge.

Hornbostel, E. M. V. (1933). The Ethnology of African Sound-Instruments. Comments on "Geist und Werden der Musikinstrumente" by C. Sachs. *Africa*, *6*(2), 129–157.

Houge, B. (2014). Food Opera□: Transforming the Restaurant Soundscape. In *Invisible Places*.

Hunt, A., & Kirk, R. (2003). MidiGrid: Past, Present and Future. *Proceedings of the 2003 Conference on New Interfaces for Musical Expression (NIME-03)*, 135–139. Retrieved from http://www.nime.org/proceedings/2003/nime2003_135.pdf

Johnson-Laird, P. N. (1991). Jazz improvisation: A theory at the computational level. *Representing Musical Structure*. Retrieved from https://psych.princeton.edu/~psych/psychology/research/johnson_laird/pdfs/1991jazzimprovisation.pdf

Johnson-Laird, P. N. (2002). How Jazz Musicians Improvise. *Music Perception*, *19*(3), 415–442. doi:10.1525/mp.2002.19.3.415

Kitahara, T., Katsura, M., Katayose, H., & Nagata, N. (2008). Computational Model for Automatic Chord Voicing based on Bayesian Network, (Icmpc 10), 395–398.

Korg inc. (2013). Korg iMS-20 (version 1.6.2) [Mobile application software]. Korg Inc. Retrieved from http://www.korg.com/ims20

Kvifte, T. (2008). What is a musical instrument?

Kvifte, T. (2011). Musical Instrument User Interfaces : the Digital Background of the Analogue Revolution (Keynote presentation). In *New Interfaces for Musical Expression*.

Lansky, P. (1990). The Architecture and Musical Logic of Cmix. In *International Computer Music Conference*. Glasgow.

Larson, S. (1987). *Schenkerian Analysis of Modern Jazz*. University of Michigan.

Larson, S. (1998). Schenkerian Analysis of Modern Jazz: Questions about Method. *Music Theory Spectrum*, *20*(2 Fall), 209–241.

Larson, S. (2009). *Analyzing Jazz: A Schenkerian Approach*. Pendragon Press.

Laurson, M., & Kuuskankare, M. (2002). PWGL: a novel visual language based on Common Lisp, CLOS and OpenGL. In *International Computer Music Conference* (pp. 142–145).

Levin, R. (2009). Improvising Mozart. In G. Solis & B. Nettl (Eds.), *Musical Improvisation: Art, Education and Society* (pp. 143–149). The University of Illinois Press.

Levine, M. (1995). *The jazz theory book*. Sher Music Co.

Levitin, D. (2006). *This is your brain on music*. New York, USA: DUTTON.

Levitt, D. A. (1981). *A melody description system for jazz improvisation*. MIT.

Loy, G. (2006). *Musimathics: the mathematical foundations of music* (Vol. 1.). MIT Press.

Machover, T. (1992). Hyperinstruments: A progress report 1987-1991. Media Lab Document. MIT, Cambridge, MA.

Magnusson, B. (1999). *The Art Of Walking Bass*. Musicians Institute Press.

Magnusson, T. (2011). The IXI Lang: A SuperCollider Parasite for Live Coding. In *International Computer Music Conference*.

Mann, A., & Edmunds, J. (1965). *The Study of Counterpoint from Johann Joseph Fux's Gradus Ad Parnassum*. (A. Mann & J. Edmunds, Eds.). Norton.

Manning, P. (2004). *Electronic and computer music* (Vol. 3). Oxford University Press.

Martin, H. (2011a). Review: More Than Just Guide Tones□: Steve Larson ' s Analyzing Jazz — A Schenkerian Approach. *The Journal of Jazz Studies*, *7*(1), 121–144.

Martin, H. (2011b). Schenker and the Tonal Jazz Repertory. *Dutch Journal of Music Theory*, *16*(1), 1–20.

Mathews, M. V., Pierce, J. R., & Guttman, N. (1962). Musical Sounds from Digital Computers. *Gravesaner Blätter*, *23/24*, 109-118-128.

McCartney, J. (1996). SuperCollider, a New Real Time Synthesis Language. In *International Computer Music Conference* (pp. 257–258). Hong-Kong.

Miranda, E. R., & Biles, J. Al. (2007). *Evolutionary Computer Music*. *Computer*. Springer Verlag.

MooCowMusic. (2013). Guitarist (version 1.9) [Mobile application software]. Retrieved from http://moocowmusic.com

Moog Music Inc. (2013). Animoog (version 2.1.0) [Mobile application software]. Retrieved from http://www.moogmusic.com

Mooney, S. (2011). *Constructing Walking Jazz Bass Lines*. Waterfall Publishing House.

Motu inc. (2013). DPControl (version 1.0.2) [Mobile application software]. Retrieved from http://motu.com

MousMuso. (n.d.). Retrieved from http://www.busker.net/mousmuso/index.html

NETTuno s. r. l. (2013). Virtual Guitar (version 2.5.1) [Mobile application software]. Retrieved from http://www.nettn.com/app.php

Nierhaus, G. (2009). *Algorithmic Composition - Paradigms of Automated Music Generation*. Springer.

Nilsson, P. A. (2008). The walking machine. In *ICMC 2008 proceedings* (pp. 1–3).

Opal Limited. (2013). Bloom (version 2.1.1) [Mobile application software]. Retrieved from http://www.generativemusic.com/bloom.html

Oxford Music online. (n.d.). Retrieved April 22, 2015, from http://www.oxfordmusiconline.com

Pachet, F. (1999). Surprising Harmonies. *International Journal on Computing Anticipatory Systems*, *4*(2), 1–20. doi:10.1002/kin.550250208

Pachet, F. (2002). The Continuator: Musical interaction with style. *Journal of New Music Research*. Retrieved from http://www.tandfonline.com/doi/abs/10.1076/jnmr.32.3.333.16861

Pachet, F., Suzda, J., & Martin, D. (2013). A Comprehensive Online Database of Machine-Readable Lead Sheets for Jazz Standards. *ISMIR, Curitiba (Brazil)*. Retrieved from http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/149_Paper.pdf

Pennycook, B., Stammen, D., & Reynolds, D. (1993). Toward a Computer Model of a Jazz

Improvisor. *International Computer Music Conference*, 15–16. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Toward+a+Computer+Model+of+a+Jazz+Improviser#0

Pickering, J. (1978). *Mel Bay's Studio: Jazz Drum Cookbook*. Mel Bay Publications, Inc.

Pink Twins. (2013). Fantastick (version 2.7) [Mobile application software]. Retrieved from http://www.pinktwins.com/fantastick/

Puckette, M. (1986). Interprocess communication and timing in real-time computer music performance. In *International Computer Music Conference* (pp. 43–46).

Puckette, M. (1988). The Patcher. *International Computer Music Association*. Retrieved from http://hdl.handle.net/2027/spo.bbp2372.1988.046

Puckette, M. (1991). Event and Signal Processing in the MAX Graphical Programming Environment. *Computer Music Journal*, *15*(3), 68–77.

Puckette, M. (2006). Preface. In C. Agon, G. Assayag, & J. Bresson (Eds.), *The OM composer's book 1* (pp. ix–xiv). Ircam Centre Pompidou.

Pulse Code Inc. (2013). RhythmStudio (version 1.08) [Mobile application software]. Retrieved from http://www.pulsecodeinc.com/rhythm-studio

ReacTable SystemsSL. (2013). ReactableMobile (version 2.1.3). Retrieved from http://reactable.com/products/mobile/

Retronyms. (2013). Tabletop (version 2.1) [Mobile application software]. Retrieved from http://retronyms.com

Roads, C. (1996). The Computer Music Tutorial. In *The Computer Music Tutorial* (p. 245). MIT Press.

Rowe, R. (1993). *Interactive music systems: machine listening and composing*. MIT Press Cambridge, MA, USA.

Rowe, R., & Singer, E. L. (1997). Two Highly Integrated Real-Time Music and Graphics Performance Systems. In *International Computer Music Conference*.

Saitara Software. (2013). AC-7 Core HD (version 1.0) [Mobile application software]. Retrieved from http://www.saitarasoftware.com

Schottstaedt, W. (1989). Automatic counterpoint. In *Current directions in computer music research* (pp. 199–244). MIT Press Cambridge, MA, USA.

Schwanauer, S., & Levitt, D. A. (1993). *Machine Models of Music*. MIT Press.

Simoni, M., & Dannenberg, R. (2013). *Algorithmic Composition: A Guide to Composing Music with Nyquist*. The University of Michigan Press.

Smudge Apps. (2017). Band (version 3.0) [Mobile application software]. Retrieved from https://itunes.apple.com/us/app/smudge-apps-band/id293003293?mt=8

Smule. (2013). Ocarina (version 1.4.5) [Mobile application software]. Retrieved from http://www.smule.com/ocarina

Sonosaurus LLC. (2013). Thumbjam (version 2.3) [Mobile application software]. Retrieved from http://thumbjam.com

Sorensen, A. (2005). Impromptu: An interactive programming environment for composition and performance. In *Australasian Computer Music Conference* (pp. 149–154).

Spiegel, L. (n.d.). Music Mouse. Retrieved April 20, 2015, from http://musicmouse.com

Steedman, M. (1984). A Generative Grammar for Jazz Chord Sequences. *Music Perception*, *2*(1), 53.

Steinberg. (2013). Cubase iC (version 1.2) [Mobile application software]. Retrieved from http://www.steinberg.net/

Stephans, M. (2013). *Experiencing Jazz: A Listeners Guide*. Scarecrow Press.

Trump, S. (2013). Orphion (version 1.5) [Mobile application software]. Retrieved from http://www.orphion.de

Ulrich, J. (1977). The Analysis and Synthesis of Jazz by Computer. *IJCAI*.

Vercoe, B. (1984). The sinthetic performer. In *ICMC '84 proceedings* (pp. 199–200).

Voicing (music). (n.d.). Retrieved from https://en.wikipedia.org/wiki/Voicing_(music)

Wang, G. (2014). The role of technology in music. Retrieved from https://www.ted.com/talks/ge_wang_the_diy_orchestra_of_the_future

Wang, G., & Cook, P. R. (2003). ChucK: A Concurrent, On-the-fly, Audio Programming Language. *ICMC 2003*, 1–8.

Watanabe, J., Watanabe, K., & Emura, N. (2008). A system generating jazz-style chord sequences for solo piano. In *International Conference on Music Perception and Cognition* (pp. 2–6). Retrieved from http://miu.i.ryukoku.ac.jp/miura/pdf_int/INT2008-08-ICMPC10-JAZZ.pdf

Winkler, T. (2001). *Composing Interactive Music, Techniques and Ideas Using Max*. MIT Press.

Xenakis, I. (1992). *FormalizedMusic*. Pendragon Press.

Zicarelli, D. (1987). M and Jam Factory. *Computer Music Journal*, *11*(4), 13–29.

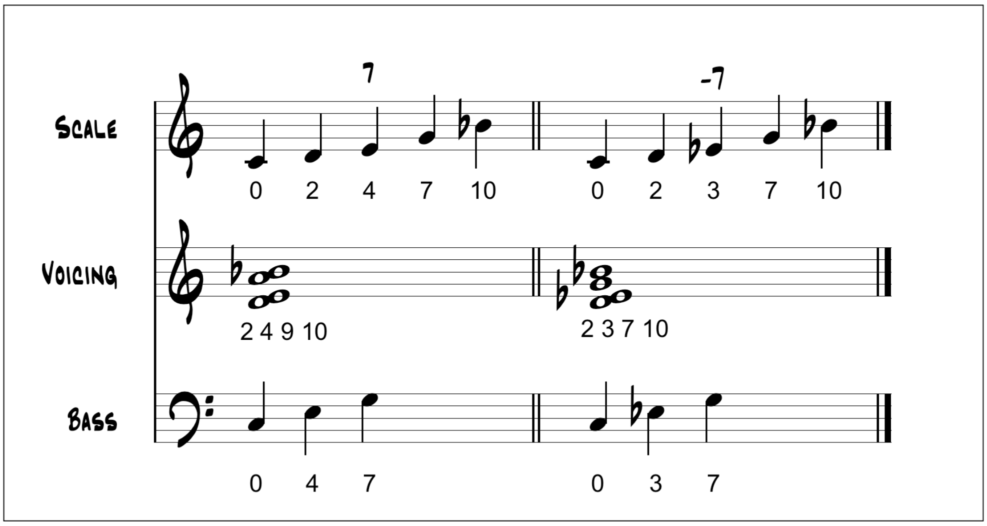
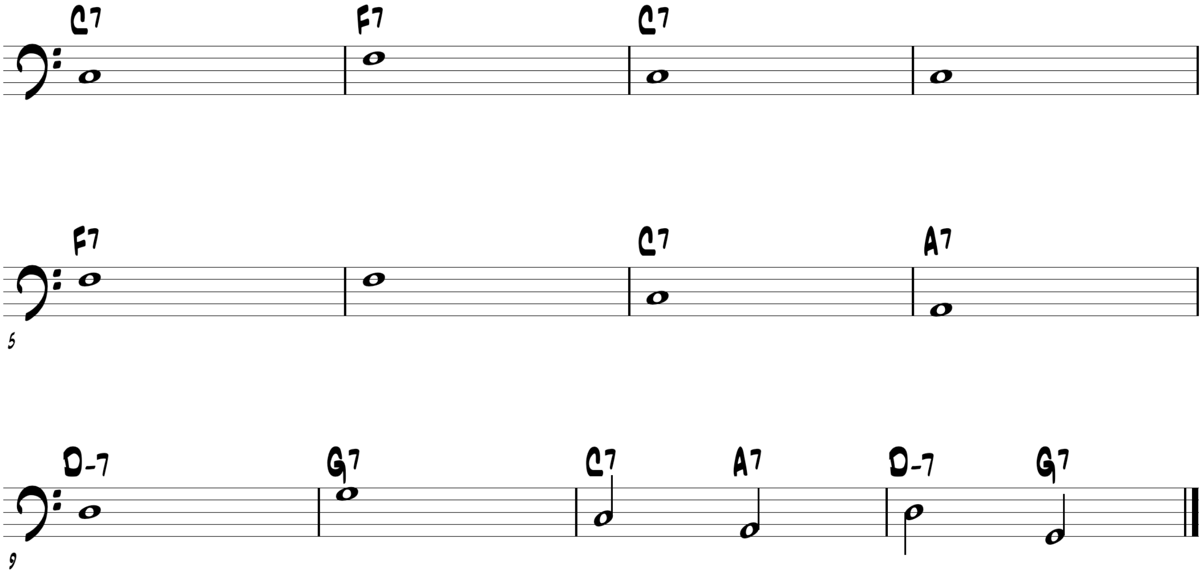
APPENDICES

APPENDIX A: GimmeDaBlues style templates

Classic Blues

Gimme 'da Blues style

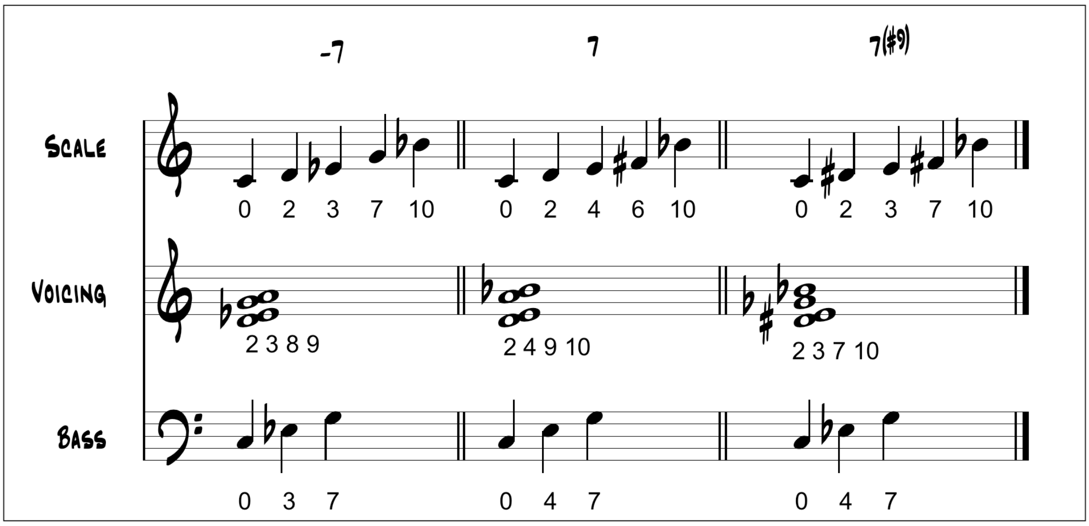
Kinetic Project

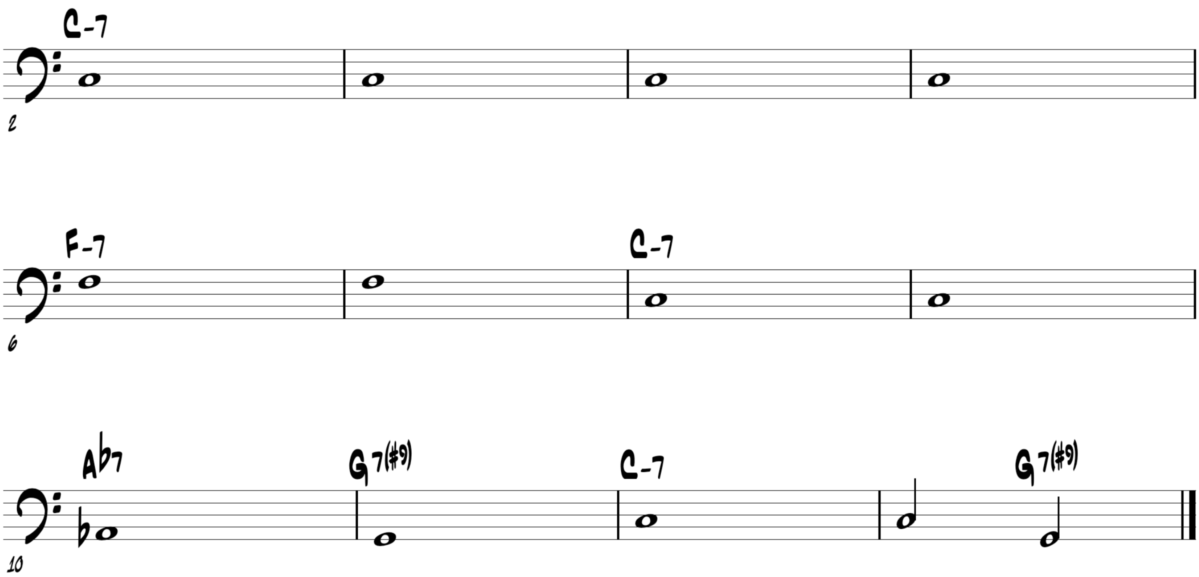


Minor Blues

Gimme 'da Blues style

Kinetic Project





APPENDIX B: *PocketBand* and *MyJazzBand* song template

PocketBand and *MyJazzBand*: song template

```
// SONG DEFINITION
song
title "Standard Blues"
key C
signature 4*4
bpm 150
swing 2
style "Advanced_Jazz.txt"
sections 3


// SECTION DEFINITION
section 1
name Intro
bars 4
// if the the section time signature and/or bpm is the same as the song
// it's not necessary to define them
// bpm 140
// signature 4 4
// transport loop

chords
1.1 C 7 mixo 1
2.1 C 7 mixo 1
3.1 C 7 mixo 1
4.1 C 7 mixo 1


// SECTION DEFINITION
section 2
name Chorus
bars 12
// transport loop

chords
1.1 C 7 mixo 1
2.1 F 7 mixo 1
3.1 C 7 mixo 1
4.1 C 7 mixo 1
5.1 F 7 mixo 1
6.1 F 7 mixo 1
7.1 C 7 mixo 1
8.1 A 7 mixo 1
9.1 D m7 minor 1
10.1 G 7 mixo 1 Db 7 mixo 1
11.1 C 7 mixo 0.2
11.3 A 7 mixo 0.2 Eb 7 mixo 0.2
12.1 D m7 minor 0.2 Ab 7 mixo 0.2
12.3 G 7 mixo 0.2 Db 7 mixo 0.2


// SECTION DEFINITION
section 3
name Ending
bars 4
```

```
// transport stop

chords
1.1 D m7 minor 1
2.1 G 7 mixo 1
3.1 C 7 mixo 1
4.1 C 7 mixo 1
```