UNIVERSITÀ DI PADOVA

TESI DI LAUREA SPECIALISTICA

AUTOMATIC EXECUTION OF EXPRESSIVE MUSIC PERFORMANCE

Laureando:
Jehu Procore
NJIKONGA NGUEJIP
Matricola: 588522-IF

Relatore:
Prof. Antonio RODÀ

Corso di Laurea Specialistica in Ingegneria Informatica

8 luglio 2014

Anno Accademico 2013-2014

Abstract

The definition of computer models to represent the expressiveness of a musical performance, is useful to try to understand how and what way anyone can express expressive intentions in a music performance. The `CaRo 2.0` is a computer model or software system that allows automatic computation in interactive way for rendering expressive musical scores. Initially, the `CaRo 2.0` was implemented using the borland Developer[1] compiler destiny to run only and exclusively on Microsoft environment, which limits the interest of the product; and the rendering in `CaRo 2.0` application is done through a MIDI synthesizer. MIDI is a wonderful format for performance applications like sequencers, but it is not so wonderful for other applications like music notation. For example, MIDI does not represent stem direction, beams, repeats, slurs, measures, and many other aspects of notation. And sharing music between computer music programs in the world used to be difficult. Herein lies the power of MusicXML. Write a song for any instrumentation group, export to MusicXML, and import into any other notation app, maintaining all of your original score elements. This thesis relates to the porting of the older `CaRo 2.0` into a cross platform ant the integration into the new software implemented of the MusicXML library.

Since 2002, a scientific initiative brings together scientists from all over the world for a competition of artificially created performances (RENCON, contest for performance rendering systems[2]). Their aim is to construct computational systems that are able to pass a kind of expressive performance Turing Test (that is, an artificial performance sounds indistinguishable from a human performance, Hiraga et al.,2004 [31]). The very ambitious goal proclaimed by the RENCON initiative is for a computer to win the Chopin competition by 2050 (Hiraga et al., 2004 [31]). It is hard to imagine that this will ever be possible, not only because the organisers of such a competition will probably not permit a computer to participate, but also because a computational model would have to take into account the complex social and cognitive contexts in which, like any human intellectual and artistic activity, a music performance is situated. But even if complete predictive models of such phenomena are strictly impossible, they advance our understanding and appreciation of the complexity of artistic behaviour, and it remains an intellectual and scientific challenge to probe the limits of formal modelling and rational characterisation.

In view of the conference on July 6, 2011 SMC Rencon, it was considered very important the improvement of the CaRo software from a point of view of its efficiency in terms of ability to infuse expressive intentions with a song music, so that the application was able to compete

[1]c++ Builder version 6
[2]http://renconmusic.org/

with the other competitors participating in the workshop. Furthermore, the choice to improve the efficiency of the program, would have made him more attractive to possible future developments.

Thanks: My goal has been achieved

My goal has been achieved after many vicissitudes. The University of Padua opened me the doors in 2004, and three years after I get my Bachelor's degree in Computer Engineering, the next year I sign up for the Master Degree in Computer Engineering, the year is marked with the beginning of the financial crisis that hit the world, anxious to integrate the world of work, beginning a growth business, and in the meantime I can give missing exams.

I thank Italy, and the University of Padua, who gave me the possibility to realize myself, if I could go back, I would do the same experience (maybe finishing the master degree in time) without any doubt.

A sincere thank you goes to the family Fabris (Montegalda, VI) for everything with heart and so much love has given me and still continue to do.

Thanks to Professor Antonio Rodà who gave me the opportunity to finish my student's career, who has been patient with me, and took into account all the discontinuities that have been in this modest thesis work.

During the thesis work, I got married, and I also wanted to thank my wife for moral support, emotional, and its presence near to me while traveling from Milan to Padova for the various discussions with my supervisor to complete this thesis.

Finally, thanks to Mom Hélène Tankoua and Dad André Nguéjip, this Master degree is for you.

Indice

Elenco delle figure

Capitolo 1

Expressive Music Performance: The State of the Art

1.1 Introduction

Like any human intellectual activity, music performance is a complex social and cognitive phenomenon with a rich context. This chapter has tried to give a comprehensive overview of the state of the art of computational modeling of expressive music performance. Four models and their approaches were presented, practical evaluations of the models on real performance data were showed. Music performance as the act of structuring and physically realizing a piece of music is a complex human activity with different aspects. This study will focus on one specific aspect which is expressive music performance.

Computational modeling is an attempt at formulating hypotheses concerning expressive performance in such a way that they can be verified on real measured performance data. Four specific models discussed in this study as: the rule- based performance model developed at KTH, the structure level models of timing and dynamics, the mathematical model of musical structure and expression by Guerino Mazzola, Machine learning model. These models focus on commonalities between performances and performers.

Scientific model defines as a familiar structure used as an analogy to interpret a natural phenomenon. Computational modeling involves embodying mathematical models in computer programs that can immediately be applied to given data for testing and prediction purposes. A mathematical or computational model is predictive in the sense that, assuming a specific fixed setting of all parameters, the model predicts the values of a specific set of variables from the values of other variables. The purpose of computational models of expressive music performance is to specify the physical parameters defining a performance, and to quantify systematic relationships between certain properties of the musical score, and an actual performance of a given piece. Models in human domain can not be expected to be correct as their predictions will always correspond to the behavior observed in humans. There are some implicit computational models that base their predictions on case-based reasoning.

The KTH model consists of a set of performance rules that predict aspects of timing, dynamics, and articulation, based on local musical context. The rule refers to a limited class of musical situations. Most of the rules operate at a rather low level, looking only at very local contexts and affecting individual notes, but there are rules that refer to entire phrases. The rules are parameterized with a varying number of parameters. The KTH model involves a professional musician directly evaluating any tentative rule brought by researcher. Musician and researcher are in a constant feed-back loop trying to find the best formulation and parameter settings for each rule. The important aspect of the model is that it is additive. This additivity is a particular problem when trying to fit the parameters to collections of real recordings. The model is a useable description language for expressive performance. In the experiment, the different emotional intentions as represented in the performances were differentiated by particular rules. The KTH model has been used to model certain emotional colorings that might not be immediately seen in the music structure. The extension to emotionality has led to a more comprehensive computational model of expressive performances (the GERM model). There is evidence that KTH rule model is a viable representation language for describing expressive performance.

The Todd model in contrast to the KTH model may be summarized under the notion of analysis by measurement, because they obtain their empirical evidence directly from measurements of human expressive performances. The essence of these models is the assumption that there is a direct link between certain aspects of the musical structure and the performance. Also this relation can be modeled by one single, simple rule. The simplistic nature of Todds model has the potential advantage that its theoretical assumptions can be tested relatively easily. Todd compared the models output with the tempo and dynamics curves from one or two performances of selected pieces by Haydn, Mozart, and Chopin. Todd model was used as an analysis tool to access the idiosyncrasies of human performance.

A different model based mainly on mathematical consideration is the Mazzola model. The Mazzola model builds on mathematical music theory that not only covers various aspects of music theory and analysis through a highly complex mathematical approach, but also involves all sorts of philosophical, semiotic, and aesthetic considerations. The Mazzola model consists of an analysis part and a performance part. The analysis part involves computer-aided analysis tools for various aspects of the music structure, as meter, melody, or harmony. Each of these is implemented in RUBBETTEs that assign particular weights to each note in a symbolic score. The performance part transforms structural features into an artificial performance is theoretically anchored in Stemma Theory and Operator Theory. There is a linear mapping between metrical weight and tone intensity to generate artificial performances.The metrical, harmonic, and melodic weights as provided by the RUBATO software served as independent variables. The overall model could explain 84% of the average tempo curve of the 28 performances, each of the three analytical components contributing equally to the model.

An alternative way of building computational models of expressive performance is to start from large amounts of empirical data and to have the computer autonomously discover significant regularities in the data by inductive machine learning and data mining techniques. These learning algorithms produce general performance rules that can be interpreted and used directly as predictive computational models. The machine learning can predict local, note-level expressive deviations and higher level phrasing patterns. These two types of models can be combined to yield an integrated multi-level model of expressive timing and dynamics. Musicians understand the music in terms of a multitude of more abstract structures, and they use tempo, dynamics and articulation to shape these structures. Music performance is a multi-level phenomenon with musical structure and performance patterns at various levels embedded within each other. There is a natural way of combining the phrase-level prediction model wit the rule- based learning model. After fitting quadratic approximation polynomials to given tempo or dynamics curve and subtracting from the original curve, what is left is residuals. Residuals are those low-level, local timing and dynamics deviations that can not be explained by phrases. The learning algorithm can be used to learn a rule based model of these local effects. Both the note- level rule model and the multi-level model have been tested on real performances. The machine processes the performance data and it produce reasonable performance pattern.

The study showed the quantification of individual styles. Predictive models like those described above generally focused on fundamental, common performance principles. The study shows the differences between artists that is aspects of personal artistic performance style. A statistical analysis revealed a number of characteristics and distinctive phrasing behaviors, some of which could be associated with certain pianists. The resulting performance data can be represented in an integrated way as trajectories in a tempo-loudness space that show the joint development of tempo and dynamics over time. Various ways toward the characterization of individual performance style are performance alphabet, and automatic identification of performers. The performance trajectories must first be converted into a form that is accessible to the automated date analysis machinery provided by data mining. The trajectories are cut into short segments. The resulting segments can be grouped into classes of similar patterns via clustering. They can see as a simple alphabet of performance, restricted to tempo and dynamics. Such performance alphabets support a variety of quantitative analysis. Another way to trying to quantify individual performance style is to develop computer programs that attempt to identify artists on the basis of their performance characteristics.

The result of this study shows that there is ample room for further research, and the field of computational performance modeling continues to be active. The new research could be finding the new control spaces and devices to control emotional aspects of music performance. The person and personality of the artist should be considered as one of the main factors in the models described in the study. There are some others aspects to be considered in predictive models, like listener expectations, performance context, artistic intentions, and personal experience.

1.2 Music performance parameter’s

The same piece of music played by different musicians will not be the same as ever , in fact this is what distinguishes a good musician to another. The attributes of the musical notes : pitch , timbre , tempo, dynamics or changes in sound intensities are not specified in the musical score and the performer must know how to use it properly. These attributes are called parameters and vary during the performance of music. A good musician is distinguished from another by his way of interpreting a song. Music performance as the act of interpreting, structuring, and physically realising a piece of music is a complex human activity with many facets: physical, acoustic, physiological, psychological, social, artistic. To express their musical intention, musicians often make gestures and moves their bodies. Body movement is an important non-verbal means of communication between humans. Body movements can help observers extract information about the course of action, or the intent of a person. This visual information provides a channel of communication to the listener of its own, separated from the auditory signal. In order to explore to what extent emotional intentions can be conveyed through musicians' movements, subjects watched and rated silent video clips of musicians performing four different emotional intentions, Happy, Sad, Angry, and Fearful. Some of this information is very robust and can be perceived even when certain parts of the moving body are occluded. It has been shown that by viewing a motion patterns, subject are able to extract a number of non-trivial features such as the sex of a person, the weight of the boxes she is carrying. It also possible to identify the emotional expression in dance and music performance, as well as the emotional expression in every-day arm movements such as drinking and lifting. Music has an intimate relationship with movement in several aspects. The most obvious relation is that all sounds from traditional acoustic instruments are produced by human movements. Some characteristics of this motion will inevitably be rejected in the resulting tones. For example, the sound level, amplitude envelope, and spectrum change during a tone on a violin has a direct relationship to the velocity and pressure during the bow gesture. Also, the striking velocity in drumming is strongly related to the height to which the drumstick is lifted in preparation for the stroke [10] [11]. Musicians also move their bodies in a way that is not directly related to the production of tones. Head shakes or bodysway are examples of movements that, although not actually producing sound, still can serve a communicative purpose of their own. In this respect the movements and the spoken words are co-expressive, not subordinate to each other. Bearing in mind that music also is a form of communication and that speech and music have many properties in common it is plausible that a similar concept applies to musical communication as well.

We prefer to think of these performer movements as a body language since, as we will see below, they serve several important functions in music performance. It seems reasonable to assume that some of the expressivity in the music is rejected in these movements.

The body movments may also be used for more explicit communications. Davidson and Correia (2002) [14] suggest four aspects that influence the body language in musical performances:

1. Communication with co-performers,
2. Individual interpretations of the narrative or expressive/emotional elements of the music
3. The performer's own exprleriences and behaviors,
4. the aim to interact with and entertain and audience

Separating the influence of each of these aspects on the specific movement may not be possible in general.

1.3 How do humans make their performances?

This section is devoted to expression as represented in all kinds of movements that occur when performers interact with their instruments during performance (for an overview, see Davidson and Correia, 2002; Clarke, 2004) [14] [8]. Performers movements are a powerful communication channel of expression to the audience, sometimes even overriding the acoustic information (Davidson, 1994) [13].

1.3.1 Extracting expression from performers' movements

There are several ways to monitor performers' movements. One possibility is to connect mechanical devices to the playing apparatus of the performer (Ortmann, 1929) [40], but that has the disadvantage of inhibiting the free execution of the movements. More common are optical tracking systems that either simply video-tape a performer's movements or record special passive or active markers placed on particular joints of the performer's body. Berstein and Poppova, introduced an active photographical tracking system (Kay et al., 2003) [32]. Such systems use light-emitting markers placed on the various limbs and body parts of the performer. They are recorded by video cameras and tracked by software that extracts the position of the markers (e.g. the Selspot System, as used by Dahl, 2004, 2005) [10] [11]. The disadvantage of these systems is that the participants need to be cabled, which is a time-consuming process. Also, the cables might inhibit the participants to move as they would normally move. Passive systems use reflective markers that are illuminated by external lamps. In order to create a three-dimensional picture of movement, the data from several cameras are coupled by software (Palmer and Dalla Bella, 2004) [38]. Even less intrusive are video systems that simply record performance movements without any particular marking of the performer's limbs. Elaborated software systems(e.g. EyesWeb[1], see Camurri et al., 2004, 2005) [5] [7] are able to track defined body joints directly from the plain video signal (see Camurri and Volpe, 2004, for an overview on gesture-related research) [6]. Perception studies on communication of expression through performers' gestures use simpler point-light video recordings (reflective markers on body joints recorded in a darkened room) to present them to participants for ratings (Davidson, 1993) [12].

1.3.2 Can a computer take advantage of this knowledge and become capable to substitute a human performer?

Let'us consider in this section five ways in which a computer can learn from human performnances

1. *Investigating human expressive performance by developing computational models.* For experimentation with human performers, models can be built which attempt to simulate elements of human expressive performance. As in all mathematical and computational modeling, the model itself can give the researcher greater insight into the mechanisms inherent in that which is being modeled.

2. *Realistic playback on a music typesetting or composing tool.* There are many computer tools available now for music typesetting and for composing. If these tools play back the compositions with expression on the computer, the composer will have a better idea of

[1]http://www.megaproject.org

what the final piece will sound like. For example, Sibelius[2], Notion[3], and Finale[4] have some ability for expressive playback.

3. *Playing computer-generated music expressively.* There are a number of algorithmic composition systems that output music without expressive performance but which audiences would normally expect to hear played expressively. These compositions in their raw form will play on a computer in a robotic way. A CSEMP to generate novel and original performances as opposed to simulating human strategies.

4. *Playing data files.* A large number of nonexpressive data files in formats like MIDI and MusicXML [Good 2001] are available on the Internet, and they are used by many musicians as a standard communication tool for ideas and pieces. Without CSEMPs, most of these files will play back on a computer in an unattractive way, whereas the use of a CSEMP would make such files much more useful.

5. *Computer accompaniment tasks.* It can be costly for a musician to play in ensemble. Musicians can practice by playing along with recordings with their solo parts stripped out. But some may find it too restrictive since such recordings cannot dynamically follow the expressiveness in the soloists performance. These soloists may prefer to play along with an interactive accompaniment system that not only tracks their expression but also generates its own expression.

[2]http://www.sibelius.com/

[3]Compose, playback, and edit music with a quality and ease of use that must be experienced. No endless level of menus to find what you need. Notion is the most efficient notation product, making it simple to write and edit your ideas quickly. http://www.notionmusic.com/

[4]Whether youre creating a simple lead sheet, making worksheets for your students, or composing your magnum opus, Finale helps you easily capture your musical ideas, produce beautiful notation, and quickly share the results. http://www.finalemusic.com/

Capitolo 2

Modelling strategies

2.1 Purpose

The purpose of computational models of expressive music performance is thus to specify precisely the physical parameters defining a performance (e.g., onset timing, inter-onset intervals, loudness levels, note durations, etc.), and to quantify (quasi-)systematic relationships between certain properties of the musical score, the performance context, and an actual performance of a given piece. Of course, models in human or artistic domains cannot be expected to be correct in the sense that their predictions will always correspond to the behaviour observed in humans.

The goal of this section is to arrive at an understanding of the relationships between the various factors involved in performance that can be formulated in a general model. Models describe relations among different kinds of observable (and often measurable) information about a phenomenon, discarding details that are felt to be irrelevant. They serve to generalise empirical findings and have both a descriptive and predictive value. Often the information is quantitative and we can distinguish input data, supposedly known, and output data, which are inferred by the model. In this case, inputs can be considered as the causes, and outputs the effects of the phenomenon. Computational models models that are implemented on a computer can compute the values of output data corresponding to the provided values of inputs. This process is called simulation and is widely used to predict the behaviour of the phenomenon in different circumstances. This can be used to validate the model, by comparing the predicted results with actual observations.

2.2 Strategies to develop the structures of model

We can distinguish several strategies for developing the structure of the model and finding its parameters. The most prevalent ones are analysis-by-measurement and analysis-by-synthesis. Recently also methods from artificial intelligence have been employed: machine learning and case based reasoning. One can distinguish local models, which operate at the note level and try to explain the observed facts in a local context, and global models that take into account the higher level of the musical structure or more abstract expression patterns. The two approaches often require different modelling strategies and structures. In certain cases, it is possible to devise a combination of both approaches. The composed models are built by several components, each one aiming to explain different sources of expression. However, a good combination of the different parts is still quite a challenging research problem.

2.2.1 Analysis by measurements

The first strategy, analysis-by-measurement, is based on the analysis of deviations from the musical notation measured in recorded human performances. The goal is to recognise regularities in the deviation patterns and to describe them by means of a mathematical model, relating score to expressive values (see Gabrielsson 1999 and Gabrielsson 2003, for an overview of the main results). [26] [27] The method starts by selecting the performances to be analyzed. Often rather small sets of carefully selected performances are used. The most relevant variables are selected and analysed. The analysis assumes an interpretation model that can be confirmed or modified by the results of the measurements. Often the assumption is made that patterns deriving from different sources or hierarchical levels can be separated and then added. This assumption helps the modelling phase, but may be overly simplistic. The whole repertoire of statistical data analysis techniques is then available to fit descriptive or predictive models onto the empirical data from regression analysis to linear vector space theory to neural networks or fuzzy logic. Many models address very specific aspects of expressive performance, for example, the final ritard and its relation to human motion (Kronman and Sundberg, 1987; [33] Todd, 1995; [50] Friberg and Sundberg, 1999; [20] Sundberg, 2000;[41] Friberg et al., 2000b); [22] the timing of grace notes (Timmers et al., 2002); [47] vibrato (Desain and Honing, 1996; [15] Schoonderwaldt and Friberg, 2001); [42] melody lead (Goebl, 2001, 2003); [29], [30] legato (Bresin and Battel, 2000); [3] or staccato and its relation to local musical context (Bresin and Widmer, 2000; Bresin, 2001). [2] [4] A global approach was pursued by Todd in his phrasing model (Todd, 1992, 1995)[49] [50]. This model assumes that the structure of a musical piece can be decomposed into a hierarchy of meaningful segments (phrases), where each phase is in turn composed of a sequence of sub-phrases. The fundamental assumption of the model is that performers emphasise the hierarchical structure by an accelerando-ritardando pattern and a crescendo-decrescendo pattern for each phrase, and that these patterns are superimposed (summed) onto each other to give the actually observed complex performance. It has recently been shown empirically on a substantial corpus of Mozart performances (Tobudic and Widmer, 2006 [63]) that this model may be appropriate to explain (in part, at least) the shaping of dynamics by a performer, but less so as a model of expressive timing and tempo.

2.2.2 Analysis by synthesis

While analysis by measurement develops models that best fit quantitative data, the analysis-by-synthesis paradigm takes into account the human perception and subjective factors. First, the analysis of real performances and the intuition of expert musicians suggest hypotheses that are formalised as rules. The rules are tested by producing synthetic performances of many pieces and then evaluated by listeners. As a result the hypotheses are refined, accepted or rejected.

This method avoids the difficult problem of objective comparison of performances, including subjective and perceptual elements in the development loop. On the other hand, it depends very much on the personal competence and taste of a few experts.

The most important model developed in this way is the KTH rule system (Friberg, 1991 [16], 1995;[17] Friberg et al., 1998 [18], 2000a[21]; Sundberg et al., 1983 [43], 1989 [44], 1991 [45]). In the KTH system, a set of rules describe quantitatively the deviations to be applied to a musical score, in order to produce a more attractive and human-like performance than the mechanical one that results from a literal playing of the score. Every rule tries to predict (and to explain with musical or psychoacoustic principles) some deviations that a human performer is likely to apply. Many rules are based on a low-level structural analysis of the musical score. The KTH rules can be grouped according to the purposes that they apparently have in music communication. For instance, differentiation rules appear to facilitate categorisation of pitch and duration, whereas grouping rules appear to facilitate grouping of notes, both at micro and macro levels.

2.2.3 The machine learning model

In the traditional way of developing models, the researcher normally makes some hypothesis on the performance aspects he/she wishes to model and then tries to establish the empirical validity of the model by testing it on real data or on synthetic performances. An alternative approach, pursued by Widmer and coworkers [Widmer, 1995a [53], b[54], 1996[55], 2000[56], 2002b[58], Widmer and Tobudic, 2003 [59], Widmer, 2003 [60], Widmer et al., 2003 [61], Widmer, 2005 [62], Tobudic and Widmer, 2006 [63]], tries to extract new and potentially interesting regularities and performance principles from many performance examples, by using machine learning and data mining algorithms. The aim of these methods is to search for and discover complex dependencies on very large data sets, without a specific preliminary hypothesis. Apossible advantage is that machine learning algorithms maydiscover new (and possibly interesting) knowledge, avoiding any musical expectation or assumption. Moreover, some algorithms induce models in the form of rules that are directly intelligible and can be analysed and discussed with musicologists. This was demonstrated ina large-scale experiment (Widmer, 2002b [58]), where a machine learning system analysed a large corpus of performance data (recordings of 13 complete Mozart piano sonatas by a concert pianist), and autonomously discovered a concise set of predictive rules for note-level timing, dynamics, and articulation. Some of these rules turned out to describe regularities similar to those incorporated in the KTH performance rule set (see above), but a few discovered rules actually contradicted some common hypotheses and thus pointed to potential shortcomings of existing theories. The note-level model represented by these learned rules was later combined with a machine learning system that learned to expressively shape timing and dynamics at various higher levels of the phrase hierarchy (in a similar way as described in Todds 1989 [48]; 1992 [49] structure-level models), to yield a multi-level model of expressive phrasing and articulation (Widmer and Tobudic, 2003 [59]).

A computer performance of a (part of a) Mozart piano sonata generated by this model was submitted to the International Performance Rendering Contest (RENCON) in Tokyo, 2002, where it won the Second Prize behind a rule-based rendering system that had been carefully tuned by hand. The rating was done by a jury of human listeners. This can be taken as a piece of evidence of the musical adequacy of the model. However, as an explanatory model, this system has a serious shortcoming: in contrast to the note-level rules, the phrase-level performance model is not interpretable, as it is based on a kind of case-based learning (see also below). More research into learning structured, interpretable models from empirical data will be required.

2.2.4 Case based reasoning

An alternative approach, closer to the observation-imitation-experimentation process observed in humans, is that of directly using the knowledge implicit in human performances. Case-based reasoning (CBR) is based on the idea of solving new problems by using (often with some kind of adaptation) similar previously solved problems. An example in this direction is the SaxEx system for expressive performance of Jazz ballads [Arcos et al., 1998 [1], which predicts expressive transformations to recordings of saxophone phrases by looking at how other, similar phrases were played by a human musician. The success of this approach greatly depends on the availability of a large amount of well-distributed previously solved problems, which are not easy to collect.

2.2.5 Mathematical theory approach

A rather different model, based mainly on mathematical considerations, is the Mazzola model (Mazzola, 1990 [**?**]; Mazzola and Zahorka, 1994 [34]; Mazzola et al., 1995 [35]; Mazzola, 2002 [37]; Mazzola and Goller, 2002 [36]). This model basically consists of a musical structure analysis part and a performance part. The analysis part involves computer-aided analysis tools, for various aspects of the music structure, that assign particular weights to each note in a symbolic score. The performance part, that transforms structural features into an artificial performance, is theoretically anchored in the so-called Stemma Theory and Operator Theory (a sort of additive rule-based structure-to-performance mapping). It iteratively modifies the performance vector fields, each of which controls a single expressive parameter of a synthesised performance. The Mazzola model has found a number of followers who studied and used the model to generate artificial performances of various pieces. Unfortunately, there has been little interaction or critical exchange between this school and other parts of the performance research community, so that the relation between this model and other performance theories, and also the empirical validity of the model, are still rather unclear.

2.2.6 Perpectives

Computer-based modelling of expressive performance has shown its promise over the past years and has established itself as an accepted methodology. However, there are still numerous open questions related both to the technology, and to the questions that could be studied with it. Two prototypical ones are briefly discussed here.

2.2.6.1 Comparing performances and models

A problem that naturally arises in quantitative performance research is how performances can be compared. In subjective comparison often a supposed ideal performance is used as a reference by the evaluator. In other cases, an actual reference performance can be assumed. Of course subjects with different background may have dissimilar preferences that are not easily made explicit.

When we consider computational models, objective numerical comparisons would be desirable. In this case, performances are represented by sets of values. Various similarity or distance measures (e.g. absolute difference, Euclidean distance, etc.) can be defined over these, and it is not at all clear which of these is most appropriate musically. Likewise, it is not clear how to weight individual components or aspects (e.g. timing vs. dynamics), or how these objective differences relate to subjectively perceived differences. Agreed upon methods for performance comparison would be highly important for further fruitful research in this field.

2.2.6.2 Common principles vs. differences

The models discussed in the previous sections aim at explaining and simulating general principles that seem to govern expressive performance, that is, those aspects of the relation between score and performance that seem predictable and more or less common to different performances and artists. Recently research has also started to pay attention to aspects that differentiate performances and performers' styles (Repp, 1992; Widmer, 2003) [39] [60]. The same piece of music can be performed trying to convey different expressive intentions (Gabrielsson and Lindstrom, 2001) [28], sometimes changing the character of the performance drastically. The CARO model (Canazza et al., 2004) [9] is able to modify a neutral performance (i.e. played without any specific expressive intention) in order to convey different expressive intentions. Bresin and Friberg (2000) [23] developed some macro rules for selecting appropriate values for the parameters of the KTH rule system in order to convey different emotions. The question of the boundary between predictability and individuality in performance remains a challenging one.

2.3 Conclusions

Quantitative research on expressive human performance has been developing quickly during the past decade, and our knowledge of this complex phenomenon has improved considerably. There is ample room for further investigations, and the field of computational performance research continues to be active. As the present survey shows, the computer has become a central player in this kind of research, both in the context of measuring and extracting expression-related information from performances, and in analysing and modelling the empirical data so obtained. Intelligent computational methods are thus helping us advance our understanding of a complex and deeply human ability and phenomenon. In addition, operational computer models of music performance will also find many applications in music education and entertainment think, for instance, of expressive music generation or interactive expressive music control in multimedia applications or games, of quasi-autonomous systems for interactive music performance, of new types of musical instruments or interfaces that provide novel means of conveying expressive intentions or emotions, or of intelligent tutoring or teaching support systems in music education. Still, there are fundamental limits that will probably be very hard to overcome for music performance research, whether computer-based or not. The very idea of a creative activity being predictable and, more specifically, the notion of a direct quasi-causal relation between the content of the music and the performance has obvious limitations. The person and personality of the artist as a mediator between music and listener is totally neglected in basically all models discussed above. There are some severe general limits to what any predictive model can describe. For instance, very often performers intentionally play the repetition of the same phrase or section totally differently the second time around. Being able to model and predict this would presup- pose models of aspects that are outside the music itself, such as performance context, artistic intentions, personal experiences, listeners expectations, etc.

Although it may sound quaint, there are concrete attempts at elaborating computational models of expressive performance to a level of complexity where they are able to compete with human performers.

Capitolo 3

Computational Models of expressive music

3.1 The KTH model

This well-known rule system has been developed at the Royal Institute of Technology in Stockholm in more than 20 years of research, starting with (Sundberg et al., 1983 [43]). In the meantime it has been extended and analysed in many ways (e.g., Sundberg et al., 1989 [44]; Friberg, 1991[16]; Friberg et al., 1998 [18]) and applied also to contemporary music (Friberg et al., 1991 [16]). A comprehensive description is given in Friberg (1995a)[17].

3.1.1 The basic model

The KTH model consists of a set of performance rules that predict aspects of timing, dynamics, and articulation, based on local musical context. The rule refers to a limited class of musical situations. Most of the rules operate at a rather low level, looking only at very local contexts and affecting individual notes, but there are rules that refer to entire phrases. The rules are parameterized with a varying number of parameters. The KTH model involves a professional musician directly evaluating any tentative rule brought by researcher. Musician and researcher are in a constant feed-back loop trying to find the best formulation and parameter settings for each rule. The important aspect of the model is that it is additive. This additivity is a particular problem when trying to fit the parameters to collections of real recordings. The model is a useable description language for expressive performance. In the experiment, the different emotional intentions as represented in the performances were differentiated by particular rules. The KTH model has been used to model certain emotional colorings that might not be immediately seen in the music structure. The extension to emotionality has led to a more comprehensive computational model of expressive performances (the GERM model). There is evidence that KTH rule model is a viable representation language for describing expressive performance.

The KTH model was developed using the so-called analysis-by-synthesis approach (Sundberg et al., 1983 [43]; Friberg & Sundberg, 1987 [19]; Sundberg et al., 1991a [46]) that involves a professional musician directly evaluating any tentative rule brought forward by the researcher. Musician and researcher are in a constant feed-back loop trying to find the best formulation and parameter settings for each rule. This method has the advantage of modelling a kind of performer listener interaction (as the authors argue, see Friberg, 1995a), but falls potentially short by placing high demands on the evaluating expert musician(s) and by not permitting reliable evaluations due to the small number of judgements. The analysis-by-synthesis method was inspired by and adapted from speech synthesis methods (Friberg, 1995a [17]).

3.1.2 Empirical evaluation

The KTH Model features a large number of free parameters that govern the relative strengths of the individual notes and a number of other aspects. To turn the rule set into a truly predictive model, these parameters must be set to fixed values. Several attempts have been made to find appropriate parameter settings. Although the analysis-by-synthesis approach involves perceptual judgements from highly skilled individual musicians as a basic method, it still needs independent empirical evaluation on real performances (Gabrielsson, 1985 [25]).

3.2 The Todd model

The Todd model in contrast to the KTH model may be summarized under the notion of analysis by measurement , because they obtain their empirical evidence directly from measurements of human expressive performances. The essence of these models is the assumption that there is a direct link between certain aspects of the musical structure and the performance. Also this relation can be modeled by one single, simple rule. The simplistic nature of Todds model has the potential advantage that its theoretical assumptions can be tested relatively easily. Todd compared the models output with the tempo and dynamics curves from one or two performances of selected pieces by Haydn, Mozart, and Chopin. Todd model was used as an analysis tool to access the idiosyncrasies of human performance.

3.2.1 The basic model

The essence of these models is the assumption that there is a direct link between certain aspects of the musical structure (i.e., grouping structure) and the performance and, secondly, that this relation can be modeled by one single, simple rule. The first approaches modelled production (Todd, 1985 [**?**], 1989a [**?**]) and perception (Todd, 1989b [**?**]) of expressive timing exclusively. Todds last contribution included the interaction of timing and dynamics as well (Todd, 1992 [49]). In Todd (1989a [**?**]), a recursive look-ahead procedure was introduced that reduced the (theoretically) high demands on working memory introduced by the first model (Todd, 1985 [**?**]). The interaction between timing and dynamics was modelled by the simple relation the faster, the louder, where intensity is proportional to the squared tempo (Todd, 1992 [49], p. 3544). Thus, the hierarchical grouping structure of the music directly controls the instantaneous tempo (the more closure and thus, the more important the phrase boundary, the slower the performed tempo) as well as the expressive dynamics (in terms of hammer velocity at a computer-controlled piano). At each level of the hierarchy, the model increases tempo and dynamics towards the middle of phrases and vice versa. A basic assumption in these models is the analogy to physical motion (see also Kronman & Sundberg, 1987 [33]; Todd, 1992 [49], 1995 [50]) that may even have a (neuro-)physiological basis in the vestibular system (Todd, 1992 [49], p. 3549).

3.3 The Mazzola model

A rather different model based mainly on mathematical considerations is the Mazzola model. The Swiss mathematician and Jazz pianist Guerino Mazzola developed his mathematical music theory and performance model from the 1980s up to the present (Mazzola, 1990; Mazzola & Zahorka, 1994a; Mazzola et al., 1995; Mazzola & Gller, 2002). His most recent book (Mazzola, 2002) extends over more than 1300 pages and gives a supposedly comprehensive survey of the vast theoretical background as well as its computational implementation. A different model based mainly on mathematical consideration is the Mazzola model. The Mazzola model builds on mathematical music theory that not only covers various aspects of music theory and analysis through a highly complex mathematical approach, but also involves all sorts of philosophical, semiotic, and aesthetic considerations. The Mazzola model consists of an analysis part and a performance part. The analysis part involves computer-aided analysis tools for various aspects of the music structure, as meter, melody, or harmony. Each of these is implemented in RUBBETTEs that assign particular weights to each note in a symbolic score. The performance part transforms structural features into an artificial performance is theoretically anchored in Stemma Theory and Operator Theory. There is a linear mapping between metrical weight and tone intensity to generate artificial performances.The metrical, harmonic, and melodic weights as provided by the RUBATO software served as independent variables. The overall model could explain 84% of the average tempo curve of the 28 performances, each of the three analytical components contributing equally to the model.

3.3.1 The basic model

The Mazzola model builds on an enormous theoretical background, namely, the mathematical music theory (Mazzola, 1990) that not only covers various aspects of music theory and analysis through a highly complex mathematical approach, but involves all sorts of philosophical, semiotic, and aesthetic considerations as well. Every step of the theory is explained in specifically mathematical terms and with a special terminology that is greatly different from what is commonly used in performance research. The whole theory is rather hermetic, if one may say so. Therefore, we restrict ourselves here to the more concrete computational facets as they have been reported in the literature. The Mazzola model basically consists of an analysis part and a performance part. The analysis part involves computer-aided analysis tools for various aspects of the music structure as, e.g., meter, melody, or harmony. Each of these is implemented in particular plugins, the so-called RUBETTEs, that assign particular weights to each note in a symbolic score. The performance part that transforms structural features into an artificial performance is theoretically anchored in the so-called Stemma Theory and Operator Theory (a sort of additive rule-based structure-toperformance mapping). It iteratively modifies the performance vector fields, each of which controls a single expressive parameter of a synthesised performance.

Capitolo 4

The CaRo software music performance: what's new?

The improvement give to the original CaRo was to be able to give additional information to the Player, in addition to the MIDI file in input, in order to be able to give greater expressiveness executive to the audio file. The most immediate way to do this was to take the corrsipondent musical score to incoming MIDI file, add the information to the notes of the score, save this information in a file, and use this information in the appropriate time. With the software FINAL, the MIDI file was open, which the user added the additional information, in the next time exported the file into MusicXML format containing all the information of the original MIDI file as well as additional ones.

The project in the past was implemented using the compiler Borland Builder C + + and is based on library MidiShare for the communication and management of MIDI files. In the past to use the CaRo the user is forced to load a MIDI file and then optionally can upload the corresponding MusicXML file or not; with the work carried out in this thesis, this requirement has been removed. i.e to this work, these improvements have been made:

- The porting of the CaRo into cross platform who will be discuss in the next chapter.

- Loading only the MIDI file, in this case, the MIDI file may contain in place of a mechanical performance, a performance neutral recorded by a human pianist or obtained by an automatic system.

- Loading the only MusicXML files, in this case the notes and the melody of the song, which were previously taken from a MIDI file, are taken directly from the MusicXML files; Indeed, this file contains all the information necessary to perform the song.

4.1 The Integration into CaRo software of MusicXML library

4.1.1 What is MusicXML?

MusicXML was developed by Recordare LLC[1], deriving several key concepts from existing academic formats. MusicXML development is currently managed by MakeMusic following the company's acquisition of Recordare in 2011. Like all XML-based formats, MusicXML is intended to be easy for automated tools to parse and manipulate. Though it is possible to create MusicXML by hand, interactive score writing programs like Finale and MuseScore greatly simplify the reading, writing, and modifying of MusicXML files. MusicXML, as XML encoding, offers all the potential of this tool:

- structuring data
- modularity
- extensibility
- exchange data back-office
- ability to query and interaction through the family of technologies related to XML

MusicXML is a music interchange format designed for notation, analysis, retrieval, and performance applications. The MusicXML library is a portable `C++` library designed close to the MusicXML format and intended to facilitate MusicXML support. The MusicXML format represents common Western musical notation from the 17th century onwards. It is an xml format that organizes the music into a header followed by the core music data. The core music data may be organized as partwise or timewise data:

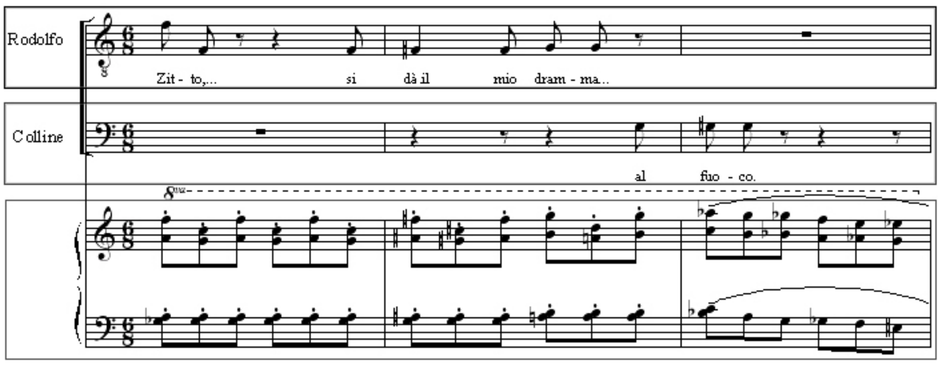
- Partwise data are organized into parts containing measures,
- Timewise data are organized into measures containing parts.

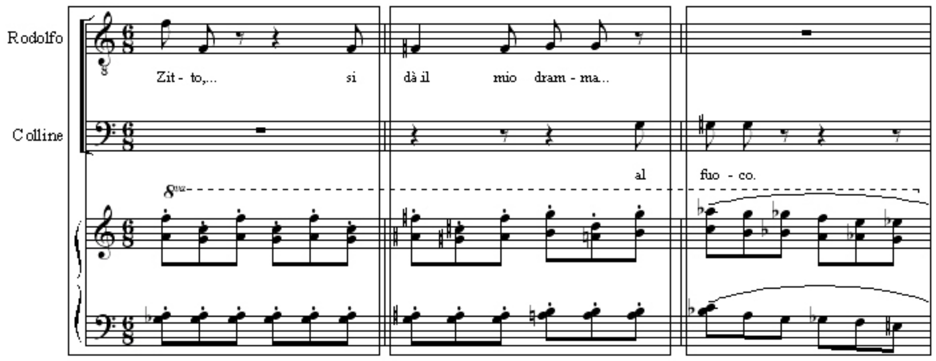
4.1.2 The MusicXML format

Writing Western music can be considered either horizontally or vertically, depending on whether the reference factor are the main parts or the measures

MusicXML has two different top-level DTDs, each with its own root element. If you use the partwise DTD, the root element is `<score-partwise>`. The musical part is primary, and measures are contained within each part. If you use the timewise DTD, the root element is `<score-timewise>`. The measure is primary, and musical parts are contained within each measure. An application reading MusicXML can choose which format is primary, and check for that document type. If it is your root element, just proceed. If not, check to see if it is the other MusicXML root element. If so, apply the appropriate XSLT stylesheet to create a new

[1]http://www.recordare.com/





MusicXML document in your preferred format, and then proceed. If it is neither of the two top-level document types, you do not have a MusicXML score, and can return an appropriate error message.

The primary definition of the file is contained in these lines:

```
<![ %partwise; [
<!ELEMENT score-partwise (%score-header;, part+)>
<!ELEMENT part (measure+)>
<!ELEMENT measure (%music-data;)>
]]>
<![ %timewise; [
<!ELEMENT score-timewise (%score-header;, measure+)>
<!ELEMENT measure (part+)>
<!ELEMENT part (%music-data;)>
]]>
```

- The %partwise; and %timewise; entities are set in the top-level DTDs partwise.dtd and timewise.dtd.

- The `<![` lines indicate a conditional clause like the #ifdef statement in C. So if partwise.dtd is used, the `<score-partwise>` element is defined, while if timewise.dtd is used, the `<score-timewise>` element is defined.

You can see that the only difference between the two formats is the way that the part and measure elements are arranged. A score-partwise document contains one or more part elements, and each part element contains one or more measure elements. The score-timewise document reverses this ordering. In either case, the lower-level elements are filled with a music-data entity. This contains the actual music in the score, and is defined as:

```
<!ENTITY % music-data
    "(note | backup | forward | direction | attributes |
      harmony | figured-bass | print | sound | barline |
      grouping | link | bookmark)*">
```

In addition, each MusicXML file contains a %score-header; entity, defined as:

```
<!ENTITY % score-header
    "(work?, movement-number?, movement-title?,
      identification?, defaults?, credit*, part-list)">
```

Sample of Score Header

```
 <work>
     <work-number>D. 911</work-number>
     <work-title>La bohme</work-title>
 </work>
 <movement-number>22</movement-number>
 <movement-title>Atto I, In soffitta. Parte di Rodolfo.</movement-title>
 <identification>
     <creator type="composer">Giacomo Puccini</creator>
     <creator type="poet">Giuseppe Giacosa</creator>
     <creator type="poet">Luigi Illica</creator>
     <encoding>
         <software>Finale 2005 for Windows</software>
         <software>Dolet for Finale 1.4.1</software>
         <encoding-date>2006-05-04</encoding-date>
     </encoding>
     <source>Based on digital edition from CD Sheet music LLC
     </source>
  </identification>
  <part-list>
    <score-part id="P1">
       <part-name>Rodolfo</part-name>
    </score-part>
    <score-part id="P2">
        <part-name>Pianoforte.</part-name>
   </score-part>
</part-list>
```

As You can see the score-header above has all five of the possible top-level elements in the score-header entity: the work, movement-number, movement-title, identification, and part-list. Only the part-list is required, all other elements are optional. Lets look at each part in turn: In MusicXML, individual movements are usually represented as separate files.

The work The work element is used to identify the larger work of which the movement is a part. The worker's works are more commonly referred to via D. numbers than opus numbers, so that is what we use in the work-number element; the work-title is the name of the larger work.

The movement If you have all the movements in a work represented, you can use the opus element to link to the MusicXML opus file that in turn contains links to all the movements in the work. Similarly, we use the movement-title element for the title of the individual song. If you have a single song that is not part of a collection, you will usually put the title of the song in the movement-title element, and not use either the work or movement-number elements.

The identification It contains basic metadata elements based on the Dublin Core. In this song, as many others, there are two creators: in this case, the composer and the poet. Therefore, we use two creator elements, and distinguish their roles with the type attribute. For an instrumental work with just one composer, there is no need to use the type attribute.

The encoding The encoding element contains information about how the MusicXML file was created. Here we are using all four of the available sub-elements to describe the encoding. You can have multiple instances of these elements, and they can appear in any order.

The part-list The part-list is the one part of the score header that is required in all MusicXML scores. It is made up of a series of score-part elements, each with a required id attribute and part-name element. By convention, our software simply numbers the parts as P1, P2, etc. to create the id attributes. You may use whatever technique you like as long as it produces unique names for each score-part. In addition to the part-name, there are many optional elements that can be included in a score-part. For more information, see at the MusicXML web site [2]

[2]http://www.makemusic.com/musicxml/

4.1.3 How to install and use the library MusicXML

The first step is to download the library in the repository web site[3] or direct link [4]. The second step according to the OS is

- **MacOS:**
  copy the libmusicxml2.framework to /Library/Frameworks
- **Windows:**
  copy the libmusicxml2.dll along with your application; For example, according to the IDE use in this thesis [5], in the directory that contain the libmusicxml.dll, there is the libmusicxml.lib to link inside the build properties of the project.
- **Linux:**
  copy the library to /usr/local/lib or to /usr/lib binaries are compiled for 64 bits architectures on Ubuntu 11.10 you should get the src code (from the package of from git) and compile on your platform

4.2 Parsing of MusicXML file

The set of five horizontal lines and four spaces that each represent a different musical pitch is extract with the code below:

```
 cout << "summary for part "<< elt->getAttributeValue("id")<< "\n";
 cout << " staves count : "<< countStaves() << "\n";

smartlist<int>::ptr voices;
smartlist<int>::ptr staves = getStaves();
for (vector<int>::const_iterator i = staves->begin(); i != staves->end();
 cout << "  staff \"" << *i << "\": " << getStaffNotes(*i) << " notes - "
<< countVoices(*i) << " voices [";

 bool sep = false;
 voices = getVoices(*i);
for (vector<int>::const_iterator v=voices->begin(); v!=voices->end(); v++
if (sep) resStream << ", ";
else sep = true;
cout << *v << ":" << getVoiceNotes(*i, *v);
}
cout << "]" << "\n";
}
```

[3]http://libmusicxml.sourceforge.net/
[4]https://code.google.com/p/libmusicxml/downloads/list
[5]Code::Blocks 13.12

```
cout << " total voices : "<< countVoices()<< "\n";
voices = getVoices();
for (vector<int>::const_iterator i=voices->begin(); i!=voices->end(); i++
{
 cout << " voice \"" << *i << "\": " <<getVoiceNotes(*i)<< " notes - ";
 staves = getStaves(*i);
 cout << staves->size() <<(staves->size() > 1 ? " staves" :" staff") << "
 bool sep = false;
for (vector<int>::const_iterator s=staves->begin(); s!=staves->end(); s++
{
if (sep) resStream << ", ";
else sep = true;
cout << *s ;
}
cout << "] main staff: " << getMainStaff(*i) << "\n";
}
}
```

Then, by taken an iterator on the set of Staves, the algorithm print the staff notes, and count all voice of the staff. After this, it show an open dialog containing all the Stave, voices, and all notes of the file MusicXML. Also, a file .txt is created cointaining all informations mentioned above.

4.3 MusicXML and MIDI

The purpose and actual use of MusicXML is the interchange of musical scores on the Internet. Over the past thirty years there have been numerous systems of representation of Western notation, but the only used on a large scale is the MIDI: Musical Instrumental Digital Interface. While the MIDI format was created as a support to musical performances, MusicXML is proposed as a standard encoding of the musical score in all its facets. Once encoded, the score can be considered as a semi-structured database and then questioned and revised. This type of approach has also been suggested, during a series of conferences organized to present the project, by Michael Good, founder of Recordare LLC. MidiShare is based on a client/server model. It is composed of six main components : an event memory manager, a time manager and synchronizer, a task manager, a communication manager, an event and task schedulerand Midi drivers.

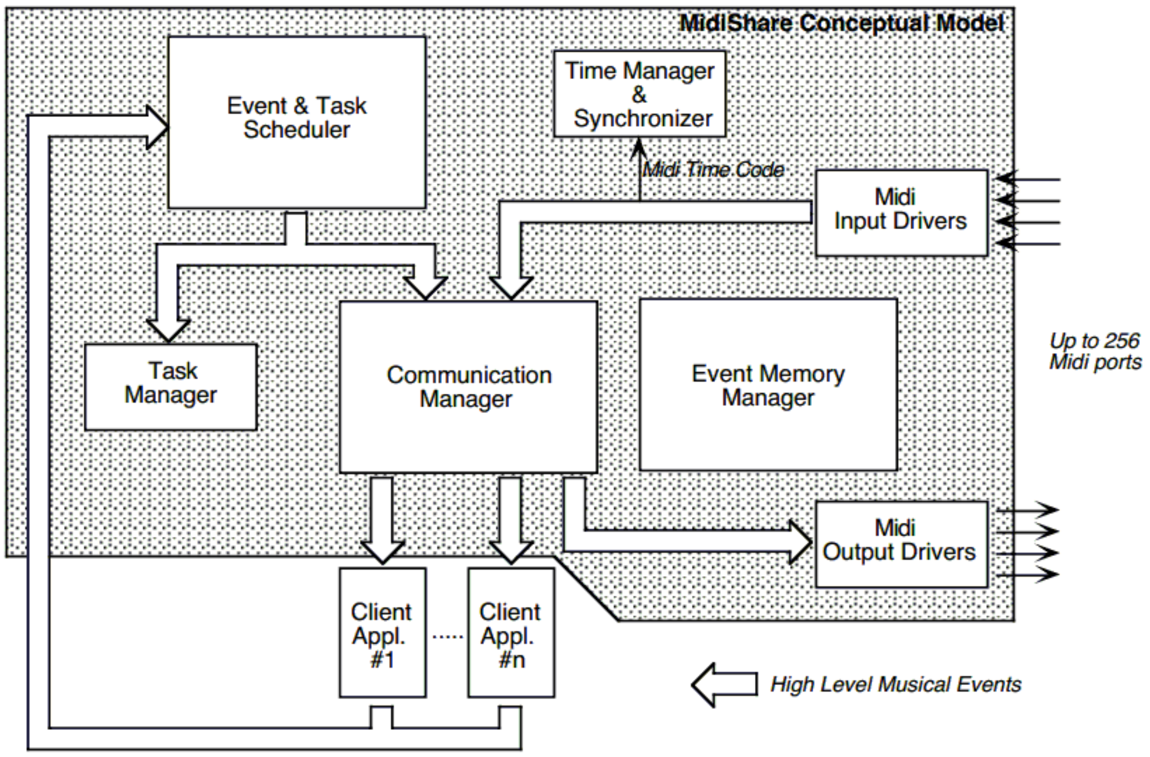


Figura 4.1: *Conceptual model of MidiShare(MIDI)*

4.4 Extracting MIDI's informations from MusicXML file

The below code is what use to extract required informations,

```
class mymidiwriter : public midiwriter {
public:
mymidiwriter() {}
virtual ˜mymidiwriter() {}

virtual void startPart (int instrCount)
{ resStream << "startPart with " << instrCount << " instrument(s)" << "\n
virtual void newInstrument (std::string instrName, int chan=-1)
{ resStream << "newInstrument \"" << instrName << "\" on chan " << chan <
virtual void endPart (long date)
{ resStream << date << " endPart" << "\n";}

virtual void newNote (long date, int chan, float pitch, int vel, int dur)
{ resStream << date << " newNote [" << chan << "," << pitch << ","
```

```
  << vel << "," << dur << "]" << "\n";
 midiAttribute << chan << "," << pitch << "," << vel << "," << dur << "\n
virtual void tempoChange (long date, int bpm)
{ resStream << date << " tempoChange " << bpm << "\n";}
virtual void pedalChange (long date, pedalType t, int value)
{ resStream << date << " pedalChange type " << t << " val " << value << "

  virtual void volChange (long date, int chan, int vol)
{ resStream << date << " volChange chan  " << chan << " vol " << vol << "
virtual void bankChange (long date, int chan, int bank)
{ resStream << date << " bankChange chan " << chan << " bank " << bank <<
  virtual void progChange (long date, int chan, int prog)
{ resStream << date << " progChange chan " << chan << " prog " << prog <<
};
```

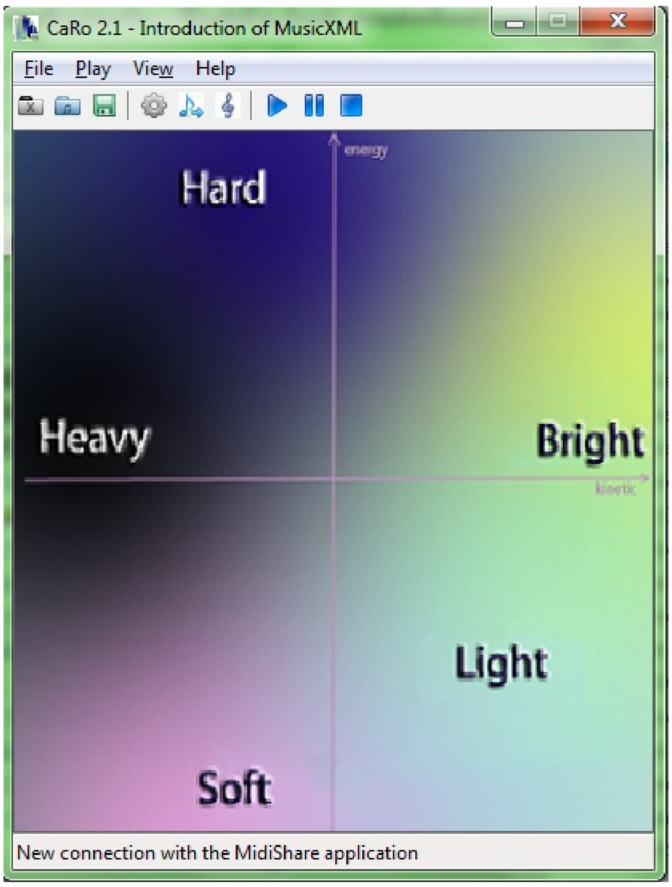
The above code is more intuitive; inside the tree node, the source is looking about the notes with main parameters:

```
<note>
        <rest measure="yes"/>
        <duration>24</duration>
        <voice>1</voice>
    </note>
         ...
    <note>
        <rest measure="yes"/>
        <duration>16</duration>
        <voice>1</voice>
    </note>
        ...
    <note default-x="59">
        <pitch>
           <step>B</step>
           <octave>5</octave>
        </pitch>
        <duration>24</duration>
        <tie type="start"/>
        <voice>1</voice>
        <type>half</type>
        <dot/>
        <stem default-y="-22">down</stem>
        <notations>
         <tied type="start"/>
        </notations>
    </note>
        ...
```

All estractions are store in a .txt file, and in the second by perform play mecchanical, the MidiShare perform the Player song, it will be explain in section 5.7

4.5 CaRo software: overview of new Features

When the application starts, the program looks like in the figure 4.2, which will be the interface of interaction between the user and the program. All operations that you can perform with the



New connection with the MidiShare application

Figura 4.2: *New version of CaRo 2.1*

CaRo 2.1 can be divided into 4 main actions:

1. Mecchanical Execution of a MIDI file (MIDI)

2. Parsing of a MusicXML file

3. Extraction principal MIDI's informations from MusicXML file
4. Mecchanical Execution of MIDI's informations extracting from MusicXML file

Will be analyzed in detail the operation and implementation behind each of these actions.

4.5.1 The main interface

For an application to be able to transmit and receive events, it must first be connected to one or more source and destination. The CaRo 2.1 firstly check if the MidiShare application is installed in our OS. Then open the connection with the MidiShare Player and store it to a reference number who will use to managed the Player, the MidiShare sequences are load on the Player by setting all the track. The description below is more intuitive:

```
if (MidiShare()) {
  "MidiShare application is installed...";
} else {
  "MidiShare application not installed...";
}
//Store refrence number of OpenPlayer
ourRefNum = OpenPlayer(AppliName);
//Connected the Player to Midishared
MidiConnect(ourRefNum, 0, true);

if (!MidiIsConnected(ourRefNum, 0)) {
  "CaRo has not been connected to MidiShare application";
} else {
  "New connection with the MidiShare application";
}

ourRefNum = MidiOpen(AppliName);
if ( ourRefNum == MIDIerrSpace ){
  "Too much MidiShare client applications";
}
```

The MidiConnect function allows the switching on or off connections between a source and destination applications and the MidiIsConnected function gives the state (on or off) of a connection. There are no restrictions in establishing connections, an application can be source or destination of as many applications as you wish and of course looping is possible. The toolbar in the figure 4.2 is added to perform quickly some operations; on opening the CaRo 2.1 main interface, some buttons on the toolbar are disabled like:

Figura 4.3: *Save the MusicXML file or MIDI file., which is action performed by saving the corripondent MIDI file or MusicXML file*

Figura 4.4: *Parse the MusicXML file Involved the libmusicxml libraries., which is action performed by reading the corripondent xml file and browse information needing by MidiShare libraries*

Figura 4.5: *Extracted the main information from MusicXML file Involved the libmusicxml libraries., which is extractiong the principal information to send to MidiShare Player.*

Figura 4.6: *Play the current song MidiShare Player is Play , which is the corrispondent executive player, maybe the Mecchanical, Neutral or Expressiveness play*

Figura 4.7: *Pause the current song MidiShare Player is Pause , which is the corrispondent executive player, maybe the Mecchanical, Neutral or Expressiveness play*

Figura 4.8: *Stop the current song MidiShare Player is Stop , which is the corrispondent executive player, maybe the Mecchanical, Neutral or Expressiveness play*

As initial toolbar, it look like:

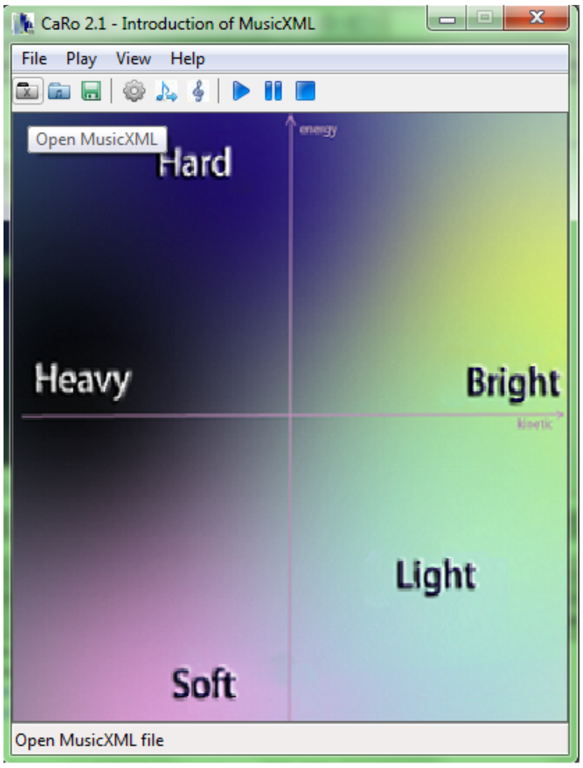


Figura 4.9: *Opening the MusicXML file by clicking button on toolbar, this operation perform the count of all notes in the file: it is like a small parse*

The possibility of open file is also given by the menu `File --> Open MusicXML`: as initial open menu file, this operation are disabled:

- Save
- Parse MusicXML
- Xml2Midi

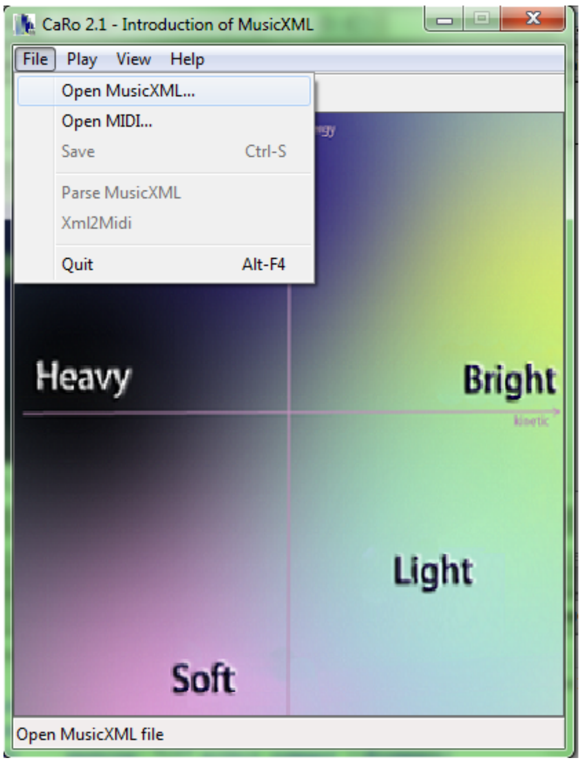
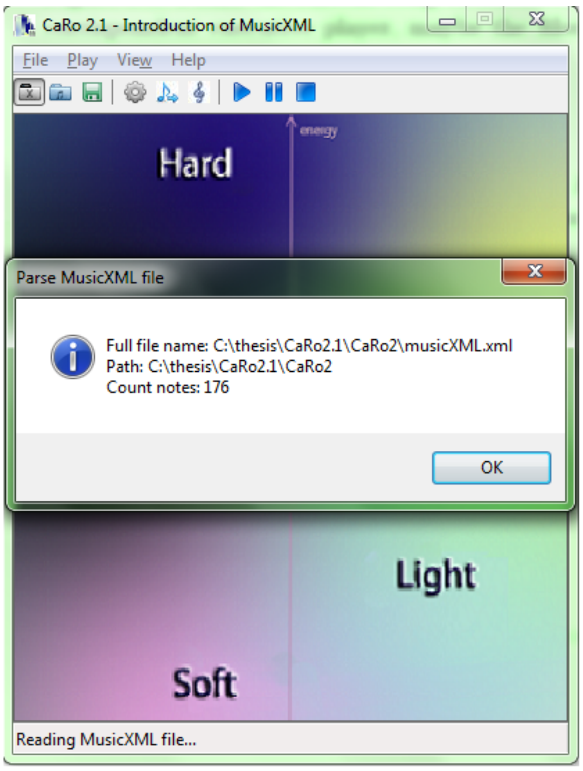


Figura 4.10: *Opening the MusicXML file by opening menu file, this operation perform the count of all notes in the file: it is like a small parse*

When the file MusicXML is correctly open, a message dialog appear and display the number of notes containing in the file:



Full file name: C:\thesis\CaRo2.1\CaRo2\musicXML.xml
Path: C:\thesis\CaRo2.1\CaRo2
Count notes: 176

Reading MusicXML file...

Figura 4.11: *Message dialog the MusicXML file readed and the number of count notes. After this dialog, the status bar of the main frame interface tell the correctly or not of the musicXML file reading*

The operation of opening MusicXML file who return the number of notes of the MusicXML file use a predicate to search inside the node of the MusicXML file all elements which type is `k_note` as the sample below:

```
struct predicate {
bool operator () (const Sxmlelement elt) const {
return elt->getType() == k_note;
}
};
```

During the perform so-called operation of `Open MusicXML` file, all the elements is browsed and stored in a vector, by using and iterator on the vector, and the predicate like the example below:

```
Sxmlelement elt = file->elements();
predicate p;
count = count_if(elt->begin(), elt->end(), p);
```

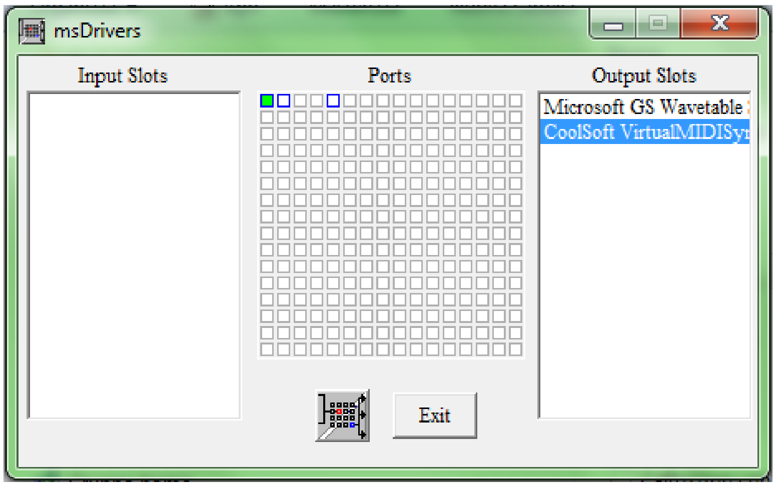
If the file is opened correctly, this two buttons below are enabled allowing the relative's operations:

Figura 4.12: *Save the MusicXML file or MIDI file., which is action performed by saving the corripondent MIDI file or MusicXML file*

Figura 4.13: *Parse the MusicXML file Involved the libmusicxml libraries., which is action performed by reading the corripondent xml file and browse information needing by MidiShare libraries*

4.5.2 Drivers connection between the MidiShare and the Player

In the directory of the project, this application dirver must be present: `msControl32`, `msDrives`, `msEcho32` and this configuration files `midishare.ini`, `msMMSystem.ini`. These files are located in directory libraries and used to set the MidiShare connections port for input/ output of MIDI files. Each dirver has slots for input/output, which correspond to the input ports and output that is used. To properly set the instructions contained within the file `.INI` just use the application `msDrivers.exe`, a driver manager that allows you to set the connections between MidiShare ports and driver's slots.



Microsoft GS Wavetable
CoolSoft VirtualMIDISyn

Figura 4.14: *Driver managers: application to set connection port between MidiShare and driver slot*

4.5.3 The Options frame

This frame is the same describe on the thesis of Massimo Barichello[6] The only improvement is reading from multiple text files and save the new informations at the relative text file. At

[6]http://tesi.cab.unipd.it/35067/1/tesiBarichello.pdf

the startup of the options frame, the information is loading from two txt file to fill the MIDI parameters and the expressive MusicXML parameters.

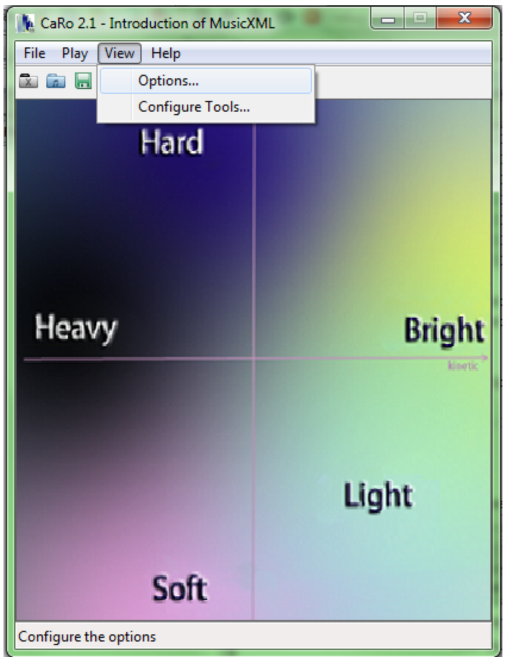
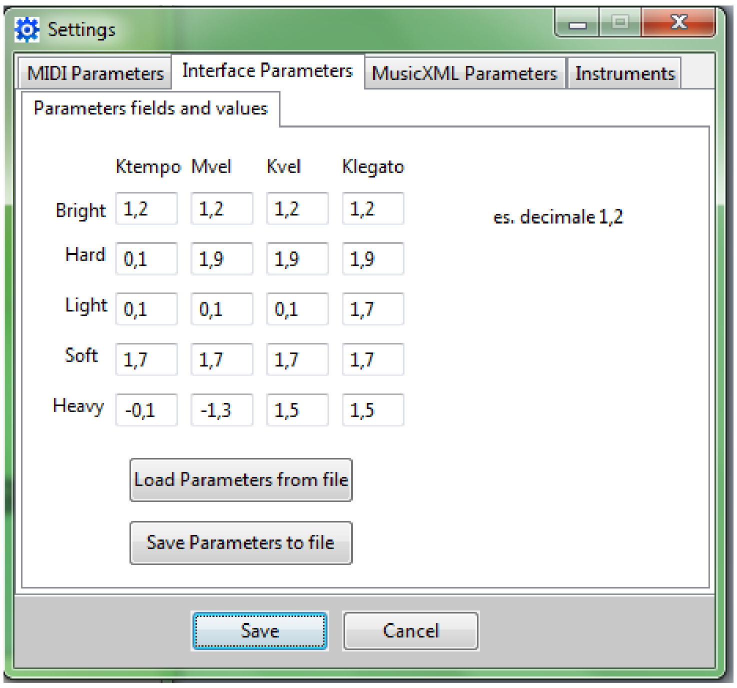


Figura 4.15: *The options frames to load parameters of the MusicXML file. The user can load prameters from any file, and save it to AdjectiveParameters.txt or MusicXMLparameters.txt*

This tab in figure **??** allow to CaRo the realtime interaction with the interface when the song is riproduced.


Settings

MIDI Parameters | Interface Parameters | MusicXML Parameters | Instruments

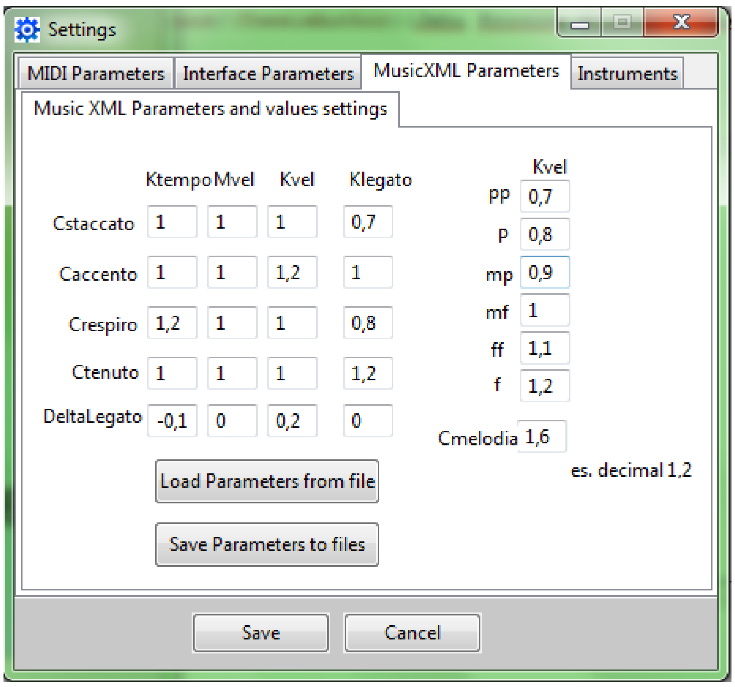
Parameters fields and values

	Ktempo	Mvel	Kvel	Klegato
Bright	1,2	1,2	1,2	1,2
Hard	0,1	1,9	1,9	1,9
Light	0,1	0,1	0,1	1,7
Soft	1,7	1,7	1,7	1,7
Heavy	-0,1	-1,3	1,5	1,5

es. decimale 1,2

Figura 4.16: *The options frames to load parameters of the MIDI file, MidiShare. The user can load parameters from any file, and save it to AdjectiveParameters.txt*

In this tab, we find the multiplicative constants related to any kind of information which is obtain through the MusicXML file, in fact any of this information goes to act in some of the parameters of the model used by the CaRo, which enables the expressiveness modification of the MIDI song.



Music XML Parameters and values settings

	Ktempo	Mvel	Kvel	Klegato
Cstaccato	1	1	1	0,7
Caccento	1	1	1,2	1
Crespiro	1,2	1	1	0,8
Ctenuto	1	1	1	1,2
DeltaLegato	-0,1	0	0,2	0

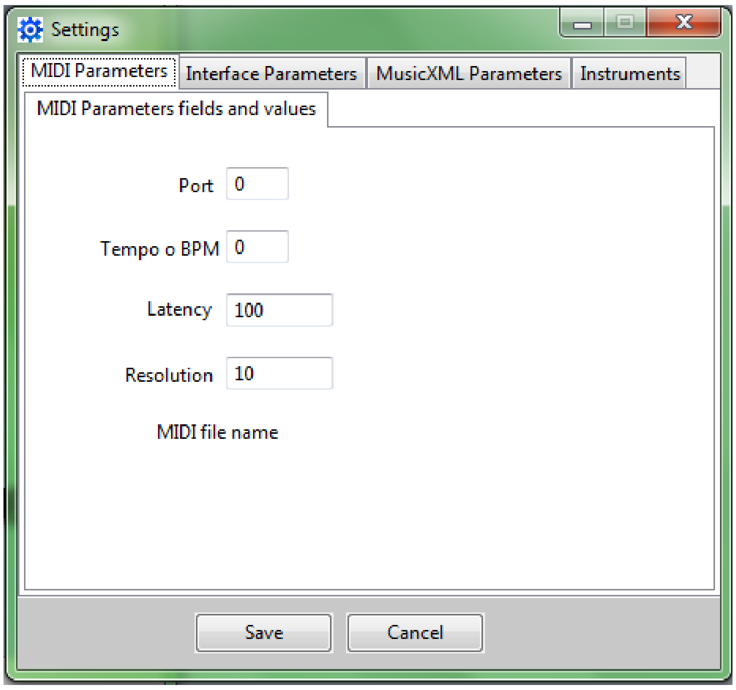
	Kvel
pp	0,7
p	0,8
mp	0,9
mf	1
ff	1,1
f	1,2

Cmelodia 1,6

es. decimal 1,2

Figura 4.17: *The options frames to load parameters of the MusicXML file, MidiShare. The user can load parameters from any file, and save it to MusicXMLparameters.txt*

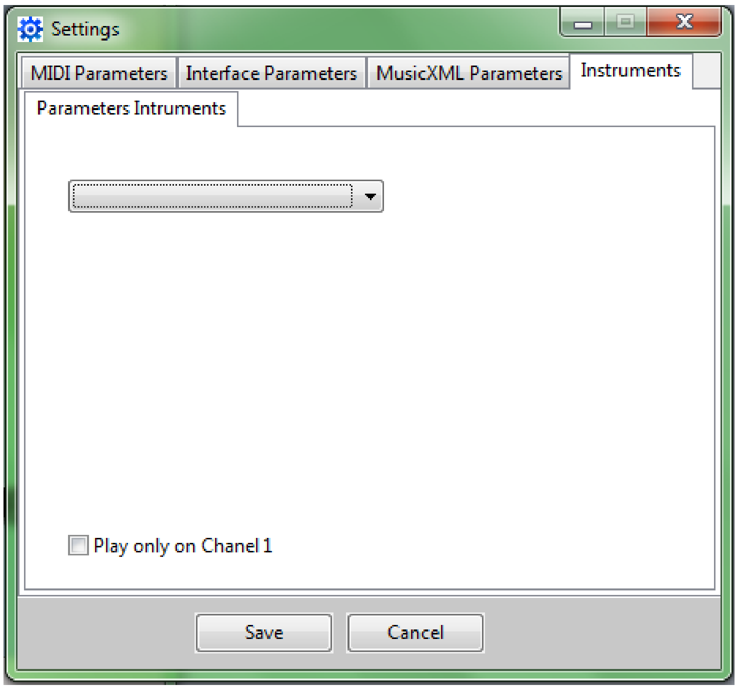
This tab contains four text boxes that identify the connection port with the MidiShare, the value of time(o Beat per Minute o Quarter note per Minute), the latency (parameter in millisecond used to move the time instants of events), and the resolution (value in millisecond of a timer that is used during the execution of a song). When loading the MIDI file, the time is instead set by default to 120bpm.



Port 0
Tempo o BPM 0
Latency 100
Resolution 10
MIDI file name

Figura 4.18: *The options frames to set MIDI parameters, MidiShare. The user can set parameters directly.*

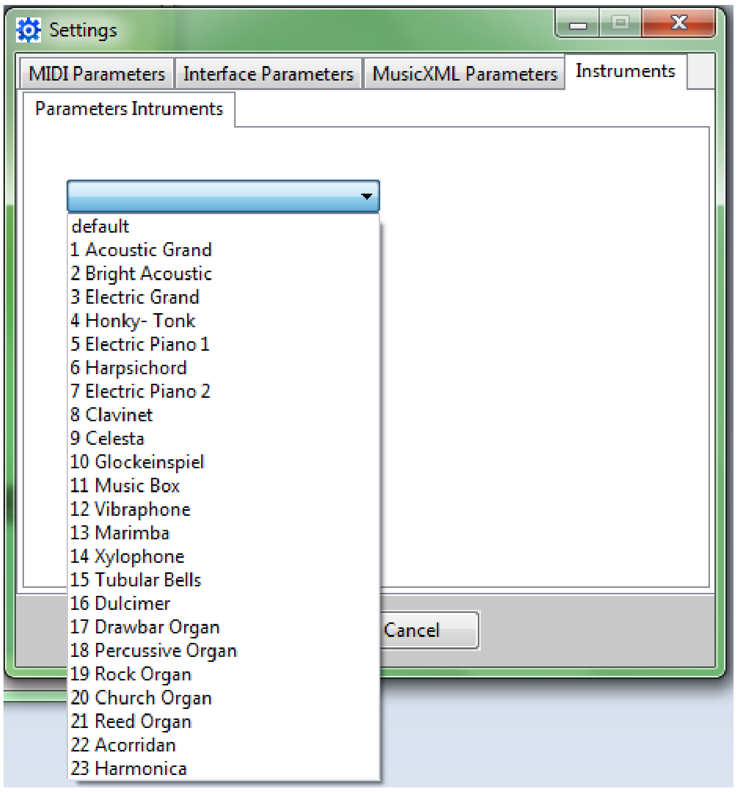
In this tab, you can select the instrument of execution of the track (the CaRo not currently charge any modification about), and click through a check button indicated with the play only on channel if the player must perform all the notes of the MIDI files with the piano.



Play only on Chanel 1 [ ]

Figura 4.19: *The options frames to set MIDI parameters, MidiShare. The user can set parameters directly.*

The Player has this setting enabled by default, so it will be added an event that changes the timbre. At the startup of the CaRo, this check in not enabled.



default
1 Acoustic Grand
2 Bright Acoustic
3 Electric Grand
4 Honky- Tonk
5 Electric Piano 1
6 Harpsichord
7 Electric Piano 2
8 Clavinet
9 Celesta
10 Glockeinspiel
11 Music Box
12 Vibraphone
13 Marimba
14 Xylophone
15 Tubular Bells
16 Dulcimer
17 Drawbar Organ
18 Percussive Organ
19 Rock Organ
20 Church Organ
21 Reed Organ
22 Acorridan
23 Harmonica

Figura 4.20: *The options frames to set MIDI parameters, MidiShare. The user can set parameters directly.*

4.5.4 The Tools frame

This frame allow to change the static image of the main interface:

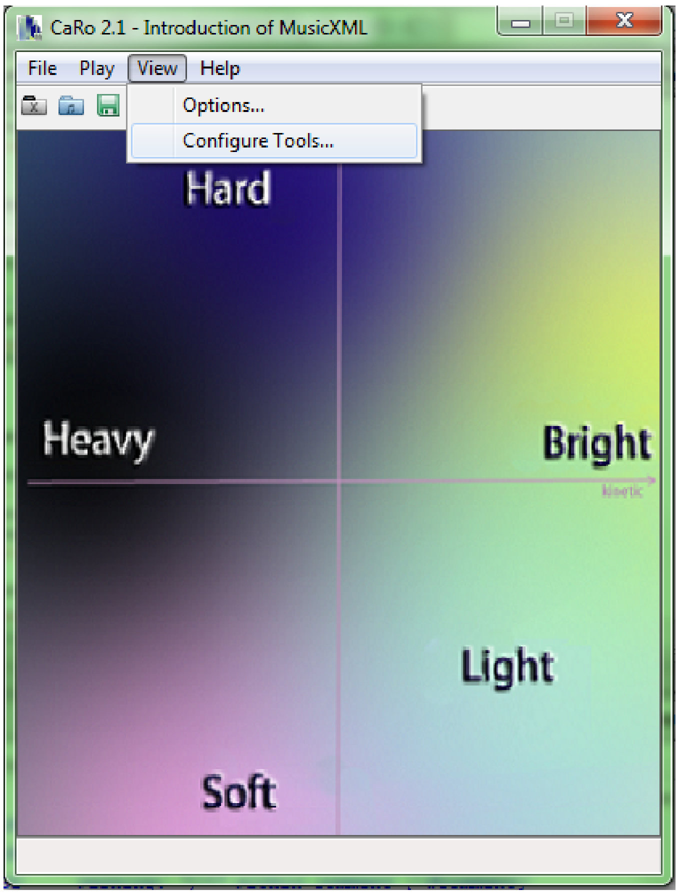


Figura 4.21: *Configure tools Involved the Change of the static image., which action performed by changing the background image of the main frame of CaRo*

When the action click is perform, a new frame is like this below is opened:

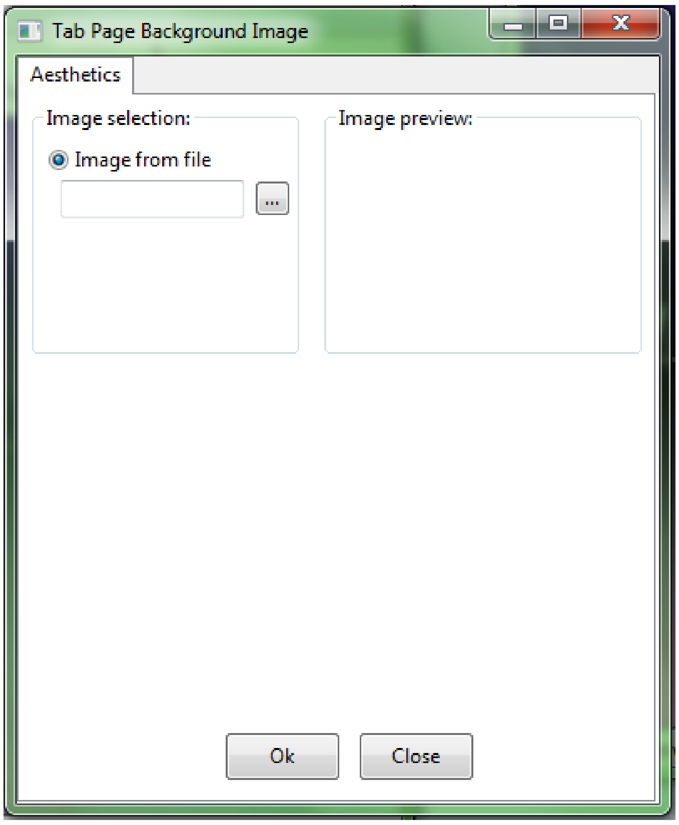


Figura 4.22: *Window who allow the choose of bacground image Choose the new static image., which action performed by choosing the new background image of the main frame of CaRo*

After choose the new image the windows mentioned above look like:

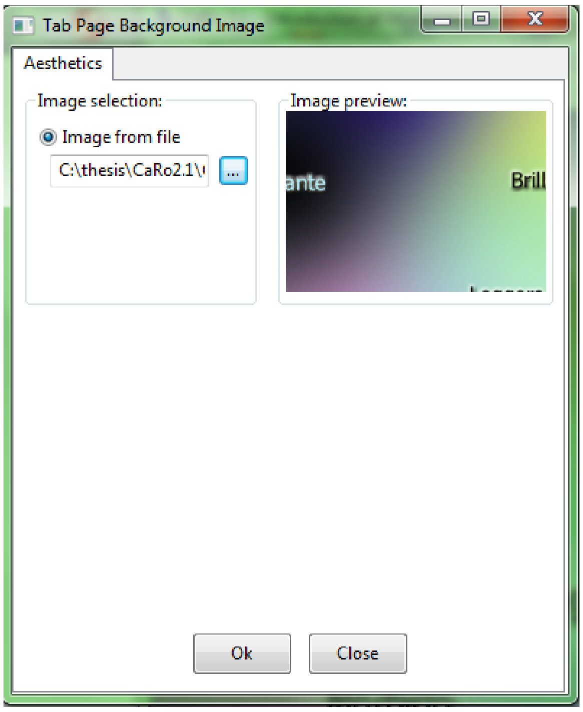
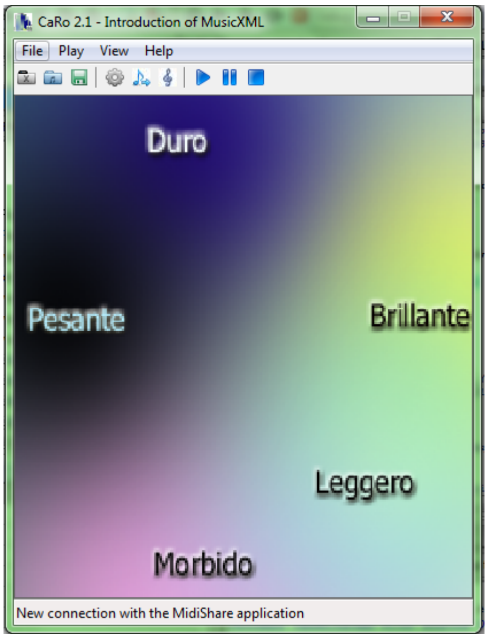


Figura 4.23: *Window with the choosing background image The new static image is choosed., which action is performed by showing the new background image choosed of the main frame of CaRo*

By perform ok button to choose image, the main frame of CaRo became like this:



New connection with the MidiShare application

Figura 4.24: *Main Window of CaRo with the new background image The new static image., which action is performed by showing the new background image of the main frame of CaRo*

4.6 Mecchanical Execution of a MIDI file: Loading and Playing

Connect the application to MidiShare physical I/Os the 3 arguments of MidiConnect are the reference number of the source, the reference number of the destination and the state of the connection (1=connected, 0=not connected). Each Player is a MidiShare application, has 256 tracks, four different synchronization mode, events chase and loop markers. A Player use MidiShare sequences. Functions are provided to read on write MIDIfiles and convert them to the MidiShare sequence format. MidiShare sequences will be loaded into a Player using the SetAllTrackPlayer function. A MidiShare sequence contains usually a tempo-map track which will be used instead of the default tempo and time signature values. After it's creation, the Player should be connected to the MIDI input/output (that is to the MidiShare application), but because the Player is manipulated using it's MidiShare reference number, it's possible to use some MidiShare functions to manipulate it. Two functions, MidifileLoad and MidifileSave allow to read and write MIDIfiles and convert them to MidiShare sequence format. When loading a multi-tracks MIDIfile, the different tracks of the MIDIfile will correspond to different tracks of the MidiShare sequence.

```
short myRefNum;
MidiseqPtr mySeq;
MidiFileInfos myInfo;
myRefNum = OpenPlayer("CaRo2Main");
MidiConnect(myRefNum,0,1);
mySeq = MidiNewSeq();
MidiFileLoad( "HD500:midifile1", mySeq, &myInfo);
SetAllTrackPlayer (myRefnum, mySeq, info.clicks);
printf("(type <ENTER> to start playing )\n");
getc(stdin);
StartPlayer(myRefNum );
printf("(type <ENTER> to stop playing )\n");
getc(stdin);
StopPlayer(myRefNum );
ClosePlayer(myRefNum);
```

Load a MIDIfile and convert it into a MidiShare sequence. When loading a multi-tracks MIDIFile (format 1 or 2) the different tracks of the MIDIfile are distinguish by the reference number which are on them. It means that all events of track 0 have a reference number of 0, all events on track 1 have a reference number of 1 and so on.

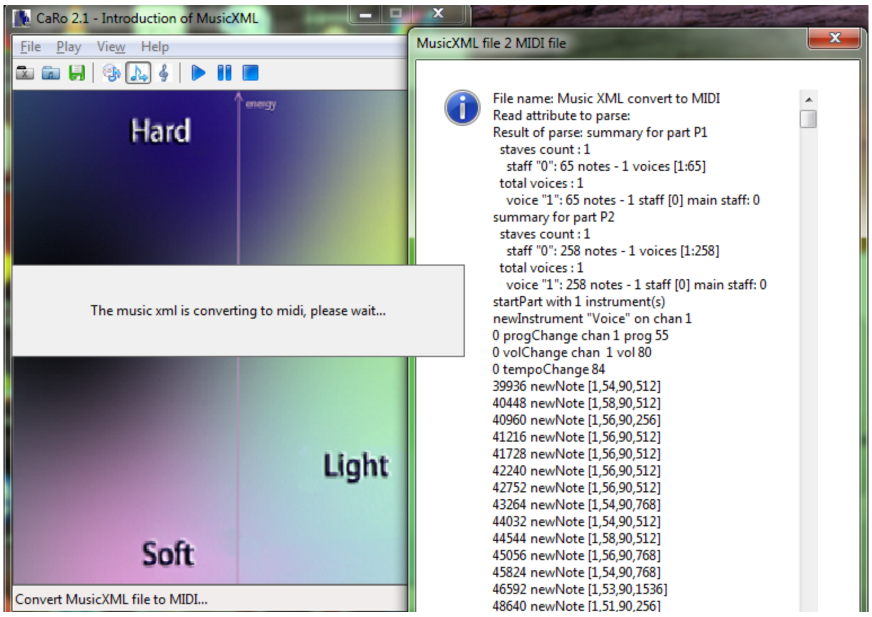
```
METHOD
long MidiFileLoad (char * filename, MidiSeqPtr s, MidiFileInfosPtr info);
ARGUMENTS
filename : a C string, the filename of the MIDIFile.
```

```
       s : a MidiSeqPtr : the sequence to be loaded.
    info : a MidiFileInfosPtr: used to record
           some information about the MIDIfile.
RESULT
The result is a MIDIfile error code.
```

4.7 Mecchanical Execution of MIDI's informations extracting from MusicXML file

```
while ( getline (infileIn, line) )
{
   std::string attr_numbers_str = line;
   //we'll put all of the tokens in here
   std::vector <std::string> numbers;
  while (line.find(",", 0) != std::string::npos)
  { //does the string a comma in it?
    //store the position of the delimiter
    size_t  pos = line.find(",", 0);
    //get the token
    std::string temp = line.substr(0, pos);
    //erase it from the source
    line.erase(0, pos + 1);
    //and put it into the array
    numbers.push_back(temp);
  }
  //the last token is all alone
  numbers.push_back(line);

  std::string pitch = numbers.at(1);
  char *cpitch = new char[pitch.length() + 1];
  strcpy(cpitch, pitch.c_str());

  std::string vel = numbers.at(2);
  char *cvel = new char[vel.length() + 1];
  strcpy(cvel, vel.c_str());

  std::string dur = numbers.at(3);
  char *cdur = new char[dur.length() + 1];
  strcpy(cdur, dur.c_str());
  std::string chan = numbers.at(0);
  char *cchan = new char[chan.length() + 1];
  strcpy(cchan, chan.c_str());

  midi << "-pitch <" << cpitch << ">" << " -vel <" << cvel << ">"
  << " -dur <" << cdur << ">" << " -chan <" << cchan << ">" << '\n';

 if (e = MidiNewEv(typeNote)) { // allocate a note event
    Pitch(e) = atol(cpitch);    // fill its fields
    Vel(e) = atol(cvel);
    Dur(e) = atol(cdur);
    Chan(e) = atol(cchan);
    Port(e) = 0;
    seq = MidiNewSeq();
    MidiSendIm(ourRefNum, MidiCopyEv(e));
    // add to the sequence all the received events
    MidiAddSeq(seq, e);
    // wait the duration of the note
    wait(atol(cdur));
 }
playSeq(ourRefNum, atol(cpitch), atol(cvel),
                  atol(cdur), atol(cchan), 0, 1024);

}
```

After playing the song, All the main information are store in a .txt file


CaRo 2.1 - Introduction of MusicXML
File Play View Help
Hard
energy
The music xml is converting to midi, please wait...
Light
Soft
Convert MusicXML file to MIDI...
MusicXML file 2 MIDI file

```
File name: Music XML convert to MIDI
Read attribute to parse:
Result of parse: summary for part P1
  staves count : 1
    staff "0": 65 notes - 1 voices [1:65]
  total voices : 1
    voice "1": 65 notes - 1 staff [0] main staff: 0
summary for part P2
  staves count : 1
    staff "0": 258 notes - 1 voices [1:258]
  total voices : 1
    voice "1": 258 notes - 1 staff [0] main staff: 0
startPart with 1 instrument(s)
newInstrument "Voice" on chan 1
0 progChange chan 1 prog 55
0 volChange chan  1 vol 80
0 tempoChange 84
39936 newNote [1,54,90,512]
40448 newNote [1,58,90,512]
40960 newNote [1,56,90,256]
41216 newNote [1,56,90,512]
41728 newNote [1,56,90,512]
42240 newNote [1,56,90,512]
42752 newNote [1,56,90,512]
43264 newNote [1,54,90,768]
44032 newNote [1,54,90,512]
44544 newNote [1,58,90,512]
45056 newNote [1,56,90,768]
45824 newNote [1,54,90,768]
46592 newNote [1,53,90,1536]
48640 newNote [1,51,90,256]
```

Figura 4.25: *Message dialog for extractiong MidiShare main information.*

```
[How is the playback using MidiShare and Player?]
```

Pitch	Velocity	Duration	Chanel
90	54	512	1
90	58	512	1
90	56	256	1
90	56	512	1

```
MidiEvPtr e; // open a MidiShare session
ourRefNum = MidiOpen("CaRo2.1"); //connect to physical Midi outputs
MidiConnect(ourRefNum, 0, true);
 if (e = MidiNewEv(typeNote)) {  //allocate a note event
     Pitch(e) = atoi(Pitch);      //fill its fields
     Vel(e)  = atoi(Velocity);
     Dur(e)  = atoi(Duration);
     Chan(e)  = atoi(Chanel);
     Port(e)  = 0;
     gSequence = MidiNewSeq();
     MidiSendIm(ourRefNum, MidiCopyEv(e));
```

```
    MidiAddSeq(gSequence, e);   //add to the sequence all the received e
    wait(atoi(Duration));       //wait the duration of the note
}
playSeq(ourRefNum, atoi(Pitch), atoi(Velocity),
        atoi(Duration), atoi(Chanel), 0, 10);
```

Capitolo 5

The Porting of CaRo in cross platform environment

5.1 The Choice: wxWidget

wxWidgets currently supports the following primary platforms:

- Windows 95/98/ME, NT, 2000, XP, Vista, 7(32/64 bits).
- Most Unix variants using the GTK+ toolkit (version 2.6 or newer)
- Mac OS X (10.5 or newer) using Cocoa (32/64 bits) or Carbon (32 only)

There is some support for the following platforms:

- Most Unix variants with X11
- Most Unix variants with Motif/Lesstif
- Most Unix variants with GTK+ 1.2
- OS/2
- Windows CE (Pocket PC)

Most popular `C++` compilers are supported;

wxWidgets[1] is a programmers toolkit for writing desktop or mobile applications with graphical user interfaces (GUIs). Its a framework, in the sense that it does a lot of the housekeeping work and provides default application behavior. The wxWidgets library contains a large number of classes and methods for the programmer to use and customize. Applications typically show windows containing standard controls, possibly drawing specialized images and graphics and responding to input from the mouse, keyboard, or other sources. They may also communicate

[1]http://www.wxwidgets.org: the wxWidgets home page

with other processes or drive other programs. In other words, wxWidgets makes it relatively easy for the programmer to write an application that does all the usual things modern applications do. While wxWidgets is often labeled a GUI development toolkit, it is in fact much more than that and has features that are useful for many aspects of application development. This has to be the case because all of a wxWidgets application needs to be portable to different platforms, not just the GUI part.

5.2 Why use wxWidgets?

One area where wxWidgets differs from many other frameworks, such as MFC or OWL, is its *multi-platform* nature. wxWidgets has an Application Programming Interface (API) that is the same, or very nearly the same, on all supported platforms. This means that you can write an application on Windows, for example, and with very few changes (if any) recompile it on Linux or Mac OS X. This has a huge cost benefit compared with completely rewriting an application for each platform, and it also means that you do not need to learn a different API for each platform. Furthermore, it helps to future-proof your applications. As the computing landscape changes, wxWidgets changes with it, allowing your application to be ported to the latest and greatest systems supporting the newest features. Another distinguishing feature is that wxWidgets provides a native look and feel. Some frameworks use the same widget code running on all platforms, perhaps with a theme makeover to simulate each platforms native appearance. By contrast, wxWidgets uses the native widgets wherever possible (and its own widget set in other cases) so that not only does the application look native on the major platforms, but it actually is native. This is incredibly important for user acceptance because even small, almost imperceptible differences in the way an application behaves, compared with the platform standard, can create an alienating experience for the user.[2]

5.3 Installing wxWidgets for Windows

This is wxWidgets for Microsoft Windows 9x/ME, Windows NT and later (2000, XP, Vista, 7, 8, etc) including both 32 bit and 64 bit versions.

5.3.1 Installation

If you are using one of the supported compilers, you can download the pre-built in binaries from

- https://sourceforge.net/projects/wxwindows/files/3.0.0/binaries/
- ftp://ftp.wxwidgets.org/pub/3.0.0/binaries/

In this case, just uncompress the binaries archive under any directory and skip to Building Applications Using wxWidgets part.

[2]http://wiki.wxwidgets.org: the wxWidgets Wiki

Otherwise, or if you want to build a configuration of the library different from the default one, you need to build the library from sources before using it.

The first step, which you may have already performed, unless you are reading this file online, is to download the source archive and uncompress it in any directory. It is strongly advised to avoid using spaces in the name of this directory, i.e. notably do *not* choose a location under `C:\Program Files`, as this risks creating problems with makefiles and other command-line tools.

After choosing the directory location, please define WXWIN environment variable containing the full path to this directory. While this is not actually required, this makes using the library more convenient and this environment variable is used in the examples below.

NB: If you checked your sources from version control repository and didn't obtain them from a release file, you also need to copy include/wx/msw/setup0.h to include/wx/msw/setup.h.

5.3.2 Building wxWidgets

The following sections explain how to compile wxWidgets with each supported compiler, see the

Building Applications

section about the instructions for building your application using wxWidgets.

5.3.2.1 Cygwin/MinGW Compilation

wxWidgets supports Cygwin, MinGW, MinGW-w64 and TDM-GCC tool chains under Windows. They can be downloaded from:

- http://www.cygwin.com/
- http://www.mingw.org/
- http://mingw-w64.sourceforge.net/
- http://tdm-gcc.tdragon.net/

respectively. Please retrieve and install the latest version of your preferred tool chain by following the instructions provided by these packages. Notice that Cygwin includes both native Cygwin compiler, which produces binaries that require Cygwin during run-time, and MinGW[-w64] cross-compilers which can still be used in Cygwin environment themselves but produce plain Windows binaries without any special run-time requirements. You will probably want to use the latter for developing your applications.

If using MinGW, you can download the add-on MSYS package to provide Unix-like tools that you'll need to build wxWidgets using configure.

`C++ 11` note: If you want to compile wxWidgets in `C++ 11` mode, you currently have to use -std=gnu++11 switch as -std=`C++ 11` disables some extensions that wxWidgets relies on. I.e. please use CXXFLAGS=-std=gnu++11.

5.4 Configuration/Build options

wxWidgets can be built using almost countless permutations of configuration options. However, four main options are available on all platforms:

5.4.1 Release versus Debug:

Debug builds include debug information and extra checking code, whereas release builds omit debug information and enable some level of optimization. It is strongly recommended that you develop your application using a debug version of the library to benefit from the debug assertions that can help locate bugs and other problems.

5.4.2 Static versus Shared

A static version of the library is linked with the application at compile time, whereas a shared version is linked at runtime. Using a static library results in a larger application but also relieves the end user of the need for installing shared libraries on his or her computer. On Linux and Mac OS X, static libraries end in `.a`, and on Windows, they end in `.lib`. On Linux and Mac OS X, shared libraries end in `.so`, and on Windows, they end in `.dll`. Note that there are certain memory and performance implications of static and shared libraries that are not specific to wxWidgets; they are beyond the scope of this book.

5.4.3 Multi-lib versus Monolithic:

The entire wxWidgets library is made up of modules that separate the GUI functionality from the non-GUI functionality as well as separate advanced features that are not frequently needed. This separation enables you to use wxWidgets modules selectively depending on your specific needs. Alternatively, you can build wxWidgets into one monolithic library that contains all of the wxWidgets modules in one library file.

Using configure

1. Open MSYS or Cygwin shell prompt.
2. Create a build directory: it is is strongly recommended to not build the library in the directory containing the sources `($WXWIN)` but to create a separate build directory instead. The build directory can be placed anywhere (using the fastest available disk may be a good idea), but in this example we create it as a subdirectory of the source one:

   ```
   $ cd $WXWIN $ mkdir build-debug
   ```
3. Run configure passing it any of the options shown by configure –help. Notice that configure builds shared libraries by default, use –disable-shared to build static ones. For example:

   ```
   $ ../configure --enable-debug
   ```

4. Build the library:

   ```
   $ make
   ```

5. Test the library build by building the minimal sample:

   ```
   $ cd samples/minimal $ make
   ```

6. Optionally install the library in a global location

   ```
   $ make install
   ```

Notice that there is not much benefice to installing under Windows so this step can usually be omitted.

Using plain makefiles NOTE: The makefile.gcc makefiles are for compilation under MinGW using Windows command interpreter (command.com/cmd.exe), they won't work if you use Unix shell, as is the case with MSYS. Follow the instructions for using configure above instead if you prefer to use Unix shell.

1. Open DOS command line window (cmd.exe, *not* Bash sh.exe).

2. Change directory to

   ```
   > mingw32-make -f makefile.gcc
   ```

   to build wxWidgets in the default debug configuration as a static library. Add BUILD=release and/or SHARED=1 to build the library in release configuration and/or as a shared library instead of the default static one.

3. To verify your build, change the directory to `samples\minimal` and run the same mingw32-make command (with the same parameters there), this should create a working minimal wxWidgets sample.

4. If you need to rebuild, use clean target first.

Capitolo 6

Conclusions and future trends

Having doing the porting of the CaRo into cross platform, all the functions performed by the previous version have not been implemented, a very important improvement would be to reimplement:

- Playing Neutral
- Playing Expressiveness
- Perform the evaluations of both playing
- Perform the action of the options frame

Another important improvement during the interaction between user and the CaRo is to allow the change of the expressiveness by clicking in the apposite area on the interface image according to the expressivity desiderated

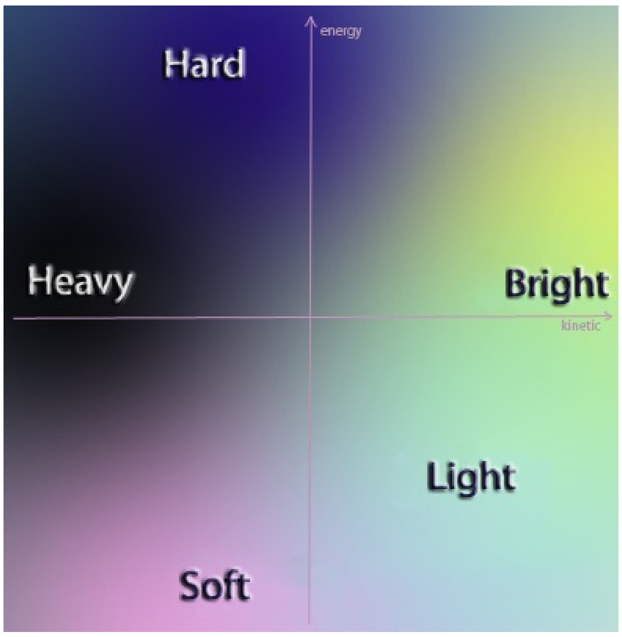


Figura 6.1: *Background image The static image of expressivity. Click an area to give the desire expressivity of song*

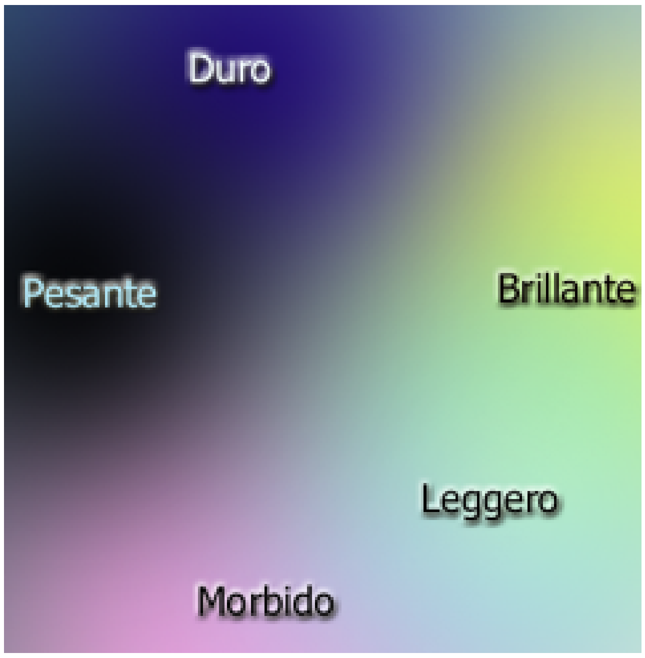


Figura 6.2: *Background image The static image of expressivity. Click an area to give the desire expressivity of song*

Other very interesting idea would be to make CaRo, an applet downloaded from the Internet, making it usable for novice users of music creativity, a musically useful tools to edit and create music to their liking.

Bibliografia

[1] ARCOS, J.L., MANTÀRAS, R., LÒPEZ DE, & SERRA, X. SaxEx: A case-based reasoning system for generating expressive performances. *Journal of New Music Research,* 27, 194210. 1998

[2] R. BRESIN AND G. WIDMER. Production of staccato articulation in Mozart sonatas played on a grand piano. Preliminary results *Speech, Music, and Hearing. Quarterly Progress and Status Report,* 2000(4):16, 2000.

[3] R. BRESIN AND G. U. BATTEL Articulation strategies in expressive piano performance. *Journal of New Music Research,* 29(3):211224, 2000.

[4] R. BRESIN. Articulation rules for automatic music performance. In A. Schloss and R. Dannenberg, editors, *Proceedings of the 2001 International Computer Music Conference, Havana, Cuba* pages 294297. International Computer Music Association, San Francisco, 2001.

[5] CAMURRI, C. L. KRUMHANSL, B. MAZZARINO, AND G. VOLPE. *An exploratory study of aniticipation human movement in dance.*. In Proceedings of the 2nd International Symposium on Measurement, Analysis, and Modeling of Human Functions. Genova, Italy, 2004.

[6] CAMURRI AND G. VOLPE, EDITORS. *Gesture-Based Communication in Human-Computer Interaction.* Springer, Berlin, 2004. LNAI 2915.

[7] CAMURRI, A., DE POLI, G., LEMAN, M., AND VOLPE, G. (2005). *Toward Communicating Expressiveness and Affect in Multimodal Interactive Systems for Performing Arts and Cultural Applications*. IEEE Multimedia, Vol.12, No.1, (pp.43-53), Jan 2005. IEEE Computer Society Press.

[8] E. F. CLARKE: *Empirical methods in the study of performance.*. In Eric F. Clarke and Nicholas Cook, editors, Empirical Musicology. Aims, Methods, and Prospects, pages 77102. University Press, Oxford, 2004

[9] S. CANAZZA, G. DE POLI, C. DRIOLI, A. RODA, AND A. VIDOLIN. Modeling and control of expressiveness in music performance. *Proceedings of the IEEE,* 92 (4):686701, 2004

[10] S. DAHL. *Playing the accent comparing striking velocity and timing in an ostinato rhythm performed by four drummers..* Acta Acustica, 90(4):762776, 2004.

[11] S. DAHL. Movements and analysis of drumming. In Eckart Altenmuller J. Kesselring, and M. Wiesendanger, editors, *Music, Motor Control and the Brain*, page in press. University Press, Oxford, 2005.

[12] J. W. DAVIDSON. *Visual perception of performance manner in the movements of solo musicians*. Psychology of Music, 21(2):103113, 1993

[13] J. W. DAVIDSON. *What type of information is conveyed in the body movements of solo musician performers?*. Journal of Human Movement Studies, 26(6): 279301, 1994.

[14] J. W. DAVIDSON AND J. S. CORREIA. BODY MOVEMENT. In Richard Parncutt and Gary McPherson, editors, The Science and Psychology of Music Performance. Creating Strategies for Teaching and Learning, pages 237250. University Press, Oxford, 2002.

[15] P. DESAIN AND H. HONING. Modeling continuous aspects of music performance: Vibrato and Portamento In Bruce Pennycook and Eugenia Costa-Giomi, editors,*Proceedings of the 4th International Conference on Music Perception and Cognition (ICMPC96).* Faculty of Music, McGill University, Montreal, Canada, 1996.

[16] A. FRIBERG *Generative rules for music performance. Computer Music Journal,* 15(2):5671, 1991.

[17] A. FRIBERG. A Quantitative Rule System for Musical Performance.Doctoral dissertation, Department of Speech, Music and Hearing, Royal Institute of Technology, Stockholm, 1995.

[18] A. FRIBERG, R. BRESIN, L. FRYDÉN AND J. SUNDBERG. Musical punctuation on the mircolevel: Automatic identification and performance of small melodic units. *Journal of New Music Research,* 27(3):271292, 1998.

[19] FRIBERG, A., & SUNDBERG, J. How to terminate a phrase. An analysis-by-synthesis experiment on a perceptual aspect of music performance. In: A. Gabrielsson (Ed.), *Action and Perception in Rhythm and Music,* (Vol. 55, pp. 4955). Stockholm, Sweden: Publications issued by the Royal Swedish Academy of Music. 1987

[20] A. FRIBERG AND J. SUNDBERG Does music performance allude to locomotion? A model of final ritardandi derived from measurements of stopping runners. *Journal of the Acoustical Society of America,* 105(3):14691484, 1999

[21] A. FRIBERG, V. COLOMBO, L. FRYDÉN AND J. SUNDBERG. Generating musical performances with Director Musices *Computer Music Journal,* 24(3):2329, 2000a.

[22] A. FRIBERG, J. SUNDBERG, AND FRYDÉN Music from motion: Sound level envelopes of tones expressing human locomotion *Journal of New Music Research,* 29(3):199210, 2000b.

[23] A. FRIBERG, V. COLOMBO, L. FRYDEN, AND J. SUNDBERG. Generating musical performances with Director Musices. *Computer Music Journal*, 24(3):2329, 2000a

[24] A. GABRIELSSON. Music performance research at the millenium. *Psychology of Music, 31(3):221272, 2003*

[25] A. GABRIELSSON. Once again: The Theme from Mozarts Piano Sonata in A Major (K.331). In Alf Gabrielsson, editor, *Action and Perception in Rhythm and Music,* volume 55, pages 81103. Publications issued by the Royal Swedish Academy of Music, Stockholm, Sweden, 1987.

[26] A. GABRIELSSON. Music Performance. In Diana Deutsch, editor, Psychology of Music, pages 501602. Academic Press, San Diego, 2nd edition, 1999.

[27] A. GABRIELSSON. Music performance research at the millenium. *Psychology of Music, 31(3):221272, 2003*

[28] A. GABRIELSSON AND E. LINDSTROM. The influence of musical structure on emotional expression. In Patrik N. Juslin and John A. Sloboda, editors, *Music and Emotion: Theory and Research*, pages 223248. Oxford University Press, New York, 2001.

[29] W. GOEBL. Melody lead in piano performance: Expressive device or artifact? *Journal of the Acoustical Society of America,* 110(1):563572, 2001.

[30] W. GOEBL. *The Role of Timing and Intensity in the Production and Perception of Melody in Expressive Piano Performance* Doctoral thesis, Institut fur Musik- wissenschaft, Karl-Franzens-Universitat Graz, Graz, Austria, 2003. available online at http://www.ofai.at/music.

[31] R. HIRAGA, R. BRESIN, K. HIRATA, AND H. KATAYOSE. Rencon 2004: Turing Test for musical expression.*In Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME04),* pages 120123. Hamamatsu, Japan, 2004.

[32] A. KAY, M. T. TURVEY, AND O. G. MEIJER. An early oscillator model: Studies on the biodynamics of the piano strike (Bernstein & Popova, 1930). *Motor Control,* 7(1):145, 2003.

[33] U. KRONMAN AND J. SUNDBERG. Is the musical retard an allusion to physical motion? In Alf Gabrielsson, editor *Action and Perception in Rhythm and Music,* volume 55, pages 5768. Publications issued by the Royal Swedish Academy of Music, Stockholm, Sweden, 1987.

[34] G. MAZZOLA AND O. ZAHORKA. Tempo curves revisited: Hierarchies of performance fields. *Computer Music Journal,* 18(1):4052, 1994.

[35] G. MAZZOLA, O. ZAHORKA, AND J. STANGE-ELBE. Analysis and performance of a dream. In Anders Friberg and Johan Sundberg, editors, *Proceedings of the KTH Symposion on Grammars for Music Performance,* pages 5968. Department of Speech Communication and Music Acoustics, Stockholm, Sweden, 1995.

[36] G. MAZZOLA AND S. GÖLLER. Performance and interpretation. *Journal of New Music Research,* 31(3):221232, 2002.

[37] G. MAZZOLA. editor. *The Topos of Music Geometric Logic of Concepts, Theory, and Performance.* Birkhauser Verlag, Basel, 2002.

[38] C. PALMER AND S. DALLA BELLA. Movement amplitude and tempo change in piano performance. Journal of the Acoustical Society of America, 115(5):2590, 2004.

[39] H. REPP. Diversity and commonality in music performance: An analysis of timing microstructure in Schumanns Traumerei. *Journal of the Acoustical Society of America,* 92(5):25462568, 1992

[40] O. ORTMANN. *The Physiological Mechanics of Piano Technique.* Kegan Paul, Trench, Trubner, E. P. Dutton, London, New York, 1929.

[41] J. SUNDBERG. . Four years of research on music and motion. *Journal of New Music Research,* 29(3):183185, 2000.

[42] E. SCHOONDERWALDT AND A. FRIBERG. Towards a rule-based model for violin vibrato. In Claudia Lomeli Buyoli and Ramon Loureiro, editors,*MOSART Workshop on Current Research Directions in Computer Music, November 15-17, 2001*, pages 6164. Audiovisual Institute, Pompeu Fabra University, Barcelona, Spain, 2001.

[43] J. SUNDBERG, A. ASKENFELT, AND L. FRYDÉN Musical performance. A synthesis-by-rule approach. *Computer Music Journal*, 7:3743, 1983.

[44] J. SUNDBERG, A. ASKENFELT, AND L. FRYDÉN Rules for automated performance of ensemble music.*Contemporary Music Review,* 3:89109, 1989.

[45] J. SUNDBERG, A. ASKENFELT, AND L. FRYDÉN Threshold and preference quantities of rules for music performance. *Music Perception,* 9(1):7192, 1991.

[46] J. SUNDBERG, A. ASKENFELT, AND L. FRYDÉN Threshold and preference quantities of rules for music performance. *Music Perception,* 9(1):7192, 1991.

[47] R. TIMMERS, R. ASHLEY, P. DESAIN, H. HONING, AND L. W. WINDSOR Timing of ornaments in the theme of Beethovens Paisello Variations: Empirical data and a model. *Music Perception,* 20(1):333, 2002.

[48] N. P. TODD. A computational model of Rubato. *Contemporary Music Review,* 3: 6988, 1989.

[49] N. P. TODD The dynamics of dynamics: A model of musical expression *Journal of the Acoustical Society of America,* 91(6):35403550, 1992.

[50] N. P. TODD The kinematics of musical expression. *Journal of the Acoustical Society of America,* 97(3):19401949, 1995.

[51] A. TOBUDIC AND G. WIDMER r. Relational IBL in classical music. *Machine Learning,* 64:524, 2006.

[52] G. WIDMER. In search of the Horowitz factor: Interim report on a musical discovery project. *In Proceedings of the 5th International Conference on Discovery Science (DS'02), Lubeck, Germany*. Springer, Berlin, 2002a.

[53] G. WIDMER. A machine learning analysis of expressive timing in pianists performances of Schumann Traumerei. In Anders Friberg and Johan Sundberg, editors, *Proceedings of the KTH Symposion on Grammars for Music Performance,* 6981. Department of Speech Communication and Music Acoustics, Stockholm, Sweden, 1995a.

[54] G. WIDMER. Modeling rational basis for musical expression. *Computer Music Journal,* Journal, 19(2):7696, 1995b.

[55] G. WIDMER. Learning expressive performance: The structure-level approach. *Journal of New Music Research, 25(2):179205,*

[56] G. WIDMER. Large-scale induction of expressive performance rules: first quantitative results. In Ioannis Zannos, editor,*Proceedings of the 2000 International Computer Music Conference, Berlin, Germany,* pages 344347. International Computer Music Association, San Francisco, 2000.

[57] G. WIDMER. In search of the Horowitz factor: Interim report on a musical discovery project. In *Proceedings of the 5th International Conference on Discovery Science (DS02), Lübeck,* Germany. Springer, Berlin, 2002a

[58] G. WIDMER. Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research,* 31(1):3750, 2002b.

[59] G. WIDMER AND A. TOBUDIC. Playing Mozart by analogy: Learning multi-level timing and dynamics strategies.*Journal of New Music Research,* 32(3):259268, 2003

[60] G. WIDMER. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence,* 146(2): 129148, 2003.

[61] G. WIDMER, S. DIXON, W. GOEBL, E. PAMPALK, AND A. TOBUDIC. In search of the Horowitz factor.*AI Magazine,* 24(3):111130, 2003.

[62] G. WIDMER. Studying a creative act with computers: Music performance studies with automated discovery methods.*Musicae Scientiae,* 9(1):1130, 2005

[63] A. TOBUDIC AND G. WIDMER. Relational IBL in classical music. *Machine Learning,* 64:524, 2006.