TENOR 2015

INTERNATIONAL CONFERENCE ON TECHNOLOGIES FOR MUSIC NOTATION & REPRESENTATION

UNIVERSITE PARIS SORBONNE / IRCAM

PARIS

First International Conference on Technologies for Music Notation and Representation
TENOR 2015
28-30 May, 2015
Université Paris-Sorbonne / Ircam
Paris

Proceedings published by :
Institut de Recherche en Musicologie, IReMus
2, rue de Louvois 75002 Paris

ISBN : 978-2-9552905-0-7
EAN : 9782955290507

Editors :
Marc Battier
Jean Bresson
Pierre Couprie
Cécile Davy-Rigaux
Dominique Fober
Yann Geslin
Hugues Genevois
François Picard
Alice Tacaille

Credits :
Nicolas Taffin (Logo design)
Nicolas Viel (Layout design)

IReMUS
institut de recherche
en musicologie

ircam
Centre
Pompidou

LAM
Institut Jean le Rond d'Alembert

GRAME
.FR
CENTRE
NATIONAL
DE CRÉATION
MUSICALE

ina
GRM

SORBONNE
UNIVERSITÉS

cnrs
dépasser les frontières

afim
Association Française d'Informatique Musicale

TENOR 2015

First International Conference on Technologies for Music Notation and Representation

---

Music notation serves the needs of representation, writing and creation. New musical forms such as electronic and/or interactive music, live coding, as well as the migration of musical instruments to gestural and mobile platforms, hybridizations with dance, design and multimedia tend to extend the notion of score in contemporary music, revisiting it through new forms of writing, and spreading it over different media. Until recently, the support provided by computer music to the field of symbolic notation remained fairly conventional. However, recent developments indicate that the tools for musical notation are today ready for a move forward towards new forms of representation.

Musical notation, transcription, sonic visualization, and musical representation are often associated in the fields of musical analysis, ethnomusicology, and acoustics. The aim of this conference is to explore these recent mutations of notation and representation in all these musical domains. The first International Conference on Technologies for Music Notation and Representation is dedicated to theoretical and applied research and development in Music Notation and Representation, with a strong focus on computer tools and applications, as well as a tight connection to musical creation.

The scholarly conference, posters and demo are taking place at Paris-Sorbonne University and Ircam.

Organizing committee

Organizing committee

Marc Battier, IReMus Université Paris-Sorbonne
Jean Bresson, Ircam-CNRS UMR STMS
Jérémie Garcia, Ircam (Posters and demo chair)
Pierre Couprie, IReMus Université Paris-Sorbonne
Cécile Davy-Rigaux, IReMus, CNRS
Dominique Fober, GRAME
Yann Geslin, INA-GRM
Hugues Genevois, LAM UPMC
François Picard, IReMus Université Paris-Sorbonne
Alice Tacaille, IReMus Université Paris-Sorbonne

Student volunteers

Fabiano Araujo
Elsa Filipe
Martin Guerpin
Marina Maluli
Daniel Rezende
Thomas Saboga

Steering committee

The Steering Committee is responsible for guiding future directions with regards to the TENOR conference. Its members currently include :
Jean Bresson, Ircam-CNRS UMR STMS
Pierre Couprie, IReMus Université Paris-Sorbonne
Dominique Fober, GRAME
Yann Geslin, INA-GRM
Richard Hoadley, Anglia Ruskin University
Mike Solomon, Ensemble 101

Scientific committee

A Carlos Agon
Andrea Agostini
Gerard Assayag

B Karim Barkati
Marc Battier
Sandeep Bhagwati
Andrew Blackburn
Alan Blackwell
Alain Bonardi
Bruno Bossis
Jean Bresson

C Elaine Chew
Michael Clarke
Pierre Couprie

D Cécile Davy-Rigaux
Frédéric Dufeu

E Simon Emmerson

F Dominique Fober
Ichiro Fujinaga

G Jérémie Garcia
Hugues Genevois
Yann Geslin
Daniele Ghisi
Jean-Louis Giavitto
Gérald Guillot

H Georg Hajdu
Keith Hamel
Richard Hoadley

J Florent Jacquemard
Guillaume Jacquemin

K Mika Kuuskankare

L Leigh Landy

M Thor Magnusson
Mikhail Malt
Peter Manning
Tom Mays
Alex Mclean

O Yann Orlarey

P Francois Pachet
François Picard

R Philippe Rigaux

S Eleanor Selfridge-Field
Mike Solomon
Marco Stroppa

T Alice Tacaille
Matthew Thibeault

V Anders Vinjar

Keynote

Does musical notation have a cultural centre?

In music encoding the expressions “common music notation” (CMN) and “common-practice period” are used freely as umbrella terms to cover European art music from 1700 to 1950. We use these terms as if a uniform understanding could be assumed. Increasingly, though, CMN is invoked to exclude music of three categories—early music, recent music, and non-Western music. If we examine CMN more closely, especially from the perspective of digital manipulation of some kind, we are inclined to keep chipping away are elements of some music within CMN to exclude particular or idiosyncratic repertories—Verdi operas, Tchaikovsky symphonies, the music of Béla Bartók and Zoltán Kodály, Western non-classical music of particular kinds, music that fulfills pedagogical needs, Braille Music Notation, and so forth. We also quickly discover that early music is not one “thing” in terms of notation but a cornucopia of notational styles, many of them less fully specified than the average score of today.

In the end, the same can be said of CMN: to the extent that written music is a compromise between a world of imagined sound and an practical means of enabling others to interpret it, the most seemingly conventional scores sometimes pose problems for which the encoder must choose between convention and reason, or else invent a new graphical means of expression. This is what has given rise in recent decades to frequent calls for new methods of notation, which in turn may threaten to undermine the usually serviceable language of CMN. While enabling music to seek new directions, we must be wary of invoking the “silo” practices of medieval scriptoria, in which music was “notated” exclusively for the use of a few individuals well known to the scribe.

E. Selfridge-Field Eleanor Selfridge-Field, Consulting Professor Music, is a musicologist and digital humanities scholar at Stanford University, where she heads the Center for Computer Assisted Research in the Humanities, an affiliate of the Packard Humanities Institute. She is the author of 16 books in digital musicology and 5 in historical musicology. Her teaching, most of it in collaboration with Craig Sapp, focuses on music representation systems and music-information retrieval.

Eleanor Selfridge-Field

E. Selfridge-Field Eleanor Selfridge-Field, Consulting Professor Music, is a musicologist and digital humanities scholar at Stanford University, where she heads the Center for Computer Assisted Research in the Humanities, an affiliate of the Packard Humanities Institute. She is the author of 16 books in digital musicology and 5 in historical musicology. Her teaching, most of it in collaboration with Craig Sapp, focuses on music representation systems and music-information retrieval.

TENOR 2015

TENOR 2015

LEADSHEETJS: A JAVASCRIPT LIBRARY FOR ONLINE LEAD SHEET EDITING

Daniel Martín
Sony CSL
dmartinmartinez@gmail.com

Timotée Neullas
Sony CSL
tneullas@gmail.com

François Pachet
Sony CSL
pachetcsl@gmail.com

ABSTRACT

Lead sheets are music scores consisting of a melody and a chord grid, routinely used in many genres of popular music. With the increase of online and portable music applications, the need for easily embeddable, adaptable and extensible lead sheet editing tools is pressing. We introduce LeadsheetJS, a Javascript library for visualizing, editing and rendering lead sheets on multiple devices. LeadsheetJS provides lead sheet editing as well as support for extensions such as score augmentation and peer feedback. LeadsheetJS is a client-based component that can be embedded from arbitrary third-party websites. We describe the main design aspects of LeadsheetJS and some applications in online computer-aided composition tools.

INTRODUCTION

A lead sheet is a specific type of music score consisting of a monophonic melody with associated chord labels (see Figure 1). Lead sheets are routinely used in many styles of popular music such as songwriting, jazz, pop or bossa nova.

With the rise of online music communities using performance or pedagogical applications, there is an increasing need for tools for manipulating music scores. In this context, music notation takes an important role, and in particular lead sheets, which are the main form of score for popular music. There is also a need for web-based tools for visualizing, playing, and editing lead sheets collaboratively. Such tools should also work on various devices, following the trend in using web applications on mobiles and tablets. Finally, these tools should intercommunicate easily with other tools, e.g. by being embeddable in third-party websites.

The most popular score editors, *Finale* and *Sibelius*, are designed as desktop applications. As such they cannot be used online, even though cloud features can be added, e.g. to share scores by exporting them to the web [9]. The open-source desktop-based editor *MuseScore*[1] provides features for sharing scores but does not provide directly online editing. There are many online tools to edit and view scores, but they do not rely on web standards, and often require the installation of a plugin on the web-browser. Some tools, such as *NoteFlight*[2], *Scorio*[3] or *Flat.io*[4], do follow standards and produce machine-readable scores, but they are not designed specifically for lead sheets. For instance, they do not support chord notations, an important feature of a lead sheet.

Besides offering basic score editing services, online lead sheet tools should provide features for *augmented editing*, e.g. to be tailored to pedagogical or social contexts. The ability of adding heterogeneous graphic objects such as colored layers, text or images, is crucial to enable collaboration between users as a way for giving feedback on certain parts of the score. *INScore* [4] supports various graphical objects, but is not easily embeddable in an online application and it is more focused on real-time rendering of interactive music scores [6] for new forms of composition and performance.

This paper presents LeadsheetJS a Javascript library for storing, visualizing, playing, editing and making graphical annotations on lead sheets. In the following section we describe the main features of the library. Then we give some hints about its implementation. We finally describe tools built on top of this library.

LEADSHEETJS

LeadsheetJS is a Javascript library for lead sheets. It enables the edition and visualization of lead sheets under conventional formats, as well as rendering, playing and storing lead sheets in a database. Figure 2 shows how LeadsheetJS interfaces with the player, the menu for editing and the rendered leadsheet.

LeadsheetJS provides tools for users to collaborate and give feedback to each other by highlighting certain parts

[1] http://musescore.org/
[2] http://www.noteflight.com
[3] http://www.scorio.com/
[4] https://flat.io/

of the lead sheet and commenting or suggesting modifications. LeadsheetJS has been implemented in Javascript, the main programming language for web browsers. This makes LeadsheetJS web-friendly and easily embeddable in third-party sites, as well as adaptable to several devices.

In the next sections we describe the main features of LeadsheetJS and we give a detailed explanation about the main design and implementation aspects.

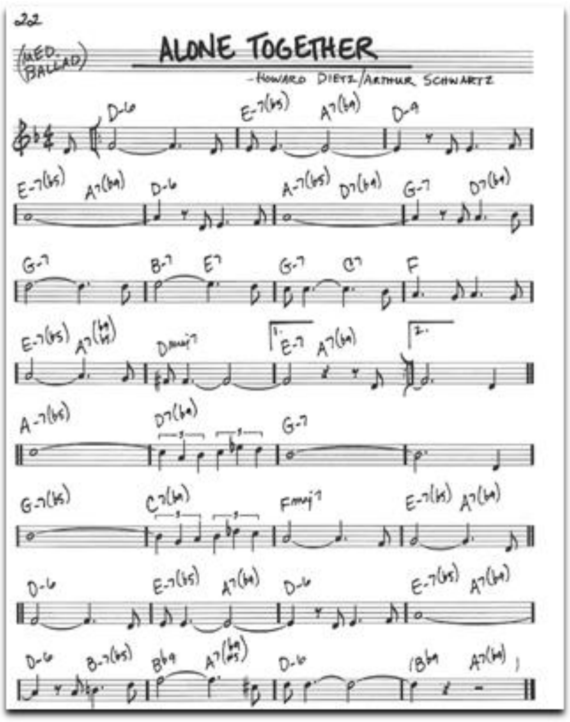


Figure 1. The lead sheet of *Alone together* by Dietz & Schwartz, as found in a typical *Fake Book*.

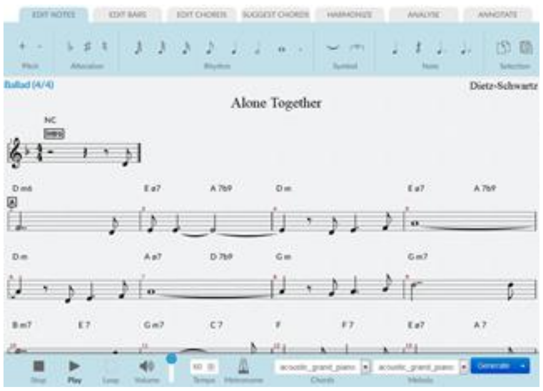


Figure 2. *Alone together* by Dietz & Schwartz, rendered in a browser with LeadsheetJS

1 Peer feedback on lead sheets

"The one true comment on a piece of music is another piece of music", Stravinsky [17].

Music composition, as well as music learning, is a domain in which *feedback* on pieces being composed plays a major role. Feedback is traditionally provided by a teacher. Nowadays, on-line learning websites provide tools for peer-feedback in which learners can produce and review feedback made by peers.

The possibility of giving feedback on the audio representation of a piece of music has been addressed in previous works, e.g. [19, 20]. However, by commenting on pure audio, i.e. on a rendered waveform, users are limited to commenting on given time spans, whereas by commenting on a lead sheet, users can refer directly to the musical elements making up lead sheets, such as notes, chord labels, chord transitions, bars or structural elements (see Figure 3).

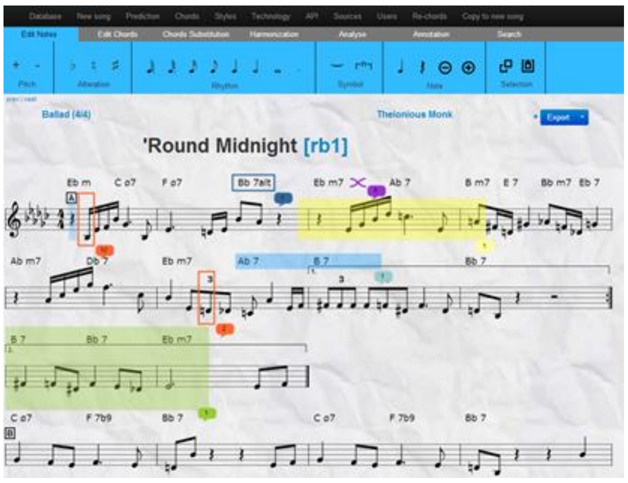


Figure 3. Examples of annotations on specific parts of a lead sheet.

In LeadsheetJS, feedback can be given at three levels:

a) Musical feedback: the basic level of feedback is musical. That is, a suggestion of a modification of a certain part of the lead sheet, such as changing certain notes, or certain chord labels,

b) Text feedback: musical suggestions can be explained with an explanation in the form of text comment,

c) Audio feedback: sometimes a musical idea is better expressed by being played in an instrument. Users can record a musical snippet, upload it and associate it to a specific metrical location in the lead sheet.

2 Embeddability

Arbitrary websites can render lead sheets by importing the LeadsheetJS library in the HTML source code. New lead sheets can be created or imported and rendered and edited from the site. As an example we show a website in the MusicCircle platform [19], displaying the lead sheet *Blue Room* by Rodgers & Hart (see Figure 4).

First, the LeadsheetJS library is imported in the HTML page. Then, the lead sheet of *Blue Room* is imported from a database (LSDB, described later) in our JSON lead sheet format through the LSDB API, which allows external sites to retrieve lead sheets. Finally, the JSON text is converted to a LeadsheetJS object and displayed in the page (see Figure 5).

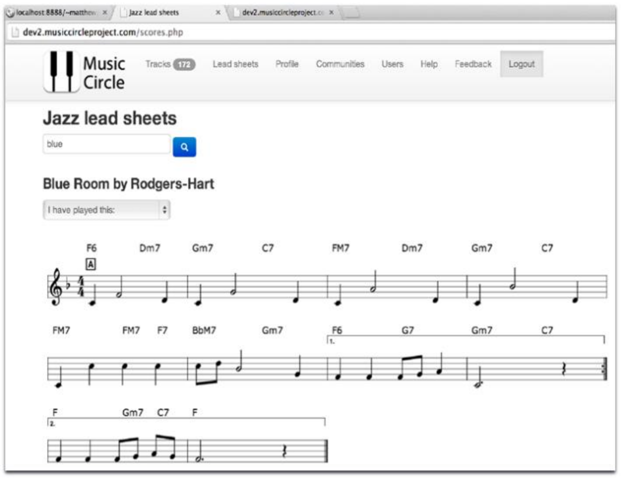
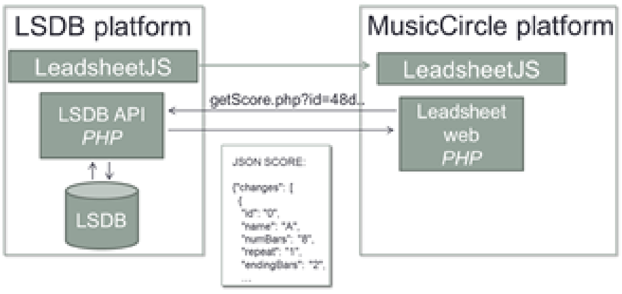


Figure 4. A lead sheet view embedded in a third party site.



```
{"changes": [
  {
   "id": "0",
   "name": "A",
   "numBars": "8",
   "repeat": "1",
   "endingBars": "2",
   ...
```

Figure 5. Architecture for embedding LeadsheetJS.

3 Multi-device

Web applications are not accessed only from a desktop computer but also from tablets and mobile phones: responsive web design has become essential for designing web applications. To that aim, LeadsheetJS resizes automatically scores depending on the width of the screen. This way it can be visualized in devices with different screen sizes such as tablets or mobile phones (see Figure 6).

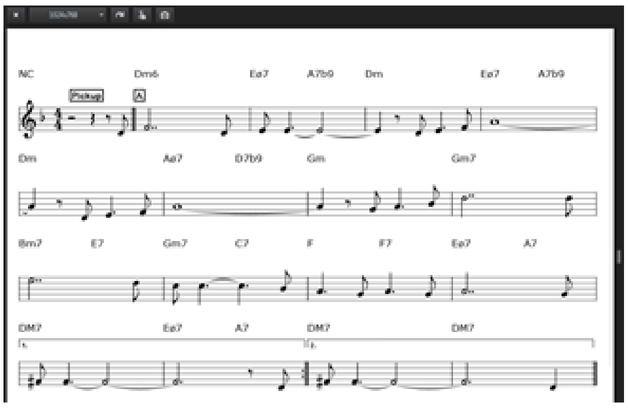


Figure 6. LeadsheetJS on a 1024x768 tablet.

4 Audio wave visualization

LeadsheetJS does not handle only symbolic information. Recordings of the performance of a lead sheet can also be associated to the lead sheet. LeadsheetJS provides visualization of the recording's waveform synchronized with the lead sheet, so that on top of each measure, the waveform of the recording part corresponding to that measure is displayed (see Figure 7). This feature is useful for musicians who record themselves performing a given lead sheet. They can then listen to their performance and see at the same time the lead sheet and the audio representation.

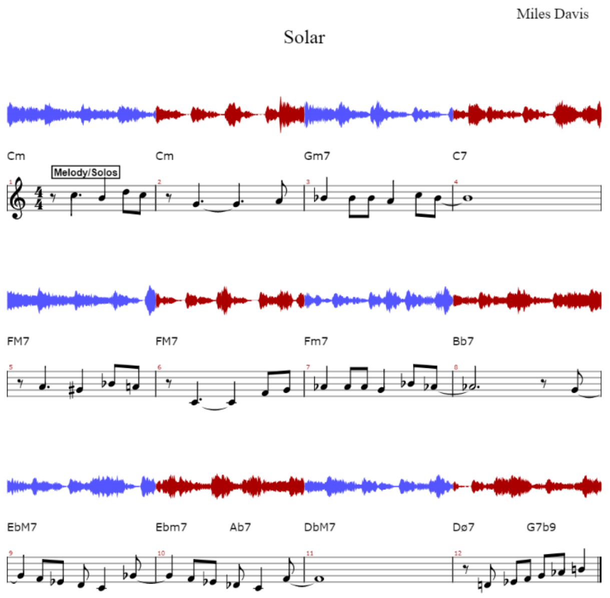


Figure 7. LeadsheetJS visualizing *Solar*, by Miles Davis, and audio recording displaying

5 Design

LeadsheetJS is a complex library that provides many functionalities (editing, visualizing, playing, storing). From an architectural point of view, it needs to be maintainable, scalable and extensible. Furthermore, modularity is required as users may need to use only certain features of LeadsheetJS. For example, a music blogger may want to visualize and play lead sheets in her blog without allowing edition or audio visualization.

The design of LeadsheetJS is module-based. It is inspired by Zakas' architecture [21] in which every module is an independent unit that does not need any other module to work. Zakas' architecture is based on the MVC (Model-View-Controller) architecture. Every module has its own model, view and controller classes. Each module is composed of a set of classes. There is one file per class. In total LeadsheetJS contains about 150 classes.

LeadsheetJS is a client-based Javascript library, i.e. it runs in the browser. However, certain functionalities require communication with a server or a database, such as storing or retrieving lead sheets. Databases and servers are not part of LeadsheetJS, yet it provides modules to communicate with them.

The architecture scheme is shown in Figure 8. The central module is *Leadsheet Model*. All modules depend on it since they need it in order to work. Modules *Viewer*, *Player* and *Interactor* provide visualization, playing and edition functionalities respectively. The *Annotation* module provides graphic annotation for peer feedback purposes. The *Format exporter/importer* modules is a converter to various formats so that the represented lead sheet can be sent to (or received from) other applications. The *Ajax* module facilitates the communication to a server. Therefore, it is used by the modules that depend on a server: the *Data Base* module, which is in charge of storing the lead sheet to a database in a given format, and the modules that are analysis tools which we describe in section 3.

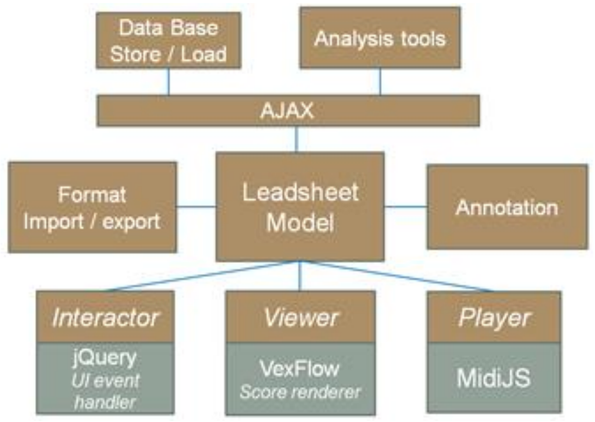


Figure 8. Module architecture of LeadsheetJS.

Thanks to its modular nature, LeadsheetJS can be easily extended by adding modules that communicate with the existing ones.

In Figure 9 we show an example of LeadsheetJS embedded within a complete system with a client/server database system where LeadsheetJS is the client part, and PHP is the language on the server side that manages user sessions and persistence (saving lead sheets into a MongoDB database). The Ajax module is in charge to send requests to the server. For example, in order to store a lead sheet in a database the Database module will send the data to the server as an HTTP request through the Ajax Module.

The core module, *Leadsheet Model*, represents a lead sheet. A lead sheet consists of a melody that is in most cases monophonic, and a chord label grid representing the harmony. From a structural point of view, a lead sheet is a hierarchical structure composed by sections, which are composed of bars, which in turn are formed by a list of notes (a melody), and a list of chord labels. Each of these levels defines specific attributes: at the top level, the lead sheet has a composer, a title, a style as well as musical attributes such as global key and time signature. Section related information attributes are section name, number of bars, number of repetitions and number of endings. Bars may also have specific time or key signature changes, as well as structure labels like coda or segno. Finally, the lowest levels of the hierarchy are notes and chord labels.

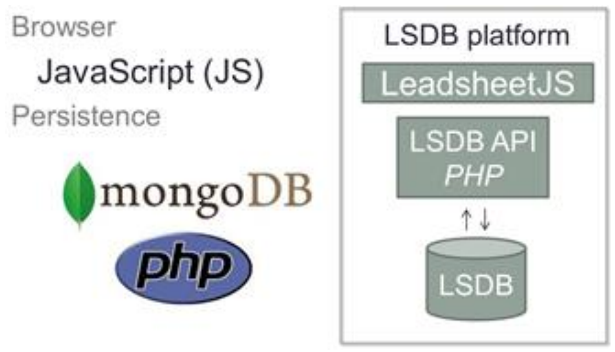


Figure 9. Example of a client-server database structure using LeadsheetJS.

The example in Figure 1 shows a lead sheet as found in a typical *Fake book*, with its attributes such as title, "Alone Together", composer "Howard Dietz and Arthur Schwarz", style "Medium Ballad". This lead sheet has two sections: the first one contains 14 bars and two endings; the second one has 12 bars.

The Leadsheet Model module enables applications to store and retrieve information about a lead sheet such as its structure, a specific bar, a chord label, or a group of notes, as well as metadata associated to it such as its title, composer, style, time signature or key signature. Typical queries include *get the notes of the first bar*, *get the number of sections*, etc. The Leadsheet Model also enables creation of new lead sheets or copies.

5.1 *Viewer*

The Viewer renders lead sheets on the web browser through an HTML5 canvas API, which allows generating graphics dynamically. The Viewer uses *Vexflow*[5], a low level score rendering Javascript library. *Vexflow* addresses low level rendering of notes and staves, whereas LeadsheetJS specifies what to draw in each bar as well as other higher level tasks such as determining how many bars to display per line.

5.2 *Interactor*

The Interactor component provides the editing part by using the library *JQuery*[6] which, among many other things, takes care of event handling. Keyboard and mouse events are caught by the Interactor to perform desired transformations on an edited lead sheet. We introduce three levels of edition: notes, chord labels and bars. Note edition works like in any traditional score editor. Chord label edition provides specific interaction schemes such as completion to suggest the most relevant chord types in a given context (see Figure 10). LeadsheetJS contains a comprehensive database of over 300 chord types, collected during the process of a lead sheet database compilation described in section 3.1.

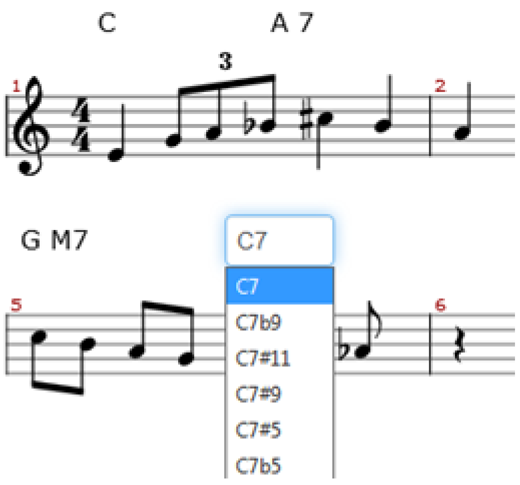


Figure 10. Chord label completion to speed up edition.

5.3 *Player*

LeadsheetJS provides a MIDI Player which uses the library *MidiJS*[7] to play a lead sheet, i.e. both the melody and the chord labels. The chord labels are transformed into MIDI chords.

The chord labels are represented by a pitch and a chord type. E.g.: in C# maj7, *C#* is the pitch and *maj7* the chord type. The chord type database provides information about the note degrees for each chord type. For instance for maj7 the degrees are I, III, V and VII.

In order to play chords, LeadsheetJS transforms chord labels into sets of MIDI notes by calculating the notes degrees of the chord type relative to the root pitch. E.g.: for C# maj7, notes are C#-E#-G#-B#. The player plays them arbitrarily in the 4th octave, so MIDI notes are 61-65-68-72. Other more refined MIDI players can easily be defined by the user.

5.4 *Javascript Module Management*

As a client-based application, LeadsheetJS runs on the browser, so each Javascript file needs to be imported in the HTML source code through the *script* tag. This may be an issue as we need to include explicitly each file and there are around 150 classes, while not all classes are always needed. For example, an instance of LeadsheetJS could only show a lead sheet and play it: in that case there is no need for editing, so the Interactor module does not need to be loaded. To optimize loading time, and ensure only needed modules are loaded, LeadsheetJS uses *RequireJS*[8], a tool to manage dependencies in Javascript.

In order to provide communication between modules in an uncoupled way we make an intensive use of the *Mediator* design pattern [12]. The Mediator pattern encapsulates the way different modules interact. It enables a module to *subscribe* to an action of another module which *publishes* it.

For example, when the Leadsheet Model module changes the pitch of a note, it *publishes* that action; that is, it sends a message to a *mediator* telling that the note's pitch has changed. The *mediator* checks which modules are interested in the action of *note pitch changed*; that is, which modules are *subscribed*, and informs them. This way, the Viewer module, which is subscribed to *note pitch changed*, knows it must redraw the score.

The advantage of using this pattern is that Leadsheet Model and Viewer do not communicate directly, which brings to uncoupled code, thus, more scalable and maintainable.

5.5 *Javascript implementation*

Javascript is a prototype-based language rather than a class-based one like C++ or Java. In order to define classes, there are mainly two approaches: to use Object literals or to use prototypes. By using object literals to define classes one can use private variables by using the *Module Pattern* [12]. The Module Pattern takes advantage of closures to simulate private variables, which are not natively supported in Javascript. On the other hand, using prototypes to define classes one cannot emulate private variables, but this approach has the advantage that it is less memory consuming, since all the methods of all instances of a class share the same memory. We have mainly used the Prototype approach as

[5] `http://www.vexflow.com`
[6] `http://jquery.com/`
[7] `http://mudcu.be/midi-js/`

[8] `http://requirejs.org/`

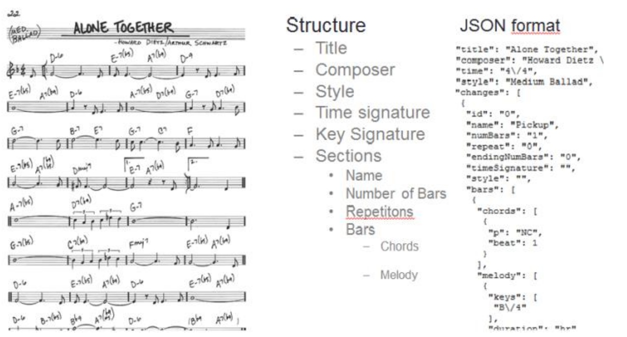
we are using multiple instances of many classes such as NoteModel or ChordModel.

6 JSON lead sheet format

LeadsheetJS provides a format to store lead sheet data in a database. The most common format for representing music scores is MusicXML [7]. LeadsheetJS does not use MusicXML for the following reasons: first, in MusicXML, chord labels' information is associated to a note, so the start beat of the chord is the same as that of the associated note. This makes it difficult to represent chords whose start beat does not match with the start beat of a note. This might not be a problem for other kinds of scores, but in lead sheets chord labels are crucial. That is why in our lead sheet format each chord label has its start beat information. Second, MusicXML provides exact formatting: it saves both musical and visualization information; e.g. for each note it saves the stem direction and the exact position in which it will be shown. LeadsheetJS only needs the musical information to render the lead sheet. The visualization aspects (stem directions, position of each element…etc.) is decided by Vexflow.

There are other human-readable music notation formats like ABC [3] and Lilypond [11]. Both are designed to let users create easily scores by writing text which is compiled by a software that produces a rendered score as an output. Therefore, they are not designed to be used in WYSIWYG[9] editors. The Guido Music Notation format [5], designed to be rendered by the Guido Engine Library [2] is similar to them, but is not only a representation format; it also supports 'functions' as instructions for transforming the score (e.g. transposing a melody). In our case, readability is not a priority as we do have a WYSIWYG editor. Instead, we have designed a JSON (JavaScript Object Notation) based format [1], as JSON is a popular lightweight format which is widely used in web APIs. For example, the GUIDO API web-service is based in JSON [18]. Further, a lead sheet has a hierarchical structure which can be very well represented by the JSON format (see Figure 11). The decision of using JSON has distanced us from using other formats like MEI [16], a notation encoding standard based on XML similar to MusicXML.

However, LeadsheetJS is compatible with MusicXML as it provides a parser to transform MusicXML to our JSON lead sheet format, and it will eventually support other formats too (Lilypond, Guido and ABC).



```
"title": "Alone Together",
"composer": "Howard Dietz \
"time": "4\/4",
"style": "Medium Ballad",
"changes": [
 {
  "id": "0",
  "name": "Pickup",
  "numBars": "1",
  "repeat": "0",
  "endingNumBars": "0",
  "timeSignature": "",
  "style": "",
  "bars": [
   {
    "chords": [
     {
      "p": "NC",
      "beat": 1
     }
    ],
    "melody": [
     {
      "keys": [
       "B\/4"
      ],
      "duration": "hr"
```

Figure 11. The lead sheet *Alone together* represented in JSON.

OTHER APPLICATIONS

This section describes applications using LeadsheetJS in various ways.

1 Lead sheet Database (LSDB)

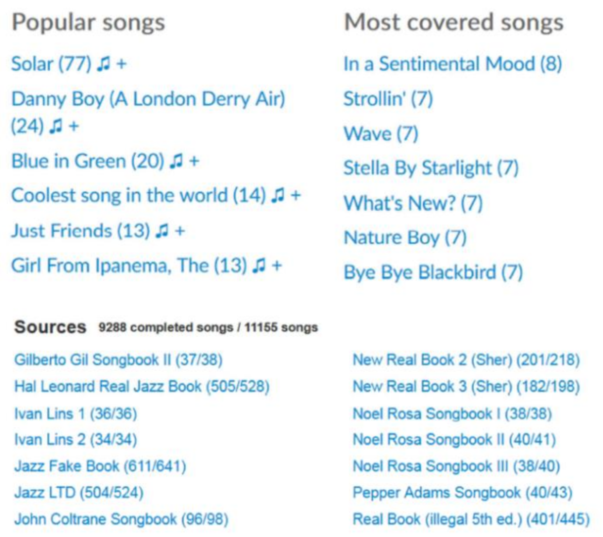
The Lead sheet Database (LSDB) [15] is a comprehensive, on-line database of lead sheets for jazz and Brazilian music. Currently LSDB contains over 10,000 songs from 76 different song books, and over 300 different chord types.

Songs are entered by professional musicians using LeadsheetJS. Average time for entering songs is about 3 minutes, thanks to the availability of many short-cuts for fast editing. An LSDB API stores/retrieves lead sheets from the database, as described in section 2.2. This database is used for musicological analysis and music generation applications such as the tools described in section 3.2

The LSDB database uses MongoDB[10], a non-relational database (NoSql). NoSql databases are based on collections that contain JSON documents, which are structures of nested arrays and objects (objects are set of key-values). The biggest drawback of using a NoSql is that some important features of SQL databases such as *joins* or referential integrity cannot be performed at the database level, and have to be managed from the code of the server that produces the queries. This can be an issue in applications with complex databases, but in our case it is not, because the database structure is quite simple: there is a main collection of lead sheets, and then other related collections like sources and composers, so integrity is not as crucial as in other more complex systems. Joins are managed from the server language's code. Moreover, the JSON structure on NoSql databases is ideal to represent tree-based structures like lead sheets, whereas representing a tree in a SQL is quite more complex.

[9] What You See Is What You Get

[10] `http://mongodb.com/`


Popular songs

- Solar (77) ♬ +
- Danny Boy (A London Derry Air) (24) ♬ +
- Blue in Green (20) ♬ +
- Coolest song in the world (14) ♬ +
- Just Friends (13) ♬ +
- Girl From Ipanema, The (13) ♬ +

Most covered songs

- In a Sentimental Mood (8)
- Strollin' (7)
- Wave (7)
- Stella By Starlight (7)
- What's New? (7)
- Nature Boy (7)
- Bye Bye Blackbird (7)

Sources 9288 completed songs / 11155 songs

- Gilberto Gil Songbook II (37/38)
- Hal Leonard Real Jazz Book (505/528)
- Ivan Lins 1 (36/36)
- Ivan Lins 2 (34/34)
- Jazz Fake Book (611/641)
- Jazz LTD (504/524)
- John Coltrane Songbook (96/98)
- New Real Book 2 (Sher) (201/218)
- New Real Book 3 (Sher) (182/198)
- Noel Rosa Songbook I (38/38)
- Noel Rosa Songbook II (40/41)
- Noel Rosa Songbook III (38/40)
- Pepper Adams Songbook (40/43)
- Real Book (illegal 5th ed.) (401/445)

Figure 12. Part of LSDB content as shown in the web.

2 Automatic Feedback on lead sheets

Feedback can sometimes be provided automatically. LeadsheetJS provides various tools that produce automatic feedback to users who are trying to compose a song. This feedback can be either in the form of an analysis of the lead sheet, or in the form of generations and transformations of a lead sheet.

For instance, a *Chord Sequence Analyzer* tries to find which *style* or styles a sequence of chords expresses. A style is defined here by a *corpus* of songs, corresponding to a given composer; e.g. the style of Miles Davis [8]. The Chord Sequence Analyzer identifies the longest subsequences that can be analyzed in the style of a given set of key composers. This analysis is performed by computing the similarity of the chord sequence with several different composers' models. These models are statistical models generated from the LSDB.

Such a tool may be used to get information about how original or similar a lead sheet is, with regards to the LSDB database. Figure 13 shows such an analysis for the chord sequences *Solar* with a map showing a time-line of the song and each composer (Pepper Adams, Charlie Parker, Duke Ellington and Michel Legrand)

Another example is the *Harmonic Analysis* tool that finds the local tonalities of a lead sheet given its chord label sequence [13]. Figure 14 shows two examples of analysis: Gm7 – C7 has been analyzed as *F Major* chords, whereas Fm7 – Bb7 are analyzed as *Eb Major*. These chords are part of *Solar*, by Miles Davis.

Other automatic feedback tools have been defined, such as a *Chord Substitution* tool which, from a given chord or chord sequence, suggests alternatives based on chord substitution rules that are learnt from a specific corpus.

The *Harmonizer* tool, given a monophonic melody, proposes a multi-voice harmonization in a given style. E.g.: one can harmonize the melody of Coltrane's jazz standard *Giant Steps* in the style of Wagner or Bill Evans [14].

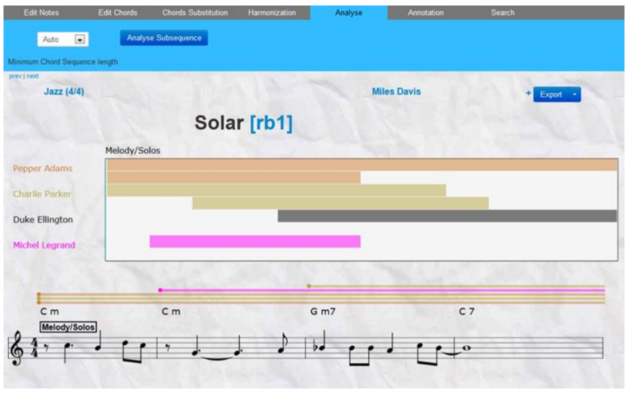


Figure 13. A chord sequence analyzer grafted on top of LeadsheetJS.

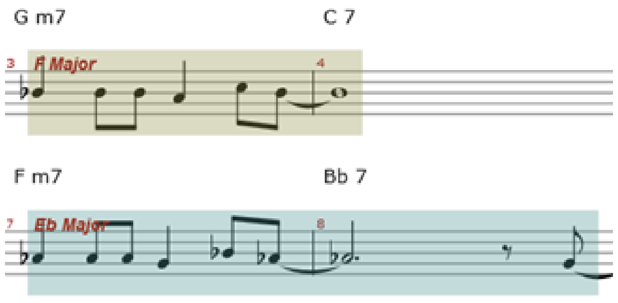


Figure 14. Harmonic analysis displayed on parts of *Solar*, by Miles Davis.

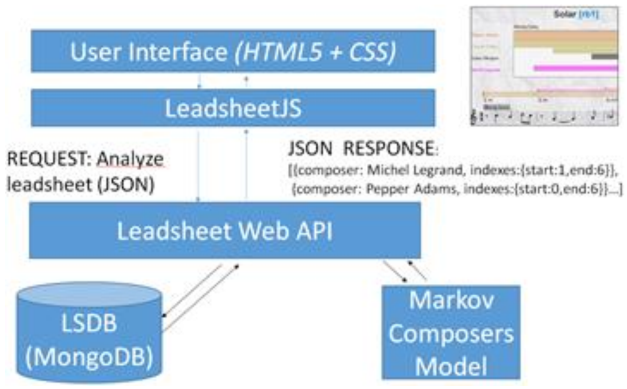

User Interface (HTML5 + CSS)
LeadsheetJS
REQUEST: Analyze leadsheet (JSON)
JSON RESPONSE:
[{composer: Michel Legrand, indexes:{start:1,end:6}}, {composer: Pepper Adams, indexes:{start:0,end:6}}...]
Leadsheet Web API
LSDB (MongoDB)
Markov Composers Model
Solar [rb1]

Figure 15. LeadsheetJS architecture and the data flow of chord sequence analyzer.

Figure 15 shows the architecture of these tools and illustrates the process for the Chord Sequence analyzer tool: The user clicks on a button 'Analyze chord sequences'. LeadsheetJS catches the user action and requests the chord sequence analysis of *Solar*, sent in JSON format through the Ajax module. The request is sent to the server where the Leadsheet Web API, which is a server extension of LeadsheetJS, computes the chord sequence analysis. The response is sent to the client, where LeadsheetJS presents it in the User Interface as a time-line map.

3 Flow Composer

In the context of the Flow Machines[11] project about style imitation, an online composition tool called Flow Composer was designed, to help a composer generate a lead sheet using different “styles”. Again, styles are defined by corpus of songs taken from the Lead sheet Database.

The main idea is that a composer can start to create a song and leave some empty measures in which there will be only silences. Then, he queries the system to fill those blanks in a given style. Those blanks can be on the melody, represented by silences, or on the chord grid, represented by *No Chords* (NC). The system will generate a melody or chord labels to fill them taking into account the style chosen by the user, and also constraints of continuity. Composers usually don’t want a whole new random song; they rather want the system to help them with certain parts of their composition. The composer can accept or reject all or part of the system’s proposition. Flow Composer tools allow composers to have at any moment a full control on the lead sheet: there is a *history* feature in which every step is saved, so they can go back to a previous state.

Flow Composer is built on top of LeadsheetJS and uses the same modular approach. LeadsheetJS is used in Flow Composer to listen, view and edit lead sheets. We show in Figure 16 how Flow Composer works. In the first image (on the top) a user is composing a bossa-nova. In the song there are two parts. The second part starts at measure 7 (with note *F* and chord *F7*) and is not shown in the figure. The second part is ok, but the composer does not know how to finish the first part so that it transitions well to the second part. So he leaves it empty with silences and *no chords* (NC), and queries Flow Composer to fill the empty part in the *bossa-nova style*. The second image (on the bottom) shows the result proposed by Flow Composer: it has filled the empty part by proposing a melody and a chord grid. Interaction may then proceed by accepting parts of the suggestions and/or querying other solutions.

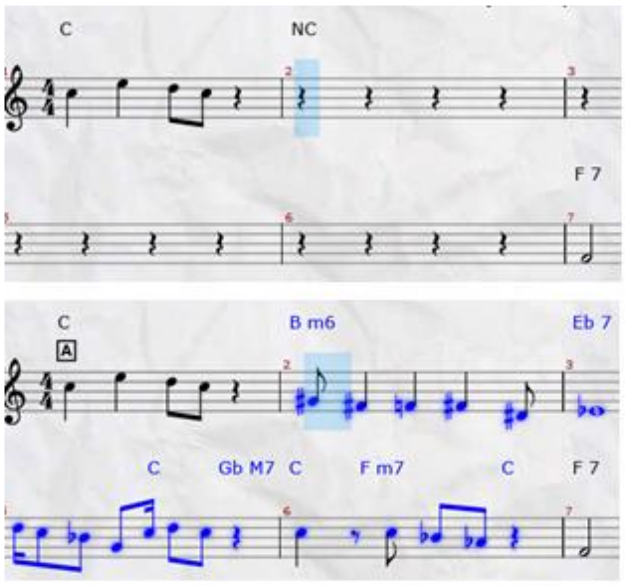


Figure 16. Flow Composer completion in blue.

4 Experiment on feedback in composition

PRAISE[12] (Practice and Performance Analysis Inspiring Social Education) is a social network for music education with tools for giving and receiving feedback in online communities. In the context of PRAISE we have built a tool for feedback in composition in which composers can compose a lead sheet and share it with other composers who can then provide feedback. This tool is based on the annotation module of LeadsheetJS.

In the PRAISE project, we designed an experiment to determine the impact of feedback in lead sheet composition [10]. We evaluate whether musical peer feedback, just like in the example explained in section 2.1, may actually improve or not the musical *quality* of a composition. In a first phase, participants are asked to compose a short song (8 bars). In the second phase they are invited to suggest modifications of other participants’ compositions. Then participants are asked to reconsider their original song and try to improve it. The point is that a group of subjects will have received feedback whereas another group will have not. We then evaluate to which extent the quality of the improved composition of those subjects who received is better than that of those who did not. The quality evaluation is estimated from a listening panel. LeadsheetJS was used to implement this experiment, including modules for editing and playing for the composition phase and the Annotation module for the feedback phase.

The composer of the lead sheet can later review suggestions and accept them or not.

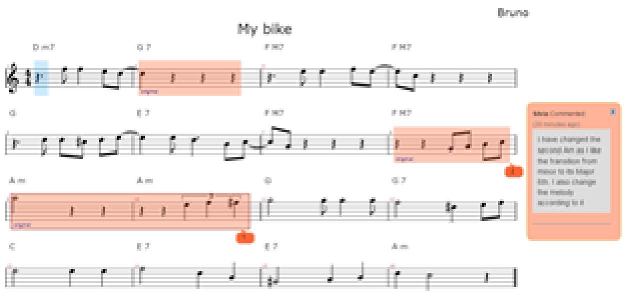
The feedback process is illustrated as follows. First, user *Bruno* composes a song and edits it with

[11] `http://www.flow-machines.com`

12 `http://www.iiia.csic.es/praise/`

LeadsheetJS. Later, user *Silvia* looks at it and plays it. She decides to make some suggestions on certain notes. As shown in Figure 17 once she has saved the suggestion, she can perform other actions, shown in the contextual menu :

- Add Comment: add an explanation of her musical suggestion,
- Upload sound: upload a sound recording related to the suggestion,
- Modify: she can decide to modify the suggestion she just saved,
- Remove: remove the suggestion.



I have changed the second Am as I like the transition from minor to its Major 6th. I also change the melody according to it

Figure 17. A user makes a suggestion on a specific part of a lead sheet.

Later on, Bruno can review all suggestions by switching between the *original* elements and *suggested* ones and listen to them. Figure 18 shows a lead sheet with three suggestions. Bruno clicks on one of them to see the associated explanation.

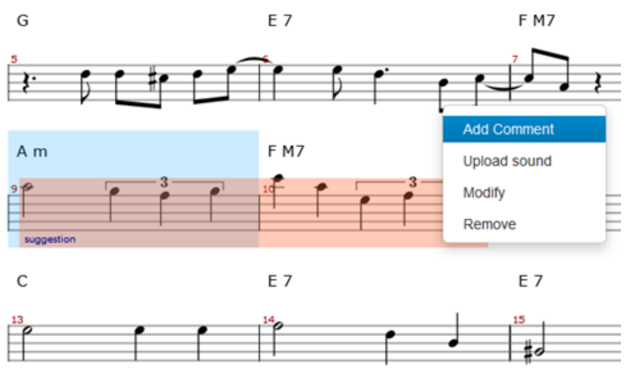


Figure 18. A user checks the suggestions received.

Finally, if Bruno likes the suggestion he can *accept* it so that the suggestion is merged with the whole song by right-clicking on the suggestion (see Figure 19).

CONCLUSION

We have presented LeadsheetJS, a Javascript library for lead sheets. By design, LeadsheetJS is compatible with multiple devices and easily embeddable. LeadsheetJS also provides various tools for music composition such as automatic analysis and peer feedback. We have illustrated how LeadsheetJS is used in several online music applications.

LeadsheetJS addresses the needs of online applications for composing, generating, sharing or teaching music on-line. New features are currently investigated such as multiple voices management, lyrics, audio based player, as well as rendering lead sheets using style-based accompaniment generation systems.

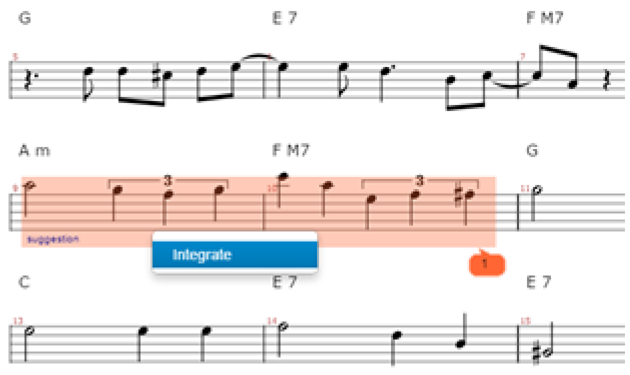


Figure 19. The user accepts a suggestion of modification.

Acknowledgments

This work is supported by the Praise project (EU FP7 number 388770), a project funded by the European Commission under program FP7-ICT-2011-8.

REFERENCES

[1] D. Crockford, “The json data interchange format”. *Technical report*, ECMA International, October 2013.

[2] C. Daudin, D. Fober, S. Letz, Y. Orlarey, “The guido engine a toolbox for music scores rendering.” In *Proceedings of the Linux Audio Conference*, 2009, pp. 105–111.

[3] G. Dyke, P. Rosen, abcjs–Project Hosting on Google Code, 2010.

[4] D. Fober, S. Letz, Y. Orlarey, F. Bevilacqua, “Programming Interactive Music Scores with INScore”. In *Proceedings of the Sound and Music Computing Conference*, 2013 july, pp. 185-190.

[5] D. Fober, Y. Orlarey, S. Letz, “Scores level composition based on the Guido Music Notation”. Ann Arbor, MI: MPublishing, University of Michigan Library, 2012.

[6] D. Fober, Y. Orlarey, S. Letz, “Augmented Interactive Scores for Music Creation.” In *Korean Electro-Acoustic Music Society's 2014 Annual Conference*, 2014 october, pp. 85-91.

[7] M. Good, “MusicXML: An internet-friendly format for sheet music.” In *XML Conference and Expo*, 2001, pp. 3-4.

[8] T. Hedges, P. Roy, F. Pachet, “Predicting the Composer and Style of Jazz Chord Progressions.” *Journal of New Music Research*, 43(3), 2014, pp. 276-290.

[9] J. Kuzmich, “The two titans of music notation.”, *School Band & Orchestra magazine*, 2008 september.

[10] D. Martín, B. Frantz, F. Pachet, “Assessing the impact of feedback in the composition process: an experiment in lead sheet composition.” In *Tracking the Creative Process in Music*, Paris, 2015, October.

[11] H. W. Nienhuys, J. Nieuwenhuizen, “LilyPond, a system for automated music engraving.” In *Proceedings of the XIV Colloquium on Musical Informatics* (XIV CIM 2003) , 2003, pp. 167-172.

[12] A. Osmani, *Learning JavaScript Design Patterns*. O'Reilly Media, Inc., 2012.

[13] F. Pachet, “Surprising harmonies.”, *International Journal of Computing Anticipatory Systems* 4, 1999.

[14] F. Pachet, P. Roy, “Non-conformant harmonization: the real book in the style of take 6.” In *International Conference on Computational Creativity*, Ljubljiana, 2014.

[15] F. Pachet, J. Suzda, D. Martín, “A Comprehensive Online Database of Machine-Readable Lead-Sheets for Jazz Standards”. In *ISMIR*, 2013, pp. 275-280.

[16] P. Roland, The music encoding initiative (mei). In *Proceedings of the First International Conference on Musical Applications Using*, Vol. 1060, 2002, pp. 55-59.

[17] I. Stravinsky, R. Craft, *Dialogues*. London: Faber and Faber, 1982.

[18] M. Solomon, D. Fober, Y. Orlarey, S. Letz, “Providing Music Notation Services over Internet”. In *Proceedings of the Linux Audio Conference*, 2014.

[19] M. Yee-King, M. d'Inverno, P. Noriega, “Social machines for education driven by feedback agents”, in *Proceedings First International Workshop on the Multiagent Foundations of Social Computing, AAMAS-2014*, Paris, 2014.

[20] M. Yee-King, M. d'Inverno, “Pedagogical agents for social music learning in Crowd-based Socio-Cognitive Systems”, in *Proceedings First International Workshop on the Multiagent Foundations of Social Computing, AAMAS-2014*, Paris, 2014.

[21] N. Zakas, Scalable “Javascript Application Architecture”.

Slides: `http://cern.ch/go/Cl6S`.

BIGRAM EDITOR: A SCORE EDITOR FOR THE BIGRAM NOTATION

Andres Perez-Lopez
contact@andresperezlopez.com

Jose M. Alcantara
pilaweto@gmail.com

Bertrand Kientz
bertrand@xipmultimedia.com

ABSTRACT

The Bigram Notation is an alternative approach to musical notation, based on the chromatic nature of Western music. As observed historically with alternative notation systems, their spread and consolidation is based on the existence of complementary and supportive tools, as ideosyncratic instruments and specific written material. Accordingly, we present the binary keyboards and the Bigram Editor, a graphical bigram score editor with automatic transcription and reproduction capabilities.

1. INTRODUCTION

It is commonly accepted that the conventional music notation system has its origin in the 11th century with the *tetragram* from Guido d'Arezzo. Since then, it has been evolving and adapting itself along with evolution of musical language [1], until conforming its modern version.

However, conventional music notation presents a number of systematic problems [1]; for instance:

- Pitch distances are not equally distributed along the vertical axis.
- Octave equivalence is not usually present in notation
- The use of accidentals might lead to a variety of signs for representing the same sound (enharmony)

In addition, conventional notation takes as a reference the C Major scale. Consequently, writing music *far* from the C Major diatonic scale might lead to understandability reduction. Figure 1 shows an excerpt from Franz Liszt's "Hungarian Rhapsody No.2", in F# Major (extracted from [2]). F# Major is the farthest diatonic scale from C Major (they only share two notes), and furthermore the passage has numerous accidentals.

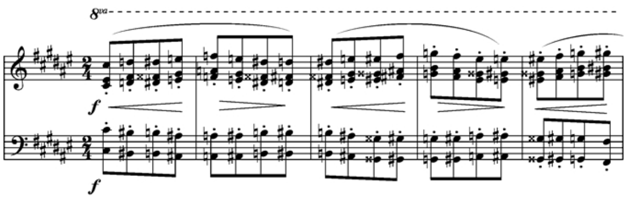


Figure 1. Score excerpt from Franz Liszt's "Hungarian Rhapsody 2"

In order to reduce the aforementioned problems, a large number of alternative notation systems has been proposed. Thomas Reed [3] gathers more than 500 different notations, being the earliest of them (by H. Richter) first documented in 1847. Reed also founded the *Music Notation Modernization Association* (MNMA) in 1985, which was the predecessor of the present *The Music Notation Project* (MNP), founded in 2008. The MNP's mission is "To raise awareness of the disadvantages of traditional music notation, to explore alternative music notation systems, and to provide resources for the wider consideration and use of these alternatives" [4].

The Music Notation Project has even presented a set of design criteria for new notation developments [5], based on the evaluation considerations of a notation comparison performed by the MNMA. The seventeen criteria emphasize the importance of concepts such as ease of writability and readability, flexibility, pitch-distance and time-distance proportionality, or octave periodicity.

However, none of those systems have been widely accepted. Parncutt proposes several explanations for that fact, highlighting the lack of a big score collection as one of the biggest potential handicaps [1, 6].

Therefore, we present a new music notation environment, called the *Bigram*, which is currently under active development. Despite its resemblance with other existing notation systems, as we will present in Section 2.2, the main strength of our proposal lies on the fact that it tries to avoid the aforementioned handicaps (lack of written material). Accordingly, the Bigram environment is divided into three main areas:

- *Bigram Notation*, a state-of-the-art notation system which meets the MNP criteria

- *Binary Keyboards*, layout-modified keyboards with high resemblance to the Bigram Notation
- *Bigram Editor*, a graphical software score editor with automatic transcription and reproduction capabilities.

Those three areas will be discussed in detail in the following Sections 2, 3 and 4, respectively.

2. BIGRAM NOTATION

2.1 Notation vs. Tablature

Traditional keyboard layout and conventional notation system share the inner structure of white keys - non-accidental notes (and, of course, *full considered* note names); therefore, conventional notation might be considered a special interpretation of keyboard tablature.

Parncutt [1] introduces the idea that, for beginners, tablature notation might be the most appropriate, due to its easiness. However, experimented interpreters might prefer conventional notation, for its resemblance with our bidimensional perception of pitch and time.

This fact gives us the opportunity to explore a new approach to musical notation. What if we could design a notation that could resemble clearly the pitch-time graph, but at the same time be an explicit representation of the finger positions in the keyboard? Such a system would be, according to Parncutt, convenient for both beginner and expert musicians, and would provide a faster learning process.

In order to reach that goal, a convenient keyboard layout should be designed. This keyboard will be discussed in Section 3.

2.2 Bigram

As a consequence of the previous idea, we developed the *Bigram Notation*. It takes its name from the fact that, in the staff, each octave presents only two equidistant lines, separated a tritone. Consequently, we preserve the octave periodicity, and minimize the cognitive overhead of counting lines to identify the note (both desired criteria from [5]).

Figures 2 and 3 show the A Major scale and the chromatic scale, respectively, written in bigram notation.

Figure 4 shows the same excerpt from Figure 1 in bigram notation.

2.2.1 Pitch representation

One of the most predominant characteristics of the bigram notation is the pitch representation by black and white noteheads. The A note was (arbitrarily) chosen to be represented over the first line, and to be black. When ascending in the chromatic scale, each new note presents a different color, alternating white and black noteheads (as in Figure

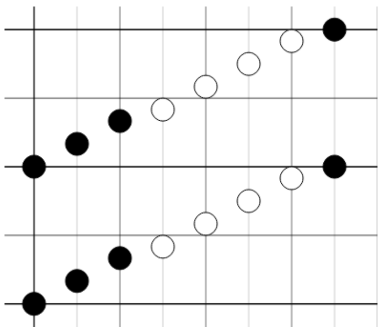


Figure 2. Bigram notation. The A Major scale

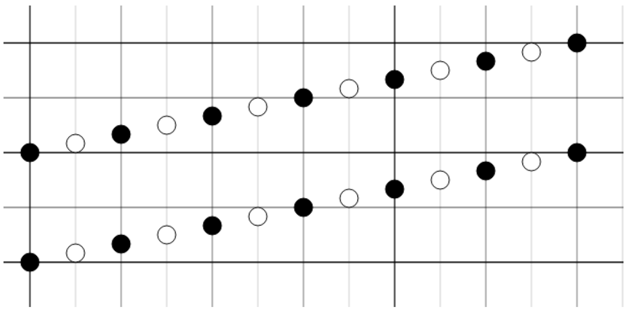


Figure 3. Bigram notation. The chromatic scale starting on A

3).
This approach causes the intervals to be color-consistent, making very explicit the inner structure of melodies and harmonies, and emphasizing intervalic reading [7]. In addition, it reduces the amount of required staff lines, facilitating note identification and minimizing cognitive overhead.

Notice that, in Figure 2, the semitone structure of the Major scale become self-evident. Furthermore, the Listz's excerpt (Figure 4) clearly reveals its structure: symmetric parallel chromatic movements, maintaining the voice's intervalic relationships.

The bigram pitch structure itself can be seen therefore as a combination of 6-6 black & white notehead systems (such as *Isomorph Notation* by Tadeusz Wójcik or *6-6 Klavar* by Cornelis Pot), with systems with staff lines separated a tritone (*MUTO Notation* by MUTO Foundation or *Express Stave* by John Keller, 2005) [3].

2.2.2 Rhythm representation

Regarding the rhythmic notation, we opted for a representation that preserves the time-distance proportionality, as suggested in the MNP criteria [5]. As in conventional notation, time is divided into bars. Each bar has a number of pulses, which have a number of divisions. Bars, pulses and divisions are represented by vertical lines, whose width is proportional to their position in the time hierarchy.

As an example, the scale in Figure 2 occupies one whole bar, with four pulses and two pulse divisions. The notes are placed in each one of the 8 bar divisions.

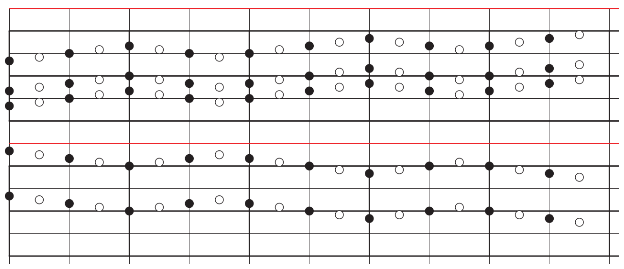


Figure 4. Bigram notation example from Franz Liszt's "Hungarian Rhapsody 2"

The notes are placed in the space that proportionally corresponds to a given pulse or division. When a irregular subdivision of pulse or division occurs, a number within a bracket or slur is used to indicate the transient subdivision.

Although notes are expected by default to last until next note, silence signs are also available. For other articulations, conventional signs are used.

2.2.3 Other Considerations

The bigram system fulfills each one of the seventeen design criteria for notation design established by the MNP. We must highlight that, although its development is subject to continuous evaluation, the potential changes that might occur will not change radically the basic ideas exposed here.

Regarding further extensions of the concept, the authors are investigating a compact and adequate way of representing harmony within the bigram context. Due to the interest of the authors on jazz, the research is focused on the most common 4-note chords and its variations.

3. MODIFIED INSTRUMENTS

3.1 Binary Keyboards

As already mentioned in Section 2.1, one of the strengths of the bigram notation is that it relies on the existence of keyboards with high resemblance to the written notation. With such instruments, it would be even possible to play a bigram notation score without knowing which notes are being represented (even though this practice is not recommended).

The authors are investigating on the prototype and fabrication of such keyboards, which are referred as *binary keyboards*. Figures 5 and 6 show two current working prototypes: a MIDI controller and a melodica, respectively. We believe that, even if the binary keyboard layout differs completely from standard layout, conventional piano playing techniques might be applied to binary keyboards, since both layouts share the two-rows key disposition.

The A notes are presented in the keyboards with a different color. This fact mimics the bigram notation, in which

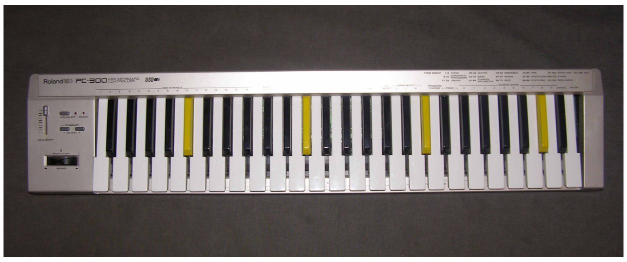


Figure 5. Binary MIDI keyboard prototype

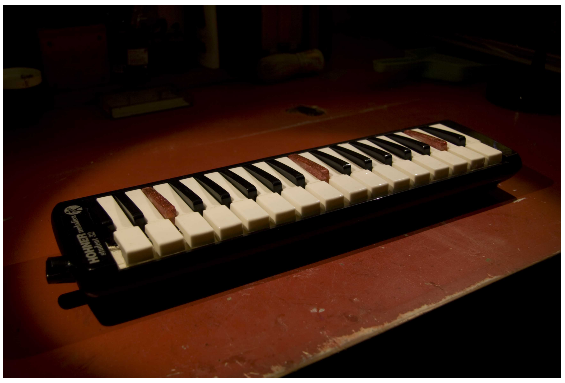


Figure 6. Binary melodica prototype

the A notes are situated over the main staff line, and therefore used as a reference.

The authors are currently investigating the appropriateness of introducing tactile feedback cues, such as using different material or introducing marks. The tritone note (D#), which occupies the central line in the staff, might also present a distinction.

Those tactile feedback cues might be helpful both for visually impaired people, and for experienced players, which might need to know their hands position without looking to the keyboard (as experienced conventional piano players usually do using the cues of black keys' absence).

From the first insight into the binary keyboard layout, it is possible to become aware of one of its main benefits. Since it is *isomorphic*, there only exists two different positions for playing any passage - starting on a white key, or starting on a black key. This fact highly contrasts with the 12 potentially different positions in conventional layouts.

3.2 Similar approaches

The presented binary keyboard layout is not a new concept; first references to the idea appeared in 1859. In his book [8], K. B. Schumann presented his binary keyboard proposal, in a chapter called *"Das natürliche Sytem"* ("The natural system"). He also described there an alternative notation system based on a chromatic approach. In the same year, A. Gould and C. Marsh patented the binary keyboard in the USA [9], with the name "Keyboard for Pianos".

Bart Willemse gathers in his website [10] some other historic binary keyboard proposals, which he calls "Balanced Keyboards".

Another relevant approach can be found in 1882 in the *Janko keyboard* [11], which featured several rows of isomorphic keys. Among others, it did not succeeded commercially because of the lack of written material, due to the reticence of publishers (motivated in turn by the musicians' reticence) [11, 12].

The *Chromatone* [13] is a modern, digital revision of the Janko keyboard.

The *Tri-Chromatic Keyboard Layout* [14] is a layout designed by R. Pertchik, and implemented in his vibraphone. The layout is identical to the binary keyboard, excepting for the colors. Three different alternate colors are present, highlighting the minor third intervals (and, consequently, the three diminished chords).

We must also mention the Dodeka approach [15]. As in our research, Dodeka presents a notation system together with a modified keyboard. The notation system follows a regular pitch-space configuration, with 3 lines per octave. The keyboard is a representation of the notation system, with colour references each major third. However, all keys are placed in a single row, which might complicate playability and standard keyboard techniques adoption.

3.3 Conventional instruments

Despite the close resemblance of bigram notation with binary keyboard, the notation is potentially suitable to all kind of conventional instruments. Isomorphic instruments, such as orchestral strings, might appear beforehand as the most accessible instruments for bigram notation, due to their intrinsic representation of pitch and intervals. However, any other instrument might be potentially capable of performing bigram scores, if the relationship between notation and instrument notes is known.

4. THE BIGRAM EDITOR

As already commented, one of the major problems that alternative notation systems and keyboard layouts faced historically for their widespread adoption was the lack of a convenient score collection. For that reason, we decided to implement a bigram notation sofware, which could both serve as a score editor, and as a automatic transcriptor. We named that tool the *Bigram Editor*.

4.1 Implementation

4.1.1 Existing software for alternative notation

The MNP provides references of music edition software which supports alternate notations [16]. Two applications are shown as potentially compatible with alternative notations: *Finale* and *LilyPond*.

Finale [17] is a well known score editor. The MNP explains the method created by John Keller to convert between notation systems [16], by using *staff templates*. Therefore, it would be possible to create a bigram template, which might have a very low developing cost, and use it for our purpose.

However, in our opinion, Finale has some drawbacks. The most important of them is that it is proprietary software. We believe that a project such as the Bigram Editor, constantly evolving and with a high educational value, should be freely available and customizable - in other words, free software. Finale's platform dependency is also a disadvantage. Furthermore, its price ($600, $350 for students) makes it potentially prohibitive.

The other proposed alternative is LilyPond [18]. It is an original, WYSIWYM approach to score edition. Lilypond is highly flexible, and thus it is possible to define the score's appearance, allowing the usage of alternative notations. In addition, it is a muliplatform, free software editor.

Nevertheless, the text-based approach to score edition of Lilypond might represent a big usability problem for those not used to code or WYSIWYM interfaces. The Bigram Editor should encourage users to create music as soon as possible, minimizing the time spent on learning how to use the software.

4.1.2 Design considerations

Therefore, we opted for implementing our own custom Bigram Editor. Despite the increase in work load, the decision gave us the opportunity to fully adapt the software to our needs. The established design criteria were the following:

- WYSIWYG paradigm metaphor for creation and edition of scores, in order to facilitate its usage
- MIDI *import* functionality for automatic transcription of existing music
- Accordingly, MIDI *export* functionality for facilitating score exchange between different notations and applications
- Score reproduction
- Multiplatform and open source

We decided to implement our system with *SuperCollider* [19]. SuperCollider is an environment and programming language for real time audio synthesis and algorithmic composition [20]. Among others, it provides inbuilt GUI management functionalities, and MIDI in/out and parsing features. Furthermore, it is free software and platform-independent.

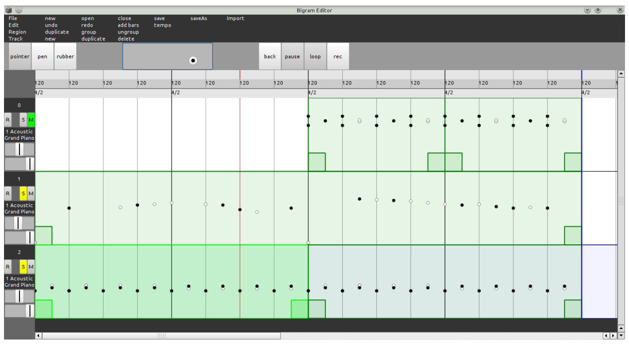


Figure 7. Bigram Editor : arrangement view

Although still in beta version, the Bigram Editor is already available at its code repository [21].

4.2 Features

The main interaction window is called the *Arrangement View* (see Figure 7) . It provides a general overview of the score in a multi-track sequencer style. Users can access from here to all available functionalities.

4.2.1 Tracks and regions

The musical material is organized into tracks or voices. Through the menus, the user can create, duplicate or delete tracks. For each track, following controls are provided:

- Track ID number
- Record/solo/mute controls
- MIDI instrument selector
- Panning and volume controls

Inside each track, users might place *regions*. A region is the structural element containing the notes. Three different tools are available for region managing:

Pointer Select a region and open the *Edit View*.

Pen Create a new region

Rubber Delete a region

Furthermore, it is possible to move, duplicate, merge and ungroup regions, through the mouse actions and/or the menus.

The *Edit View* (Figure 8) provides access to edit the music material. Users can insert, delete, duplicate or move notes using the *Input (I)* and *Edit (E)* controls. A binary keyboard reference is shown at the left margin of the score, along with the octave number.

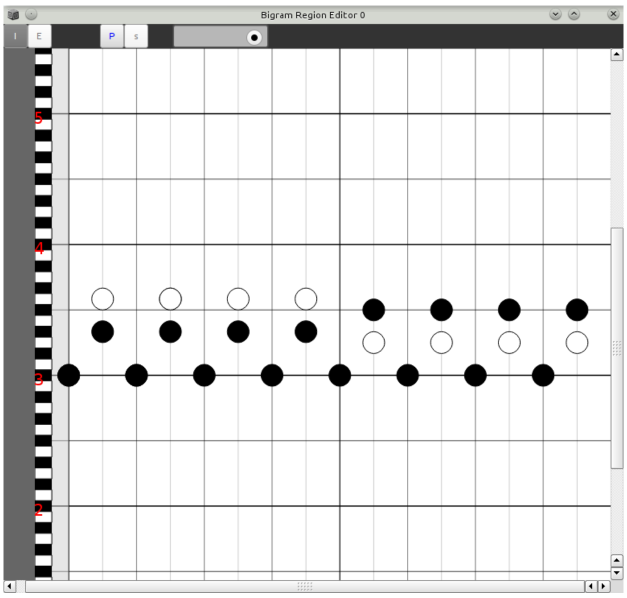


Figure 8. Bigram Editor : edit view

4.2.2 Reproduction

The *Arrangement Window* provides play/stop and loop reproduction controls; these are managed by the *reproduction bar* and the *loop bar* (vertical red and blue lines in Figure 7, respectively).

Sound is not synthesized by SuperCollider. Instead of that, the score is translated to MIDI and streamed in real-time to a MIDI synthesizer, which is platform-dependent. Currently, the system is using FluidSynth [22] for Linux, and default internal synthesizers for Windows and OSX.

4.2.3 File managing

The Bigram Editor provides file save and load functions. The score state is translated into a simple and custom description file based in XML. These files are generated automatically in the temporary folder every time a change in the score occurs; the undo/redo functions are built upon this functionality.

Furthermore, it is possible to import multi-track MIDI files from the menu in the *Arrangement Window*.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the basis of the Bigram Notation, and the holistic approach to our alternative notation considering the notation theory itself, the modified keyboards, and the score editor.

Several experiments might be run in order to assess the usability of the Bigram Editor, in terms of Human-Computer Interaction. However, its usefulness is provided by the fact that it is currently the only available score editor for the bigram notation.

The authors have received good preliminary qualitative impressions from individual users that already started studying with the bigram system, using the software and the binary keyboards. Those impressions were specially remarkable in the case of people with few or very limited previous musical background or keyboard skills. We must remark that due to the current limited availability of binary keyboards, these test experiences cannot still be carried in a regular basis.

In the near future, an experimental case-study is planned, in order to evaluate the learning curve and the acquisition of musical skills in beginners, using the the bigram notation. That experiment would be a variant of the Parncutt's proposal [1], which has never been carried out. Such experiment would consist of two control groups of musical untrained subjects learning piano, one using conventional keyboard and notation, and the other using bigram notation and binary keyboards. The subjects' acquired musical knowledge (in terms still to be defined) would be evaluated over a broad enough period.

Regarding the Bigram Editor, a number of improvements might be implemented. One of the most relevant features would be the possibility of editing and exporting the score in a graphical format. That feature might allow to obtain high-quality scores in a printable version, for its usage without the computer.

Another potential improvement might be the adoption of the MusicXML markup language [23] for the description files. MusicXML is used by most of the score editors and Digital Audio Workstations; therefore, its adoption might widen considerably the range of available compositions for the bigram notation, and the score exchange possibilities.

6. REFERENCES

[1] R. Parncutt, "Systematic evaluation of the psychological effectiveness of non-conventional notations and keyboard tablatures". In Zannos, I. (Ed.), *Music and Signs* (pp. 146-174). Bratislava, Slovakia: ASCO Art & Science. 1999

[2] The Music Notation Project, "Chromatic Staves Example" [online], http://musicnotation.org/tutorials/chromatic-staves-example/ (Accessed: January 2015)

[3] T. S. Reed, "Directory of music notation proposals". Notation Research Press. 1997

[4] The Music Notation Project, "Introducing The Music Notation Project" [online], http://musicnotation.org/blog/2008/01/introducing-the-music-notation-project/ (Accessed: January 2015)

[5] The Music Notation Project, "Desirable Criteria for Alternative Music Notation Systems" [online], http://musicnotation.org/systems/criteria/ (Accessed: January 2015)

[6] R. Parncutt, "Psychological testing of alternative music notations" (Research project ideas for students of systematic musicology and music psychology), [online] 2014, http://www.uni-graz.at/~parncutt/fk5_projekte.html#Psychological_testing_of_alternative (Accessed: January 2015).

[7] The Music Notation Project, "Intervals in 6-6 Music Notation Systems" [online], http://musicnotation.org/tutorials/intervals-in-6-6-music-notation-systems/ (Accessed: January 2015)

[8] K. B. Schumann, "Vorschläge zu einer gründlichen Reform in der Musik durch Einführung eines höchst einfachen und naturgemässen Ton-und Noten-Systems, nebst Beschreibung einer nach diesem System construirten Tastatur für das Fortepiano". 1859. http://hdl.handle.net/1802/15314

[9] A. Gould, "Arrangement of keyboard for pianos" [patent], US Patent 24, 021. 1859. http://www.google.com/patents/US24021 (Accessed: January 2015).

[10] B. Willemse, "People and resources relating to the Balanced keyboard" [online], 2013 http://balanced-keyboard.com/PeopleAndResources.aspx

[11] A. Dolge, "Pianos and Their Makers" [book], Covina/Dover. pp. 7883. ISBN 0-486-22856-8. 1911

[12] K. K. Naragon, "The Jankó Keyboard" [PhD Thesis], M. West Virginia University. 1977

[13] tokyo yusyo inc, "Chromatone" [online], 2014 http://chromatone.jp (Accessed: January 2015).

[14] The Music Notation Project, "Tri-Chromatic Keyboard Layout" [online], http://musicnotation.org/wiki/instruments/tri-chromatic-keyboard-layout/ (Accessed: January 2015).

[15] crea-7, "Dodeka" [online], http://www.dodeka.info (Accessed: January 2015).

[16] The Music Notation Project, "Software" [online], http://musicnotation.org/software/#ftn1 (Accessed: January 2015)

[17] MakeMusic, Inc., ”Finale” [online], 2015, `http://www.finalemusic.org` (Accessed: January 2015)

[18] H. W. Nienhuys & J. Nieuwenhuizen, ”LilyPond, a system for automated music engraving”. In Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003) (pp. 167-172). May 2003

[19] J. McCartney, ”Rethinking the computer music language: SuperCollider”. Computer Music Journal, 26(4), 61-68. 2002

[20] SuperCollider, `http://supercollider.sourceforge.net/` (Accessed: January 2015)

[21] A. Perez-Lopez, ”Bigram” [online], 2014, `https://github.com/andresperezlopez/Bigram` (Accessed: January 2015)

[22] FluidSynth, ”FluidSynth” [online], 2015, `http://www.fluidsynth.org/` (Accessed: January 2015)

[23] M.Good, ”MusicXML for notation and analysis.” The virtual score: representation, retrieval, restoration 12: 113-124. 2001

EXPRESSIVE QUANTIZATION OF COMPLEX RHYTHMIC STRUCTURES FOR AUTOMATIC MUSIC TRANSCRIPTION

Mauricio Rodriguez
Superior Conservatory of Castile and Leon
marod@ccrma.stanford.edu

ABSTRACT

Two quantization models for 'expressive' rendering of complex rhythmic patterns are discussed. A multi-nesting quantizer captures expressivity by allowing fine-grained/high-quality resolution, thus covering the automatic transcription of a wide range of rhythmic configurations, yielding from simple to rather complex music notations. A look-up table quantizer is discussed as another model to attain expressivity and musical consistency; input is quantized by comparison of 'rhythmic similarity' from a user-defined data-set or look-up 'dictionary'.

Both quantizers are presented as computing assisting tools to facilitate the transcription of rhythmic structures into the symbolic domain (i.e. music notation).

Keywords: Computer Assisted Composition, Rhythmic Quantization, One-Level Quantizer, Multi-Level or Multi-Nesting Quantizer, Look-Up Table Quantizer, Accumulative or Sequential Quantizer.

1. INTRODUCTION

The process of quantizing numeric series representing onset/durations of rhythmic patterns, is a facility commonly employed in computer assisted composition environments to ease the rendering of input data into symbolic music representation (i.e. music notation). Robust algorithms for rhythmic quantization not only aim to produce an acceptable form of music notation, but to 'interpret' the input data in a categorical form while producing 'intelligent' and interesting musical results. An expressive quantizer model should take into consideration the former aspects, namely, it should allow for a logical representation to capture the way music material is structured while preserving a rather accurate and readable notational rendering of the input data, that ultimately, depicts the composer's notational intentions. Neither of these two features are easily achieved through the use of computing models when it comes to the problem of rhythmic quantization. There have been however, interesting quantizing models to categorize input data either taken from performance situations or generated through algorithmic processes. The Connectionist [1], the Bayesian [2] and the 'Kant-Quantizer' [3], are some of those effective quantizing models that logically "filter-out" and provide "structuring" of the input data to ease the rendering process of an automated music notation. Even though the logical data parsings of some of those aforementioned quantizers allow for effective music transcriptions, their notated results are generally circumscribed to rhythmic notations of a fairly conventional fashion. For instance, the connectionist quantizer is a model that operates through a *cell-network* system where an initial onset/duration set interacts with *activation cells* to gradually converge to an equilibrium state. The equilibrium state seeks for simple-ratios between adjacent durations of the initial values from the onset/duration sequence; if no simple tuplet-ratios are found, the activation mechanism adjusts the original sequence until its values are rounded to evenly complete a subdivided beat. Within this mechanism, an irregular tuplet (if found) must follow an equivalent rhythmic figure until the full subdivided beat is completed, voiding the case of sequentially having, for instance, one irregular tuplet after another irregular tuplet of a different species, which is the core principle of a multi-nested rhythmic notation.

Most professional "general-purpose" quantizers, such as the "omquantify" (Open Music), the "gquantify" (PatchWork & PWGL) and the "ksquant" (PWGL) among others, treat their input data as linear arrays of numbers where durations follow to the next ones without determining any relational scheme between them. This principle, instead of being a disadvantage, permits to generalize and apply the quantization process to the most varied sets of data, from arbitrarily defined inputs to different algorithmic-generated numeric values. As a previous step for final notation output, the user of these quantizers can calibrate some quantization parameters, thus facilitating a more personal notational result; however, the limitations of "general-purpose" quantizers lacking the supervising of "logical/intelligent" algorithms or user input inspection, are evident when, for instance, trying to quantize a simple rhythmic pattern of a *ritardando* figure, whose notational result would be most likely quantized with lots of tied irregular tuplets, without this resulting quantization necessarily showing the simple gesture-figure of a deceleration.

A general-purpose multi-nesting quantizer would not necessarily overcome the limitations shared by previous "non-logical" quantizers, however, it is claimed that a refined level of musical expressivity is achieved when the problem of quantization is generalized to capture and render complex rhythmic structures such as those present in multi-nested (fine-grained) rhythmic patterns.

2. XA-LAN MULTI-LEVEL QUANTIZER

Xa-Lan is a LISP-based software to generate expressive music scores based on graphic transformations that control the different symbolic/notational elements of a music score [4]. Xa-Lan relies on three modules (or engines) to produce output. The third of these modules is a multi-level or multi-nesting quantizer that allows to transcribe the onsets and durations of rhythmic patterns into accurate, complex, and expressive rhythmic notations.

The multi-nesting quantizer is a recursive adaptation of a simple one-level quantizer [5]. The algorithm of the one-level quantizer compares equal subdivisions of a given temporal segment to the original onset-duration sequence, searching for minimal-error differences. Since larger number of subdivisions reduce error-difference values, the minimal-error curve is compared to an ideal fitting curve whose maximum difference is chosen as the earliest optimal quantization (Figure 1).

The multi-level quantizer in Xa-Lan recursively applies the former quantizing process to different portions of a temporal segment, from the measure level to the beat and the beat subdivision levels, therefore, yielding nested rhythmic figures when necessary. The Xa-Lan multi-nesting quantizer is unique in its functionality. No other quantizer available in common computer-assisted composition environments, such as PatchWork, OpenMusic, Symbolic Composer, or PWGL (to name some), can render automated nested rhythmic figures to allow a rather refined quantizing resolution. The Xa-Lan quantizer can also be aligned to any user-defined metrics-grid that works as a “structural container” of the quantization. The maximum subdivision for irregular tuplets at any nesting level, can also be arbitrarily set by the user, allowing for different notational resolutions of the same input.

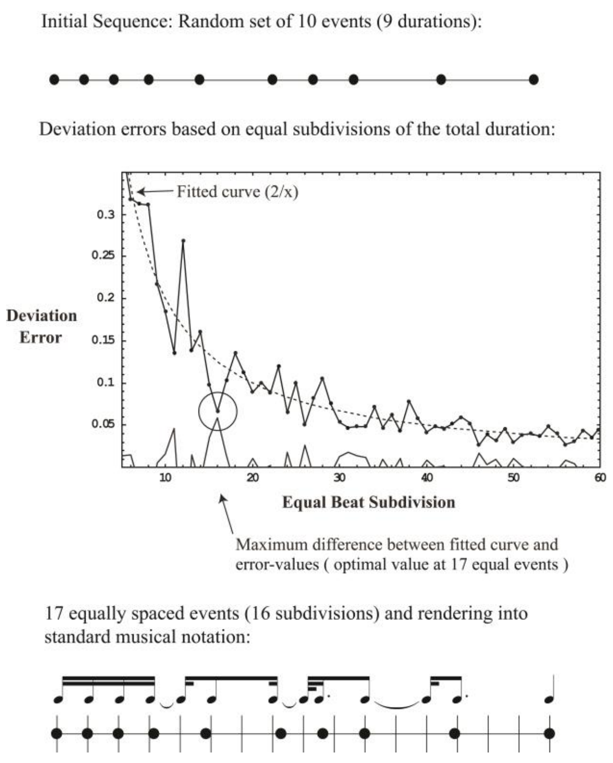


Figure 1. One-Level Quantizer.

To dynamically interact with the facilities of the multi-level quantizer, Xa-Lan uses the “Expressive Notation Package” (ENP) from the visual language of PWGL [6] for final display (Figure 2).

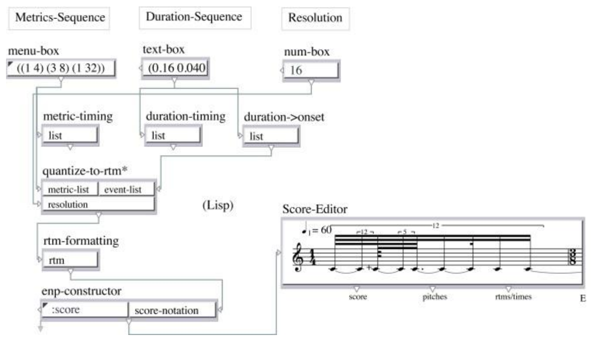


Figure 2. Multi-Level Quantizer interface on ENP.

To observe some of the possible results that can be obtained by this mutli-nesting quantizer, consider the following duration sequence: 0.16 0.3 0.25 0.040000007 0.58000005 0.37 0.45000005 0.29999995 0.17499995. This array of numbers will be quantized using the following arbitrary metric container: 1/4, 3/8, and 1/32. The maximum number of subdivisions per nesting level is also arbitrarily set to 12 (Figure 3):

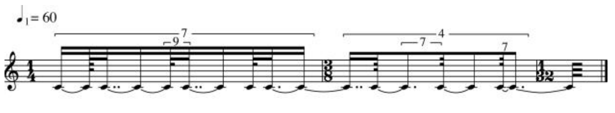


Figure 3. Multi-Level Quantization I.

In the following figure, the resolution subdivision is downsampled by half (to 6) of the previous quantization (Figure 4) :

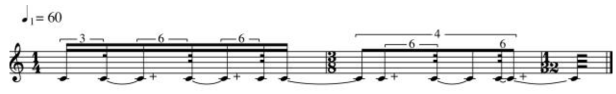


Figure 4. Multi-Level Quantization II.

Lastly, the same array of durations are quantized with a different metric container (6/32, 1/4, and 7/32) and the maximum number of subdivisions per nesting level is 12 again (Figure 5) :

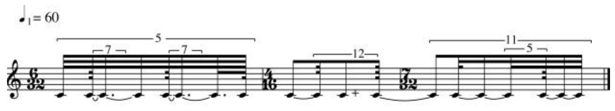


Figure 5. Multi-Level Quantization III.

3. LOOK-UP TABLE QUANTIZER

Another quantization model that attempts to capture complex rhythmic structures and render them into a meaningful and expressive musical context, is through the use of a look-up table quantizer. In a look-up quantizer, a 'dictionary' with predefined rhythmic-patterns uses each of its entries as place-holders where input onset/durations sequences are sieved-on, choosing the place-holders or "rhythmic grids" with minimal error-difference as the best quantized approximations. The real advantage of a look-up table quantizer is that the user can define a unique and precise dictionary of rhythmic configurations from where quantization takes place, ensuring an idiomatic behavior that easily conforms to a compositional system of temporal organization.

The internal workings of the look-up quantizer are similar to the ones of the multi-level quantizer, except that the searching space to compare and get the optimal error- difference is manually introduced by the user, instead of being algorithmically generated. Once the user includes a new "rhythmic word" in the dictionary, by using a symbolic "rhythmic-tree" representation, the first task for the look-up algorithm is to convert any "word" into its equivalent timing equivalent (e.g. (1 (1 1 (2 (1 1 1))) is equivalent to 0.25, 0.25, 0.5/3, 0.5/3 and 0.5/3 seconds, assuming a quarter-note is equal to one second). From there, the comparison of the original time-input sequence with the time-converted "words" is straight. The next step is to do the proper rhythmic configuration groupings of the "word" that is chosen as optimal quantization. If for instance, the original time input is 0.25, 0.58 and 0.17 seconds, the place-holder word of (1 (1 1 (2 (1 1 1))) would be output as (1 (1 1 (2 (2.0 1))), being this result the best quantization among the given words of that dictionary.

The idea of a user-predefined rhythmic dictionary might appear burdensome at first, but this quantizing model is essentially as effective as any other general purpose quantizer, with the invaluable advantage of rendering rhythmic results that fully conform to a precise selection of rhythmic configurations that are previously input by its users, and therefore, the expressivity of the resulting transcriptions completely accommodate to the idiomatic and aesthetic needs of composers.

```
(defvar *rhythm-reservoir*
'(                            (1 (5 (1 1)) 1 1)
(1)                           (1 (5 (1 1 1)) 1 1)
(1 1)                         (1 (5 (1 1 1 1)) 1 1)
(1 1 1)                       (1 1 (5 (1 1)) 1)
(1 1 1 1)                     (1 1 (5 (1 1 1)) 1)
(1 1 1 1 1)                   (1 1 (5 (1 1 1 1)) 1)
(1 1 1 1 1 1)                 ((3 (1 1)) (5 (1 1)))
(1 1 1 1 1 1 1)               ((3 (1 1)) (5 (1 1 1)))
(1 1 1 1 1 1 1 1)             ((3 (1 1)) (5 (1 1 1 1)))
(1 (7 (1 1)))                 ((5 (1 1)) (3 (1 1)))
(1 (7 (1 1 1)))               ((5 (1 1 1)) (3 (1 1)))
(1 (7 (1 1 1 1)))             ((5 (1 1 1 1)) (3 (1 1)))
(1 (7 (1 1 1 1 1)))           (1 1 (4 (1 1 1)) 1 1)
(1 (7 (1 1 1 1 1 1)))         (1 1 (3 (1 1)) (3 (1 1)))
((7 (1 1)) 1)                 ((3 (1 1)) (3 (1 1)) 1 1)
((7 (1 1 1)) 1)               ((3 (1 1)) 1 1 (3 (1 1)))
((7 (1 1 1 1)) 1)             (1 1 1 (3 (1 1)) 1 1)
((7 (1 1 1 1 1)) 1)           (1 1 (3 (1 1)) 1 1 1)
((7 (1 1 1 1 1 1)) 1)         (1 (3 (1 1)) 1 1 1 1)
(1 (6 (1 1 1 1)) 1)           (1 1 1 1 (3 (1 1)) 1)
(1 1 (6 (1 1 1 1)))           (1 (3 (1 1)) (4 (1 1 1)))
((6 (1 1 1 1)) 1 1)           ((4 (1 1 1)) (3 (1 1)) 1)
(1 (6 (1 1 1 1 1)) 1)         (1 (4 (1 1 1)) 1 1 1)
(1 1 (6 (1 1 1 1 1)))         (1 1 1 (4 (1 1 1)) 1)
((6 (1 1 1 1 1)) 1 1)         (1 (4 (1 1 1)) (3 (1 1)))
(1 1 1 (5 (1 1)))             ((3 (1 1)) (4 (1 1 1)) 1)
(1 1 1 (5 (1 1 1)))           ((3 (1 1)) 1 (4 (1 1 1)))
(1 1 1 (5 (1 1 1 1)))         ((4 (1 1 1)) 1 (3 (1 1)))
((5 (1 1)) 1 1 1)             ((4 (1 1 1)) 1 1 1 1)
((5 (1 1 1)) 1 1 1)           (1 1 1 1 (4 (1 1 1)))
((5 (1 1 1 1)) 1 1 1)         ))
```

Table 1. User-defined 'rhythmic-grid' dictionary

When working with the look-up table quantizer, it is important to keep in mind that varied and fine-grained quantizations can only take place if there is a comprehensively large data-set of place-holder rhythmic words in the dictionary, otherwise one or several input values in some cases could not be quantized, in which case, the output of the algorithm will indicate the number of non-quantized events. An interesting compositional strategy to use this quantizer can be forcing the quantization process to a limited set of dictionary-words, and by gradually changing, or rather expanding the searching space where quantization takes place, different resolutions of the quantization would show the kind of transcriptions that fit more naturally to the original input data. To facilitate this compositional methodology, there is an additional routine in this quantizer to compare and sort the deviation-error similarity (in an ascending to descending order) of one chosen word in relation to all the other words of the dictionary. Additionally, the similarity comparison among words can be truncated to show only the ones that present the same "rhythmic profile" from a reference word being compared; for example, the rhythmic tree (1 (3 1 2)) could be output as similar to (1 (4 1 3)) since both share the same "rhythmic profile", meaning that the first duration of the group is larger than the second, and the second being shorter than the third. The following figures show the similarity rankings from the rhythmic tree (1 (2 1 1 4)), which is indeed a grouped version of the simple rhythmic tree (1 (1 1 1 1 1 1 1 1)); first, similarity is presented regardless rhythmic profile (Figure 6), and then truncated to show words with equivalent profiles (Figure 7). The results of these comparisons are based on the following searching-space dictionary:

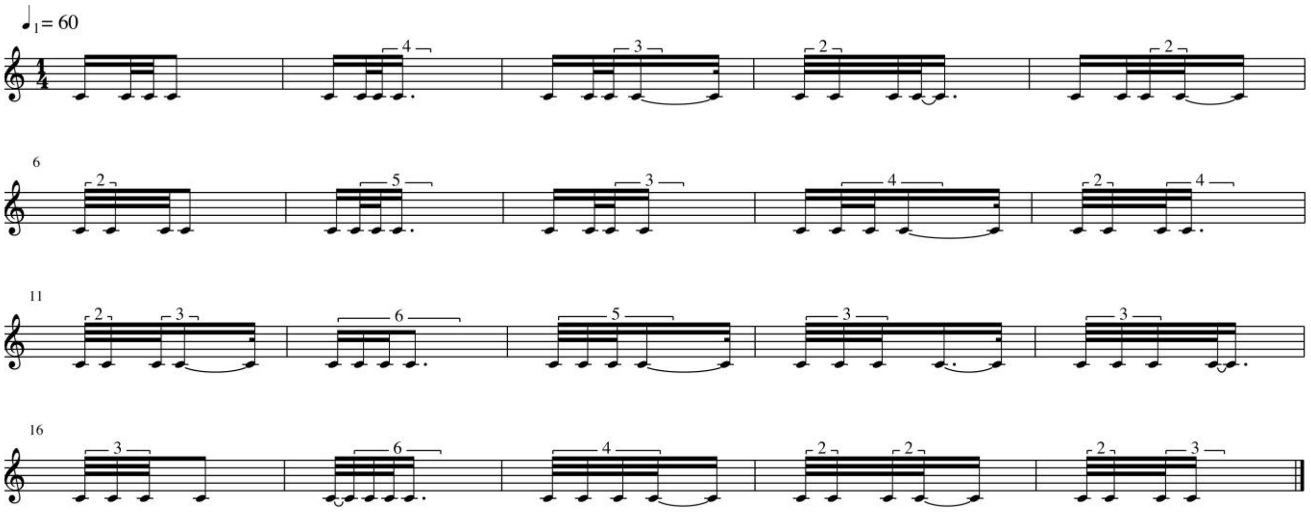


Figure 6. Rhythmic similarity

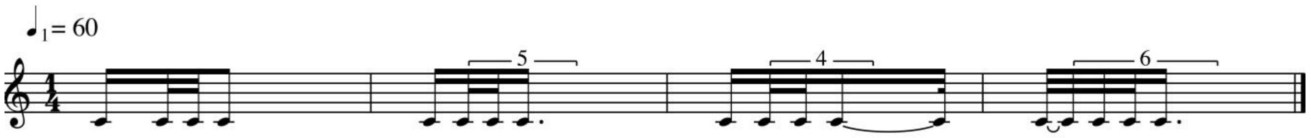


Figure 7. Rhythmic similarity with equal 'profile'.

Further implementations

In order to make more flexible and expressive the quantized results of the look-up table quantizer, a changing metrics-grid that would serve as a variable structural container, could enhance the quantization results as it happens with the metric flexibility already implemented on the multi-nesting quantizer. Another interesting feature to implement could be the automatic or algorithmic generation of additional dictionary-words or "rhythmic grids", based on the analysis of rhythmic similarities equivalent to those manually defined. Lastly, an additional feature to consider in any real expressive quantizer, is the possibility to render exceptional notation cases, as it happens when "incomplete" subdivisions of the beat are sequentially linked or concatenated, without necessarily rounding with the reminders of the previous tuplet subdivisions. The results of such an "accumulative" or "sequential quantizer" would reduce the compromising filtering of the input data that occurs during the output of a general-purpose quantizer.

4. CONCLUSIONS

Automatizing the rendering of an *expressive* rhythmic notation is a task that demands, on the one hand, a logical structuring or shaping of the input data to provide the resulting notation with musically consistent results, and on the other hand, quantization results should conform to the aesthetic and notational idiosyncrasy of a given user. Two general-purpose quantizing models have been presented to aim for notational expressivity from different perspectives. A multi-level or multi-nesting quantizer achieves 'expression' by fine-grained / high-quality resolution output, while preserving an uncompromising general-purpose applicability (i.e. no pre/post filter processes are applied to input data). A look-up table quantizer guarantees expressivity through a user-defined data-set that works as a closed searching-space from where quantization takes place. These two quantizers aim to be used as computing tools to facilitate and assist the composition *and writing* or notational rendering of music works.

5. REFERENCES

[1] P. Desain, and H. Honing, “Quantization of musical time: a connectionist approach” in *Music and Connectionism*, MIT Press Cambridge, 1991, pp. 150-167

[2] A. T. Cemgil, P. Desain, and B. Kappen, “Rhythmic Quantization for Transcription”, Computer Music Journal, vol. 2, n^o. 24, 2000, pp. 60-76.

[3] C. Agon, G. Assayag, J. Fineberg, and C. Rueda, “Kant: A critique of pure quantification”, in *Proceedings of the International Computer Music Conference*, *International Computer Music Association*, Aarhus Denmark, 1994, pp. 52–9.

[4] M. Rodriguez, “Xa-lan: Algorithmic Generation of Expressive Music Scores Based on Signal Analysis and Graphical Transformations” in *Proceedings of the International Workshop on Musical Metacreation - 8th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Stanford University, 2012, pp. 83-85.

[5] C. Saap, “Rhythmic Quantizer”, *unpublished paper*, Center for Computer Assisted Research in the Humanities (CCARH), Stanford University, 2011.

[6] M. Laurson, and M. Kuuskankare, “PWGL: A Novel Visual Language based on Common Lisp, CLOS, and OpenGL, in *Proceedings of the International Computer Music Conference*, San Francisco, 2002, pp. 142-145.

COMPUTER-AIDED MELODY NOTE TRANSCRIPTION USING THE TONY SOFTWARE: ACCURACY AND EFFICIENCY

Matthias Mauch
Queen Mary University of London

Chris Cannam
Queen Mary University of London

Rachel Bittner
New York University

George Fazekas
Queen Mary University of London

Justin Salamon
New York University

Jaijie Dai
Queen Mary University of London

Juan Bello
New York University

Simon Dixon
Queen Mary University of London

ABSTRACT

We present Tony, a software tool for the interactive annotation of melodies from monophonic audio recordings, and evaluate its usability and the accuracy of its note extraction method. The scientific study of acoustic performances of melodies, whether sung or played, requires the accurate transcription of notes and pitches. To achieve the desired transcription accuracy for a particular application, researchers manually correct results obtained by automatic methods. Tony is an interactive tool directly aimed at making this correction task efficient. It provides (a) state-of-the art algorithms for pitch and note estimation, (b) visual and auditory feedback for easy error-spotting, (c) an intelligent graphical user interface through which the user can rapidly correct estimation errors, (d) extensive export functions enabling further processing in other applications. We show that Tony's built in automatic note transcription method compares favourably with existing tools. We report how long it takes to annotate recordings on a set of 96 solo vocal recordings and study the effect of piece, the number of edits made and the annotator's increasing mastery of the software. Tony is Open Source software, with source code and compiled binaries for Windows, Mac OS X and Linux available from `https://code.soundsoftware.ac.uk/projects/tony/`.

1. INTRODUCTION

Our goal is to make the scientific annotation of melodic content, and especially the estimation of note pitches in singing, more efficient. A number of well-known digital signal processing methods have been successfully applied to measuring singing pitch precisely and unambiguously, e.g. [1,2]. While their accuracy is sufficient for many applications, arriving at a satisfactory annotation often requires significant manual adjustment on the part of the researcher. This need for adjustment is even more pronounced when the aim is to transcribe discrete notes. Performing such adjustments take much time and effort, especially in the absence of a user-friendly interface.

The main contributions of this paper are (1) the presentation of the Tony user interface aimed at streamlining the melody annotation process, (2) the new note transcription algorithm it uses (implemented in the pYIN Vamp plugin), and (3) an evaluation of Tony's utility in terms of note transcription accuracy and the effort required for note annotation in a real-world use case. Features and design described in this paper reflect Tony version 1.0 except where noted.

2. BACKGROUND

Music informatics researchers, music psychologists and anyone interested in the analysis of pitch and intonation routinely use software programs to annotate and transcribe melodies in audio recordings. The two main objects of interest are the pitch track, which traces the fundamental frequency (F0) contours of pitched sounds in smooth, continuous lines, and the note track, a sequence of discrete note events that roughly correspond to notes in a musical score. In order to find out which tools are used we conducted an online survey that was sent out through several channels including the *ISMIR Community*, *Auditory* and *music-dsp* mailing lists. [1]

We obtained 31 responses with a strong bias towards researchers in music informatics (see Table 2). Most of the participants were from academic institutions (27; 87%), of which students were the greatest contingent (11; 35%). Four participants were from industry (13%). Experience with pitch and note representations was nearly evenly distributed (58% and 52%, respectively, including those who had experience with both kinds of annotation).

We asked the participants which tools they are aware of. Responses included a large variety of tools, which we separated into user-interface-based software and signal processing software without user interfaces (see Box 1). [2]

Our first observation is that despite the wide range of tools, there are some that were mentioned many more times than others: in the case of user interfaces these are

[1] The survey questions are given in Appendix A.
[2] We are furthermore aware of tools for pitch track annotation [1] and pitch track and note annotation [2] that are not publicly available.

software	URL
Tony	`https://code.soundsoftware.ac.uk/projects/tony`
pYIN	`https://code.soundsoftware.ac.uk/projects/pyin`
Pitch Estimator	`https://code.soundsoftware.ac.uk/projects/chp`
Sonic Visualiser Libraries	`https://code.soundsoftware.ac.uk/projects/sv`

Table 1. Software availability.

Field of work		Position	
Music Inf./MIR	17 (55%)	Student	11 (35%)
Musicology	4 (13%)	Faculty Member	10 (32%)
Bioacoustics	3 (10%)	Post-doc	6 (19%)
Speech Processing	2 (5%)	Industry	4 (13%)
Experience			
Pitch track	18* (58%)		
Note track	16* (52%)		
Both	7 (23%)		
None	3 (10%)		

*) includes 7 who had experience with both pitch and note tracks.

Table 2. Participants of the survey. Top four responses for participant makeup.

The tools with graphical user interfaces mentioned by survey participants were: Sonic Visualiser (12 participants), Praat (11), Custom-built (3), Melodyne (3), Raven (and Canary) (3), Tony (3), WaveSurfer (3), Cubase (2), and the following mentioned once: AudioSculpt, Adobe Audition, Audacity, Logic, Sound Analysis Pro, Tartini and Transcribe!.
The DSP algorithms mentioned by survey participants were: YIN (5 participants), Custom-built (3), Aubio (2), and all following ones mentioned once: AMPACT, AMT, DESAM Toolbox, MELODIA, MIR Toolbox, Tartini, TuneR, SampleSumo, silbido, STRAIGHT and SWIPE.

Box 1. Survey Results.

Sonic Visualiser [3] [3] and Praat [4] [4], and in the case of DSP tools it is YIN [5]. None of the tools with user interfaces are specifically aimed at note and pitch transcription in music; some were originally aimed at the analysis of speech, e.g. Praat, others are generic music annotation tools, e.g. Sonic Visualiser and AudioSculpt [6]. In either case, the process of extracting note frequencies remains laborious and can take many times the duration of the recording. As a consequence, many researchers use a chain of multiple tools in custom setups in which some parts are automatic (e.g. using AMPACT alignment [7]), as we have previously done ourselves [8]. Commercial tools such as Melodyne, [5] Songs2See [6] and Sing&See [7] serve similar but incompatible purposes. Melodyne in particular offers a very sleek interface, but frequency estimation procedures are not public (proprietary code), notes cannot be sonified, and clear-text export of note and pitch track data is not provided.

In summary, the survey further corroborated the impression gained during our own experiments on note intonation: a tool for efficient annotation of melodies is not available, and the apparent interest in the scientific study of melody provides ample demand to create just such a tool. We therefore set out to create Tony, a tool that focusses on melodic annotation (as opposed to general audio annotation or polyphonic note annotation). The Tony tool is aimed at providing the following components: (a) state-of-the art algorithms for pitch and note estimation with high frequency resolution, (b) graphical user interface with visual and auditory feedback for easy error-spotting, (c) intelligent interactive interface for rapid correction of estimation errors, (d) extensive export functions enabling further processing in other applications. Lastly, the tool should be freely available to anyone in the research community, as it already is (see Table 1). This paper demonstrates that the remaining requirements have also been met.

Any modern tool for melody annotation from audio requires signal processing tools for pitch (or fundamental frequency, F0) estimation and note transcription. We are concerned here with estimation from monophonic audio, not with the estimation of the predominant melody from a polyphonic mixture (e.g. [9, 10]). Several solutions to the problem of F0 estimation have been proposed, including mechanical contraptions dating back as far as the early 20th century [11]. Recently, the area of speech processing has generated several methods that have considerably advanced the state of the art [4, 5, 12, 13]. Among these, the YIN fundamental frequency estimator [5] has gained popularity beyond the speech processing community, especially in the analysis of singing [14, 15] (also, see survey above). Babacan *et al.* [16] provide an overview of the performance of F0 trackers on singing, in which YIN is shown to be state of the art, and particularly effective at fine pitch recognition. More recently, our own pYIN pitch track estimator has been shown to be robust against several kinds

[3] `http://www.sonicvisualiser.org/`
[4] `http://www.fon.hum.uva.nl/praat/`
[5] `http://www.celemony.com/`
[6] `http://www.songs2see.com/`
[7] `http://www.singandsee.com/`

of degradations [17] and to be one of the most accurate pitch transcribers, especially for query-by-singing applications [18] (alongside the MELODIA pitch tracker [10]).

The transcription of melodic *notes* has received far less attention than pitch tracking—perhaps because *polyphonic* note transcription [19, 20] was deemed the more exciting research problem—but several noteworthy methods exist [2, 21, 22]. We have implemented our own note transcription method intended for use in Tony, of which a previous version has been available as part of the pYIN Vamp plugin [17]. This is the first time pYIN note transcription has been presented and evaluated in a scientific paper.

3. METHOD

Tony implements several melody estimation methods: fully automatic pitch estimation and note tracking based on pYIN [17], and custom methods for interactive re-estimation. Tony resamples any input file to a rate of 44.1 kHz (if necessary), and the signal processing methods work on overlapping frames of 2048 samples ($\approx$46 ms) with a hop size of 256 samples ($\approx$6 ms).

3.1 Pitch Estimation

We use the existing probabilistic YIN (pYIN) method [17] to extract a pitch track from monophonic audio recordings. The pYIN method is based on the YIN algorithm [5]. Conventional YIN has a single threshold parameter and produces a single pitch estimate. The first stage of pYIN calculates multiple pitch candidates with associated probabilities based on a distribution over many threshold parameter settings. In a second stage, these probabilities are used as observations in a hidden Markov model, which is then Viterbi-decoded to produce an improved pitch track. This pitch track is used in Tony, and is also the basis for the note detection algorithm described below.

3.2 Note Transcription

The note transcription method takes as an input the pYIN pitch track and outputs discrete notes on a continuous pitch scale, based on Viterbi-decoding of a second, independent hidden Markov model (HMM). Unlike other similar models, ours does not quantise the pitches to semitones, but instead allows a more fine-grained analysis. The HMM models pitches from MIDI pitch 35 (B1, $\approx$61 Hz) to MIDI pitch 85 (C♯6, $\approx$ 1109 Hz) at 3 steps per semitone, resulting in $n = 207$ distinct pitches. Following Ryynänen [21] we represent each pitch by three states representing attack, stable part and silence, respectively. The likelihood of a non-silent state emitting a pitch track frame with pitch q is modelled as a Gaussian distribution centered at the note's pitch p with a standard deviation of σ semitones, i.e.

$$P(n_p|q) = v \cdot \left(\frac{1}{z}\left[\phi_{p,\sigma}(q)\right]^{\tau}\right) \tag{1}$$

where n_p is a state modelling the MIDI pitch p, z is a normalising constant and the parameter $0 < \tau < 1$ controls how much the pitch estimate is trusted; we set $\tau = 0.1$. The probability of unvoiced states is set to $P(\text{unvoiced}|q) = (1 - v)/n$, i.e. they sum to their combined likelihood of $(1 - v)$ and $v = 0.5$ is the prior likelihood of a frame being voiced. The standard deviation σ varies depending on the state: attack states have a larger standard deviation ($\sigma = 5$ semitones) than stable parts ($\sigma = 0.9$). This models that the beginnings of notes and note transitions tend to vary more in pitch than the main, stable parts of notes.

The transition model imposes continuity and reasonable pitch transitions. Figure 1a shows a single note model, with connections to other notes. Within a note we use a 3-state left-to-right HMM consisting of Attack, Stable and Silent states. These states are characterised by high self-transition probability (0.9, 0.99 and 0.9999 for the three note states, respectively), to ensure continuity. Within a note, the only possibility other than self-transition is to progress to the next state. The last note state the Silent state, allows transitions to many different Attack states of other notes. Like the musicological model in Ryynänen and Klapuri's approach [21] we provide likelihoods for note transitions. Unlike their approach, we do not deal with notes quantised to the integer MIDI scale, and so we decided to go for a simpler heuristic that would only take into account three factors: (1) a note's pitch has to be either the same as the preceding note or at least 2/3 semitones different; (2) small pitch changes are more likely than larger ones; (3) the maximum pitch difference between two consecutive notes is 13 semitones. A part of the transition distribution to notes with nearby pitches is illustrated in Figure 1b.

3.3 Note Post-processing

We employ two post-processing steps. The first, amplitude-based onset segmentation helps separate consecutive notes (syllables) of similar pitches as follows. We calculate the root mean square (RMS, i.e. average) amplitude denoted by a_i in every frame i. In order to estimate the amplitude rise around a particular frame i we calculate the ratio of the RMS values between the frames either side

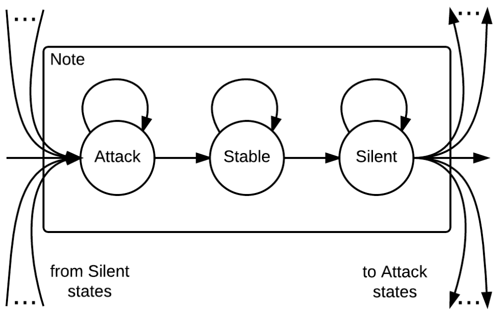
$$r = \frac{a_{i+1}}{a_{i-1}} \tag{2}$$

Given a sensitivity parameter s, any rise with $1/r < s$ is considered part of an onset,[8] and the frame $i - 2$ is set to unvoiced, thus creating a gap within any existing note. If no note is present, nothing changes, i.e. no additional notes are introduced in this onset detection step. The second post-processing step, minimum duration pruning, simply discards notes shorter than a threshold, usually chosen around 100 ms.

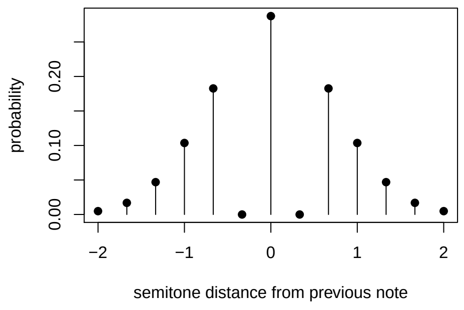
3.4 Semi-automatic Pitch Track Re-estimation

In addition to fully manual editing of notes (Section 3.4.2), the user can also change the pitch track. However, since human beings do not directly perceive pitch tracks, Tony offers pitch track candidates which users can choose from.

[8] The inverse $1/r$ is used in order for s to correspond to sensitivity.



(a) Excerpt of the pYIN note transition network.



(b) Central part of the note transition probability function.

Figure 1. Transition model in pYIN note transcription module.

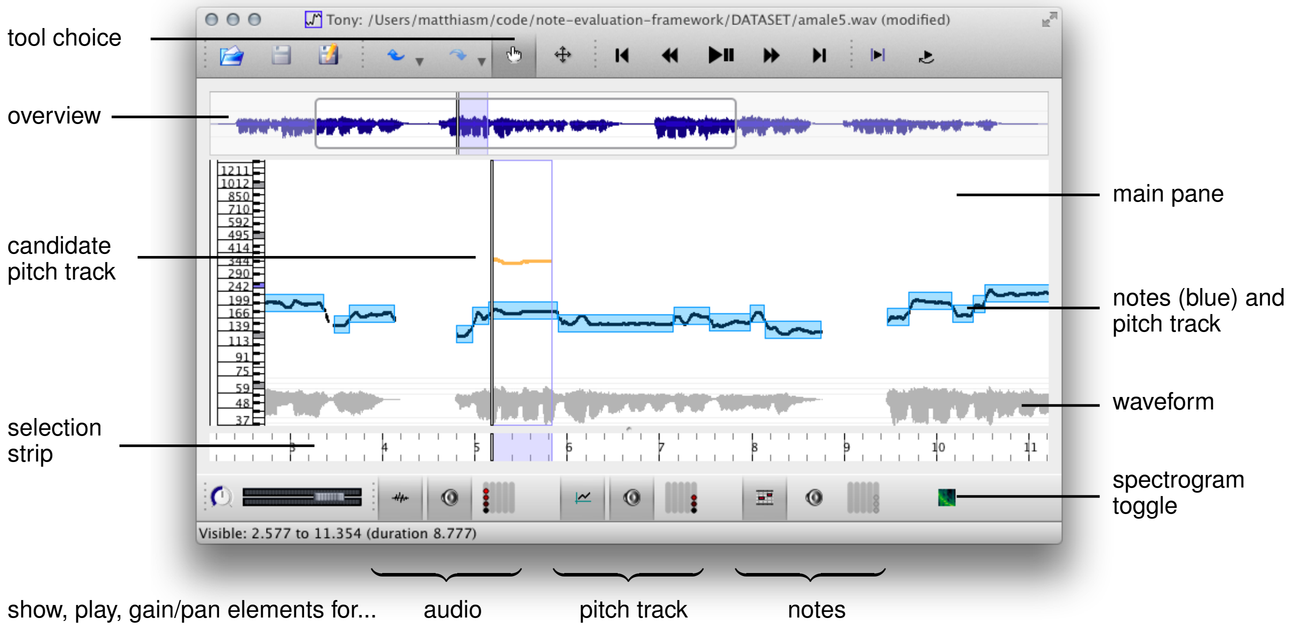

tool choice
overview
candidate pitch track
selection strip
main pane
notes (blue) and pitch track
waveform
spectrogram toggle
show, play, gain/pan elements for...
audio
pitch track
notes
Tony: /Users/matthiasm/code/note-evaluation-framework/DATASET/amale5.wav (modified)
Visible: 2.577 to 11.354 (duration 8.777)

Figure 2. Graphical User Interface with key elements highlighted.

Two methods are available: multiple alternative pYIN pitch tracks on a user-selected time interval, and a single pitch track on a user-selected time-pitch rectangle.

3.4.1 Multiple pYIN pitch tracks

In order to extract multiple pitch tracks, the pYIN method is modified such that its second stage runs multiple times with different frequency ranges emphasised. The intended use of this is to correct pitches over short time intervals. As in the default version, the first pYIN stage extracts multiple pitch candidates m_i (given in floating point MIDI pitches) for every frame, with associated probabilities p_i. Depending on the frequency range, these candidate probabilties are now weighted by a Gaussian distribution centered at $c_j = 48 + 3 \times j$, $j = 1, \ldots, 13$, for the j^{th} frequency range, i.e. the new candidate pitch probabilities are

$$p_{ij} = p_i \times \phi_{c_j,\sigma_r}(m_i), \qquad (3)$$

where $\phi(\cdot)$ is the Gaussian probability density function and $\sigma_r = 8$ is the pitch standard deviation, indicating the frequency width of the range. With these modified pitch probabilities, the Viterbi decoding is carried out as usual, leading to a total of 13 pitch tracks.

Finally, duplicate pitch tracks among those from the 13 ranges are eliminated. Two pitch tracks are classified as duplicates if at least 80% of their pitches coincide. Among each duplicate pair, the pitch track with the shorter time coverage is eliminated.

3.4.2 Pitch track in time-pitch rectangle

In some cases, the desired pitch track is not among those offered by the method described in Section 3.4.1. In such cases we use a YIN-independent method of finding pitches based on a simple harmonic product spectrum [23]. When using this method, the user provides the pitch and time range (a rectangle), and for every frame the method returns the pitch with the maximum harmonic product spectral value (or no pitch, if the maximum occurs at the upper or lower boundary of the pitch range). This way even subtle pitches can be annotated provided that they are local maxima of the harmonic product spectrum.

4. USER INTERFACE

Figure 2 is a screenshot of the Tony user interface. The basic interface components as well as the underlying audio engine and other core components are well tested as they come from the mature code base of Sonic Visualiser (see also Table 1). Tony differs from the other tools in that it is designed for musical note sequences, not general pitch events, and intentionally restricted to the annotation of single melodies. This specialisation has informed many of our design choices. Below we highlight several key aspects of the Tony interface.

4.1 Graphical Interface

While graphical interface components from Sonic Visualiser have been re-used, the focus on a single task has allowed us to combine all relevant visualisation components into a single pane: pitch track, note track, spectrogram and the waveform. Visibilty of all can be toggled. The focus on single melodies meant that we could design a special note layer with non-overlapping notes. This averts possible annotation errors from overlapping pitches.

As soon as the user opens an audio file, melodic representations of pitch track and notes are calculated using the methods described in Sections 3.1 and 3.2. This contrasts with general tools like Praat, Sonic Visualiser or AudioSculpt, which offer a range of processing options the user has to select from. This is avoided in Tony, since the analysis objective is known in advance. However, the user has some control over the analysis parameters via the menu and can re-run the analysis with the parameters changed.

Editing pitch tracks and notes is organised separately. Note edits concern only the placement and duration of notes in time, and their pitch is calculated on the fly as the median of the underlying pitch track. Any corrections in the pitch dimension are carried out via the pitch track.

In order to select pitches or notes the user selects a time interval, either via the Selection Strip or via keyboard commands. Both pitch track and note track can then be manipulated based on the selection. The most simple pitch track actions are: choose higher/lower pitch (by octave) in the selected area; remove pitches in the selected area. For more sophisticated pitch correction, the user can request alternative pitch tracks in a selected time interval (see Section 3.4.1), or the single most likely pitch track in a time-pitch rectangle (see Section 3.4.2). Note actions are: Split, Merge, Delete, Create (including "form note from selection"), and Move (boundary). The note pitch is always the median of the pitch track estimates it covers and is updated in real-time.

4.2 Sound Interface

Tony provides auditory feedback by playing back the extracted pitch track as well as the note track alongside the original audio. Like the visual pitch track and note representations, playback (including that of the original recording) can be toggled using dedicated buttons in a toolbar (see Figure 2), giving users the choice to listen to any combination of representations they wish.

Sonification of the notes is realised as a wave table playback of an electric piano sound. The sound was especially synthesised for its neutral timbre and uniform evolution. Unlike other programs, synthesis in Tony is not constrained to integer MIDI notes, and can sonify subtle pitch differences as often occur in real-world performances. The pitch track is synthesised on the fly, using a sinusoidal additive synthesis of the first three harmonic partials.

5. EVALUATION

To assess the utility of Tony as a note transcription system, we conducted two experiments. First, we compared the underlying note transcription method to existing methods, using a publicly available dataset [24]. Second, in a real-world task an expert annotated notes for an intonation study using the Tony software, and we measured the time

	Group.1	Overall. Acc.	Raw. Pitch. Acc.	Vo. False Alarm	Vo. Recall	F COnPOff	F COnP	F COn
1	melotranscript	0.80	0.87	0.37	0.97	0.45	0.57	0.63
2	ryynanen	0.72	0.76	0.37	0.94	0.30	0.47	0.64
3	smstools	0.80	0.88	0.41	**0.99**	0.39	0.55	0.66
4	pYIN s=0.0, prn=0.00	0.83	**0.91**	0.37	0.98	0.38	0.56	0.61
5	pYIN s=0.0, prn=0.07	0.84	**0.91**	0.34	0.98	0.40	0.59	0.64
6	pYIN s=0.0, prn=0.10	0.84	**0.91**	0.33	0.97	0.41	0.60	0.64
7	pYIN s=0.0, prn=0.15	0.84	0.90	0.32	0.96	0.41	0.60	0.63
8	pYIN s=0.6, prn=0.00	0.84	**0.91**	0.35	0.98	0.38	0.56	0.61
9	pYIN s=0.6, prn=0.07	0.84	**0.91**	0.32	0.97	0.43	0.62	0.67
10	pYIN s=0.6, prn=0.10	**0.85**	**0.91**	0.31	0.97	0.44	0.62	0.67
11	pYIN s=0.6, prn=0.15	**0.85**	0.90	0.29	0.95	0.44	0.62	0.65
12	pYIN s=0.7, prn=0.00	0.83	0.90	0.33	0.97	0.39	0.54	0.61
13	pYIN s=0.7, prn=0.07	**0.85**	**0.91**	0.30	0.97	0.46	0.63	0.69
14	pYIN s=0.7, prn=0.10	**0.85**	0.90	0.29	0.96	0.47	0.64	0.69
15	pYIN s=0.7, prn=0.15	**0.85**	0.89	0.27	0.94	0.47	0.64	0.67
16	pYIN s=0.8, prn=0.00	0.84	0.89	0.28	0.96	0.39	0.52	0.61
17	pYIN s=0.8, prn=0.07	**0.85**	0.89	0.25	0.95	0.48	0.66	**0.73**
18	pYIN s=0.8, prn=0.10	**0.85**	0.89	0.24	0.94	0.49	**0.68**	**0.73**
19	pYIN s=0.8, prn=0.15	**0.85**	0.87	**0.22**	0.91	**0.50**	0.67	0.71

Table 3. Results for fully-automatic melody note transcription.

taken and the number of notes manipulated. The experimental results are given below.

5.1 Accuracy of Automatic Transcription

We used a test set of 38 pieces of solo vocal music (11 adult females, 13 adult males and 14 children) as collected and annotated in a previous study [24]. All files are sampled at 44.1 kHz. We also obtained note transcription results extracted by three other methods: Melotranscript [22], Gómez and Bonada [2], Ryynänen [21]. We ran 16 different versions of Tony's note transcription algorithm, a grid search of 4 parameter settings for each of the two post-processing methods. Minimum duration pruning was parametrised to 0 ms (no pruning), 70 ms, 100 ms and 150 ms. The amplitude-based onset segmentation parameter was varied as $s = 0$, 0.6, 0.7 and 0.8.

For frame-wise evaluation we used metrics from the evaluation of pitch tracks [25] as implemented in `mir_eval` [26], but applied them to notes by assigning to every frame the pitch of the note it is covered by. The results are listed in Table 3. The pYIN note transcriptions reach very high overall accuracy rates (0.83–0.85) throughout. The highest score of the other methods tested is 0.80. [9] Among the pYIN versions tested, the best outcome was achieved by combining pruning of at least 100 ms and an onset sensitivity parameter of at least $s = 0.6$. The efficacy of the system results from high raw pitch accuracy (correct when there *is* a pitch), and low rate of voicing false alarm. There is, however, a tradeoff between the two: better raw pitch accuracy is achieved with low values of s, and lower false alarm rates with higher values of s. The algorithm smstools achieves perfect voicing recall at the price of having the highest voicing false alarm rate.

The results for note-based evaluation expose more subtle differences. The metric "COnPOff" [24], which takes into account correct note onset time (±5 ms), pitch (±0.5 semitones) and offset (± 20% of ground truth note duration), is the most demanding metric; "COnP" (correct onset and pitch) and "COn" (correct onset) are relaxed metrics. Here, we report F measures only. We observe that—without post-processing—the pYIN note transcription achieves values slightly worse than the best-performing algorithm (melotranscript). Considering the post-processed versions of pYIN, minimum duration pruning alone does not lead to substantial improvements. However, a combination of onset detection and minimum duration pruning leads to COnPOff F values of up to 0.50, compared to 0.38 for the baseline pYIN and 0.45 for the best other algorithm (melotranscript). This carries through to the more relaxed evaluation measures, where F values of the post-processed versions with at least 0.10 seconds pruning are always higher than the baseline pYIN algorithm and the other algorithms tested. Figure 3 shows all 100 ms-pruned pYIN results against other algorithms.

5.2 Effort of Manual Note Correction

In order to examine the usability of Tony we measured how editing affects the time taken to annotate tunes. We used recordings of amateur singing created for a different project, and one of us (JD) annotated them such that each final note annotation corresponded exactly to one ground truth note in the musical score matching her perception of the notes the singer was actually performing. The dataset consists of 96 recordings, with 32 singers performing three tunes from the musical *The Sound of Music*. The annotation was performed with an earlier version of Tony (0.6).

Tony offers five basic editing operations: Create, Delete, Split, Join, and Move (either left or right note boundary). We estimated the number of edits required, considering only timing adjustments (i.e. ignoring any changes to the pitch of a note). [10] The estimate is a custom edit distance implementation. First, we jointly represent the actual state of the note track (after automatic extraction) and the desired state of the note track as a string of tokens. Secondly, we define transformation rules that correspond to the five possible edit operations. The estimate of the number of edits performed by the user is then an automated calculation of a series of reductions to the source string in order to arrive at the target. In particular, if pYIN happened to perform a completely correct segmentation "out of the box",

[9] Note that Ryynanen's method outputs only integer MIDI notes, so for the fine-grained analysis required here it may be at a disadvantage.

[10] At the time of the experiment we were not able to record the actual actions taken.

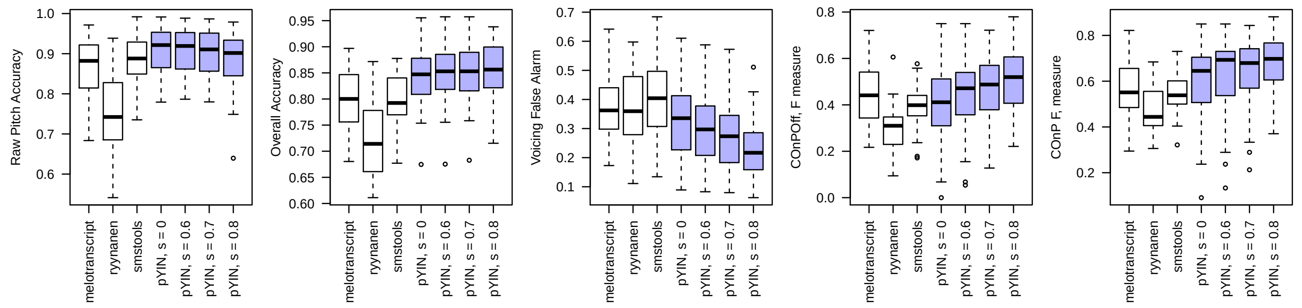
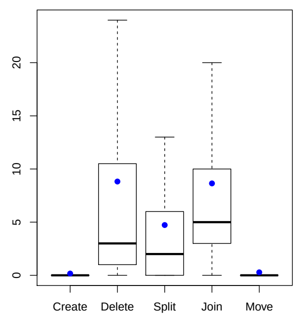
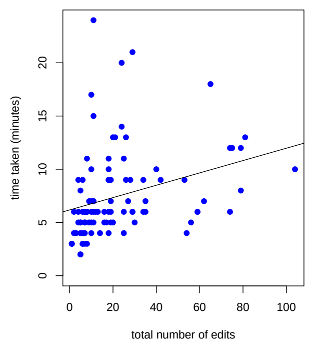


Figure 3. Results of existing algorithms and pYIN note transcription with minimum duration pruning at 0.1 s, showing, from left to right, raw pitch accuracy, overall accuracy, voicing false alarm, COnPOff F measure and COnP F measure.



(a) Number of edits per recording. Box plots with mean values (dots).



(b) Time taken against edits.

Figure 4. Edit operations.

the edit count would be zero.

Figure 4a illustrates the distributions of edit counts in a box plot with added indicators of the mean. First of all, we notice that very few notes had to be Created (mean of 0.17 per recording) or Moved (0.28), and that Join (8.64) and Delete (8.82) are by far the most frequent edit operations, followed by Splits (4.73). As expected, the total number of edits correlates with the time taken to annotate the recordings (see Figure 4b).

Which other factors influence the annotation time taken? We use multivariate linear regression on the number of Creates, Deletes, Splits, Joins, Moves and familiarity with the Tony software (covariates), predicting the annotation time (response). As expected, the results in Table 4 show that any type of editing increases annotation time, and that familiarity reduces annotation time. The baseline annotation time is 437 seconds, more than 7 minutes. (The mean duration of the pieces is 179 seconds, just under 3 minutes.) The result on Familiarity suggests that every day spent working with Tony reduces the time needed for annotation by 2.3 seconds.[11] The time taken for every Create action is 145 seconds, a huge amount of time, which can only be explained by the fact that this operation was very rare and only used on tracks that were very difficult anyway. Similar reasoning applies to the (boundary) Move operations, though the p value suggests that the estimate cannot be made with much confidence. The distinction

[11] This is clearly only true within a finite study, since the reduction cannot continue forever. Annotations happened on 14 different days.

	Est. (seconds)	Std. Error	p value
(Intercept)	437.20	51.87	<0.01
Creates	145.08	42.77	<0.01
Deletes	3.51	1.82	0.06
Splits	5.58	2.95	0.06
Joins	3.18	2.35	0.18
Move	45.51	39.61	0.25
Familiarity	-2.31	0.82	0.01

Table 4. Effects on annotation time taken.

between the remaining three edit operations is more helpful: each Delete and Join accounts for 3.5 seconds time added, but splits take much longer: 5.7 seconds. This is likely to result from the fact that the user has to position the play head or mouse pointer precisely at the split position, whereas joins and deletes require far less precise mouse actions. As Table 4 shows, most of the effects are at least moderately significant ($p < 0.1$), with the exception of number of Joins. The variance explained is $R^2 = 25\%$.

6. DISCUSSION

The results of the second experiment may well have impact on the design of future automatic melody transcription systems. They confirm the intuition that some edit actions take substantially more time for a human annotator to execute. For example, the fact that Merges are much cheaper than Splits suggests that high onset recall is more important than high onset precision.

We would also like to mention that we are aware that the accuracy of automatic transcription heavily depends on the material. The tools we evaluated (including existing algorithms) were well-suited for the database of singing we used; in other annotation experiments [27] it has become obvious that some instruments are more difficult to pitch-track. Furthermore, it is useful to bear in mind that the dataset we used is predominantly voiced, so the voicing false alarm outcomes may change on different data.

As evident from our survey (Box 1), early versions of Tony have already been used by the community. This includes our own use to create the MedleyDB resource [27], and some as yet unpublished internal singing intonation and violin vibrato experiments.

7. CONCLUSIONS

In this paper we have presented our new melody annotation software Tony, and its evaluation with respect to two aspects: firstly, an evaluation of the built-in note transcription system, and secondly a study on how manual edits and familiarity with the software influence annotation time.

The note transcription results suggest that the pYIN note transcription method employed in Tony is state-of-the-art, in terms of frame-wise accuracy and note-based evaluation. The study of manual edits shows the relative effort involved in different actions, revealing that Splits and Creates are particularly expensive edits. This suggests that for the task of note annotation, transcription systems should focus on voicing recall and note onset/offset accuracy.

In summary, we have presented a state-of-the-art note annotation system that provides researchers interested in melody with an efficient way of annotating their recordings. We hope that in the long run, this will create a surge in research and hence understanding of melody and intonation, especially in singing.

7.1 Acknowledgements

Thanks to Emilio Molina for kindly sharing the results of the experiments in his recent evaluation paper [24], and to all survey participants. Matthias Mauch is supported by a Royal Academy of Engineering Research Fellowship.

A. SURVEY QUESTIONS

- Have you ever used software to annotate pitch in audio recordings? (multiple choice)
- What software tools/solutions for pitch annotation exist? List tools that you are aware of. (free text)
- What characteristics of the tools would need to be improved to better suit your use case? (free text)
- Comments (free text)
- Your field of work (multiple choice)

B. REFERENCES

[1] S. Pant, V. Rao, and P. Rao, "A melody detection user interface for polyphonic music," in *National Conference on Communications (NCC 2010)*, 2010, pp. 1–5.

[2] E. Gómez and J. Bonada, "Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing," *Computer Music Journal*, vol. 37, no. 2, pp. 73–90, 2013.

[3] C. Cannam, C. Landone, M. B. Sandler, and J. P. Bello, "The Sonic Visualiser: A visualisation platform for semantic descriptors from musical signals." in *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006, pp. 324–327.

[4] P. Boersma, "Praat, a system for doing phonetics by computer," *Glot International*, vol. 5, no. 9/10, pp. 341–345, 2001.

[5] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[6] N. Bogaards, A. Röbel, and X. Rodet, "Sound analysis and processing with AudioSculpt 2," in *Proceedings of the International Computer Music Conference (ICMC 2004)*, 2004.

[7] J. Devaney, M. Mandel, and I. Fujinaga, "A study of intonation in three-part singing using the automatic music performance analysis and comparison toolkit (AMPACT)," in *13th International Society of Music Information Retrieval Conference*, 2012, pp. 511–516.

[8] M. Mauch, K. Frieler, and S. Dixon, "Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory," *Journal of the Acoustical Society of America*, vol. 136, no. 1, pp. 401–411, 2014.

[9] M. Goto, "A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication*, vol. 43, no. 4, pp. 311–329, 2004.

[10] J. Salamon and E. Gòmez, "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, 2010.

[11] C. E. Seashore, "The tonoscope," *The Psychological Monographs*, vol. 16, no. 3, pp. 1–12, 1914.

[12] D. Talkin, "A robust algorithm for pitch tracking," in *Speech Coding and Synthesis*, 1995, pp. 495–518.

[13] H. Kawahara, J. Estill, and O. Fujimura, "Aperiodicity extraction and control using mixed mode excitation and group delay manipulation for a high quality speech analysis, modification and synthesis system STRAIGHT," *Proceedings of MAVEBA*, pp. 59–64, 2001.

[14] J. Devaney and D. Ellis, "Improving MIDI-audio alignment with audio features," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009, pp. 18–21.

[15] M. P. Ryynänen, "Probabilistic modelling of note events in the transcription of monophonic melodies," Master's thesis, Tampere University of Technology, 2004.

[16] O. Babacan, T. Drugman, N. D'Alessandro, N. Henrich, and T. Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, 2013, pp. 7815–7819.

[17] M. Mauch and S. Dixon, "pYIN: a fundamental frequency estimator using probabilistic threshold distributions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, 2014, pp. 659–663.

[18] E. Molina, L. J. Tardón, I. Barbancho, and A. M. Barbancho, "The importance of F0 tracking in query-by-singing-humming," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 277–282.

[19] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: Challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[20] T. Cheng, S. Dixon, and M. Mauch, "A deterministic annealing EM algorithm for automatic music transcription," in *14th International Conference of the Society of Music Information Retrieval (ISMIR 2013)*, 2013, pp. 475–480.

[21] M. P. Ryynänen and A. P. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, vol. 32, no. 3, pp. 72–86, 2008.

[22] T. De Mulder, J. Martens, M. Lesaffre, M. Leman, B. De Baets, and H. De Meyer, "Recent improvements of an auditory model based front-end for the transcription of vocal queries," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, vol. 4, 2004, pp. iv–257–iv–260.

[23] M. R. Schroeder, "Period histogram and product spectrum: New methods for fundamental-frequency measurement," *The Journal of the Acoustical Society of America*, vol. 43, no. 4, pp. 829–834, 1968.

[24] E. Molina, A. M. Barbancho, L. J. Tardón, and I. Barbancho, "Evaluation framework for automatic singing transcription," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 567–572.

[25] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[26] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "`mir_eval`: A transparent implementation of common MIR metrics," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 367–372.

[27] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "MedleyDB: a multitrack dataset for annotation-intensive MIR research," in *15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 155–160.

UNDERSTANDING ANIMATED NOTATION

Christian M. Fischer
Estonian Academy of Music and Theater
mail@c-m-fischer.de

ABSTRACT

Alternative notation approaches become more and more popular. Animated notation is one of them. It is popular mainly because it seems easy to apply. On the other hand, practice shows that classically trained musicians, composers and musicologists tend to reject or misunderstand this kind of music notation. Furthermore some musical performances based on animated notation should face the question whether a regular notation would not have been more efficient. As a researcher, performer and composer working with animated notation, I experienced that there is still a lack of knowledge and some misconceptions when it comes to animated notation, its advantages and its disadvantages and foremost its practical application. A brief look into the development of animated notation, actual fields of application, an attempt of a typology, an examination of the visual communication process and a closer look at two different animated score examples will shed a little light into the darkness and support utilizing this tool in contemporary music practice.

INTRODUCTION

After a peak in the musical avant-garde of the 1950s and 1960s, approaches of alternative music notation face a renaissance recently. Although there was some interest before, especially from visual artists, Theresa Sauer's book *Notations 21* from 2009 as a direct successor of John Cage's *Notations* from 1969 seems to have been somehow the starting signal. Of course the same problems regarding alternative notation arise now as they did in the mid of the twentieth century. Questions regarding the applicability, the accurateness and whether it is music notation at all, came up. In the recent years various papers and professional literature appeared. The December 2014 issue of *Organised Sound*, this very conference and of course contemporary music practice reveals a still growing interest in the field. New technologies continuously find their way into music performance and music notation and all its manifestations like gesture notation, screen scores, various forms of extended notation or live generated scores. One kind of notation that is used more frequently in the recent years, but at the same time remains a kind of mystery is animated notation. Animated notation serves in this text as an umbrella term for various approaches, where abstract graphics (avoiding images, symbols or pictograms with an inherent meaning) are put into motion for music notational purposes and manifest as fixed media. Hence, any kind of interaction or live generated and live manipulated scores are excluded. In practice animated scores are often shown simultaneously to performers and audience. As a score, it communicates the music and supports the understanding of the structure of the piece. However, to show it to the audience is neither obligatory nor important for the understanding of animated scores in general. The most common form of music notation in the Western world is regular staff notation. In this paper staff notation serves as a kind of reference, to support the understanding of animated notation.

A BRIEF LOOK INTO HISTORY

As many of our contemporary music practices, animated notation is rooted in ideas and works of the musical avant-garde between 1950 and 1970 [5]. In that time many famous composers were exploring alternative music notation. Publications of that time reveal that those approaches were quite diverse. John Cage in the USA and Erhard Karkoschka in Europe published widely recognized books in the late 1960s that collected various works of that time [3, 9]. In these compilations one can find for instance notations that were merely musical scores. Musical graphics, a term coined later by Roman Haubenstock-Ramati [5], were considered to work rather as a trigger for improvisation than to be a proper musical score. Earle Browns' piece *December 1952* [8] is the first musical graphic, although it appears in some writings mistakenly as a graphic notation. Composers like John Cage, Morton Feldman, Mauricio Kagel, Karlheinz Stockhausen or Roman Haubenstock-Ramati [18, 19], to name but a few, were mainly driven by the limitations of staff notation to communicate their musical ideas [9]. Some composers even experimented with video scores [5]. The diversity of appearances and the desire to overcome restrictions is common for avant-garde graphic notation and animated notation today.

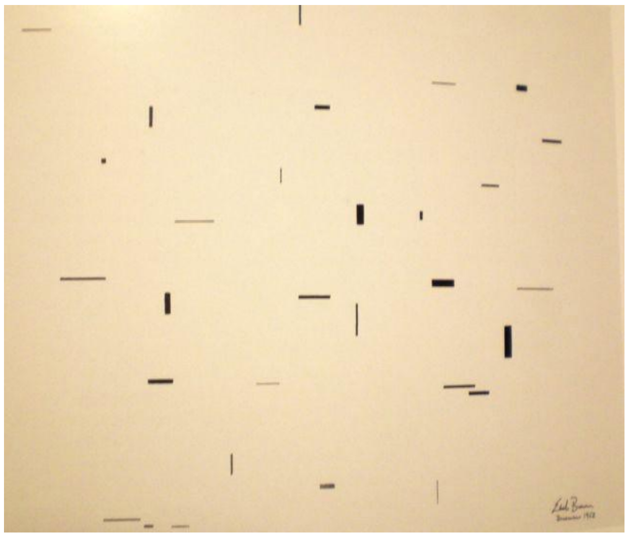


Figure 1. Musical Graphic, *December 1952* by Earle Brown [21].

From the 1970s onwards, composers seem to lose interest in the graphic notation. According to Julia H. Schröder visual artists developed ideas further as "their interest in the individual handwriting manifesting itself in musical graphics is greater than that of composers, who were concerned with the establishment of a new, normative graphic canon" [5]. Schröders analysis reveals two important distinctions regarding graphic and animated notation. First, avant-garde composers wanted to develop a generally applicable kind of graphic notation, implying a certain framework and rules to be able to work with it like with staff notation. As this did not work out, they lost interest. Second, avant-garde composers' self-conception and position within music history regarding the development of a new notation was entirely different from the situation of animated notation today. In *"Darmstädter Beiträge zur neuen Musik – Notation"* [19] composers like Brown, Kagel and Haubenstock Ramati wrote about their practices using graphic notation. For them it was clear and self-evident that the composition of new music required a new music notation. Furthermore this new notation could only come to life by somehow overcoming regular staff notation [19]. Today, animated notation can be considered a tool. It extends possibilities of notating contemporary music without neglecting other techniques or abandoning staff notation. Thereby animated notation or people using it respectively, are not aiming to establish a rigid framework and generally applicable rules.

Since the 1970s very different connections of sound or music and visuals came to life. Visual music, VJing and especially music video shaped our everyday culture like film, art, advertisements and of course music itself [10]. Technological progress, manifesting for instance in ubiquitous computer power, had a major impact on music production, performance and consumption [4]. Regular staff notation on the other side underwent only minor changes in the last 50 years, while its core system, meaning how music is principally notated, remained the same. Surely influences of the developments of the avant-garde can be traced in today's notation practice. Very often staff notation is extended by individual signs and symbols to indicate sounds or techniques that are otherwise not communicable. In 2013 Christian Dimpker published his book *Extended Notation* that develops a consistent notation system for extended instrumental playing techniques and electro acoustic music, based on the common practice [6]. Generally staff notation remains surely satisfyingly expressive. However, compared to the influence of the computer on music itself, music notation (apart from notation software like *Sibelius* or *Finale*) seems to be almost unaltered by technological progress. Only in the recent years, with concepts of interdisciplinarity, inter-media and hybrid arts, a growing interest in alternative notation utilizing computational power can be found. Practice shows there are multiple areas of application that feature new ways of music making and composition. Animated notation is just one amongst many. Yet, the utilization of screens and animation techniques for notational purposes is in its early stages. Even a commonly used term for this kind of notations can hardly be found. Australian composer and researcher Lindsey Vickery generally calls them screen scores [20] while Severin Behnen talks in his PhD thesis about motion graphic scores with its subdivisions animated, interactive and plastic score [1]. An online collection of several works by composer Pall Ivan Palsson [24] or the website `animatednotation.com` by Ryan Ross Smith [26] display a wide range of different scores and approaches. Thereby animated scores use various techniques and styles and are created with various software. In animated notation, graphical attributes are not strictly mapped with specific sounds or actions. There are no symbols or a syntax. Although animated scores often share common features, for instance a 'play-head' that indicates the actual position within a scrolling score [20], none of these features are obligatory or generally valid. Basically each score looks different. On one hand this seems to be a deficiency. On the other hand this freedom is the bases for individual artistic and musical expression and the possibility to create new music [9, 19], just like in the 1960s.

SPECIAL FEATURES

1 Two Areas of Application

Let's take a look at two areas of actual application to show two major features of animated notation. The area where animated notation can demonstrate its intuitive applicability the best is education. *Dabbledoo Music* for instance is a project by Shane Mc Kenna and Kilian Redmond from Ireland [22]. They call it "a new multimedia experience for young musicians… It aims to encourage creative music making and group performance in a new and exciting way." [22] Various types of animated notation, varying from simple to complex ones, are used to encourage and educate children to improvise and compose within a structured framework. Thereby especially timing and interaction can be practiced without the necessity of learning a complicated notational system. Another interesting example is the artistic research project *Voices of Umeå* at Umeå University Sweden by Anders Lind. He utilizes *The Max Maestro*, a standalone application programmed in Max/MSP that features an animated notation which can be controlled in real-time [23]. A choir of musically untrained people is conducted via *The Max Maestro* to produce vowels and other sounds. The length of each vowel, dynamics and structure over time are indicated. It basically allows participants to perform prima vista. Thereby performers become a part of the real-time compositional process [23]. Again the intuitiveness and simplicity of the animated score, in relation to the high quality of the musical performance, is remarkable.

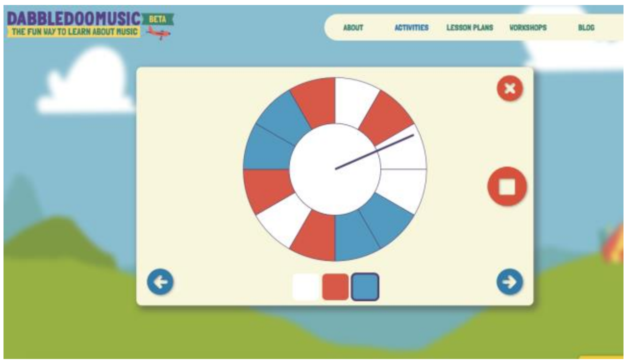


Figure 2. Screenshot of *Dabbledoo Music* website (beta version) [22].

A second area of application are musical genres or works that utilize alternative instruments, a mix of various instruments (like in live electronic music with acoustic and computer instrument) or are composed for indeterminate instrumentation. As there is no common practice, the notation of alternative instruments or objects can be accomplished on a very individual bases by the composer. For instance abstract computer sounds cannot be adequately represented in regular staff notation. By using abstract graphics, which can be mapped to musical parameters in a customized manner, animated notation can create a common ground, a kind of musical communication platform for all instruments involved [7]. Furthermore music like live electronic music is often improvised. Apart from offering a score that is able to generally structure and define musical improvisation, animated notation manifests usually in a video (file) and is therefore time-based media [2]. This allows especially to structure events accurately over time, and the score is as long as the piece. Hence, frequently used techniques like score following, stop watches or other means of triggering musical events and synchronizing acoustic and computer instruments, with their known drawbacks become obsolete.

2 Tackling a Typology

After examining the development of contemporary scores, composer and researcher Lindsay Vickery suggested four different types of what he calls screen scores. Namely scrolling score, permutation, transformative and generative scores [20]. Vickery's terminology was introduced in an historical context. Furthermore his subdivisions describe mainly the visual appearance of animated scores, like scrolling score, as the score actually scrolls. Additionally, in practice many scores mix techniques. They might not be described accurately by one of the four different types. Therefore this rather strict distinction is not truly useful for a categorization of animated scores. Still the used terminology proves very useful when discussing the appearance of animated notation in general. As mentioned earlier, the generative type is neglected in this paper.

A frequently used type of animated score is the scrolling score [20] (e.g. see figure 4). These kind of scores have several advantages. They support western reading habits as they scroll usually from left to right. These scores often work with a play zone or another indication that signals the performer which part to play. Many use a so called play-head, which is usually a line that graphics have to cross to indicate when to play them (see fig. 4). However, the most important feature of a scrolling score is the possibility to read ahead. Performers are of course used to this from staff notation. A lack of this feature might therefore cause considerable problems for musicians to utilize an animated notation [7]. Scrolling scores often utilize preliminary knowledge of the performers, for instance that a relative pitch height is indicated on the vertical axes.

Second, there are permutation or coherent scores, like for instance some Ryan Ross Smith's research studies [26]. These scores usually focus on the sequence of sound events and are therefore actional. Those scores appear as circular shapes, like clocks, grids or other networks of (sometimes multilayered) objects, that change sequentially (permutation) over time and indicate precisely when and sometimes even how long to play. Often also the number of players is clearly indicated. Depending on the graphic design of the score, it is possible for the performer to read ahead (see fig. 2). Generally, these scores convey structure of events over time and not specific sounds. This allows them to be very precise regarding the sequence of events. If the sequences are not too fast, these scores could be even played prima vista by experienced musicians. Of course there are also other permutation scores where performers have clear instructions not only when to play, but also what to play. Then these scores can be regarded as the most accurate type of animated notations, where the least interpretational effort and least amount of improvisation for the performer is required.

Finally, there are transformative or morphing scores. They are usually highly associative in character. Graphics move on the screen or change their overall appearance from one distinct graphical object to another (e.g. morphing). Movements in any direction along X, Y and Z axes are possible. This does not allow performers to look ahead. Therefore these scores require profound involvement by the performer. Without further instructions or guidelines by the composer, these scores are musical graphics in motion in the sense of *December 1952*. Nevertheless it is possible to connect visual and musical attributes. For instance the overall appearance, the design of graphics, color, shape and of course the speed of the score can be mapped by the composer to convey specific sonic attributes.

When analyzing contemporary animated notations, various mixed types of the above mentioned appearances can be found. Furthermore, as there are no generally valid and commonly accepted rules for the design and use of animated scores, a strict categorization using Vickery's terms is difficult. Therefore I propose a three dimensional coordinate system, where scores can be positioned in a more flexible manner. For instance a scrolling score can be a rather associative score that works instructive and is actional. Or anything in between. Hence, this typology does not say anything about the visual appearance or the usability of the score.

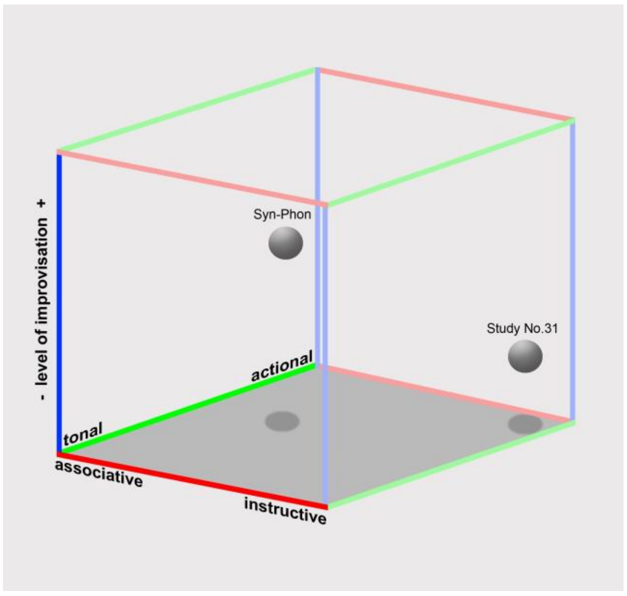


Figure 3. 3D-coordinate system to categorize animated scores. Example scores *"SYN-Phon"* and *"Study No.31"*

x-axes (red) : *associative - instructive*. This distinction refers to the appearance and possible interpretation of an animated score. A purely associative score can be regarded as a sheer trigger for improvisation, similar to a musical graphic. This means musical or acoustic parameters are not clearly mapped to graphical ones by the composer. What color, size or motion of a graphic indicate, is not defined. Rather the overall look and appearance of the score should influence the improvisation of the performer. An instructive score on the other hand indicates what to do and often precisely, when to do it. The score communicates instructions. The clock-like score on the *Dabbledoo* website (fig. 2) is a rather instructive score. The clock hand indicates when to play, and the color indicates the instrument group (red or blue) or a pause (white).

y-axes (blue) : *level of improvisation.* The position on the y-axes indicates overall how much improvisation is needed to perform the score. It is very likely that associative score requires a lot of improvisation by the performer. Nevertheless there are associative scores, where very few musical parameters are clearly mapped with graphical parameters. For instance performers simply play, when graphics are moving. On the other hand an instructive score can be very precise with certain parameters while other parameters need to be improvised.

z-axes (green) : *tonal - actional.* If not specified by the composer, the distinction between tonal and actional can be sometimes difficult. Tonal and actional refers to whether a graphic concerns sound or the means of execution. In other words, tonal graphics describe what to play, while actional graphics indicate when to play or what to do. Again the example of the clock in figure 2. This score is rather actional. The color refers to the instrument group involved. For the music itself, shapes, colors and motion have no meaning. What to play is not indicated. The example *SYN-Phon* in figure 4 is tonal and actional. The red play-head indicates when and how long to play, while at the same time, for instance the white curvy line at the right side of the picture also indicates a kind of slow vibrato.

VISUAL COMMUNICATION PROCESS

The visual communication process describes how graphical elements (e.g. staff and notes on paper or motion graphics on the screen) are understood by the receiver (e.g. a violin player). Understanding the visual communication process of animated notation is crucial for understanding animated notation itself. Many problems derive of misconceptions and wrong expectations about how information, like playing instructions, are communicated in animated notation. An example : as mentioned in paragraph 2, avant-garde composers lost interest in graphic notation as they could not establish a new normative graphic canon. This loss in interest had several reasons that can't be discussed in detail here. However, one important point was exactly this misconception of the visual communication process. Avant-garde composers regarded graphic notation as the successor of regular staff notation [5]. Therefore they assumed that it would work the same way. However there is a disparity in the communication process of western staff notation and animated notation. Animated notation consists of abstract graphics or objects in motion. Usually it is a video or in other words moving pictures. According to visual communication theory, the logic of an image (or a video) is different from the logic of a text. It is not bound to a certain framework or rules. Therefore we cannot read and understand a picture in the same way as we would read and understand a text [13]. Pictures cannot be read. They can only be analyzed and interpreted. The more unspecific, unclear or abstract the image, the more sketchy and difficult the interpretation. In this context, there is no right or wrong interpretation as long as it is coherent and comprehensive. Surely scores in staff notation need also a certain level of interpretation. Still staff notation can be read. Similar to a text using words, one has to learn signs, modes and rules of staff notation first, to be able to read and execute them. Therefore the visual communication process of animated notation and the visual communication process of staff notation work entirely different. In consequence, avant-garde composers were disappointed of the potential of graphic notation regarding the "storage" of a musical idea, because a score could be interpreted in so many different ways. Their desire to establish a new normative canon had to remain unfulfilled.

German communication theorist Heinz Kroehl, discusses sign systems and visual communication in connection to semiotics and the theories of Charles Sanders Peirce [14]. According to Kroehl there are three major communication systems : Art, everyday life and science [11]. The everyday life system refers to real objects that surround us. It is not applicable when discussing music notation. Things have a name and we can assume that we are understood by others if we use the right name for the right object. When I say “bread”, everybody, capable of English language, will know what I mean. In the scientific system, signs refer to definitions and rules. Staff notation consists of a system of specific rules, syntax and modes that need to be learned and understood to be able to apply them for musical performance. In other words, there is a (pre-)defined connection between sign and sonic result. This connection was shaped through the centuries, from neumes in the early middle-ages to western staff notation, that we know and use today. Someone able to read staff notation knows exactly which key to press on a piano keyboard when reading a specific note in a score e.g. a C4. Another musician reading the very same score will therefore press exactly the very same key on the piano keyboard when reading C4. To interpret this C4 as a completely different pitch and therefore pressing any key apart the C4 would be regarded as wrong. Therefore the transfer of knowledge, the visual communication process in staff notation can be called scientific according to Kroehls distinction [11]. Animated notation works entirely different. The interpretation of one graphic could sound different every time it is performed. Opposite to staff notation, animated notation operates in the artistic system [11]. The artistic system conveys possibilities. It is not possible that two people, in our case musicians, interpret or understand a graphic in exactly the same way and thus play identically. An animated notation is an invitation for composers and performers to start their own so called mapping process. They need to connect or map visual attributes with sonic attributes. In staff notation the mapping by composer and performer are basically congruent. In animated notation the mapping process is done individually, first by the composer and then by the performer.

It is important to understand the peculiarities of animated notation in the visual communication process to be able to comprehend its advantages and disadvantages as a tool for composition. Animated scores are intuitively applicable. Any musical parameter, like pitch, dynamics or even timbre and any other playing instruction can be conveyed. Animated notations can be simple and utilized by children and musically untrained people. On the other hand, animated scores can be quite sophisticated and require experienced and skilled musicians. The advantages of animated notations are at the same time the reasons for its drawbacks. This type of notation cannot store music in the way staff notation does. It is not possible to communicate distinct pitches, harmonics or rhythm in a way that they can be repeated in a similar manner in each performance. Still animated notation is music notation. It does not lead to a random performance or purely free improvisation. The composer defines the limits. Animated notation is simply a different approach to music composition and interpretation.

1 Design, Mapping and Guidelines

The design of the score is of course a crucial part that requires some knowledge in graphic design and motion graphics in order to be able to compose and not „to be composed“ by a software. In other words, it is possible that a lack of experience and limitations of a certain software have a significant impact on the design process of a score. This influence should be strictly avoided. However, the major difficulty in animated notation is the connection or mapping of visual and musical parameters [7]. Most musicians are used to western staff notation. For them it is clear how notes should be interpreted. But how does a red square sound compared to a green triangle? As described before, there cannot be a clear answer to that as animated notation communicates artistically. As mentioned already, animated notation needs to be interpreted and this interpretation might vary. This leads us to the mapping process. Clef, key, lines, bars and notes indicate precisely what (e.g. pitch) to play. In staff notation the major mapping process has been done already as it relies on a set of specific universally accepted rules. In graphic and animated notation meaning needs to be created individually by interpreting graphics. The mapping processes, describes the creation of meaning by connecting graphics and graphical attributes with sounds and sonic attributes. This process is divided in two separate steps. First step is the mapping done by the composer (c-mapping). The composer tries to create a score, which allows comprehensible connections between graphics and sounds or graphics and actions. Comprehensibility is the key. It is advisable to build up on previous knowledge and commonly accepted relationships. For instance western color coding, the Cartesian coordinate system with pitch on the y-axes and time on the x-axes, connecting the size of graphics with musical dynamics or utilizing the inherent motion of graphics on the screen for displaying a phrase or motive. Second step is the more delicate mapping done by the performer (p-mapping). Now, the performer interprets the score and tries to find connections between the visuals and playing music. P-mapping might vary significantly from c-mapping. However, the more precise, distinct and comprehendible the c-mapping, the more definite the score and the less interpretation work (and improvisation) by performers is required. The p-mapping can be also supported using additional guidelines. In those guidelines the composer talks about the work itself, clarifies how to read the score, explains the meaning of certain graphics or offers other means to facilitate the interpretation and mapping process for the performer. For instance one major distinction that can be made by composers and that contemporary notation struggles with for quite some time (however in a slightly different context [17]) is the distinction of graphics in either tonal or actional types. Tonal means the graphics convey sound characteristics. They refer directly to the sound and its acoustic parameters. Actional concerns the means of playing or execution. Actional graphics do not convey what to play or how it should sound but what to do or foremost when to play. Another possibility is to map instruments to a certain color. Like the design of the score, the use of additional guidelines or other explanations is of course completely up to the composer.

2 Two Examples

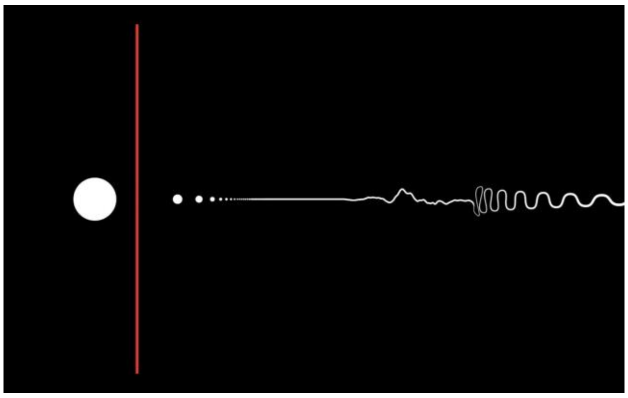


Figure 4. Screenshot of a scrolling score *SYN-Phon* by Candaş Şişman featuring a red playhead [25]

SYN-Phon by Candaş Şişman [25] (see fig.4). On Şişman’s website you will find a video of the score with a recording of a performance. There one can hear one possible interpretation of the score. *SYN-Phon* is a scrolling score, featuring a red play-head. The instrumentation is trumpet, cello and electronics/objects. Şişman himself calls it a graphical notation. White graphics on a black background scroll from right to left indicating when to play and what to play. These graphics are tonal and actional graphics at the same time. The X-axes is clearly indicating time, while Y-axes is indicating a relative pitch. There is no clear indication within the graphics that refers to a specific instrument. Therefore it is up to the performers to decide who plays certain graphics or parts of the score. The image in figure 4 shows the very beginning of the piece. The big white ball that just passed the play-head, was interpreted as a presumably electronic, gong-like sound while the smaller dots that follow are short strokes by the cello that become a continues tone changing pitch according to the curves of the line. Later the score displays several different types of objects at the same time. They are interpreted by different instruments. When watching the video on Şişman’s website one can state that the score generally works very accurately regarding the structure of events over time. The mapping of visuals and music also works out well. Most graphics find a comprehensive acoustic equivalent. What can be a little distracting sometimes is the inconsistency of the mapping. For example, some uniquely defined graphics (dots connected with thin lines) are played by the trumpet and the live electronics. The cello repeats similar playing techniques and sounds although the graphics look quite versatile. Furthermore performers do not interpret graphics consistently. The snake like line on the very right in figure 4 is played by the cello as a tone, slowly rising and falling in pitch. Visually, the interval modulates around a kind of center frequency and should be larger in the beginning of the snake. While at the end the interval should be smaller. In the performance, the cellist plays the interval modulating around a rising center frequency, which does not correspond properly to the visuals. It could be discussed whether this is a misinterpretation of the score by the performer, or whether it is unprohibited by the composer to interpret the score more freely, though.

Study No. 31 for 7 triangles and electronics by Ryan Ross Smith [26]. This piece belongs to the permutation/coherent type and comes with few explanatory guidelines by the composer. There are seven imaginary circles with cursors that indicate which part to play. One cursor/circle for each triangle player. Each circle features four attack/mute event nodes connected by an arc. The graphics are actional as they indicate when to hit a triangle and how long it should ring. The nodes and the arcs change over time. A standalone Max/MSP patch is triggered by the score. It records the triangle sounds, manipulates them and plays them back automatically. Hence, there is no need to indicate the live electronics in the score. The animated notation hardly requires any interpretational work by the performers. The way the score is designed indicates directly that the piece is about structure or patterns respectively. The patterns change over time while the overall form of the piece remains the same. The score is very intuitive. With very few explanations even musicians with limited skills are able to perform the work in a satisfactory way. Since the score is instructive, the graphics are actional and not much improvisation is demanded, the score constitutes a kind of minimal music approach that unfolds vividly how simple and precise animated notation can work.

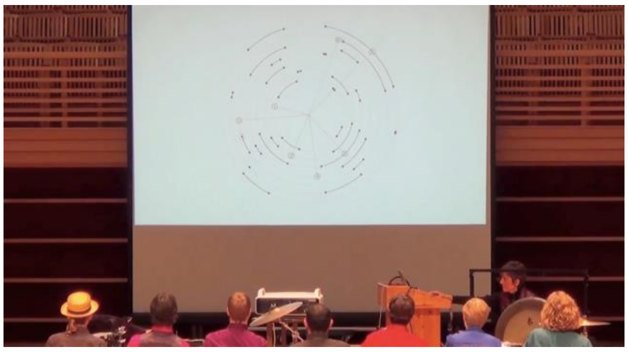


Figure 5. Screenshot of a performance documentation video featuring the score of *Study No.31* by Ryan Ross Smith [26]

CONCLUSIONS

Animated notation is an alternative approach to contemporary music composition and performance. Its intuitive applicability and the possibility to notate any kind of sound source or non-musical instruments are the major advantages of this kind of music notation. However, the visual communication process, meaning the transfer of a musical idea in general and of playing instructions in particular, is significantly different from regular staff notation. Animated scores cannot be read, they can only be interpreted. And this interpretation might vary significantly. Composers have to understand these differences to be able to utilize the advantages of animated notation. The future development of hardware and software will surely influence the evolution of animated notation and the possibilities to interconnect it to other techniques. As a creative tool, it has by no means reached its limit, yet. There is still a lot to research and to explore in the field of animated notation.

REFERENCES

[1] S. H. Behnen. „*The Construction of Motion Graphics Scores and Seven Motion Graphics Scores*“ doctoral dissertation, University of California, 2008.X. Someone and Y. Someone, *The Title of the Book*. Springer-Verlag, 2004.

[2] M. Betancourt, *The History of Motion Graphics - From Avant-Garde to Industry in the United States*. Rockville : Wildside Press, 2013

[3] J. Cage, *Notations.* New York : Something Else Press, 1969.

[4] N. Collins, *The Cambridge Companion to Electronic Music.* Cambridge MA : University Press, 2007.

[5] D. Daniels, S. Naumann, *See this Sound - Audiovisuology Compendium*. Cologne : Walther Koenig, 2009.

[6] C. Dimpker, *Extended Notation - The depiction of the unconventional.* Zürich : LIT Verlag, 2013.

[7] C. Fischer, “Motion Graphic Notation - a tool to improve live electronic music practice”. *Emille Vol.11 - Journal of the Korean Electro-Acoustic Music Society*. 2013.

[8] C. Gresser, “Earle Brown’s Creative Ambiguity and Ideas of Co-Creatorship” in *Selected Works. Contemporary Music Review* 26 (3), 2007, pp. 377-394.

[9] E. Karkoschka, *Das Schriftbild der neuen Musik.* Celle: Hermann Moeck Verlag, 1966.

[10] H. Keazor, T. Wübbena, *Rewind Play Fastforward - The Past, Present and Future of the Music Video*. Bielefeld: transcript Verlag, 2010

[11] H. Kroehl, *Communication design 2000 - A Handbook for All Who Are Concerned with Communication, Advertising and Design.* Basel: Opinio Verlag AG, 1987.

[12] A. Logothetis, *Klangbild und Bildklang.* Vienna: Lafite, 1999.

[13] M. Müller, *Grundlagen der visuellen Kommunikation*. UVK, 2003.

[14] C. S. Peirce, *Phänomen und Logik der Zeichen.* Frankfurt an Main: Suhrkamp Verlag, 1983.

[15] T. Sauer, *Notations 21*. London: Mark Batty, 2009.

[16] R. M. Schafer, *The Composer in the Classroom.* Scarborough: Berandol Music Limited, 1965.

[17] C. Seeger, “Prescriptive and Descriptive Music-Writing.” In *The Music Quarterly Vol.44*. Oxford: University Press, 1958.

[18] K. Stockhausen, *Nr.3 Elektronische Studien - Studie II*. London: Universal Edition, 1956.

[19] E. Thomas, *Darmstädter Beiträge zur neuen Musik - Notation.* Mainz: B. Schott, 1965.

[20] L. Vickery, “The Evolution of Notational Innovations from Mobile Score to Screen Score”. In *Organized Sound 17(2).* Cambridge MA: University Press, 2012.

Online Resources

[21] Artlicker Blog. *December 1952*. Retreived from `https://artlicker.wordpress.com/tag/earle-brown/` on January 8, 2015.

[22] Dabbledoo Music. *Activities – The Clock.* Retreived from `http://beta.dabbledoomusic.com/clock/level1-section2.html` on January 8, 2015.

[23] Umeå University. *Voices of Umeå Project*. Retrieved from `http://www.estet.umu.se/konstnarligforskning/` on January 8, 2015.

[24] Palsson, Pall Ivan. *Animated Notation*. Retrieved from `http://animatednotation.blogspot.com/` on January 8, 2015.

[25] Şişman, Candaş. *SYN-Phon*. Retrieved from `http://www.csismn.com/SYN-Phon` on January 8, 2015.

[26] Smith, Ryan Ross. *Complete Studies*. Retrieved from `http://ryanrosssmith.com/` on January 8, 2015.

AN ATOMIC APPROACH TO ANIMATED MUSIC NOTATION

Ryan Ross Smith
Rensselaer Polytechnic Institute
ryanrosssmith@gmail.com

ABSTRACT

Since the turn of the century, and in particular the last 15 years,[1] the discourse surrounding dynamic scoring techniques and practices has increased dramatically, while leading to an increasingly disparate terminological melee. With an awareness of what implications may exist in the premature analysis and theorization of an emerging field of practice, the author argues that in order to further develop the discourse surrounding dynamic scoring techniques and practices, it may be useful to take a reductionist approach toward defining the various low-level elements of dynamic scoring, in the case of this paper those elements that feature prominently in Animated Music Notation [AMN]. By targeting a set of low-level elements, and isolating the actualized indicators of contact and intersection as the primary functional components of AMN, the author will propose a working definition of AMN supported by examples drawn primarily from the author's work,[2] and the descriptive language generally employed during the author's compositional, rehearsal and performance experiences. To this end, this definition is not intended to entirely satisfy the broad range of dynamic scoring techniques that implement AMN, but to highlight prevalent methodologies, point toward the extension of existing taxonomies, and distinguish AMN as a notational methodology contained by the more general entity of the dynamic score, a methodology meant to clarify two basic compositional parameters: what to do and when to do it.

INTRODUCTION

In *Preface: Virtual Scores and Real-Time Playing*, Arthur Clay and Jason Freeman define real-time notation as "any notation, either traditional or graphic, which is created or transformed during an actual musical performance," and qualify this term by noting that within this particular issue of "Contemporary Music Review" alone "dynamic music notation, live scoring, virtual scoring, and reactive notation" are used by authors in describing their work, and are more or less particular to their specific approaches [1]. For the sake of this paper, I will use the term dynamic score to describe real-time scores with a collection of symbols that feature visual dynamism beyond performer interaction, this dynamism actualized as perceptible movement. At the risk of being overly pedantic, by 'beyond performer interaction' I mean to distinguish the difference between a score that generatively displays or activates notation in real-time as the result of some process autonomous from the performers, as opposed to the physical gesture of turning pages on a music stand, for instance, or the automated or hands-free turning of digital pages.

I also mean to distinguish between scores rendered for performance a priori by the performer through some process provided by the composer. John Cage's *Variations II*, for instance, requires the performer to create a unique version of the score before performance, and while this process is dynamic, in that the work *Variations II* is a set of constrained possibilities with no fixed state, its actualization as the score is ultimately fixed. Similarly, scores that are performer-determinant in real-time at the formal level (or, beyond conventional notions of interpretation) must still be considered from the score object itself as a fixed entity. Earle Brown's *December 1952* and Karlheinz Stockhausen's *Klavierstucke XI* (as graphically and conventionally notated examples respectively) are often cited in this regard. The score is a fixed entity, its dynamism or

[1] Due in large part to *Contemporary Music Review*, Vol. 29, No. 1, *Organized Sound*, Vol. 19, Special Issue 03, *Leonardo Music Journal*, Vol. 21, and animatednotation.blogspot.com. It is also important to note that dynamic scoring practices can be traced back well into the 20th century, but given the scope of this paper cannot be covered in detail.

[2] The author here acknowledges the potential downside of an analysis that focuses largely on the author's work, but contends that the concepts put forth are, while contextually-limited, available for expansion and generalization.

mobility largely conceptual, not perceptibly actualized [2].

Within these constraints, certain dynamic scoring practices present problematic actualization models. The scroll scores of Andy Ingamells feature long strips of paper, populated by small, multicolored circles that represent sonic events. In performance, the unrolled scroll is physically pulled, or scrolled, past the ensemble by two assistants. While the element of human interaction is clearly present, the assistants are not performers per se, but simply provide the mechanics necessary toward Ingamells' dynamic requirements autonomous of the performers, the theatricality of it all notwithstanding.

Similarly, works that involve real-time human-computer interaction to influence the score, including Harris Wulfson's *LiveScore*, in which the audience, through their interaction "becomes a part of the performance," but "never exactly cross over into the 'proper' domain inhabited by the ensemble performers" [3], or Nick Didkovsky's *Zero Waste*, in which the pianist in tandem with the score application creates "the composition through the act of performance" clearly displays actualized notational dynamism in real-time [4]. The performers do not lead in the conventional sense, but are led through the score by an actualized dynamic process, interactive or otherwise. Returning to Stockhausen, *Klavierstucke XI* (or any conventional score for that matter) may be considered dynamic in terms of its mobility [2], but the cursor, represented here by the performer's eye, is virtual, not actual, or actualized. Simply put, agency lies primarily with the performer to activate or *dynamize* the conventional score, whereas the dynamic score has agency over the performer; movement is perceptible, not *of* the eye, but *to* the eye. While further discussion of the various distinctions between methods of real-time scoring practices may be warranted, it is beyond the scope of this paper. However, within the dynamic score exist the potential for a variety of dynamic representations. AMN will be considered as a form of real-time notation in which the actualization of contact and intersection, which provide perceptible indications as to the specific temporal location of sonic events, are its primary distinguishing feature.

BASIC ELEMENTS OF ANIMATED MUSIC NOTATION

"A graphical method is successful only if the decoding is effective. No matter how clever and how technologically impressive the encoding, it fails if the decoding process fails." – Cleveland and McGill [5]

Introduction

Several high-level analyses and aesthetic reflections regarding the ontology of dynamic scores have provided foundational terminologies with which to describe the global functionalities of dynamic scoring techniques, including of course those represented by the wide variety of notational practices [3]. Lindsay Vickery has most recently extended existing score distinctions to include the Rhizomatic, 3D, and Animated scores respectively, distinctions based in part on their high-level functionality and visual design. What is of primary interest in Vickery's current project is the investigation into the perceptible qualities of the dynamic score, including an in-depth account of sight-reading studies, contingent on the "natural constraints based on the limitations of human visual processing," and the impact these constraints may have on communicative clarity, symbolic and functional design [6]. Similarly, David Kim-Boyle has recently investigated issues regarding the impact notational design may have on the relationship between score functionality and audience perception. [7]. These observations begin to enhance the distinction between not only high-level dynamic scoring approaches, and low-level functionalities that lead to their actualization, but suggest that analytics regarding the functional and perceptible effectiveness can be assessed at the symbolic and micro-functional level. To this end, an in-depth, low-level account of AMN specifically is largely absent, its admittedly pedantic particulars assumed, rendering the term AMN itself unfortunately colloquial.[4] I believe that to suggest particular delineations and definitions will lead toward a more rich discourse regarding AMN specifically, and distinguish AMN as a distinct methodology within the broad category of dynamic scoring, while also, through a deliberate focus on the author's own creative practice, suggest that these distinctions may be limited to particular compositional practices. To this end, a reductionist, atomic approach will be used to unpack and define the low-level elements of AMN. This reductionist analysis will not focus on musical content or concept, but target the nuts and bolts, so to speak, including prevalent symbologies and their respective dynamisms, symbol design and interaction, and an examination of actualized indication, including contact and intersection. As a global mapping of AMN practices is beyond the scope of this paper, those

[3] Scholarly contributions can be largely attributed to the work of Cat Hope, Lindsay Vickery, David Kim-Boyle, Jason Freeman, Pedro Rebelo and Gerhard E. Winkler, among many others, while their artistic contributions, and those within the field of dynamic scoring in general [Páll Ivan Pálsson's animatednotation.blogspot.com and the authors animatednotation.com provide numerous examples] continue to make significant contributions.

[4] It has been my admittedly contrary intention with `animatednotation.com`, following the model of `animatednotation.blogspot.com`, to be inclusive regarding the diversity of dynamic scoring practices, regardless of those low-level symbolic and functional requirements I will put forth here.

notational approaches that most clearly represent a clearly defined symbology, perceptible functionality, and actualized indication will be prioritized.

The symbolic elements of AMN, with which dynamic functionalities are actualized, can often be reduced to four increasingly complex entities: geometric primitives [primitives], semantically and visually integrated primitives [compound primitives], structures, and aggregates.

Primitives

A primitive is an irreducible static or dynamic symbol.[5] A primitive is irreducible when no aspect of its design can be removed without limiting its intended communicative potential. Channeling Goodman to some degree, Vickery writes "One important factor contributing to the efficacy of notation is semantic soundness – the degree to which the graphical representation makes inherent sense to the reader, rather than necessitates learning and memorization of new symbols." [6]. To this end, a primitive, which may be of any shape or size, is often cast as small geometric primitives [circles, squares, rectangles, lines (straight and curved)], favoring extensible clarity over verbose ambiguity. [7] As Gerhard E. Winkler notes, "the different parts of the score to be *reduced* to a number of elements, which can be learned and 'trained' in advance, and which can be seized with 'one glance' immediately during a performance." [8]

A stationary, or static primitive is referred to as a node, while a stationary or static *line* is referred to as an attack line or play head. A non-line dynamic primitive is referred to as a cursor or attack cursor, while a dynamic line is often referred to as a dynamic attack line or a *swiping* play head (see Figure 1) [9]. Screen boundaries, the physical (or projected) limitations of the score may or may not be treated symbolically, but are necessarily static.[6] Representative images [frogs, spaceships, etc.] are less common, and often serve higher-level purposes, as a visual representation of a particular action to be performed or instrument to be activated, as opposed to the more robust, contextually-variable symbol.[7]

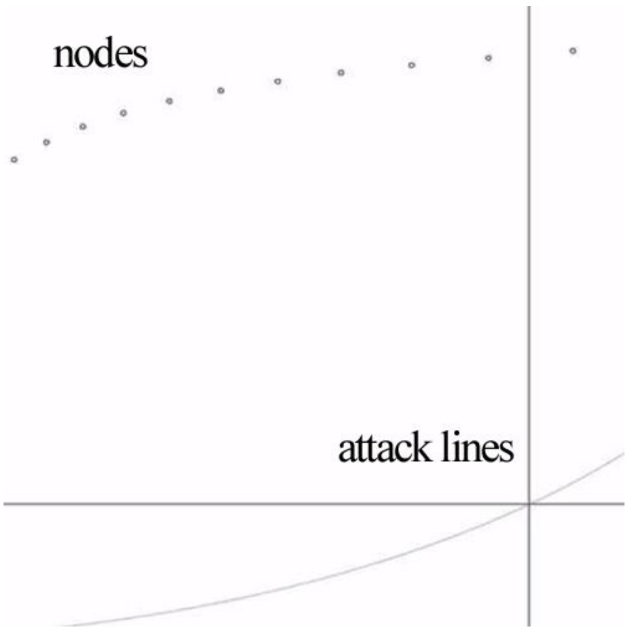


Figure 1. *y = f(x)* (2012) by Þráinn Hjálmarsson [detail] Example of sonic events represented as static circular nodes, their temporality denoted by the crossing of the dynamic attack lines/swiping play heads.

Two or more primitives can be seamlessly combined in such a way that a secondary primitive enhances or embellishes the primary, creating a compound primitive. For instance, a vertical line intersecting a circular primitive in order to clarify the moment of intersection with a static attack line.

The visual qualities of a primitive, including size and color, can also be modified to denote changes to the sonic qualities of the corresponding sonic event, insofar as it can still be 'decoded' by the performer [5]. Changes of this type are, from the visual perspective, necessarily linked to the ontology of the irreducible primitive, and so would not be considered compound (see Figure 2).

Cases where information regarding the qualities of a particular sonic event as prescribed by a primitive appear in conjunction with the primitive, but not visually embedded within it, can still be considered a compound primitive, so long as it clearly references a single instance of a primitive (see Figure 3), as opposed to a modifier, which applies to two or more primitives, and is thus not integrated.

Regions describe a subset of both static nodes and dynamic attack cursors, and are represented by a large primitive, often functionally integrated by intersecting a line (see Figure 5), or its intersection *by* a line (see Figure 6). Regions generally represent an event that is sustained, and/or modified over time. In K. Michael Fox's *Accretion* (2014), the ADSR curve is cast as a notational region, representing relative dynamics in its relation to the static attack line and vertical boundaries (see Figure 4).

[5] The focus here is on those symbols abstracted from, or distinct from conventional symbologies, but this should not presuppose their exclusion in practice.

[6] This refers to the *physical* limitations of the score, not boundaries that may result from letterboxing, for instance, which may be treated dynamically.

[7] In *The Limitations of Representing Sound and Notation on Screen*, Lindsay Vickery develops this through a continuum ranging from the spectrogram [detailed image] to the text score [distilled image]. References to frogs and spaceships is in regards to the particularly interesting experiments in notational design by the S.L.A.T.U.R. collective in Reykjavík, Iceland.

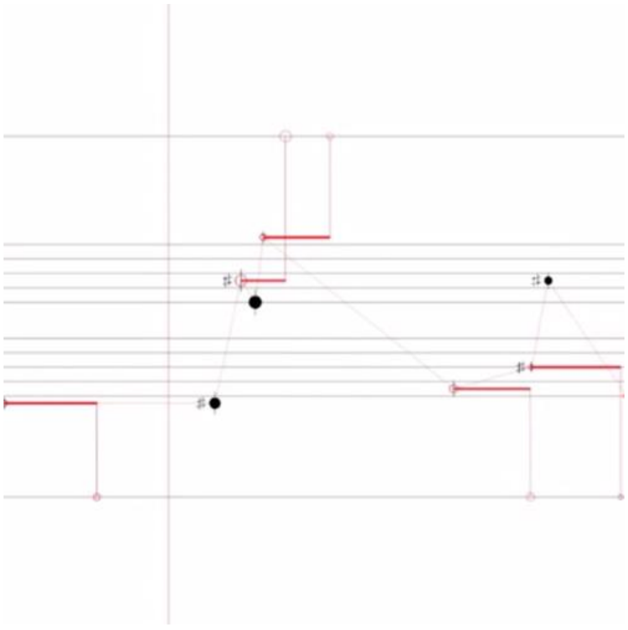


Figure 2. *Study no. 10* (2012) by Ryan Ross Smith [detail] Dynamics are embedded within each primitive, represented by relative size.

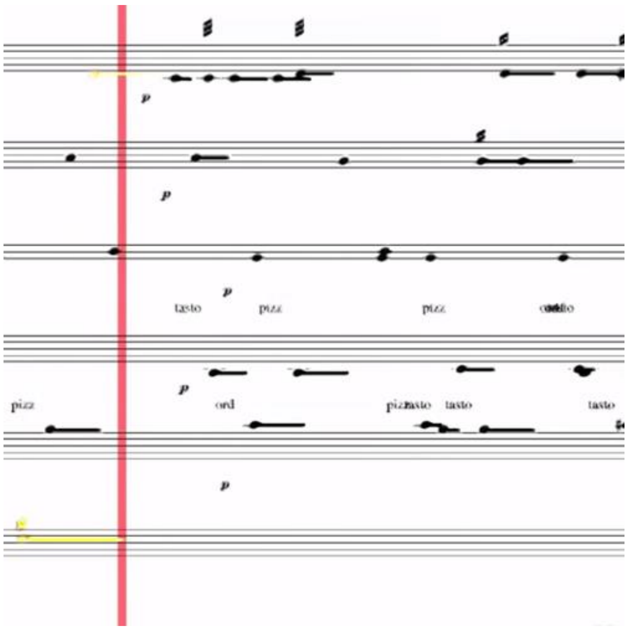


Figure 3. *Spam* (2009) by Luciano Azzigotti [detail] Dynamic markings follow the same speed and trajectory as the symbol they are applied to.

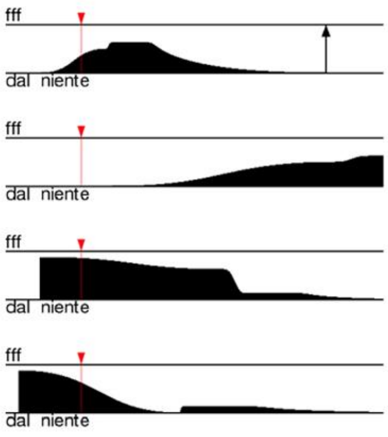


Figure 4. *Accretion* by K. Michael Fox (2014).

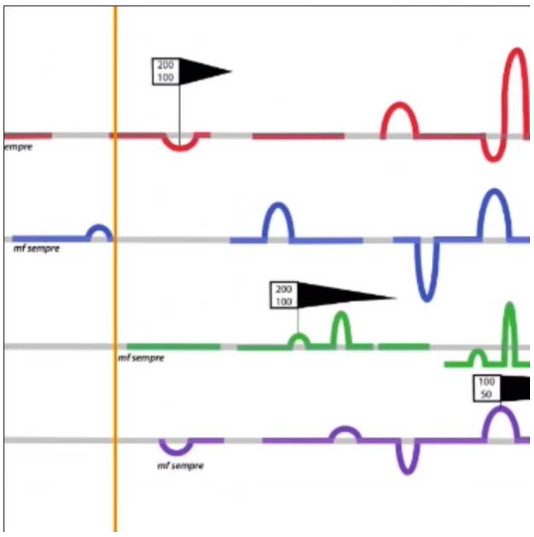


Figure 5. *Cruel and Usual* (2011) by Cat Hope.

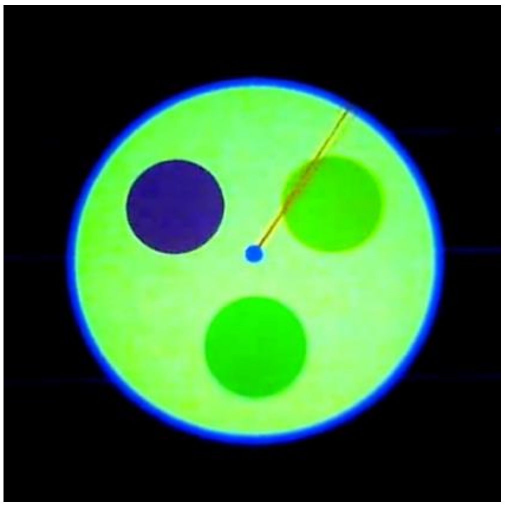


Figure 6. *Spooky Circle* (2012) by Jesper Pedersen.

Structures

A structure refers to two or more primitives in some interrelated relationship. This may be represented by an object, for example a line connecting two circular primitives (see Figure 7 [left]), or created through some dynamic relationship between symbols (see Figure 7 [right]). A structure may contain one or more primitives that are not functionally symbolic, but *clarify* functionality and "semantic soundness." [6] Many of the author's radial scores incorporate a rotating line that connects a rotating attack cursor to a central static node. This line has negligible value regarding its notational functionality, but clarifies moments of contact and intersection (see Figure 8). At the lowest level, a single structure may contain the elements necessary to produce an actualized indication of contact or intersection, an AMN capable of determining the temporal location and quality of a sonic event. To this end, an instantiation of AMN will contain at least one structure, which will in turn contain two or more primitives, at least one of which will exhibit dynamic qualities (see Figure 7 [right]).

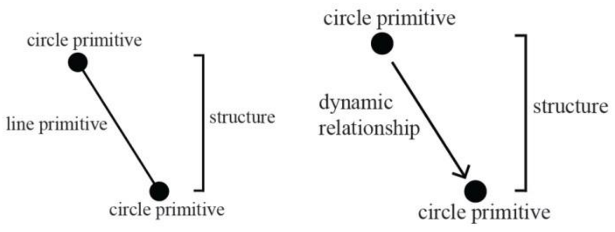


Figure 7. [left] Two circular primitives in a static relationship with one another form a structure. [right] Two circular primitives in a dynamic relationship with one another form a structure.

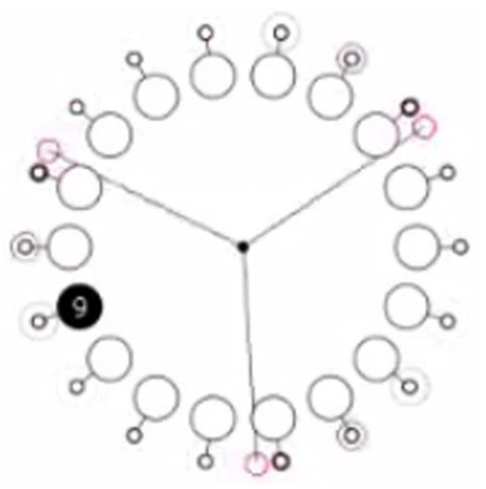


Figure 8. *Study 40.1* [Pulseighteen] (2014) by Ryan Ross Smith [detail] Each of the 18 outer nodes is activated by the intersection by the three attack cursors. The functional structure includes the rotating attack cursors and nodes. The line connecting the attack cursor to the center is a non-essential aspect of the structure, but may improve legibility and clarify functionality.

Aggregates

An aggregate is the collection of primitives, structures, and their respective dynamisms that corresponds to a single player. Aggregates may be visually displaced or integrated, and may be functionally autonomous (see Figure 9) or dependent regarding its relation to other aggregates (see Figure 10). Aggregates range in complexity from a single, simple structure (see Figure 9) to a set of integrated structures, each comprised of several primitives (see Figure 11 & 12).

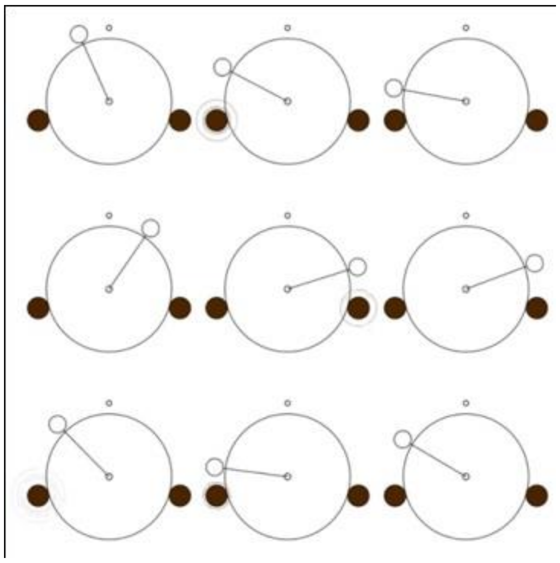


Figure 9. *Study no. 8* [15 Percussionists] (2012) by Ryan Ross Smith [detail]. Visually displaced, functionally autonomous.

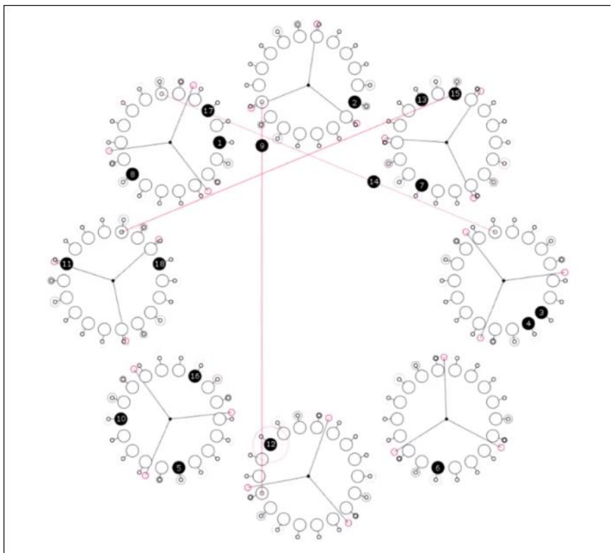


Figure 10. *Study 40.1* [Pulseighteen] (2014) by Ryan Ross Smith. Visually displaced, functionally dependent.

It is important to note that autonomous aggregates that appear to be visually integrated with other aggregates does not necessarily imply any functional integration, dependence or influence (see Figure 9).

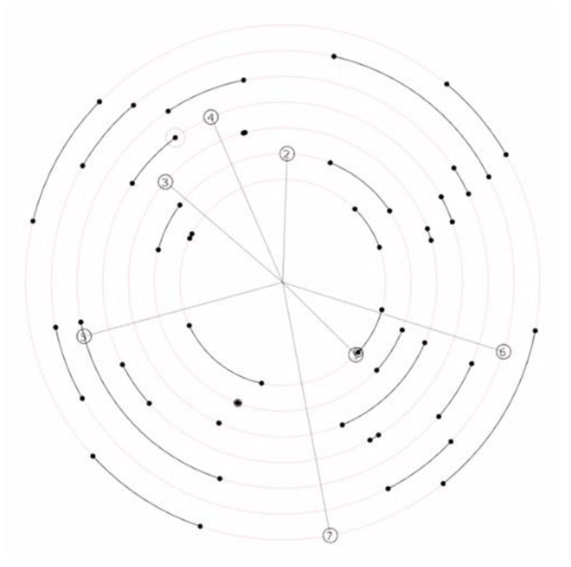


Figure 11. *Study no. 31* (2013) by Ryan Ross Smith. Each aggregate (including one of the seven concentric circles, four dynamic 'barbells,' and single rotating attack cursor) is functionally autonomous, but visually integrated, in that each aggregate seems to *encapsulate* smaller aggregates.

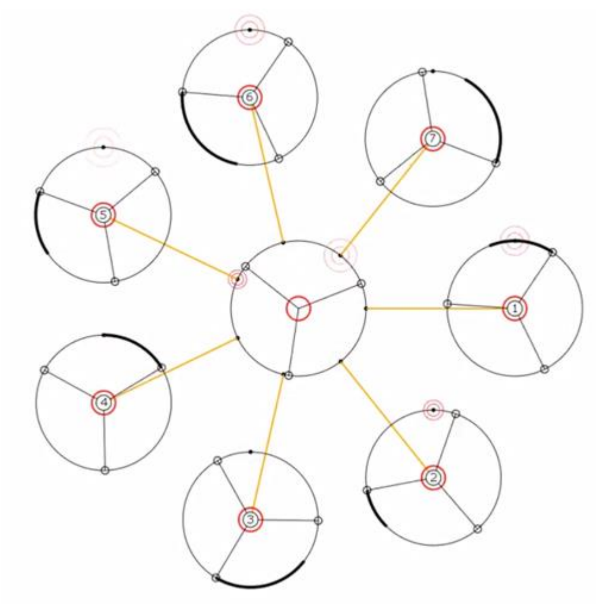


Figure 12. *Study no. 40.3* [pulseven] (2014) by Ryan Ross Smith [detail] Each numbered aggregate (numbers corresponding to players) is dependent on the central aggregate for particular functionalities throughout the piece. The central aggregate is a *collective* aggregate, in that it is accessible by more than one player.

Furthermore, the distinction between autonomous and dependent aggregates is necessarily independent from any global functionality imposed by the score generator, as all elements of the score are necessarily dependent on the score generator for their actualization.

Traversal Duration

Traversal duration refers to the time it takes for an attack cursor to move from its starting point to the point of contact or intersection. Traversal offset refers to the distance a cursor, or line, travels over the course of the traversal duration (see Figure 13). Cursor traversal must be perceptible, or trackable, in order that the performer can clearly gauge the arrival of an incoming cursor and prepare for the moment of attack, and traversal duration and cursor offset must be considered in conjunction toward this end. Lindsay Vickery considers these issues in depth, suggesting that "at scroll rates greater than 3 cm per second the reader struggles to capture information" [6]. A concatenation of nodes or cursors may extend the potential ranges of both the traversal duration and cursor offset, due in part to the regularity or *feel* that concatenation may evoke (see Figure 8). Furthermore, these particular limitations of legibility can be exploited to create, as Winkler notes "'stress' or even 'frustration'" for the players, a music *and* theatrical disruption [8], and explore the extremities of such real-time practices [10].

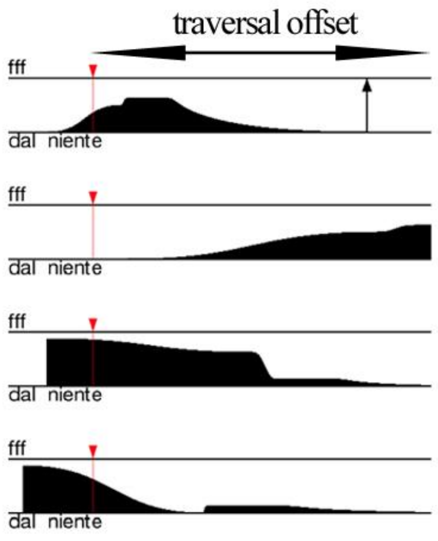


Figure 13. *Accretion* (2014) by K. Michael Fox [detail] In this example, traversal duration impacts not only onset, but the performer's current 'location' within a sustained or continuously-modified event, represented here as a region.

ACTUALIZED INDICATION

Contact

"...the true nature of things may be said to lie not in things themselves, but in the relationships which we construct, and then perceive, between them." – Terence Hawkes [11]

Actualized indication refers to a particular methodology by which the temporal location of a sonic event can visually represented with a high degree of specificity. While the history of notation provides myriad ways to locate a sonic event, this section will deal with only those that best distinguish those functionalities necessary to AMN: contact and intersection.

Contact is the "union or junction of surfaces" [12], and 'surfaces' will here refer to the boundaries of any object, visually defined by its own delineated boundaries [13]. In *Features and Objects in Visual Processing*, Anne Treisman writes "…boundaries are salient between elements that differ in simple properties such as color, brightness, and line orientation but not between elements that differ in how their properties are combined or arranged" [14]. In other words, in order for two objects, or symbols as it were, to appear to come into contact with one another, their respective visual representation must be well defined, differentiated, and at least one must demonstrate dynamic qualities.

The physical gestures of performers and conductors alike most clearly represent the *concept* of contact as a meaningful, perceptible action. The conductor's baton 'bouncing' off a virtual or imaginary boundary elicits a predetermined response based on score location and intensity; The violinist's quick breath and head snap cues an upcoming unison entrance; the guitar player jumps off the drum kit at the correct time in order to make contact with the floor at the following downbeat. These physical gestures of contact, their necessary 'setup,' as (un)subtle as they may be, within virtual and physical constraints, more or less clearly convey a bundle of performance instructions in reference to, but beyond any conventional notion of notation; in other words, the speed at which the violinist snaps her head back, and the amplitude of 'sniff volume' may determine not only the moment of attack, but relative dynamic, tempo, and other less quantifiable parameters (smooth or jagged, heroic or melancholic, etc.); A set of dynamic qualities represented by perceptible movement.

The moment of contact as a notational indicator is not new, nor dependent on digital media,[8] but does suggest a method whereby these interactions can be actualized with a high degree of temporal specificity, even in a generative context, and effectively transfer temporal agency from the performer to the score.

Contact in the context of AMN is represented by the collision of two symbols, actualized as surface juncture. Contact can occur between objects of any shape or size, with at least one exhibiting dynamic qualities. The moment at which contact occurs signifies that some sonic event is to be performed by the player.

One of the most common methods of contact includes a [dynamic] attack cursor making surface contact with a [static] node or play head. In these cases, contact occurs at the moment the cursor's boundary collides with the node or play head's boundary, followed by the cursor reversing its previous trajectory, appearing to bounce of the node, moving away in some other trajectory or simply disappearing. The cursor will not penetrate the node's boundary, and often follows a consistent trajectory (see Figure 14).

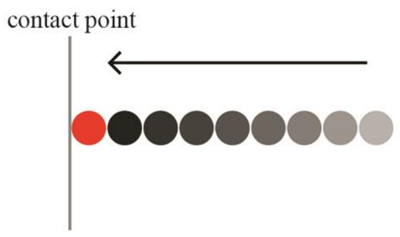


Figure 14. *Contact*: Dynamic attack cursor and static play head.

Intersection

Intersection, as an actualized indicator, consists of a [dynamic] attack cursor intersecting a [static] node or play head. This functionality requires the cursor to penetrate the node or play head, the cursor often continuing on in the same direction following intersection (see Figure 15). Intersection is often utilized for sustained or continuously modified events, and is regularly represented by a region. For continuously modified events, the alignment of the centroid is not applicable, but the position of the attack point (line or node) within the region. In Cat Hope's *Cruel and Usual* (2011), sustained tones are represented by regions in the form of straight and curved lines, their position in relation to the fixed attack line determining the relative degree to which the current pitch is detuned (see Figure 5).

Related to this functionality is the aforementioned dynamic attack line, or swiping play head, in which the nodes are rendered static, the moment of attack determined by the attack line *intersecting* the node, although the general functionality is similar (see Figure 16) [5].

[8] From Max Fleischer to Karaoke, player piano rolls to Guitar Hero, contact and intersection have been the basis for a variety of media applications of real-time notational approaches throughout the 20th century.

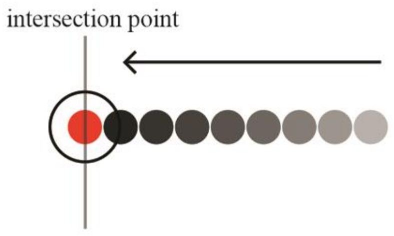


Figure 15. Intersection: Dynamic attack cursor and static play head.

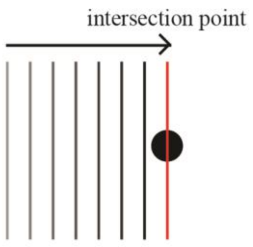


Figure 16. Intersection: Dynamic attack line, or swiping play head, and static node. Similar to the previous example, an event occurs at the moment the line aligns with the node's center.

Certain design schemes and functionalities may render these distinctions negligible. For instance, a node and cursor of relatively small size may make the *exact* moment of contact or intersection difficult to perceive, which often occurs with a concatenation of nodes or cursors [6].

A less common but similarly effective actualized indication includes the convergence by a dynamic cursor on an encapsulated static node. This describes the relationship between a dynamic cursor of similar shape to a static node sharing the same center, beginning larger, and diminishing in size until it makes contact with the node. Contact occurs when the inner boundary of the cursor reaches the outer boundary of the node (see Figures 17, 18 & 19).

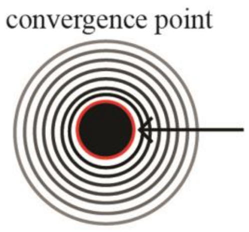


Figure 17. Convergence: Dynamic attack cursor and static node.

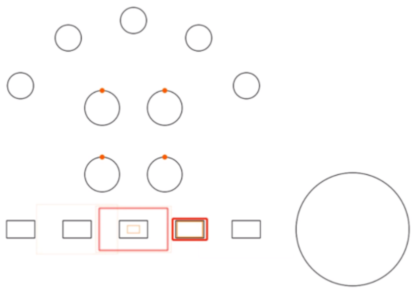


Figure 18. *Study no.16* [NavavaN] (2013) by Ryan Ross Smith. Red rectangles [attack cursor] converge on the black rectangles [static node] to denote the moment of attack.

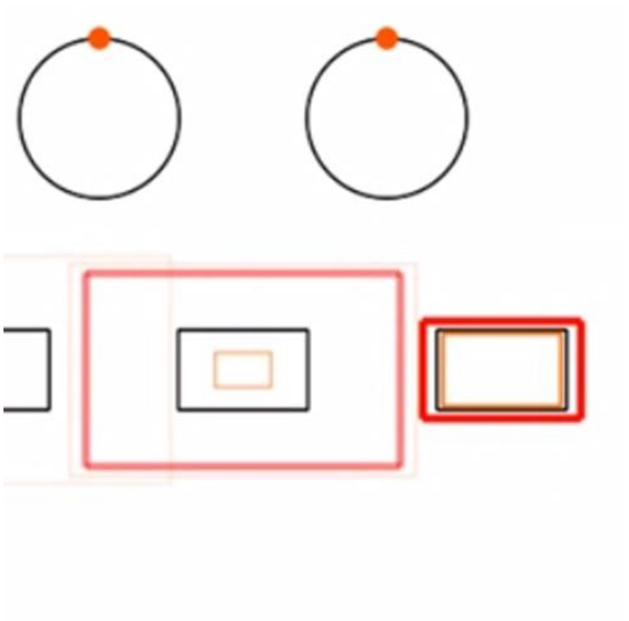


Figure 19. *Study no.16* [NavavaN] (2013) by Ryan Ross Smith [detail].

CONCLUSION

AMN is a form of dynamic notation that utilizes actualized contact and intersection between two or more symbols to denote the temporal location of sonic events. The purpose of this paper has been to propose a distinction between the low level elements [primitives, structures, aggregates, and actualized indication] that distinguish AMN as a particular notational methodology, and the dynamic score as a container which AMN and other approaches are realized, largely framed its utilization by the author to obtain temporal specificity. The continued expansion of this reductive analysis may lead to not only further this distinction, but to suggest a terminological and functional foundation from which one can clearly and consistently explain "how the system works" [8], and present possibilities for tactical subversion.

Acknowledgments

Thank you to those composers and researchers who have inspired my initial and continued interest in this field of practice, in particular the work of Páll Ivan Pálsson, Jesper Pedersen, Cat Hope, Lindsay Vickery, David Kim-Boyle, K. Michael Fox, and the composers of the S.L.A.T.U.R. collective.

REFERENCES

[1] A. Clay, J. Freeman, "Preface: Virtual Scores and Real-Time Playing," *Contemporary Music Review* 29, no. 1 (2010): 1.

[2] L. Vickery, "Increasing the mobility of Stockhausen's Mobile Scores." 2010. http://www.slideshare.net/lindsayvickery/increasing-the-mobility-of-stockhausens-mobile-scores-2010-lindsay-vickery

[3] G. Douglas Barrett, M. Winter, "LiveScore: Real-Time Notation in the Music of Harris Wulfson," *Contemporary Music Review* 29, no. 1, 2010, pp. 55-62.

[4] G. Hajdu, N. Didkovsky, "On the Evolution of Music Notation in Network Music Environments," *Contemporary Music Review* 28, n^{os}. 4/5, 2009, pp. 395-407.

[5] W. S. Cleveland, R. McGill, "Graphical Perception and Graphical Methods for Analyzing Scientific Data," *Science* 229, n^{o}. 4716, 1985, pp. 828-833.

[6] L. Vickery, "The Limitations of Representing Sound and Notation on Screen," *Organized Sound* 19, n^{o}. 3, 2014, pp. 215-227.

[7] D. Kim-Boyle, "Visual Design of Real-Time Scores," *Organized Sound* 19, no. 3, 2014, pp. 286-294.

[8] G. E. Winkler, "The Realtime Score. A Missing-Link in Computer-Music Performance" (Proceedings of Sound and Music Computing 2004, IRCAM, Paris, FR).

[9] C. Hope, L. Vickery, "Screen Scores: New Media Music Manuscripts" (Proceedings of the International Computer Music Conference 2011, University of Huddersfield, UK, July 31 – August 5, 2011).

[10] J. Freeman, "Extreme Sight-Reading, Mediated Expression, and Audience Participation: Real-Time Music Notation in Live Performance," *Computer Music Journal* 32, no. 3, 2008, pp. 25-41.

[11] T. Hawkes, *Structuralism and Semiotics*. Berkeley: University of California Press, 1977.

[12] "Contact," *Merriam-Webster's Collegiate Dictionary*, accessed January 31, 2015, http://www.merriam-webster.com/dictionary/contact.

[13] J. Feldman, "What is a visual object?," *TRENDS in Cognitive Sciences* 7, n^{o}. 6, 2003, p. 252.

[14] A. Treisman, "Features and objects in visual processing," *Scientific American* 255, n^{o}. 5, 1986, pp. 114-125.

WORKS CITED

Y = f(x) by Þráinn Hjálmarsson (2012): https://vimeo.com/96485535

Study no. 10 by Ryan Ross Smith (2012): http://ryanrosssmith.com/study10.html

SPAM by Luciano Azzigotti (2009): https://www.youtube.com/watch?v=9U0Jb-7jRs4

Accretion by K. Michael Fox (2014): http://www.kmichaelfox.com/works/accretion/

Cruel and Usual by Cat Hope (2011): https://www.youtube.com/watch?v=CtNccMuPg4w&feature=youtu.be

Spooky Circle by Jesper Pedersen (2012): https://www.youtube.com/watch?v=NN5Z9c5lrac&feature=youtu.be

Study no. 40.1 [pulseighteen] by Ryan Ross Smith (2014): http://ryanrosssmith.com/study40_1.html

Study no. 8 [15 percussionists] by Ryan Ross Smith (2012): http://ryanrosssmith.com/study8.html

Study no. 31 by Ryan Ross Smith (2013): http://ryanrosssmith.com/study31.html

Study no. 40.3 [pulseven] by Ryan Ross Smith (2014): http://ryanrosssmith.com/study40_3.html

Study no. 16 [NavavaN] by Ryan Ross Smith (2013): http://ryanrosssmith.com/study16.html

SEMAPHORE: CROSS-DOMAIN EXPRESSIVE MAPPING WITH LIVE NOTATION

Richard Hoadley
Anglia Ruskin University
richard.hoadley@anglia.ac.uk

ABSTRACT

This paper describes research, investigations, creative experiments and performances undertaken by the author in collaboration with practitioners in different creative and performance domains. The research focuses on the translation of expression between these domains and its implementation using technology. This paper focuses primarily on the role of notation in this process. The domains involved include music (audio and notation), movement (dance) and text (poetry). The data arising from performers' movements are collected and investigated; consideration is given to the use of image and graphics enabling elementary algorithmically generated dance notation.

These implementations are taken to be a part of the creative process. This research is about creating and investigating stimulating experiences where connections between one domain and the other are perceivable and where this connection itself provides an aesthetic experience. They are not intended to be fixed and permanent (although may remain so for the duration of a composition). The research is about creating dynamic environments, not musical instruments or general purpose tools.

1. THREE STREAMS

1.1 Algorithmic generation of material

The practice-led research described here is the result of the concatenation over time of a number of research strands, the first of which is the algorithmic generation of material. My primary interests involve music notation but through collaborative work these have widened to include text-based material - mainly poetry - as well as the consideration of image and graphics-based work involving notations such as dance (e.g. labanotation) and the more graphical components of music notation.

It is important to note that this work does not currently attempt to use artificial intelligence, only relatively simple algorithms and physical data to generate music in ways that one might compare to traditional composition techniques.

1.2 Physical computing

The use of physical computing - physical performance in computing environments - forms a second research strand. It is necessary for the implementation of embodied expression and translation between expressive domains as well as other factors such as synchronisation in live performance and within groups. It plays an essential role in domains such as music and dance where physical effort is of significance.

1.3 Live notation

A third strand and the main focus of this paper is notation (in this case music and text) and in particular with regard to live environments. In part due to the growth of popularity of middleware such as Open Sound Control (OSC) which facilitate bespoke communications between hard and software environments, and also because of technological and in particular network-based innovations, there are increasing technologies allowing live control over a variety of notations. One of the most visible examples of these is *Google Docs*, but software such as INSCORE [1] provides a variety of specialised notational and graphic tools, designed to be solely controllable using OSC (and therefore over networks). Related software includes MaxScore [2], the Bach Project [3] and Quintet.net [4]. While these packages each has their own advantages, they do not share INSCORE 's focus on control over and flexibility in graphical presentation which is particularly important in the author's implementation of notation synthesis for live performance.

By concentrating on the presentation and interpretation of notation, INSCORE encourages freer, more intuitive methods of composition using small, 'local' algorithms that together generate material such as that shown in Figure 1 - material generated in response to dancers' physical movements. These phrases are *not* generally pre-composed (although they could be - this is a choice made driven by aes-

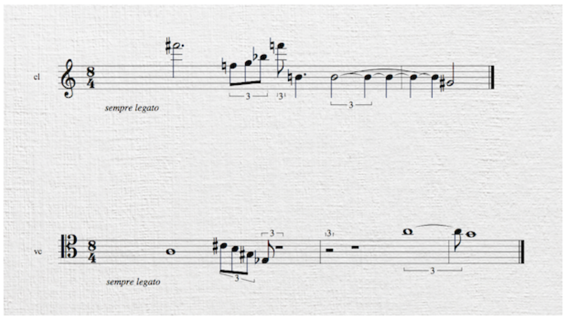


Figure 1. Dynamic notation from *Semaphore*, scene 1

thetic and practical considerations: the musicians are quite happy to encounter the music in this way). INSCORE also allows considerable control over the presentation of notation, an important feature for those composers who, like the author, find the appearance of notation reflects its expressivity (while being mindful of notation devotee Cornelius Cardew's warning that 'a musical notation that looks beautiful is not a beautiful notation, because it is not a function of a notation to look beautiful' [5]).

1.3.1 Live text

Unsurprisingly, 'liveness' has different consequences in different domains. For those working in the domain of text the ability of Google Docs to update material synchronously for all users is literally a demonstration of the editing of material as 'performance'. Inevitably some creative artists have used this platform as a way of interrogating particular methods of creating, viewing and performing with text [6]; others have used features of Skype and Twitter in similar ways [7].

Book publishing tends to emphasise the finished product - the messy processes of writing and editing are obscured by the impeccable published item. There have been a number of projects making use of electronic and networked resources, including novel-writing as performance [8] and as real-time performance [9], writing as performance art [10], writing as a contest against time [11] and against other authors on-line in the Penguin Books competition 'We Tell Stories' [12].

Of course text can also be created and manipulated generatively rather than collaboratively. This is less prevalent in text-based media (although 'off-line' methods such as *Oulipo* [13] are well known and understood). One of the first practical references to the possibility of the algorithmic generation of meaningful text was by Alan Turing [14]. In this famous test Turing replaces the question "can machines think" with "are there imaginable digital computers which would do well in the imitation game?" (The imitation game is one possible implementation of the Turing test.) While the test is for intelligence, in effect a major factor in communication is the requirement for the proper parsing of grammar through algorithms.

This apparently simple idea has been highly influential as well as controversial. In 2014 the press reported 'the first computer ever to pass the Turing Test' [15] - a claim quickly disputed [16]. Eugene Goostman [17] joins a long list of attempts at the algorithmic generation of meaning, stretching back through chatterbots such as ELIZA [18].

More recently there has been interest in the generation of robotic or virtual algorithmic creatures, for instance examples of real-time animation Larry the Zombie [19], or Milo from Kinect [20].

Through these examples and others it is clear that live action requires a particular aesthetic - books, films, art and music are all based on planning *or* improvisation. Live action/live art tends to be based on forms of guided improvisation or semi-improvisation with forms that were not previously available, so allowing hybrid creative structures involving group and real-time coordination through generative notations.

1.3.2 Live notation in music

Music, drama and dance are temporal art forms having significant improvisatory and/or interpretive components.

Over the last fifty years particular emphasis, even reverence [21], has been placed on the 'urtext' - most obviously in 'classical' musics where the score is, or has become, a fundamental element. This contrasts with many popular musics and jazz where the skilful variation or personalisation of an existing 'standard' is frequently considered central (witness Bob Dylan's own increasingly inventive variations in his performances of *Like a Rolling Stone*). In classical musics performers have been vilified for veering too far away from the original instruction or a 'classic' interpretation [22]. In forms where scores are less definitive - pop, jazz and other oral, aural and more improvised forms, 'liveness' is not in the form of notation, but in musical signals passing between musicians. (It may be significant that so-called tribute bands - replicas of older pop acts - now exist for whom authenticity is now a main criteria.) All of these factors make the live generation of music notation a particularly hybrid form. Classically-trained instrumentalists are readily able to create dynamic and exciting performances from carefully constructed live notation - they are used to creating performance in deplorably short spaces of time from fearsome scores, after all. In this case, the live notation should not be too difficult and proper thought must be given to its format and presentation (how to judge when to 'turn a page' - whatever that means digitally - for instance). The author's experience is that under these conditions musicians find performing from live scores exciting and exhilarating [23].

In the technical operation of algorithmically structuring notation it is of prime importance to achieve a satisfactory balance between the maintenance of musical style and the creation of notation straightforward and clear enough to enable the musician to give an expressive performance even when almost sight-reading. For this reason the author has made the choice to stick primarily to common practice notation. In addition, the notation has been kept as simple as possible bearing in mind the modernist style of the music. These choices have been made in order to facilitate the skills of classically-trained performers who have, through years of experience, a particular relationship with notation and they are able to transform it into dynamic, expressive performance.

Nonetheless, the live generative use of music notation has been generally less visible. While software for music notation has been developing for many years (Notator and Finale in 1988, Sibelius publicly released in 1993), there has been little apparent interest in methods of using notation both generatively and in live environments. More recently, Lilypond (e.g. [24]) has been used extensively as a platform for non-real-time generation of notation and systems such as PGWL [25] and Slippery Chicken [26] have added very sophisticated notation facilities to computer-aided-composition software. As mentioned in section 1.3 there are now a number of options available to composers working in live music notation ([2–4, 27]), although the emphasis of both remains on computer-aided composition.

Prominent 'historical' examples of live notation in music include Baird [24], Wulfson [28] and Kim-Boyle [29]. The use of notation in these cases mainly consists the manipulation of image files or the generation of large quantities of material - for instance through the algorithmic coding of Lilypond files [30]. However there are some more significant uses of live generated scores [31, 32]. Volume 29:1 of Contemporary Music Review (2010) is given over entirely to a review of live notation.

Unsurprisingly in a comparatively new field there are significant issues yet to be dealt with in the practical implementation of live notation. These include bridging the technical and aesthetic divide between notation and signals [31], general complications with synchronisation and timing, practical difficulties such as when to 'page turn', how to achieve the correct balance between reading and improvisation as well as inherent issues such as sight-reading and how difficult-to-play notation can become before it requires practice. As Lukas Foss commented on, "the precise notation which results in imprecise performance" and that "to learn to play the disorderly in orderly fashion is to multiply rehearsal time by one hundred" [33].

1.3.3 Live notation in dance and graphics

Prominent extant forms of dance/movement notation include *Labanotation*, or *Kinetographie Laban* by Rudolf von Laban [34], *Benesh Movement Notation* (graphical representation of human bodily movements), *Eshkol-Wachman Movement Notation* (graphical representation of bodily movements of other species in addition to humans, and indeed any kind of movement (e.g. aircraft aerobatics)) as well as others. These forms are primarily graphical reflecting their main focus on movement rather than textual or symbolic meaning.

While some forms of music notation have had a long and varied history, dance notation has not been so prominent. One of the reasons for this lies in the different functions that exist for dance notation. It is usually considered as a way of storing and passing on existing dances rather than as a way of expressing oneself, making the adoption or even exploration of dance or movement notation more difficult. It is rarely used in the communication of new dance work, and in spite of Albrecht Knust's suggestion that in Labanotation "the symbols must speak directly to the eyes of the reader so that he can perform the movements without, or at least without too much, reflection" [35], there are questions as to how easily and quickly it can be read and digested. Text and music notations are generally so well understood by performers that this is not a problem (although it usually requires some time to 'digest' them (see section 5)). Some musics have tests for sight-reading ability, implying that financial considerations are very likely to reduce the capacity for detailed rehearsal!

A further difference is that dance notation is generally considered such a specialised field that professional notators need to be employed, limiting its take-up in live work.

Finally, a problem specifically associated with the live use of this notation is how it can be communicated to the dancers. Most commonly this is via a data projector, but this limits the dancer's movements significantly.

Recent developments linking live notation and dance have included a variety of instances of 'hacking choreography' and 'live coding' involving dance and other forms of embodied expression. While predominantly extensions of the physical computing methods mentioned above, the use of live coding as a form of notation has been imaginatively investigated by Alex McLean and Kate Sicchio in [36–38] and demonstrated in 2013 [39].

While there are some practical problems with these systems - mainly around communicating the notation to the dancer, McLean's version of Texture, demonstrated in [39] is both visually striking and expressive. It does however, become increasingly complex as the dance progresses, making interpretation a particularly vital part of the interaction.

While the present condition of dance notation can appear to be quite frustrating, particularly in its lack of standardisation, the field is open for further developments in notation systems.

2. CROSS-DOMAIN EXPRESSION

These three research streams together allow for the practice-led investigation of cross-domain expression. Cross-domain ways of thinking are so natural to us that it is difficult to imagine expression without them. Performed music is itself a cross-domain activity utilising both physical and mental dexterity. (Arguably the use of mixed metaphors (such as my own use of the phrase 'mental dexterity' in the previous sentence) is another example, as are metaphors and analogies themselves.)

Writing about music often requires the use of metaphors and particularly when we are seeking to analyse or describe less embodied musical forms, such as acousmatic music, we are even more reliant on other domains such as language and image [40].

Most expressive domains themselves comprise of a number of linked sub-domains. A lot of music, for instance, can be described as expression through pattern enabled by physical effort. This research leans heavily on these interdependencies, seeking to maximise expression and interaction through the exploitation of musicians' learned performance skills articulated through common practice notations.

3. SEMAPHORE

Semaphore is a collaborative music-dance-text piece composed using research which seeks to translate between expressive domains using technology. An expressive domain is a form of artistic expression such as music, dance or text. Uniquely, information is taken from one domain and translated into another in real-time so allowing simultaneous performance. The music, environment and programming is by the author, choreography is by Jane Turner and text is by the novelist and poet Phil Terry. The music is performed live from code in the SuperCollider audio programming environment [41, 42], a combination of preprepared functions and structures and including some methods related to live coding.

3.1 A cross-domain sequence explained

Semaphore is composed of patterns of interactive cross-domain scenes and sequences. The following is an example of a single synchronous sequence:

A dancer's physical movement triggers and modulates the computer generation of a text phrase, which is displayed and performed. This performance is recorded and the recording is analysed spectrally. The results of the analysis then trigger and modulate a musical phrase presented as music notation which is then played by an instrumentalist. A dancer responds to the performed phrase with a physical gesture.

This set of actions might take place over a period from a few milliseconds to one or two seconds, or over an even more extended period of time. We find that the only significant latency occurs as performers consciously respond to newly displayed notations.

Alongside its creative potential, this research enables people working in one domain to generate material in another. These people might be expert performers in another domain or members of the public with no particular expertise.

There are many examples of movement-based interfaces for music, but this work is unique in its facilitation of translations from one domain into the notation of another: music, text, dance or graphics. The use of notation allows us to preserve performance interpretation that many audience members find so fundamental in live art.

Of course, the creative problem of how to create meaningful expression from these technical procedures remains as crucial as ever.

4. TECHNICAL PROCEDURES

In the following sections ways in which the parts of the sequence described above were implemented technically outlined in more detail.

4.1 A dancer's physical gesture...

The ubiquitous Microsoft Kinect (Xbox 360 version) is used to capture a dancer's physical movements. The software used for programming the audio environment, SuperCollider, is also used to perform some rudimentary movement detection. Gesture recognition is not central to this research and the software does not seek to make precise distinctions between different gestures but it is used to detect the speed and range of the movements of certain body parts. Effective though the Kinect is, the *Loie Fuller Apparition* dress which is used in part of the performance (see Figure 2) proved too concealing skeletally for the Kinect. For the next rehearsal, we used a bespoke ultrasound sensor device, the *Gaggle* [43] to gauge proximity and movement.

4.2 ...triggers and modulates the computer generation of a text phrase...

Figure 3 shows a screenshot from Semaphore showing the results of a variety of text-based manipulations of the original text displayed in INSCORE using its ability to parse HTML text and formatting. The original text was prepared

Figure 2. Loie Fuller Apparition costume. Photo © Chris Frazer-Smith 2014.

in collaboration with the team by the writer and poet Phil Terry especially for this performance. One of the key questions was how to achieve an expressive balance between sound and meaning in the text. Terry is well-versed in Oulipo techniques [13] and was aware of many possible technical textual procedures and their results - we wanted something focused and related to the Semaphore concept. Eventually, we decided on material that fell in between sound and semantics, and which also enabled some algorithmic manipulation. (Apparently by chance - or euphony - the word 'semantic' appears in the poem, linked sonically to 'semaphore'.)

Semaphore or some are for just as elsewhere
some are against
Some fear to offer or seem to fear
Afar a fir so that through the undergrowth and
across the map
A flare or a car

Soars to see the same semantic dance
Oars soar with ease or seem to soar
The same flares through the firs
Seem to spore

Ears arms too as a sheer harm
Verse as shame same sheep
Sham spheres or spare harems reap hope
Marsh shears or fennel ash

When we discovered that the original poem was too short, Terry expanded it, using a pantoum structure derived from the Malay pantum verse form which repeats lines in a pattern, effectively doubling the original length:

A B C D
B E D F
E G F H
G I/A/C H J/A/C

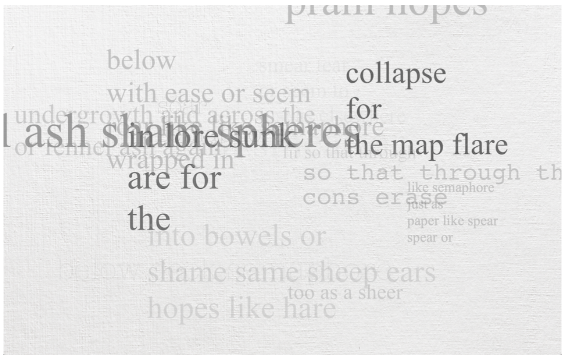


Figure 3. Semaphore, scene 3

This produces verses with a gentle, somewhat zen-like quality, emphasising the rather surreal nature of the original verse:

Some fear to offer or seem to fear
Soars to see the same semantic dance
A flare or a car
Oars soar with ease or seem to soar

Soars to see the same semantic dance
The same flares through the firs
Oars soar with ease or seem to soar
Seem to spore

Ears arms too as a sheer harm
The same flares through the firs
Seem to spore
Verse as shame same sheep

While the final part of Semaphore revolves around a prewritten poem (see section 4.3), an introductory, more abstract section (figure 3) originally involved direct interaction between dancers and text. As an example we arranged a passage where if the movements of one of the dancers was faster/higher than a given threshold, a trigger is sent to an algorithm which then chooses one from a group of selected words from the poems (such as flashing, shear, roar, billows, swelling, etc.).

Although the metaphors chosen here seem rather trite or simplistic, the scenario proved expressive, successful and full of potential.

4.3 ...the recording is analysed...

For the last part of Semaphore, we recorded Terry reading the poem. As we needed to mix between dry and wet audio streams we used a recording, although the use of a live voice (at least in part) reading live generated text is a important goal.

The software analyses the frequency and amplitude components of the vocal. The base frequency generates a series of sustained chords accompanying the voice gently in the

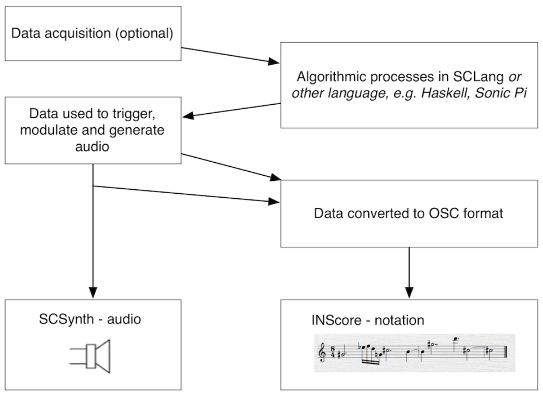

Data acquisition (optional)
Algorithmic processes in SCLang *or other language, e.g. Haskell, Sonic Pi*
Data used to trigger, modulate and generate audio
Data converted to OSC format
SCSynth - audio
INScore - notation

Figure 4. Conversion process from data to audio and notation formats

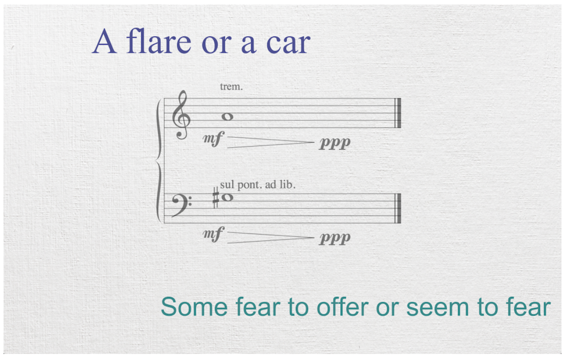


Figure 5. Semaphore, scene 4

background. If the frequency pushes over a certain threshold, a small melisma is triggered. Similarly, if an amplitude threshold is broken, a sharper, more dissonant chord is generated.

4.4 ...a musical phrase presented as notation...

At specific times during this episode - after about every thirty seconds or so - a snapshot is taken of the voice's frequency. This frequency is used in the generation of the notation of sustained notes for the clarinet and 'cello (see screenshot in Figure 5). These are arranged to create an effect in imitation of the sound of the bell of a navigational ocean buoy. In all these cases INSCORE is used to present the notation. INSCORE is controlled through OSC messages, allowing a tight integration between the language used for algorithmic control (in this case SCLang, but it could be any other OSC compatible environment) and processes synthesising the notations (see Figure 4).

4.5 ...performed by an instrumentalist and interpreted by dancers

As these notes are performed instrumentally, they are interpreted by the dancers as a port of the overall choreography. In turn, these movements may contribute further to the process of text and music notation generation. In future, we plan to use this 'audio feedback' to modulate the generation of dance notation (section 1.3.3).

5. LATENCY

The subject of latency frequently arises during discussions concerning performance using these technologies. Latency is defined as the time taken from the moment one event happens - in this case, the movement of a dancer - to the moment that the *effect* of that event is perceived - in this case, the generation of the notation and its subsequent performance [44]. The origin of the problem of latency in digital systems lies in the field of audio production and reproduction - it is the (inevitable) result of digital systems where data must be read from memory to be converted into sound. The larger quantity of data that can be read, the more efficient and to that extent the faster the system, but the higher the potential latency. Designers of digital audio instruments must find a balance between these two incompatible goals. There are, of course, examples of non-digital or mechanical latency, the time that an organ pipe (especially very large lower pitched ones) take to activate following the mechanical pressing of the key in common with many other larger acoustic instruments (double bassoon, baritone saxophone, etc.) for instance.

5.1 Causes of Latency in Semaphore

When using the Kinect, apart from the unit itself, once data is transferred to SuperCollider there are a number of additional factors that can cause latency. Most algorithmic processes dealing with symbolic musical structures (such as notation and musical pitch) will involve rather minimal processing and so will not usually cause any delay. However, the production of the notation itself can have a significant effect.

Mirroring the description of digital latency above, synchronisation with physical events requires a 'sampling' of those events in order to process them. Any system then has to balance the accuracy of this sampling against other system requirements. When including physical movements, especially those created through skilled dancers, we usually wish to identify general gestures rather than small movements - the upward rapid sweep of an arm, for instance. In order to achieve this we need to average the incoming data so smoothing out any sudden extraneous movements. (Of course, in some circumstances this is not wanted, in which cases the sampling windows must be kept small.)

These movements must then be mapped to musical gestures in one way or another. The author has chosen to develop these mappings [45–47] as an integral part of the creative process. They may be very straightforward one-to-one mappings [45, 46] - for instance an upwardly moving arm might produce an upwardly proceeding arpeggio or scale - or it may be used as a form of gesture - a fast movement may produce a fast moving string of notes (see notes 2-5 in Figure 1 above). Equally the mapping may include some aspects of real-world behaviours and gestures [48].

In some cases it is not possible to conclude a musical phrase without synchronous information, again meaning that some form of latency is inevitable.

Finally, the involvement of humans and human perception and notation is itself probably the greatest cause of latency. Rehearsals with live notation suggest that ideally performers need a second or so from the moment that the new notation is displayed to properly digest and respond to it.

5.2 Effects of Latency

Stimulating creative results seem to arise from these developmental, even compositional choices, sometimes emphasising a direct, easily perceivable relationship between movement and result, sometimes confounding expectations with a melismatic flurry as if from nowhere.

One of the difficulties some have with high levels of latency is that there is perceived to be a lack of control, even a lack of feeling of cause and effect. This implies that our main aim should be the creation of musical instruments in the best traditions of the New Interfaces for Musical Expression conference [49]. However, the design of musical instruments is not the main focus in this research. One aim in Semaphore is to investigate whether expert expressive movement can find a mapped reflection in another domain, in this case music or text. Latency might be a feature of the systems, but is not an issue for the team. If precisely timed responses are required, solutions are easily available, such as strict pre-planning of rhythm, movement and display or even the simple playback of recordings.

6. AUDIENCE RESPONSE

6.1 Universities' Week

Universities' Week [1] provided a particularly successful occasion for about 60 members of the public of all ages to interact with our system voluntarily. Although interactions produced somewhat modernist music without clear melody or rhythm and although it is likely that only a relatively few of the participants understood music notation it was clear that most enjoyed the experience immensely. Children in particular seemed able to relax and expressed themselves

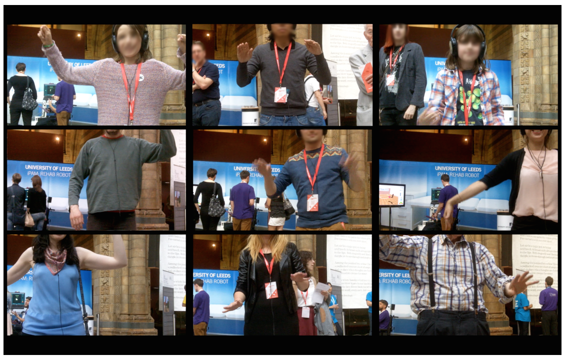


Figure 6. Universities' week interactions

Paper	**Size** (mm)	**Area** (mm^2)
A4	210 x 297	62370
15" Screen	332 x 204	**67728**
foolscap	216 343	74088
'common' size	241 x 318	76638
B4	250 x 353	88250
music part	260 x 365	94900

Table 1. Paper and screen sizes compared

enthusiastically and with none of the self-consciousness so typical of their parents. A video recording of these interactions is available - please contact the author for access (see Figure 6 for an example screenshot).

6.2 Rehearsal and acquaintance with the system

Feedback on all aspects of the composition and the notation system was gathered from the participants throughout the rehearsal process. This included two early rehearsals during which the author worked with one student dancer to properly ascertain basic functionality such as sensor ranges and sensitivities. While the Kinect can be quite sensitive to some movements it is also the case that its basic design is to recognise simple bodily movements usually associated with sports and gaming rather than the sometimes delicate and gentle movements used in contemporary dance. These factors were also linked to allowances made for latency and reliability (see section 5). In Semaphore there are relatively few requirements for absolute and precise temporal coordination, although we are optimistic that more precise synchronisation can be achieved as the systems develop.

Performers were encouraged to provide informal feedback throughout the rehearsal process and, as has happened in the past, it was soon apparent that the main problems emerged not from the generated music but rather how it was displayed.

A quick comparison of paper sizes and areas (table 1) shows that the screen area of a 15" MacBook Pro is quite small - resolution is rather irrelevant as quite a large size

[1] Universities' Week 2014 provided research groups within UK universities to showcase their research to the public. We were invited to demonstrate the work behind Semaphore during the event held at the Natural History Museum in London in June 2014.

of notation is needed. Traditional music paper sizes are far from standardised, but tend to be quite significantly larger. The laptop's screen also only allows for the viewing of one 'page' at a time and this small screen is in landscape mode rather than portrait. All these factors mean that it is a very different experience reading from a laptop's screen rather than from pieces of paper.

Another problem relating to screen size and presentation is when 'page turns' should occur and in this new environment exactly what a page turn is. In paper parts page turns, particularly those parts where frequent or near constant playing is demanded, are planned carefully, maximising the time available to turn the page at the most convenient moment. This also means that when a musician turns the page they can consciously 'discard' previous information. Semaphore attempts a variety of experimental solutions, none of which are optimal as yet.

At the moment it is clear that the use of live notation requires compromise in how it is implemented and used. For some composers these compromises may simply be too radical to consider at present.

Jonathan Eacott [50] suggests that there is a *requirement* in live notation for 'a metronome or cursor to keep musicians in sync' and that there 'must be a way of continually scrolling the music so that musicians can look ahead' - these features would certainly be very useful. However, they are not essential, depending on the nature of the material presented. If the music appears note by note as it is being created this has the advantage that it can give a fairly clear indication of the 'tempo' at which it should be played, and any further synchronisation can be achieved between instrumentalists as usual: paper parts do have cursors or metronomes.

Apart from these issues, all involved with Semaphore and earlier pieces such as *Calder's Violin* have been very positive about their experience with. Although some have displayed confusion and anxiety on first acquaintance, after some rehearsal and after realising that they are not required or expected to play every note with perfect accuracy, they relax and even enjoy the experience [23].

7. CONCLUSIONS

All who have been involved in Semaphore have been gratified by the response received from audiences and workshop visitors. The audience were offered the chance of completing a general questionnaire; fourteen were completed. These were uniformly positive; a number also contained free text comments. Below are included a selection of these, included not in a spirit of self-congratulation, but in order to demonstrate the connection felt between audience, the dancers' physical movements and the resulting music, both audio and notation:

- "I really enjoyed the performance... it was interesting to watch the dancers 'create' the music."
- "I came because of a fondness for dance but ... there is so much to take in here that it was useful to have to have two performances of the piece... Another couple of renditions would have permitted me to take in fully the choreography, the score, the text and the interaction of all the elements."
- "Thanks, it was beautiful"
- "Very interesting, would attend another similar event"
- "Really engaging and interesting... [the] performance was captivating"
- "It was great, and I wish more events had a discussion and then second performance format, that worked well"
- "Brilliant!"

Those who took part during the Universities' week also clearly demonstrated that people find generating music in this way very enjoyable and rewarding. There would also appear to be a deep link between the domains of physical movement and music. Semaphore shows that it is also possible to create and manipulate translations between music, movement and text and that both performers and audience find this expressive and stimulating. We very much hope to continue to develop these systems to enable expression and experimentation between domains. There are many possibilities that we have not even yet begun to explore.

Acknowledgments

The author would like to thank Anglia Ruskin University, Arts Council England and Turning Worlds Dance for their support in this project. In addition I would particularly thank the Jane Turner, Phil Terry, Kate Romano, Cheryl Frances-Hoad, Ann Pidcock, Gwen Jones and Ciara Atkinson for their support.

8. REFERENCES

[1] D. Fober, Y. Orlarey, and S. Letz, "Inscore: An environment for the design of live music scores," in *Proceedings of New Interfaces for Musical Expression*, Oslo, Norway, 2011.

[2] N. Didkovsky and G. Hajdu, "Maxscore: music notation in max/msp," in *Proceedings of International Computer Music Conference*, ICMA, Ed., SARC, Belfast, 2008.

[3] A. Agostini and D. Ghisi, “Bach: an environment for computer-aided composition in Max,” in *Proceedings of the International Computer Music Conference*, Ljubljana, 2012, pp. 373–377.

[4] G. Hajdu, K. Niggemann, A. Siska, and A. Szigetvari, “Notation in the context of quintet.net projects,” *Contemporary Music Review*, vol. 29, no. 1, pp. 39–53, 2010.

[5] C. Cardew, “Notation: Interpretation, etc.” *Tempo*, vol. 58, no. Summer 1961, pp. 21–33, Summer 1961.

[6] (2012, October). [Online]. Available: `http://theperformanceartinstitute.org/2012/07/27/october-19-2012-the-artist-is-elsewhere/`

[7] (2012, January). [Online]. Available: `http://visualisecambridge.org/?p=132`

[8] C. Ng. Novel-writing as performance art. [Online]. Available: `http://fictionwritersreview.com/shoptalk/novel-writing-as-performance-art/`

[9] R. Sloan. Writing as real-time performance. [Online]. Available: `http://snarkmarket.com/2009/3605`

[10] E. James. Writing as performance art. [Online]. Available: `http://www.novelr.com/2009/10/10/writing-as-performance-art`

[11] 3 day novel contest. [Online]. Available: `http://www.3daynovel.com`

[12] Penguin. We tell stories. [Online]. Available: `http://www.wetellstories.co.uk`

[13] P. Terry, *Oulipoems 2*. Ontario, Canada: aha-dada books, 2009.

[14] A. Turing, “Computing machinery and intelligence.” *Mind*, vol. 59, pp. 433–460, 1950.

[15] A. Griffin, “Turing test breakthrough as super-computer becomes first to convince us it’s human,” June 2014. [Online]. Available: `http://goo.gl/7SNpi7`

[16] I. Sample and A. Hern, “Scientists dispute whether computer ‘Eugene Goostman’ passed Turing test,” June 2014. [Online]. Available: `http://goo.gl/wdQI1a`

[17] V. Veselov, E. Demchenko, and S. Ulasen. Eugene Goostman. [Online]. Available: `http://www.princetonai.com`

[18] J. Weizenbaum, “Eliza–a computer program for the study of natural language communication between man and machine,” *Communications of the ACM*, vol. 9, no. 1, pp. 36–55, January 1966.

[19] Motus Digital. Real-time animation. [Online]. Available: `http://www.motusdigital.com/real-time-animation.htm`

[20] P. Molyneux. Meet Milo, the virtual boy.

[21] H. Cole, *Sounds and Signs*. London: Oxford University Press, 1974.

[22] I. Stravinsky, *Conversations with Igor Stravinsky*. Faber and Faber, 1959.

[23] R. Hoadley, “Calder’s violin: Real-time notation and performance through musically expressive algorithms,” in *Proceedings of International Computer Music Conference*, ICMA, Ed., 2012, pp. 188–193.

[24] K. C. Baird, “Real-time generation of music notation via audience interaction using python and gnu lilypond,” in *Proceedings of New Interfaces for Musical Expression*, Vancouver, BC, Canada, 2005, pp. 240–241.

[25] M. Laurson, M. Kuuskankare, and V. Norilo, “An overview of pwgl, a visual programming environment for music,” *Computer Music Journal*, vol. 33, no. 1, pp. 19–31, Spring 2009.

[26] M. Edwards, “An introduction to slippery chicken,” in *Proceedings of International Computer Music Conference*, Ljubljana, 2012, pp. 349–356.

[27] A. Agostini, E. Daubresse, and D. Ghisi, “Cage: a high-level library for real-time computer-aided composition,” in *Proceedings of the International Computer Music Conference*, ICMA, Ed., ICMA. Athens, Greece: ICMA, September 2014, pp. 308–313.

[28] H. Wulfson, G. Barrett, and M. Winter, “Automatic notation generators,” in *Proceedings of New Interfaces for Musical Expression*, New York, 2007.

[29] D. Kim-Boyle, “Real-time score generation for extensible open forms,” *Contemporary Music Review*, vol. 29, no. 1, pp. 3–15, 2010.

[30] M. M. Lischka and F. Hollerweger, “Lilypond: music notation for everyone!” in *Linux Audio Conference*, 2013.

[31] S. W. Lee and J. Freeman, “Real-time music notation in mixed laptop-acoustic ensembles,” *Computer Music Journal*, vol. 37, no. 4, pp. 24–36, 2014.

[32] J.-B. Barriére, “Distant mirrors,” 2013.

[33] L. Foss, “The changing composer-performer relationship: A monologue and a dialogue,” *Perspectives of New Music*, vol. 1, no. 2, pp. 45–53, Spring 1963.

[34] S. Barbacci, “Labanotation: a universal movement notation language,” *Journal of Science Communication*, vol. 1, no. 1, 2002.

[35] A. Knust, “An introduction to kinetography laban (labanotation),” *Journal of the International Folk Music Council*, vol. 11, pp. 73–76, 1959.

[36] K. Sicchio, “Hacking choreography: Dance and live coding,” *Computer Music Journal*, vol. 38, no. 1, pp. 31–39, Spring 2014.

[37] ——, “Coding choreography: Twists in language, technology and movement,” *Torque: Mind, Language and Technology*, vol. 1, no. 1, pp. 145–151, 2014.

[38] A. McLean and G. Wiggins, “Texture: visual notation for live coding,” in *Proceedings of the International Computer Music Conference*, ICMA, Ed. ICMA, 2011, pp. 621–628.

[39] K. Sicchio and A. McLean. Sicchio and mclean - sound choreography body code. [Online]. Available: `https://vimeo.com/62323808`

[40] M. Blackburn, “The visual sound-shapes of spectromorphology: an illustrative guide to composition,” *Organised Sound*, vol. 16, no. 1, pp. 5–13, February 2011.

[41] J. McCartney, “Rethinking the computer music language: Supercollider,” *Computer Music Journal*, vol. 26, no. 4, 2002.

[42] S. Wilson, D. Cottle, and N. Collins, Eds., *The SuperCollider Book*. Cambridge, MA: MIT Press, 2011.

[43] R. Hoadley, “Sculpture as music interface,” in *Proceedings of International Computer Music Conference*, M. Adkins and B. Isaacs, Eds. Huddersfield: International Computer Music Association, 2011, pp. 441–444.

[44] D. Bragg, “Synchronous data flow modeling for dmis,” in *Proceedings of the International Conference on New Interfaces for Musical Expression*, vol. 2013. NIME, 2013, pp. 237–242.

[45] A. Hunt, M. Wanderley, and R. Kirk, “Towards a model for instrumental mapping in expert musical interaction,” in *Proceedings of the International Computer Music Conference*, ICMA, Ed. ICMA, 2000.

[46] A. Hunt and M. Wanderley, “Mapping performer parameters to synthesis engines,” *Organised Sound*, vol. 7, no. 2, pp. 97–108, 2002.

[47] C. Salter, M. Baalman, and D. Moody-Grigsby, “Between mapping, sonification and composition: Responsive audio environments in live performance,” *Computer Music Modeling and Retrieval. Sense of Sounds*, pp. 246–262, 2009.

[48] R. Godoy and M. Leman, *Musical Gestures: Sound, Movement and Meaning*. New York and London: Routledge, 2010.

[49] NIME, “New instruments for musical expression - http://www.nime.org/nime2014/,” April 2014. [Online]. Available: `http://www.nime.org`

[50] J. Eacott, “Flood tide see further: Sonification as musical performance,” in *Proceedings of International Computer Music Conference*, ICMA. ICMA, 2011.

THE DECIBEL SCOREPLAYER - A DIGITAL TOOL FOR READING GRAPHIC NOTATION

Cat Hope
Edith Cowan University
c.hope@ecu.edu.au

Lindsay Vickery
Edith Cowan University
l.vickery@ecu.edu.au

ABSTRACT

In 2009, the Decibel new music ensemble based in Perth, Western Australia was formed with an associated manifesto that stated "Decibel seek to dissolve any division between sound art, installation and music by focusing on the combination of acoustic and electronic instruments" [1]. The journey provided by this focus led to a range of investigations into different score types, resulting in a re-writing of the groups statement to "pioneering electronic score formats, incorporating mobile score formats and networked coordination performance environments" [2]. This paper outlines the development of Decibel's work with the 'screen score', including the different stages of the 'Decibel ScorePlayer', an application (App) for reading graphic notation on the iPad. The paper proposes that the Decibel ScorePlayer App provides a new, more accurate and reliable way to coordinate performances of music where harmony and pulse are not the primary elements described by notation. It features a discussion of selected compositions facilitated by the application, with a focus on the significance of the application to the author's own compositional practices. The different stages in the development, from prototype score player to the establishment of a commercialized 'Decibel ScorePlayer', are outlined in the context of practice led investigations.

INTRODUCTION

The Decibel new music ensemble is made up of six renowned exponents of new music in Perth, Western Australia. Three of these performers are also composers, and one of the performers has a mathematical computer programming background. The other two performers are supportive of workshopping processes and a variety of approaches to new music, including working with electronics and improvisation. Decibel have sought to support Australian, and specifically, Western Australia new music practice, and have commissioned over eighty Australian works since their inception. A large proportion of these works are from composers within the group, but many are from significant Australian composers, electronic artists and songwriters. There is also an international aspect in their repertoire, with the group having presented monograph concerts of works by US composers Alvin Lucier and John Cage, as well as works by the late Italian composer Giacinto Scelsi and French *musique concrete* artist Lionel Marchetti. All the Decibel commissions feature acoustic and electronic components, and the group perform these works without a standard public amplification set up or live engineer. All electronics are generated from the stage, and a collection of powered monitor type speakers are used to present the electronic components throughout, which may vary from electronic playback to interactive and spatialised electronics. The rationale for this approach is to enable electronics to behave more like acoustic instruments, by using directional monitor speakers on the stage, giving a focus to the source of sound, and the way the sound is controlled and manipulated created by an operator [3]. This approach has lent itself to music scores that use graphic and extended notations, and included parts where electronics are scored quite specifically, and often, read on a computer. Decibel ensemble member Lindsay Vickery calls these 'screen scores' - music presented on and read from a computer screen. He classifies these scores into four types: real-time, scrolling, mobile and traditional [4]. Decibel engages all of these types of score in their repertoire, with a focus on real-time and scrolling scores - but also developing new categories.

In 2009, the composers within the group, Cat Hope, Lindsay Vickery and Stuart James, worked together to develop a solution that would enable the presentation of screen scores for Decibel to perform. The entire ensemble has been involved in a process of creation and interpretation of musical works in where new ideas and techniques are conceptualised, tested, evaluated, revised and disseminated in performances, recordings and archiving [5]. Through this process, the group developed a system for reading scrolling scores that was prototyped in MaxMSP. With the assistance of programmer (and Decibel viola player) Aaron Wyatt, these systems evolved into an iOS App, the Decibel ScorePlayer for the Apple iPad. It is now available on the iTunes Store internationally.

Decibel are of course not the first to engage with screen scores - previous work by Dannenberg [6], Clay and Freeman [7], Kim-Boyle [8] and others have examined the possibilities for real time score generation on computers, and a variety of propriety score generators for traditional notation are available, two examples being INscore [9] and MaxScore [10]. However the use of graphic notation - newly composed and extant - in screen scores has been limited, and often tied to traditional notation. The digital format offers a range of possibilities to develop graphic notation practice - through the incorporation of aspects such as colour, real time generation, video and interactivity. Decibel's score player investigations have focused primarily on this area of development, and in providing a 'reading mechanism' for performance, rather than a score generation tool.

THE DEVELOPMENT OF A SCROLLING SCORE PLAYER

The iTunes store describes the Decibel ScorePlayer as software that "allows for network-synchronised scrolling of proportional colour music scores on multiple iPads. This is designed to facilitate the reading of scores featuring predominantly graphic notation in rehearsal and performance" [11]. It works best for music that needs to be coordinated in a "timed" way, with proportional pitch structures. It is particularly useful for music that is pulseless, or requires pulse to be removed from the reading mechanism. The Decibel ScorePlayer is very good at presenting scores that in the past would have required a clock to coordinate multiple performers.

The Decibel ScorePlayer began as a bespoke solution to the problem of reading certain graphic scores, specifically those by author Cat Hope, who is a composer and ensemble director of Decibel. In 2008, before Decibel had began, Hope's *Kingdom Come* (2008) for laptop duet featured A graphic notation read from left to right. The image was put in motion in a movie program, and the performers read the score at the point just before it passed off the screen. This was not particularly accurate but provided an approximation of coordination that facilitated the performance. The score had been created on a computer, and did not exist in any real "physical" dimension. In preparation for the first Decibel concert in September 2009, Hope presented a score consisting of a computer print out of ten landscape A4 pages stuck together, a kind of coloured line graphic score for five instruments - one of which was a turntable - again with the problem of how to read the music in a coordinated manner.

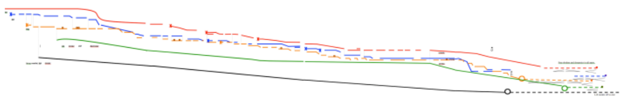


Figure 1. Cat Hope's score *In The Cut* (2009).

This piece was *In The Cut* (2009) for violin, cello, bass clarinet, bass guitar and turntable with sub woofer and is shown in **Figure 1**. The piece does not treat harmony or meter in any 'traditional' way, adopting graphic notation as a way to better reflect a proportional approach to music composition [12].

A solution to the problem of reading *In the Cut* was provided through the creation of a MaxMSP patch, where the digitally created score file (a JPEG or PNG) was read by passing under a vertical line over a pre prescribed period of time, in the case of *In The Cut*, seven and a half minutes, as shown in **Figure 2.** A control panel was built to adjust specifications for each performance, and was shown on the same screen as the score.

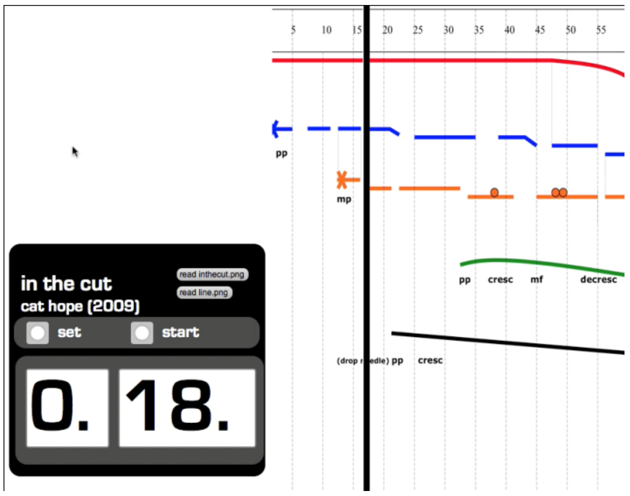


Figure 2. Lindsay Vickery's control panel for the score player built in Max MSP.

This vertical line came to be known as the playhead, referencing the tape head on tape players. Musicians would play their part as it passed by the playhead, providing an accurate way of coordinating the performers together by reading the same part in the score at the same time. The playhead was placed slightly in from the left side of the score image, so that the performers could see the material approaching the playhead in advance, but also so a small amount of material already performed, which would often assist in referencing the upcoming material. The coloured parts provided easy identification for the different performers, and the piece itself was proportional in its representation of pitch across all the instruments. The score presents each instruments part as a long, slowly descending line, representing a very smooth sound quality that uses glissandi to move between different pitches. Simply, the score looks very much as it sounds, and this is supported by a number of audio spectrograms made of different performances, such as the example provided in **Figure 3.**

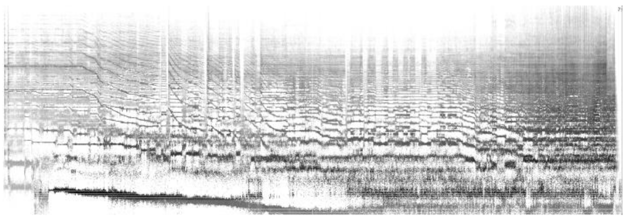


Figure 3. Spectrogram of a performance of Cat Hope's score *In The Cut* (2009) [13].

Vickery built the MaxMSP patch in consultation with Hope and ensemble. It usually required the performers to have access to a full version of MaxMSP to run the program, though it was later made workable on Max Runtime. A number of works were written for this software player prototype, some for other ensembles, and some without electronics. One example is Hope's *Kuklinski's Dream* (2010) for instrumental trio, carving knives and electronics. Like *In The Cut*, the work is characterised by a lack of pulse, proportional pitch relationships, colour representations for different instruments and unusual instruments (in particular, carving knives bowed and amplified). A notated electronic part was also featured, required programming by the ensemble's electronics operator prior to performance. Another work by Hope, *Wolf at Harp* (2011) for four drum kits, used blocks of notation to describe fields of activity on certain parts of percussion kits, in this case the bass drum, cymbals and toms. The scrolling nature of these scores effectively communicate the composer's intention a kind of pulseless music characterized by long sustained sounds. They also allow careful ensemble interactions enabling an accurate reading of the proportional nature of the score.

READING AND NETWORKING

The first Decibel scrolling scores were projected onto a screen in the performance space, to facilitate musicians reading the score in performance. Whilst providing a straightforward solution to coordinating a performance, the performers mostly had their backs to the audience, hardly a desirable performance presentation format. The score was also a very predominant feature in the space. Many audience members would comment on the nature of the score and follow it intently during the performance. Whilst this brought a new audience to our concerts seeking to 'understand' the practice of new music, it had become more of a focus than the music itself. To overcome this, Decibel member Stuart James added networking capacity, so that multiple laptop computers could be connected and coordinated over cabled Ethernet. This meant that each performer had their own score player coordinated with the others in the ensemble. The patch was further developed by Vickery to fast-forward to different parts of a score, and to slow the speed of the piece for rehearsal purposes.

These developments made the software more workable in rehearsal situations, and some fifteen works were composed for this version of the player. The ensemble also began adapting a range of other composer's scores to be read by the ensemble using the patch, including Earl Brown's *December 1952* for open instrumentation and Giacinto Scelsi's *Aitsi* (1974) for piano and electronics among others. Works from Percy Grainger's Free Music project, namely his *Free Music No. 1* (1936) for four Theremins and *Free Music No. 2* (1937) for six Theremins were put into the player. The pages of Grainger's hand drawn score were joined together and scanned into a single file, the different parts traced over in different colours and a playhead designed to include the list of pitches represented by the undulating lines that are a feature of this composition, as shown in **Figure 4** [14].

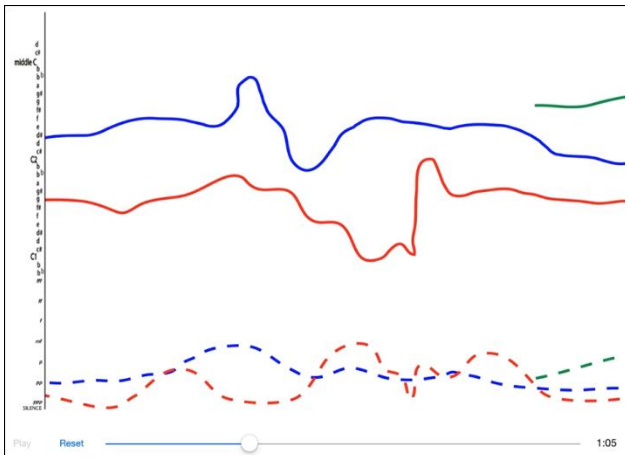


Figure 4. Percy Grainger *Free Music No. 1* (1931) adapted for the iPad Decibel ScorePlayer. This image shows the playhead replaced by a chromatic meter, and the scrub function along the bottom of the image, with the time elapsed on the right.

Other screen scores were being developed within the ensemble that included variations on the theme of scrolling presentation. Vickery's *Ghosts of Departed Quantities* (2011) for bass flute, bass clarinet, cello, keyboard and live electronics, for example, features music notation that subtly appears and disappears to the reader as it passes a playhead. **Figure 5** shows the presentation of two instrumental parts, bass flute and bass clarinet. The musical information passes from left tor right across the playhead.

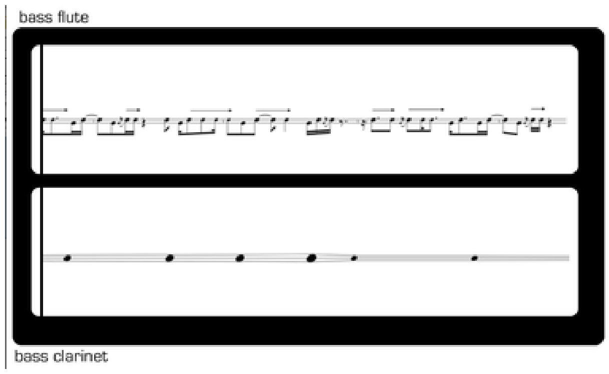


Figure 5. Lindsay Vickery's *Ghosts of Departed Quantities* (screen shot) excerpt.

In *Ghosts of Departed Quantities*, each performer has unique score activity, unlike Hope's scores, which required a tightly coordinated presentation of fixed materials. Vickery's screen scores presented materials that would arrive in a different order and quantity each time the piece was performed. Scores such as *In the Cut* provide performers with the possibility of choosing different starting notes for each performance, but require them to maintain the same pitch relationships each time.

The score player patch continued to be adjusted and developed to incorporate a range of new behaviors, including changes in the direction of the score. Hope's *Liminum* (2010) features a score that musical material goes backwards and forwards, and the play head jumps to different parts in the score at certain points. Again, each player's score is independent in this process, whilst being coordinated to start and finish together. In *Juanita Neilsen* (2012) these 'jumps' are coordinated to occur in random places, but coordinated with all players. These scores have been categorized as 'Variable Scrolling Scores'. In a collaborative work between Hope and Vickery, *Talking Board* (2011), circles traverse a larger than the screen image, serving as the guide for musicians to read said image, as shown in **Figure 6**. The movements of the circles provide information to an electronics operator for generative, interactive and spatialised electronic parts. *Talking Board* was a radical departure from the scrolling score format used on the score player up until that point, completely breaking away from the linear, left to right presentation and reading of the score. The circles have a series of different behaviors, including swarming, following, getting larger and smaller, appearing and disappearing [15]. It also required the transmission of data generated by movements on the score to another sound generating computer, signaling the need for the score player to send more than score data, leading to investigations around the incorporation of Open Sound Control (OSC).

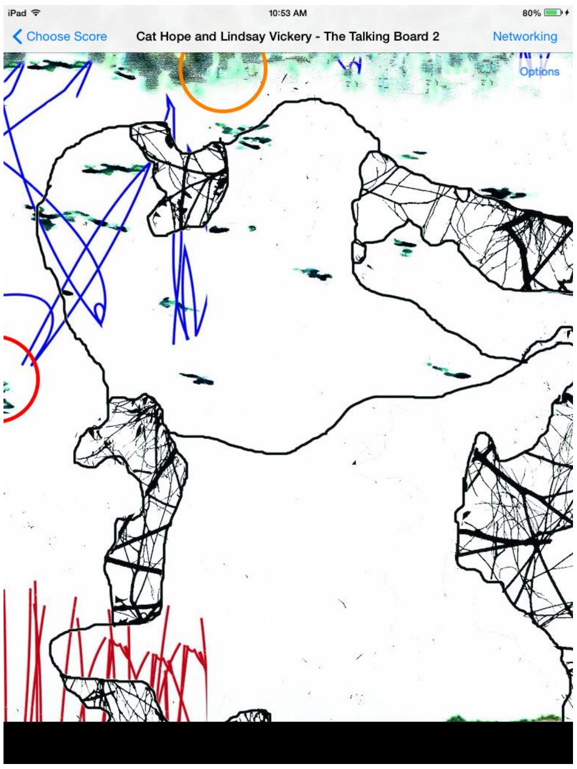


Figure 6. Cat Hope and Lindsay Vickery, *The Talking Board* (2011), screen shot of score excerpt. Here, two circles are visible - one at the top of the score, the other to the left - each half off the screen.

EXTENDING THE PARADIGM

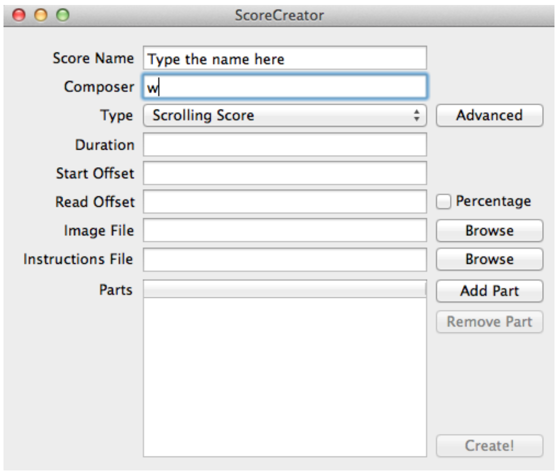
The score player project involved a number of other developments for reading scores other than graphic notations that are worth mentioning here. Automated page turning and synchronised click tracks were adopted and used in performances of pieces such as Thomas Meadowcroft's *Pretty Lightweight* (2001) and Lindsay Vickery's *Night Fragments* (2011). Mauricio Kagel's *Prima Vista* (1967), is a piece designed to feature slides shuffled and presented in a slide projector at random order. This 'score play' technique too was automated in a MaxMSP patch.

Decibel also performed other MaxMSP generated screen scores written specifically for the ensemble. Sam Dunscombe's *West Park* (2010) provided a range of changing score slides that would connect with the live electronic processing. In David Kim Boyle's *Point Studies No. 1* (2011), a beautiful spiraling colour video score produces sine tones as a result of the generative activity in the patch producing the score [8]. Between 2010 and 2012, a number of pieces were written for the scrolling score player by a range of composers, often characterised by the inclusion of non traditional instruments, that would otherwise be difficult to notate using conventional notations.

From laptops to tablets

Despite moving to wireless networking in 2011, the laptop presented a number of limitations for presentation of the scores. Most performers laptops were used for other purposes than score reading - leading to issues with different operating systems, networking protocols and personal settings. Despite the development of a network utility developed in MaxMSP to monitor network activity, the collection of IP address and constant monitoring of who was on and off the network provided ongoing problems. A European tour in late 2011 featuring Decibel repertoire in the prototype score player provided a turning point in the development of the score player. It was decided to move the score player project to portable tablet computers. Funding was secured in early 2012 to purchase five iPads and to develop the score player on the iOS platform.

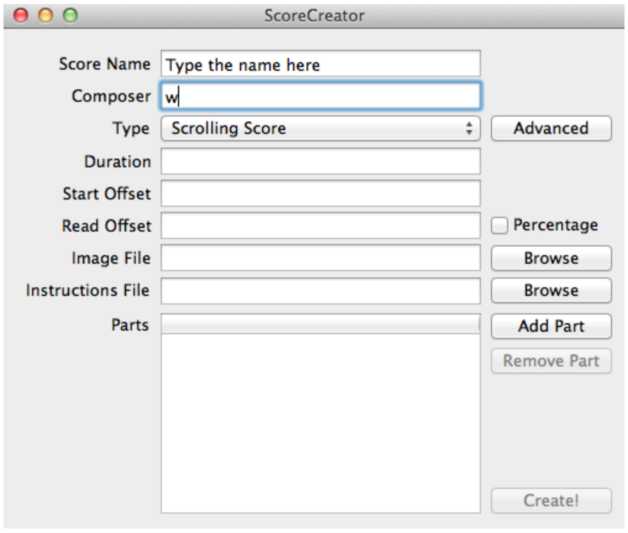
Decibel members Aaron Wyatt, Malcolm Riddoch and Stuart James set about developing what was to be called the Decibel ScorePlayer for iPad in early 2012, and the first release was issued on the Apple App store later that year. This release come with packaged with two scores each by Hope and Vickery, and provided a link to a free desktop application, the Decibel Score Creator, developed by Wyatt to enable users to create their own scores in the format required for uploading to the player, a *.dsz* file. The Decibel Score Creator is where important elements of the piece are assembled and stored into the file, and the interface is shown in **Figure 7.** In addition to naming the piece by title and composer, the length of the piece, the position of the play head, extra (separated out) parts and any instruction notes for performance can be added. Any instructions would appear in a drop down menu on the ScorePlayer when the piece is selected from a menu listing all the compositions in the player. These elements all constitute the *.dsz* file



Score Name	Type the name here
Composer	w
Type	Scrolling Score
Duration	
Start Offset	
Read Offset	☐ Percentage
Image File	
Instructions File	
Parts	

Figure 7. The Score Creator interface built by Aaron Wyatt and designed by Decibel composers in conjunction with him.

The iPad Decibel ScorePlayer provided a number of benefits over the laptop version. A much easier networking facility, native to iOS meant each iPad user could join any network agreed on by the ensemble, and users could see who else was on the network at any time using a network tab [16]. Once *.dsz* files are created, users can add scores to the Player by uploading them in the sharing facility of iTunes, as seen in **Figure 8.**

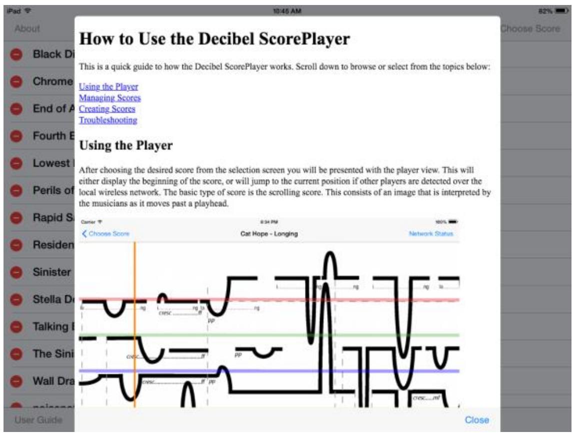


Score Name	Type the name here
Composer	w
Type	Scrolling Score
Duration	
Start Offset	
Read Offset	☐ Percentage
Image File	
Instructions File	
Parts	

Figure 8. Screenshot the sharing facility in iTunes, showing the Decibel score player (red for testing version, black for current commercially available version) and the place to add scores.

Whilst the lengths of each piece were set in the Score Creator, they could be altered for rehearsal purposes, and would reset to the original speed if the score was re-opened. A scrub button along the bottom of the screen provided easy access to any part of the score, and an information tab provided a drop down note for any instructions required for each individual score, as in Figure 9.

A User Guide is provided on the App to explain how it works, how to set up network, and how to create your own scores for the App. This includes a contact email for any enquiries or bug fix suggestions to be made, and point the user to a web site where instructional videos are provided [17]. On the iPad ScorePlayer, you can choose to see the score as a whole, or as individual parts. This function was first used on Hope's piece *Juanita Nielsen* for two violas, two cellos, piano, electric guitar and electronics, at the premiere performance of the Decibel ScorePlayer in September 2012 at the Perth Institute of Contemporary Arts. It became evident in rehearsals of *Juanita Nielsen* that the complex nature of the diagrams in the piece required magnification to be read accurately, and so the idea of providing separate parts was born. These can be added in the score creator in addition to a master score. The parts are coordinated with each other, even when you use the finger drag up and down on the screen to change between different parts.



How to Use the Decibel ScorePlayer

This is a quick guide to how the Decibel ScorePlayer works. Scroll down to browse or select from the topics below:

Using the Player
Managing Scores
Creating Scores
Troubleshooting

Using the Player

After choosing the desired score from the selection screen you will be presented with the player view. This will either display the beginning of the score, or will jump to the current position if other players are detected over the local wireless network. The basic type of score is the scrolling score. This consists of an image that is interpreted by the musicians as it moves past a playhead.

Figure 9. The 'User Guide' pop up, as seen over the list of works in the player (screen shot).

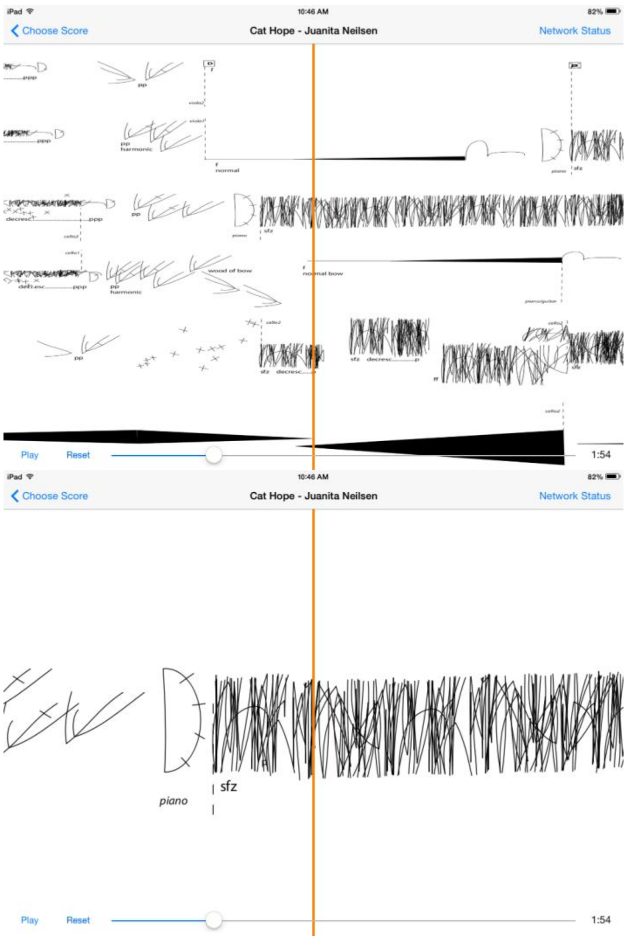


Figure 10. Hope's *Juanita Nielsen*. The top image shows the full score in the player. The lower image shows one part - in the same point of the piece, visible. The playhead is in the middle of the screen as the score goes in different directions. I red light in the top right flashes twice as a warning that the direction is about to change.

Figure 10 shows one of the parts at the same part in and next to the master score on the Decibel score player.

Early testing versions of the Decibel ScorePlayer were deployed using a program entitled Test Flight [18], which enabled Decibel to test new developments to the App. The composers for the ScorePlayer could make a standard scrolling score and parts in the Score Creator and test these in the player themselves. Whilst all the scrolling scores for the prototype player were adapted for the iPad player, new types of scores continued to be created for the Player, with the group using a 'developer' version of the App as new works, and updates to the player, could be tested before updates to the App on the iTunes store would be made.

Some scores were designed to read up and down, rather than left to right. This is useful when an instrument or group of instruments needs to be referred to spatially in the score. The shift can be done by simply locking the rotation on the iPad and turning it to a portrait, instead of landscape, view, so the score flows upwards, rather than from left to right. The Hope's piece *Broken Approach* (2014) for solo percussionist is read across a horizontal playhead, reflecting the spatial arrangement of the different percussion instruments in the performers set up, and is seen in **Figure 11**. Likewise, Hope's piano works *Chunk* (2010) and *Fourth Estate* (2014) use the playhead to reflect the horizontal presentation of the piano keyboard to the performer, the latter providing a shuffling mechanism that presents the composition differently each time, with eight different score images joining seamlessly in a different order each time the piece is opened on the ScorePlayer, using a 'tiling' approach for the different images. These scores have been named 'vertical scrolling scores'.

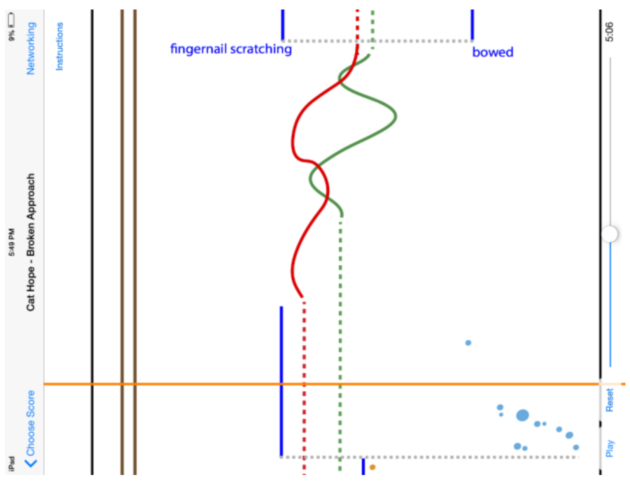


Figure 11. *Broken Approach* (screen shot). Note the presentation of the kit on the horizontal access, which is how it should be read.

Score Materials

The scores that can profit from being read in the Decibel ScorePlayer on the iPad are quite diverse. These include pieces that feature some elements of traditional notation, such as James Rushford's *Espalier* (2012) (also featured at the premiere concert of the ScorePlayer), featuring a stave and pitched note heads throughout, as shown in Figure 12.

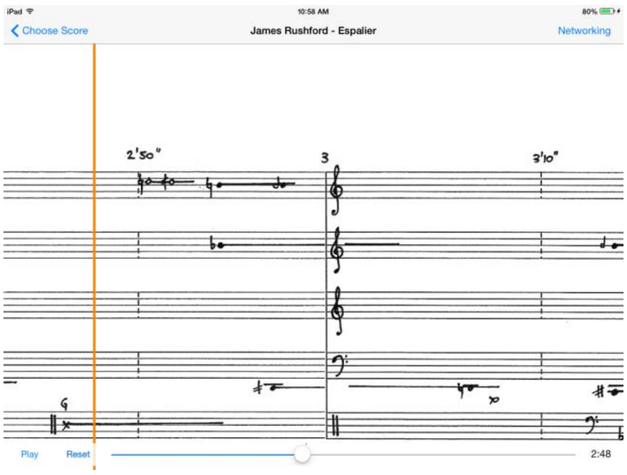


Figure 12. James Rushford's *Espalier* in the Decibel ScorePlayer (screen shot). Note the times on the top of the score - rendered superfluous by the ScorePlayer.

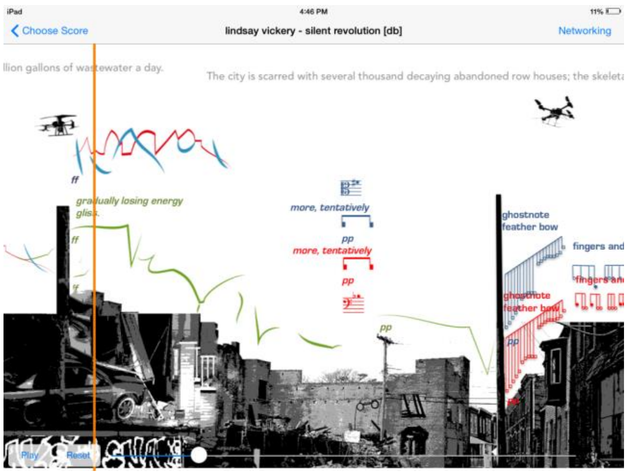
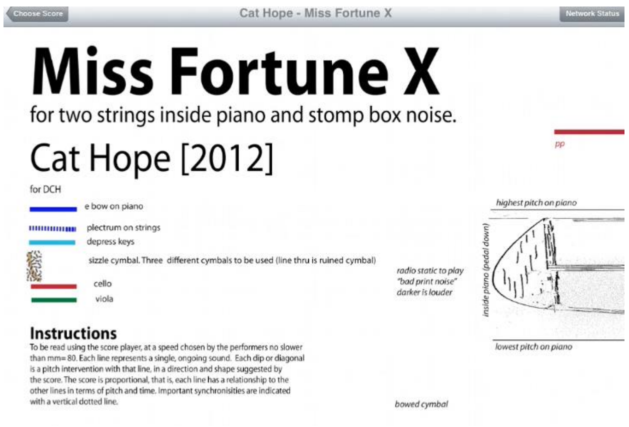

Choose Score
lindsay vickery - silent revolution [db]
Networking
llion gallons of wastewater a day.
The city is scarred with several thousand decaying abandoned row houses; the skelet
ff
gradually losing energy
gliss.
more, tentatively
pp
more, tentatively
pp
ghostnote
feather bow
fingers and
Play
Reset

Figure 13. Lindsay Vickery's *Silent Revolution* (screenshot) showing pictorial elements that are not read literally as part of the score.

An interesting development has been the use of pictorial imagery in the scores. Vickery's *Silent Revolution* (2013) includes images that are not 'read' by the musicians as such, but still provide useful information to the interpretation of the notations, as shown in **Figure 13.** These scores have come to be known as 'poctorial'. Hope's 'Miss Fortune X' (2012) uses the photocopy 'noise' from an old copy of a model aircraft plan as notation for radio static, as shown in Figure 14.

A variety of techniques have been engaged to generate the actual scores images - from Computer Assisted Design (CAD) software in Joe Stawarz's *Cells* (2012), coloured pencils in Mace Francis's *When Traffic Rises* (2012) and shades of graphite in Lyndon Blue's *Decabell* (2012). Chris Cobilis's *Forever Alone Together Or* (2012) features freehand text and interspersed with hand drawn colour shapes and written pitch suggestions, as shown in Figure 15.


Choose Score — Cat Hope - Miss Fortune X — Network Status

Miss Fortune X

for two strings inside piano and stomp box noise.

Cat Hope [2012]

for DCH

e bow on piano

plectrum on strings

depress keys

sizzle cymbal. Three different cymbals to be used (line thru is ruined cymbal)

cello

viola

Instructions

To be read using the score player, at a speed chosen by the performers no slower than mm= 80. Each line represents a single, ongoing sound. Each dip or diagonal is a pitch intervention with that line, in a direction and shape suggested by the score. The score is proportional, that is, each line has a relationship to the other lines in terms of pitch and time. Important synchronisities are indicated with a vertical dotted line.

Figure 14. Hope's *Miss Fortune X* score excerpt, (screen shot) showing the first issue Decibel ScorePlayer's welcome screen for the piece. This information was later replaced with an information dropdown tab. Note the copy 'noise' on the right hand side of the image.

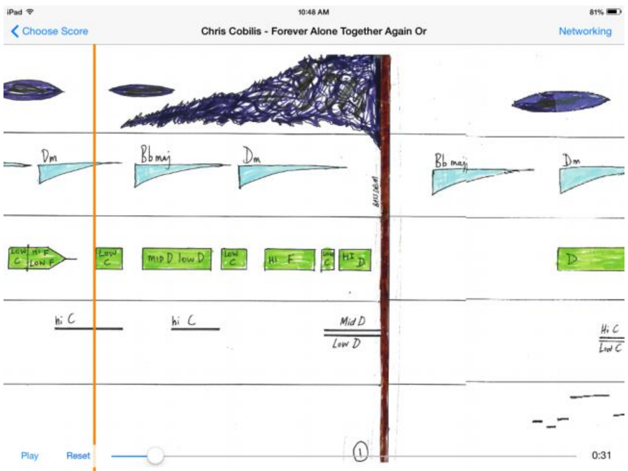


Figure 15. Chris Cobilis *Forever Alone Together Or* score excerpt (screen shot). Showing chords, notes and textural information.

Cobilis is an experimental electronics/singer songwriter who does not read or write traditional notation, and who created a work by recording it on a home recorder then 'drawing it' out over time. His work provides an excellent example of the wide variety of approaches to the design of scores that are featured in the Decibel Score Player, and potential it offers musicians who do not read or write conventional music notation.

ONGOING DEVELOPMENTS

The ScorePlayer paradigm has served as a springboard for other works. Decibel celebrated the centenary year of John Cage's birth by creating a score player for their 'Complete John Cage Variations Project' in 2012. This began as a laptop prototype, but was soon adapted to the iPad as a stand alone App. The score player involved the development of score generators for Variations I, II, III, IV, V and VI and packaging them with the remaining two Variations into the John Cage Variations App, in consultation with Cage's publishers, Peters Edition, and the John Cage Foundation in New York. Scheduled for release in conjunction with the groups recordings of the eight Variations on US label MODE in 2015, the App takes aspects of the Decibel ScorePlayer and applies them to the Variations, creating graphic scores by following and automating Cage's detailed processes. The result is very accurate and easy to read notations for each of the Variations, an example of which can be found in Figure 16**.** This example shows the graphic representation selected by Decibel of the data generated according to Cage's specifications around the placement of dots, lines and other shapes.[1] It also shows the similarity of the presentation on the iPad to the Decibel Score Player.

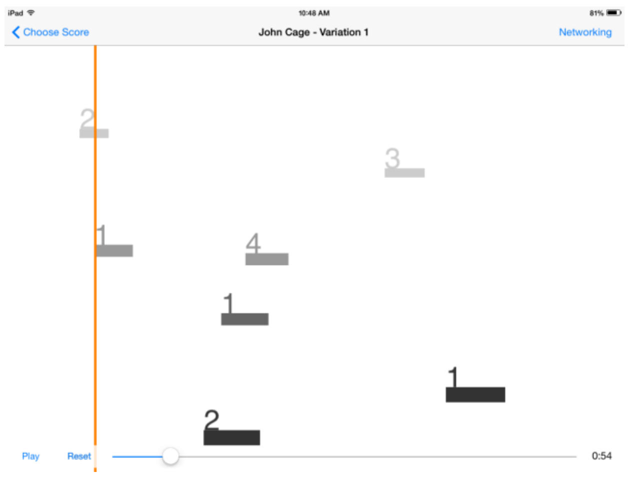


Figure 16. John Cage *Variation 1* score excerpt (screen shot) showing the graphic representation that scrolls in the Decibel 'The Complete John Cage Variations' ScorePlayer.

Australian sound poet Amanda Stewart's *Vice Versa* (2001) is a one-page text for live performances. Decibel adapted the work as a variable scrolling score by typesetting the text in the score player, facilitating reading from different directions, at different times. A range of differently coloured parts are provided, and occasionally text would appear scrubbed over, leaving the instruments to play the resulting shapes. Figure 17 shows the original score in the player, beside and a screen shot of how scrubbed over version. Experiments such as this one highlight the number of ways the simple reading device of the playhead can be used to create readable scores for different kinds of composition.

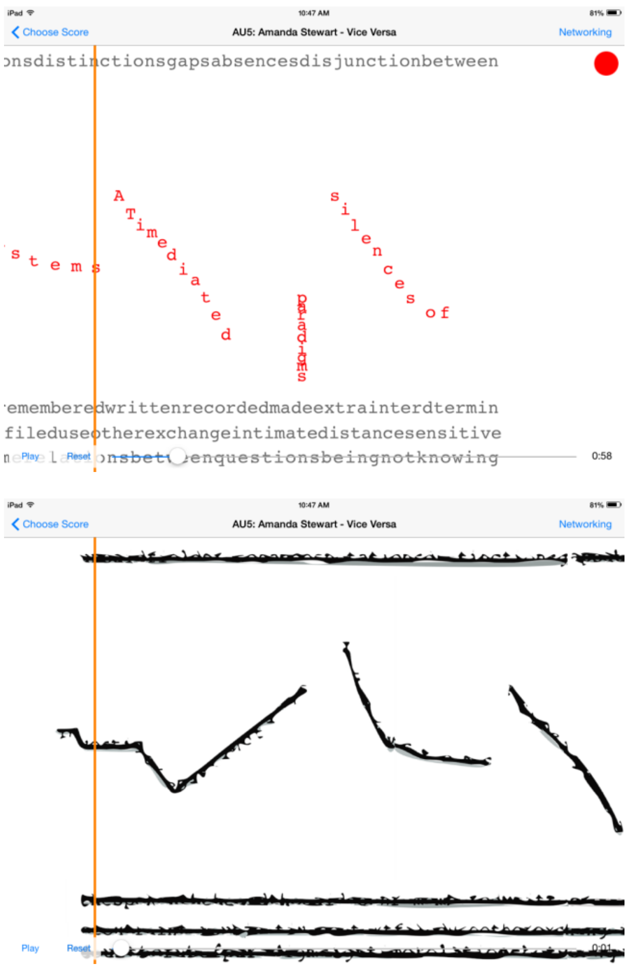

iPad
10:47 AM
81%
Choose Score
AU5: Amanda Stewart - Vice Versa
Networking
onsdistinctionsgapsabsencesdisjunctionbetween
A Timediated
silences of
stems
paradigms
ememberedwrittenrecordedmadeextrainterdtermin
fileduseotherexchangeintimatedistancesensitive
onsbetweenquestionsbeingnotknowing
Play
Reset
0:58
iPad
10:47 AM
81%
Choose Score
AU5: Amanda Stewart - Vice Versa
Networking
Play
Reset
0:01

Figure 17. Amanda Stewart's *Viceversa* (excerpt screen shot). The top image shows the score part (a different colour for each performer. The lower image shows the 'scrubbed out' text for instruments to play. The image goes left to right, and right to left in the player.

There are ongoing updates and bug fixes to the Decibel ScorePlayer, but the most recent developments have included the ability to create score files that embed a full quality audio track into the *.dsz* format, opening the possibilities for a huge range of works for instrument and tape that could be adapted for the Decibel ScorePlayer. Vickery created a score player for his 2009 performance of Denis Smalley's piece *Clarinet Threads* (1985) for clarinet and tape that enabled the score to be read accurately alongside playback [22]. Hope's *Signal Directorate* (2014) for bass instrument/s and prerecorded sounds, prototyped in MaxMSP by Vickery, is the first piece to use the iPad ScorePlayer to deliver the score synchronized with audio playback from within the iPad, and contained within the *.dsz* file. The Score Creator will

[1] A more detailed discussion of the implentatoin and the other Cage Variations can be found in a paper in the 2013 *Malaysian Music Journal* [19] and papers by Lindsay Vickery [20] and Cat Hope [21].

be updated to enable the most recent facilities enabled by the player. The next release will feature OSC compatibility and extra options for the *Talking Board* circle reading paradigm, allowing users to insert their own image and select the number of circles required for a performance, as shown in Figure 18. OSC will enable the data required to drive the electronics in this piece to be sent to another computer running the audio manipulation software.

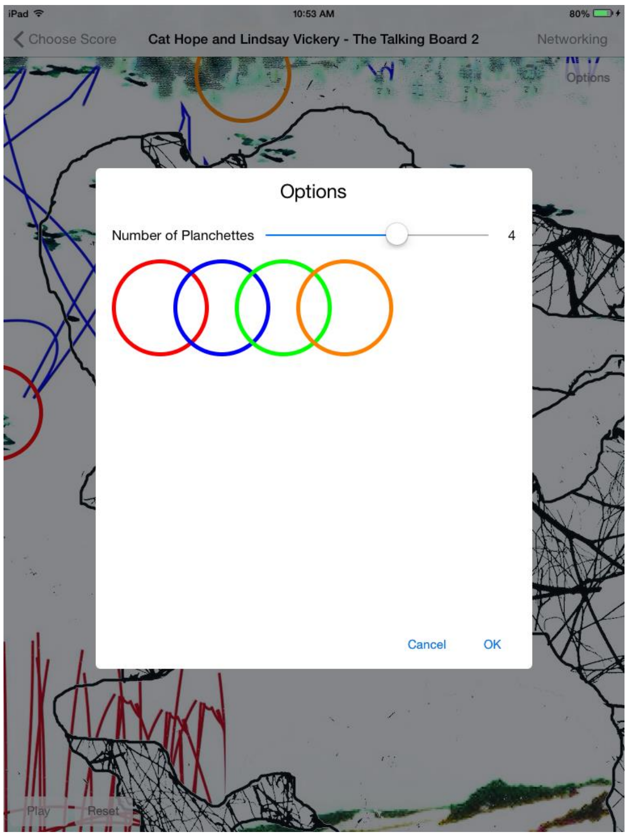


Figure 18. The 'circle selector' for *The Talking Board*, available when pressing the options tab.

In 2012, the first survey of Australian graphic music notation was curated by Cat Hope in two Australian cities, and featured a number of the scores for the scrolling score player presented as movies on a screen in a gallery [23]. These movie representations of scrolling scores are a fixed alternative for the reading of the scores, when a single projection is desirable. Synchronised with a live performance, they can also provide useful illustrations to how the works may be performed. However, in for larger ensembles or more complex parts, it is sometimes difficult to see the required level of detail and no variation of speed is easily possible.

CONCLUSIONS

Without any marketing support other than a few Facebook posts to the DecibelNewMusic page, and showcasing though tours, the Decibel ScorePlayer has sold 140 copies to date at AUD$2.99, not including the free copies the Decibel composers can access for the performances of their works. A visit to Malaysia by Decibel performing the 'John Cage Variations Project' using the bespoke application brought into sharp focus the need to make an Android version of the application, as Android appears to dominate the tablet computer market in large areas of Asia. However, funding for this development is yet to be found.

The potential for the Decibel ScorePlayer is substantial. There has been a recent resurgence of interest in graphic notation with some detailed examinations of practice [24] [25] [26] and an awareness of animated notations disseminated by online services such as YouTube and Vimeo. Yet it is quire remarkable how few of these developments engage with the full potential of digital representation. Further negotiations with publishers could result in a number of approaches for digital publication of extant works, and currently any composer can put their work in the ScorePlayer and publish it.

Research into the impact of reading different kinds of screen scores has recently commenced. Using eye-tracking equipment, Vickery has been comparing traditional paper notations and the different kinds of score formats developed in Decibel [27], leading to detailed examinations of the way readers process colour and movement in music notation.

The Decibel ScorePlayer embraces the possibilities of colour and graphic notations in digital score reproduction, as well as the interactive possibilities inherent in digital score creation and composition. Whilst currently a relatively simple device, the possibilities for its development are considerable. It does not claim to solve problems for all types of graphic notation, but makes certain types more efficient to read. Screen scores are in their infancy, and the way we understand colour and shape as musical information, as well as our ability to process moving information on computer screens requires further investigation [28]. The Decibel ScorePlayer represents the potential of group projects where composers, musicians, programmers and music curators can work together to extend the possibilities of available technologies.

Acknowledgments

Decibel new music ensemble consists of Cat Hope (artistic director, flutes, composer), Lindsay Vickery (composer, reeds, programmer), Stuart James (composer, piano, drum set, electronics, networking, programming), Aaron Wyatt (viola, violin and iOS programming), Tristen Parr (Cello, testing), Louise Devenish (percussion, testing). Lindsay Vickery created the first score player prototype. Stuart James built the Network Utility and lead the team for the Decibel 'Complete John Cage Variations' ScorePlayer. Aaron Wyatt is the programmer the iOS iPad Decibel ScorePlayer. The Decibel ScorePlayer project, and the Complete John Cage Variations Project were funded with assistance from Edith Cowan University.

REFERENCES

[1] C. Hope (Ed), *Audible Designs,* PICA Press, 2011. p. 6

[2] Decibel (n.d.). Decibel CV `http://www.decibelnewmusic.com/` (accessed 24 Jan, 2015).

[3] C. Hope (Ed), *Audible Designs*, PICA Press, 2011. p. 7

[4] L. Vickery, "Screening the Score" in C. Hope, (ed) *Audible Designs,* PICA Press, 2011, p. 86.

[5] H. Smith and R.T. Dean, *Practice-led research, research-led practice in the creative arts*, Edinburgh University Press, 2009, p. 56.

[6] R. B. Dannenberg, "Music representation issues, techniques and systems", *Computer Music Journal*, Vol 17, No.3, 1993, p 20–30.

[7] A. Clay, & J. Freeman, J. "Preface: Virtual Scores and Real-Time Playing", *Contemporary Music Review*, Vol 29 No. 1, 2010, p. 1.

[8] D. Kim-Boyle, "Real-time Score Generation for Extensible Open Forms.", *Contemporary Music Review*, Vol 29, No. 1, 2010, p. 3-15.

[9] D. Fober, Y. Orlarey, S. Letz, "INscore: an Environment for the Design of Live music scores". From `http://www.grame.fr/ressources/publications/INScore-ID12-2.pdf`

[10] Maxscore for MAX/msp and Abelton Live. `http://www.computermusicnotation.com`

[11] The Decibel Score Player `https://itunes.apple.com/au/app/decibel-scoreplayer/id622591851?mt=8`

[12] C. Hope and L. Vickery, "Visualising the score: Screening scores in real-time performance." *IME Journal,* Murdoch University, 2012.

[13] C. Hope, A. Wyatt, l. Vickery, "Reading Free Music" *Australasian Musicological Jounal*, 2015, in review.

[14] L. Vickery, "The Evolution of Notational Innovations from the Mobile Score to the Screen Score," *Organised Sound.* Vol 17 No. 2, 2012, p. 130.

[15] C. Hope, L. Vickery, "Screen Scores: New Media Music Manuscripts", *International Computer Music Conference.* Monty Adkins, Ben Isaacs. Huddersfield, UK. The International Computer Music Association, 2011, p. 224-230.

[16] A. Wyatt, C. Hope, L. Vickery, S. James, "Animated Music Notation on the iPad (Or: Music stands just weren't designed to support laptops)". *Proceedings of the International Computer Conference*, Perth, WA, 2013, p. 201- 207.

[17] The Decibel Score Player `http://www.decibelnewmusic.com/decibel-scoreplayer.html`

[18] Test Flight, `https://www.testflightapp.com`

[19] C. Hope, L. Vickery, A. Wyatt, S. James, "Mobilising John Cage: The Design and Generation of Score Creators for the Complete John Cage Variations I - VIII". *Malaysian Music Journal,* vol. 2 no. 1, 2013, p. 34-45.

[20] L. Vickery, C. Hope, S. James, "Digital adaptions of the scores for Cage Variations I, II and III". *International Computer Music Conference*. Ljubljana. International Computer Music Association, 2012, p. 426-432.

[21] C. Hope, S. James & L. Vickery, "New digital interactions with John Cage's Variations IV, V and VI.", *Proceedings of the 2012 Australasian Computer Music Conference*. Griffith University, Brisbane, Australia. Australasian Computer Music Association, 2012, p. 23-30.

[22] L. Vickery, "Mobile Scores and Click Tracks: Teaching Old Dogs New Tricks". *The Proceedings of the Australasian Computer Music Conference,* Australian National Unversity, 2010.

[23] C. Hope (Ed.), *Drawn from Sound,* Tura New Music, 2103.

[24] F. Feaster, *Pictures of sound: One thousand years of educed audio: 980-1980*. Dust to Digital, 2012.

[25] R. Johnson (Ed.), (1981). *Scores: An anthology of new music*. Schirmer, 1981.

[26] T. Suaer, *Notations 21,* Mark Batty, 2009.

[27] L. Vickery, "Exploring a Visual/Sonic Representational Continuum," in *Proceedings of the International Computer Conference*, Athens, Greece, 2014 in press.

[28] L. Vickery, "The Limitations of Representing Sound and Notation on Screen," *Organised Sound,* Vol 19, 2014 p. 226.

SPECTROMORPHOLOGICAL NOTATION: EXPLORING THE USES OF TIMBRAL VISUALIZATION IN ETHNOMUSICOLOGICAL WORKS

Mohd Hassan Abdullah
Sultan Idris Education University
mohd@upsi.edu.my

Andrew Blackburn
Sultan Idris Education University
andrew@fmsp.upsi.edu.my

ABSTRACT

Ethnomusicologists often face problem in precisely describing characteristic of a sound recorded in the fieldwork. Written explanation normally use metaphoric words to represent the timbral characteristics of a sound produced by ethnic musical instruments. But to what extent will the reader understand and perceive the sound based on the writer's explanation ? This study will explore the possibilities of using timbral visualization in the recognize of Malaysian traditional musical instruments. We introduce an instrument recognition process in solo recordings of a set of Malay traditional instruments (*gedombak*), which yields a high recognition rate. A large sound profile is used in order to encompass the different sound characteristic of each instrument and evaluate the generalization abilities of the recognition process.

INTRODUCTION

Ethnomusicology is a field of music which dealing with any musical activities and perspectives related to the specific music in a certain ethnic group. One of the perspectives of the study in this field is the organology of traditional instruments, and an evaluation of the sound produced by the instrument. Researchers who study in this field will normally describe in details about the sound and music performed with any particular instrument in a community.

Qualitative data gathered or recorded during the fieldwork is often be presented in scholarly printed publications in descriptive way. Researchers will try to describe the characteristic of a sound and try to make the reader to understand the sound without listens to the recording materials. Often the readers misunderstood the sound and perceive it differently from what the researcher mean. In short, the sound which is described in writing maybe perceived differently from the actual sound that the readers listen to. This project is a part of a larger research project (Spectromorphological notation: Notating the UnNotatable) exploring the creation of possible models of timbral notation. Using spectrograms allowing specific quantitative information of the timbre of traditional Malaysian instruments, relating them to the instruments organology has not been undertaken.

PROBLEM STATEMENT

For the past few decades, many ethnomusicologists had been trying to precisely describe the sound of any musical activities in many different ways. Some of them describe the sound of music in narrative way while some of them giving some meaning and using metaphor or other type of sound representation to describe the characteristic of a sound. Being as an ethnomusicologist, I also face difficulty in describing a sound of music from my fieldworks. The sound that I describe based on my understanding maybe perceived differently by other people. How could I overcome this situation ? Spectrogram have been used to objectively describe the organology of instruments of other culture but not in Malaysia.

In the field of ethnomusicology, we, the researchers are normally describing a sound base on what we perceive or using a local terminology to describe a particular sound. Most of the indigenous musical instruments are not constructed to any standard pic. Generally, almost all the ethnic musical instruments have different timbre and pitches. For example, in the Kompang (frame drum) ensemble of the Malay people, the sound of the kompang depends on the tautness and thickness of a skinhead as well as the size of the instrument. However, the kompang is also need to be tuned to the "*Bunyi yang diterima*" (acceptable sound) before it being played. A kompang ensemble normally consists of 15 to 25 players who performed on the similar instrument in interlocking rhythmic patterns to celebrate joyful occasions in the Malay community.

All the kompangs used in an ensemble is tuned to a certain pitch as closest possible from one to another. Even though there is no standard tuning set for the kompang, but an experienced kompang player is able to tell the "acceptable sound" of a kompang. The "acceptable sound" of a kompang to the players is described as (*kuat*) loud, (*gemersik*) penetrating, (*tajam*) sharp and (*tegang*) taut. How can one precisely understand and perceive the sound of a kompang as loud, penetrating, sharp and taut ? Can one precisely describe the sharpness sound of the kompang ? As the sound of any indigenous musical instruments are mostly not standardize in nature, there is a need to find ways on how to identify and recognize the "acceptable sound" of any particular musical instruments especially for the beginners and who are not expert in that field.

Moreover, contemporary Western arts and traditional music notation is usually linked to an analysis and the semiotic representation of the musical elements of melody and harmony (vertical and horizontal pitches) using common music notation. Precise pitch indications are "rounded out" into the twelve semitones of this system, unable to accommodate more the precise subtleties of sound that are inherent in all music tradition. Further, Musical parameters such as articulation (attack, decay, sustain and release) and dynamics (volume or intensity) are loosely indicated through the use of staccato or phrase markings for articulations or dynamic marks (forte, piano, crescendo, diminuendo, ect.).

Representation of other significant musical elements such as tone and colour (timbre) are largely limited to instrumental naming or specific performance directions (sul ponticello – play near the bridge for string instruments). The lack, along with the difficulties of definition and understanding of timbre are increasingly recognized within both new music and traditional music fields.

AIMS OF RESEARCH

This project will explore the creation of a model for the timbral and performance notation of acoustic music that notates more content details of the various elements of sound. Of significance for ethnomusicologists who working in this field, will be the use of spectrographic notation leading to the creation of an authentic and precise transcription library and catalogue inclusive of all musical elements. Such a catalogue will lead to a greater understanding of the individual and unique spectral and tuning characteristics of traditional Malay musical instruments. This method will be applied to instruments such as kompang, gedombak, gendang, serunai, and rebab. Knowledge and experience of creating spectrograms of the Malay traditional instruments will then be applied into forefront of music making using these possible model and system.

RESEARCH QUESTIONS

In exploring the possibilities of using the spectrographic features in ethnomusicological study, there are many related questions can be addressed.

i. How can an ethnomusicologist describe the sound of a musical instrument ?

ii. What are the elements that ethnomusicologists require from a notation system and how can these be represented ?

iii. What kind of notational/transcription system can possibly describe precisely the musical sound of traditional instrument ?

iv. What organological elements are common or exclusive to each instrument and how can they best be identified and analyzed ?

v. Can spectrographic analysis and software be used to provide a method for defining and identifying unique qualities of Malay traditional Instruments ?

vi. Can this information be used to describe and notate the specific individuality of sounds materials and performance methods in ways that expand the range and musical vocabulary of the ethnomusicologist ?

vii. What parameters of analysis can be defined to provide useful and universally understood symbols using spectrographic software ?

viii. How can this notational system help scholars, musicians, instrument makers and others in identifying a prefer timbre for any particular Malay traditional instrument ?

ix. What other knowledge can be drawn from this ?

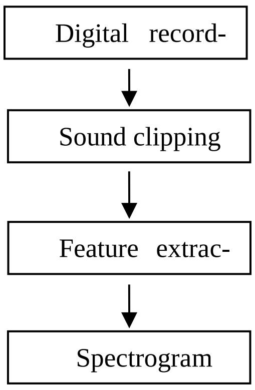
METHODOLOGY

In conducting this study, various methods will be utilized in getting the useful data and information to answer the research question. Generally, methods will be grounded in practice. While exploring all the possibilities of using spectrographic as a tool to describe the characteristic of a sound, researchers will analyze and think through practice. This method is also always referred as practice-led research. Three phases will cumulatively document, analyze, apply and reflect on project activities and outcomes. Critical reflection is a key criterion of the research, supported by textual analysis.

Research activities include identifying the sound characteristic of a few selected Malay traditional musical instruments such as *gedombak* (goblet drum), *gendang* (cylindrical drum), *kompang* (frame drum), *serunai* (double-reed oboe type instrument), and *rebab* (spike-fiddle). Each of them will be performed by the expert players for the recording purpose. A few software packages will be utilized to visualize the sound characteristic of each instrument. From the spectrograms, the researchers will then think on how it can be applied in ethnomusicological works.

THE RESULT

Many samples of Malay traditional instrument sound have been recorded in the form of wave file. The instruments include the *gedombak*, *gendang*, *serunai*, *geduk and gong* have been performed by the expert players both solo and ensemble for the recording purpose. Three software packages – Eanalyse, Sonic Visualiser and Praat- have been utilized to visualize the recorded clips.



A series of recording done on the instrument demonstrated that the underlying phonetic representation of an unknown utterance can be recovered almost entirely from a visual examination of the spectrogram. The most common format is a graph with two geometric dimensions: the horizontal axis represents time; as we move right along the x-axis we shift forward in time, traversing one spectrum after another, the vertical axis is frequency and the colors represent the most important acoustic peaks for a given time frame, with red representing the highest energies, then in decreasing order of importance, orange, yellow, green, cyan, blue, and magenta, with gray areas having even less energy and white areas below a threshold decibel level.

Figure 1 shows the spectrogram of a gedombak beaten in a series of single tapping in the middle of the skinhead. What can we learn from this spectrogram ?

After receiving clarification from the expert player, the 4th beat of the sound is the most preferred sound by the expert player. One can analyze from the colours and density of the spectrogram to tell the characteristic of the preferred sound.

Different filters have been applied to the one recording of the gedombak. The results show different features of the sound performed on the same instrument. Below are the example of different spectrograms show different features and characteristic of a sound performed on Malay traditional instrument.

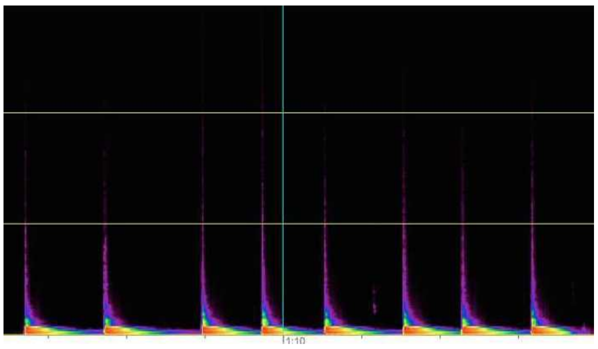


Figure 1. Spectrogram of a gedombak

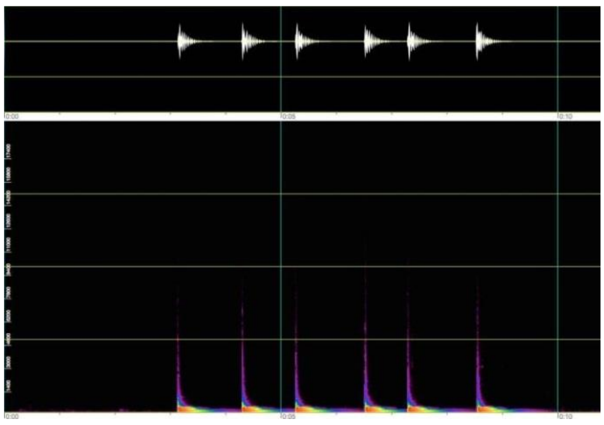


Figure 2. Spectrogram of a Gedombak with waveform.

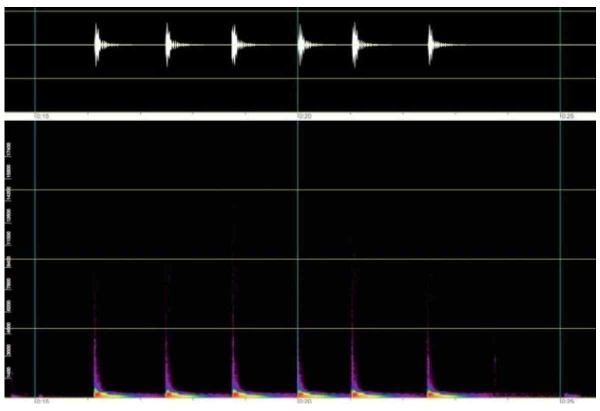


Figure 3. Spectrogram of a smaller size of Gedombak.

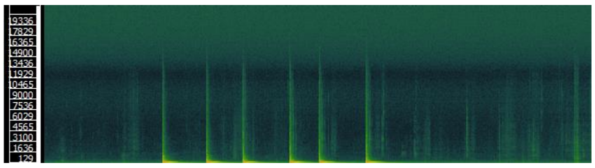


Figure 4. Zoom in Spectrogram of a Gedombak

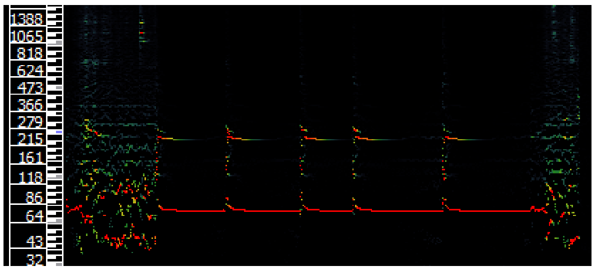


Figure 5. Spectrogram of a Wayang Kulit ensemble

DISCUSSION AND SUGGESTION

The spectrograms of gedombak (goblet drum) in Wayang Kulit ensemble (Shadow puppet play) above are an initial attempt to explore the potential of a spectrogram as a performative notation. The gedombak is indicated as the large regularly spaced columns of Figure 4. In Figure 5, the horizontal lines represent the melodic lines of the serunai. The pitch variations and arabesque ornamentation so characteristic of the instrument is also visible. This begins to plan to use spectrograms of individual instruments to identify preferred timbral quality of instruments for use in specific musical/dramatic contexts–why a Wayang Kulit 'master' selects one instrument over another in a given performance ?

Just what is timbral notation - gestural, purely tonal, semiotic etc. ? This opens the potential for different forms and styles. In the ethnomusicological context - instrumental profiling of timbre, linked to the organology of the instrument is both applicable in Malaysia and opens ideas that appear to inform ideas and practices in the other sub-projects of the overall research project.

CONCLUSION

In this paper, we dealt with recognition of sound samples and presented several methods to improve recognition results. Tones are extracted from a database of Malaysian traditional musical instruments (gedombak, gendang, serunai, etc.). We use two different parameters in the analysis. From the experiments, we could observe evident results for spectrogram and autocorrelation. Maximum and minimum values of amplitude for autocorrelation for all musical instruments have different ranges. Spectrogram of gedombak is much larger than those of gendang and serunai. Result shows that the estimation of spectrogram and autocorrelation reflects more effectively the difference in musical instrument.

Acknowledgements

We would like to thanks the Ministry of Education, Malaysia for awarding the research grant under the Fundamental Research Grant Scheme. We also wish to thanks the Sultan Idris Education University through their Research Management and Innovation Centre who coordinating all the research activities. We also would like to express our gratitude to the dean and all management team in the Faculty of Music and Performing Arts at the same university for providing all the facilities used in this research. Lastly, we thanks to all the people who either directly or indirectly involve in this research. Without that, this study would not have been possible.

BIBLIOGRAPHY

[1] K. D. Martin : *Sound-Source Recognition: A Theory and Computational Model*, Ph. D. thesis, MIT, 1999.

[2] A. Livshin, X. Rodet : "Musical Instrument Identification in Continuous Recordings", *Proc. of the 7th Int. Conference on Digital Audio Effects* (DAFX-04), Naples, Italy, October 5-8, 2004

[3] A. Eronen, A. Klapuri: "Musical Instrument Recognition Using Cepstral Coefficients and Temporal Features", *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP* 2000, pp. 753-756

[4] T. Kitahara, M. Goto, H. Okuno: "Musical Instrument Identification Based on F0-Dependent Multivariate Normal Distribution", *Proc. of the 2003 IEEE Int'l Conf. on Acoustic, Speech and Signal Processing (ICASSP '03)*, Vol.V, pp. 421-424, Apr. 2003

[5] A. Eronen: "Musical instrument recognition using ICA-based transform of features and discriminatively trained HMMs", *Proc. of the Seventh International Symposium on Signal Processing and its Applications, ISSPA* 2003, Paris, France, 1-4 July 2003, pp. 133-136

[6] G. De Poli, P. Prandoni: "Sonological Models for Timbre Characterization", *Journal of New Music Research*, Vol 26(1997), pp. 170-197, 1997

DENM (DYNAMIC ENVIRONMENTAL NOTATION FOR MUSIC): INTRODUCING A PERFORMANCE-CENTRIC MUSIC NOTATION INTERFACE

James Bean
University of California, San Diego
jsbean@ucsd.edu

ABSTRACT

In this paper, I describe the state of development for an automatic music notation generator and tablet-based graphical user interface. The programs currently available for the automatic generation of music notation are focused on the compositional and theoretical aspects of the music-making process. **denm** (dynamic environmental notation for music) is being designed to provide tools for the rehearsal and performance of contemporary music. All of the strategies underlying these tools are utilized by performers today. These strategies traditionally involve the re-notation of aspects of a musical score by hand, the process of which can be detrimentally time-consuming. Much of what performers re-notate into their parts is composed of information latent in the musical model—the musical model which is already being represented graphically as the musical score. **denm** will provide this latent information instantaneously to performers with a real-time music notation generator.

1. BACKGROUND

Commercial music typesetting software, such as Finale and Sibelius, are the most common tools for creating musical scores, which require a musician to manually enter musical information via graphical user interfaces. There are cases, for example when compositions are algorithmically generated, where the process of manually entering musical information in this manner is inefficient. As such, programs have been designed to create musical scores where the input from the user is text-based, generated by algorithmic processes, or extracted from spectral analyses.

Most Automatic Notation Generators (ANGs) [1] create a static image, either to be read by musicians from paper, or from a screen displaying it in a Portable Document Format (PDF) representation. LilyPond [2] and GUIDO [3] and convert textual descriptions of music into musical scores. Abjad [4] and FOMUS [5] generate musical score information that can be graphically rendered by LilyPond. OpenMusic [6], Bach [7], PWGL [8] / ENP [9], and JMSL / JSCORE [10] are software tools for composers that generate music notation as part of the compositional process. Belle, Bonne, Sage [11] is a vector graphics library for music notation that enables the drawing of advanced notational concepts not offered by traditional music typesetters. Music21 [12] is a music analysis program that creates score information to be graphically rendered in LilyPond. Spectmore [1] maps spectral analysis information onto a musical score. A few newer ANGs, such as INScore [13] and LiveScore [14], generate animated musical notation for screen representation.

Thus far, ANGs generate static scores that are useful to composers and theorists, and animated scores that are useful for those performing in real-time (described as the *immanent* screen score paradigm by Hope and Vickory [15]). No ANGs specifically target the rehearsal processes of contemporary music performers (a process described as *interpretive* by Hope and Vickory).

I have found that the most critical period for the success of my own works is the rehearsal processes with performers. Performers spend a considerable amount of time in individual rehearsal and group rehearsal settings, and have developed extensive strategies to comprehend, embody, and execute the propositions of composers (see: [16], [17], [18], [19]). Many of the cues that performers notate into their parts are composed of information latent in the musical model—the musical model which is already being represented graphically as the musical score.

denm is software written for iOS devices in the Swift language using the Core Frameworks that enables performers to reap the benefits of these rehearsal strategies without the high cost normally associated with preparing them. Both the musical model and graphical rendering engine are built from scratch to best utilize the touch interfaces of tablet computers. The initial development of **denm** began in the

Javascript language to control the graphical output of Adobe Illustrator. This first phase of development served as research into the systematic organization of musical scores and the programmatic drawing of musical symbols. The vision of this project necessitates animated graphical content, which ultimately required the rewriting of all source code. There are certain features that were prioritized in the initial phase of development [1] that will ultimately be rewritten in an animated context.

2. GRAPHICAL USER INTERFACE

More and more performers are reading music from tablet computers. Software applications like ForScore display a PDF representation of a musical score, allowing a performer to turn pages with a Bluetooth footpedal, as well as to annotate scores with handwritten or typed cues. Performers are able to store many scores on a single device, simplifying the logistics of performing many pieces. Because the PDF contains no reference between graphical musical symbols and their musical functions, the degree to which a player is able to interact with this medium in a musical context is limited.

Many of the cues that performers handwrite in their parts are simplified versions of other players' parts [20]. These types of cues are being reentered by the performer, even though this information is already retrievable from the data that is being graphically represented by the score. The primary objective of **denm** is to expose the structures underlying the music to performers with little cost of access.

2.1 Graphic Design Priorities

The graphic design style of **denm** is minimalist, with as few graphical ornaments as possible. Rather, variations in color, opacity, line-thickness, and other graphical attributes are used to differentiate an object from its environment. In some cases, the variations in graphical attributes serve to differentiate an object's current state from its other potential states. Basic musical symbols, such as clefs and accidentals, have been redesigned to implement this universal design philosophy.

Many of the design choices of standard music notation generators are made with printing in mind. The choices made in **denm** are optimized for display on a screen. The use of thin lines and color is problematic for printers to represent, though these techniques are quite successful with high quality displays.

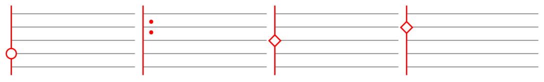


Figure 1. Design of clefs: treble, bass, alto, tenor.

2.1.1 Clef Design

Traditional clefs take up a considerable amount of horizontal space. The width of traditional clefs is problematic for the spacing of music, particularly when the preservation of proportionate music spacing is a high priority. The minimalist clefs in Fig. 1 take up very little horizontal space. Clefs are colored specifically to enable a differentiation of the clef from the surrounding context and subtle breaks are made in the the staff lines to accentuate the clefs' presence. Staff lines are gray, rather than black, enabling the creation of a foreground / background relationship between musical information carrying objects (notes, accidentals, articulations, etc.) and their parent graph.

2.1.2 Accidental Design

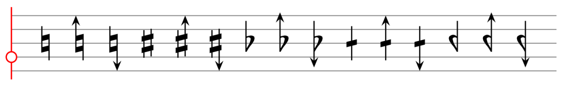


Figure 2. Design of accidentals.

Accidentals, as can be seen in Fig. 2, are drawn programmatically, as opposed to being instances of glyphs from a font. The advantage to uniquely drawing each accidental is that small vertical adjustments can be made to individual components of the object (e.g. body, column(s), arrow) in order to avoid collisions in a more dynamic fashion than is usually implemented in other music notation software [2]. Burnson's work with collision detection of musical symbols [22] serves as an example for the similar work to be approached in continued development.

2.1.3 Rhythm Design

In cases of embedded tuplets, beams are colored by the events' depth in the metrical hierarchy. Ligatures, as seen in Fig. 3, connect tuplet brackets to their events to clarify jumps in depth.

[1] Automatically generated woodwind fingering diagram, string tablature to staff pitch notation conversion, and automatically generated cues.

[2] The initial development of **denm** in Adobe Illustrator-targeted Javascript prioritized this dynamic accidental collision avoidance. Extending the traditional process of avoiding of accidental collisions by stacking accidentals in multiple vertical columns [21], individual adjustments are made to the graphics of the accidentals themselves. Many accidental collisions that traditionally warrant horizontal movement of the objects can be avoided with a single or several small adjustments to individual components of each accidental. Avoiding unnecessary horizontal movement of accidentals makes retaining proportionate music spacing more feasible. More rigorous study of the effects of readability of slightly adjusted accidental graphics is to be undertaken throughout the near-term development of **denm**.

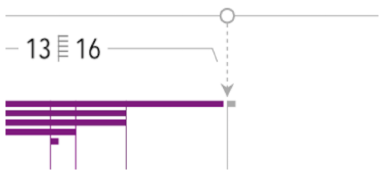


Figure 3. Design of beams and tuplet bracket ligatures

Small graphics, as seen in Fig. 4, indicate the subdivision value that clarify the values of a tuplet. The style of the straight beamlets in the tuplet bracket subdivision graphics mirror the straight beams of rhythms, without the visual noise of traditional flags. Further, the line-thicknesses of beams in the graphics are inversely proportional to their subdivision value, aiding in their visual differentiation. Left edges of tuplet brackets are straight, while right edges of tuplet brackets are angled.

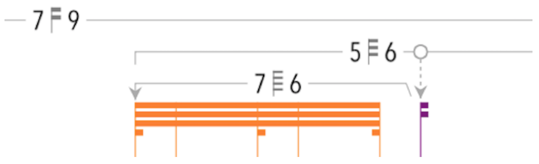


Figure 4. Design of tuplet bracket label graphics

2.2 User Interaction Design Priorities

Many cues that performers notate into their scores are useful at certain points of the learning and rehearsal process, but become less useful at different points in the process. The user interaction design style of **denm** enables performers to determine what musical information is displayed at any point. Performers touch the screen score directly to show or hide certain objects. More advanced user interface strategies will be developed as the underlying analytical procedures (some of them seen in Sec. 3) are implemented.

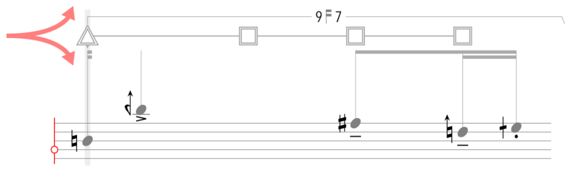


Figure 5. Screenshot of Metronome Graphic revelation.

For example, when a user decides to show a stratum of Metronome Graphics (described further in Sec. 2.3.2), as can be seen in Fig. 5, the entire page recalculates its dimensions, to ensure that the Metronome Graphics take up only the space that they need to. When the user decides to hide that stratum of Metronome Graphics, the layout is recalculated once again, the Metronome Graphics are hidden, and the space they were occupying disappears.

The layout of **denm** is organized as a hierarchy of embedded boxes that recalculate their heights based on what the user elects to show or hide within them. Fig. 6 shows these vertically accumulating boxes. Each box defines its own padding, keeping layout separation consistent for each object.

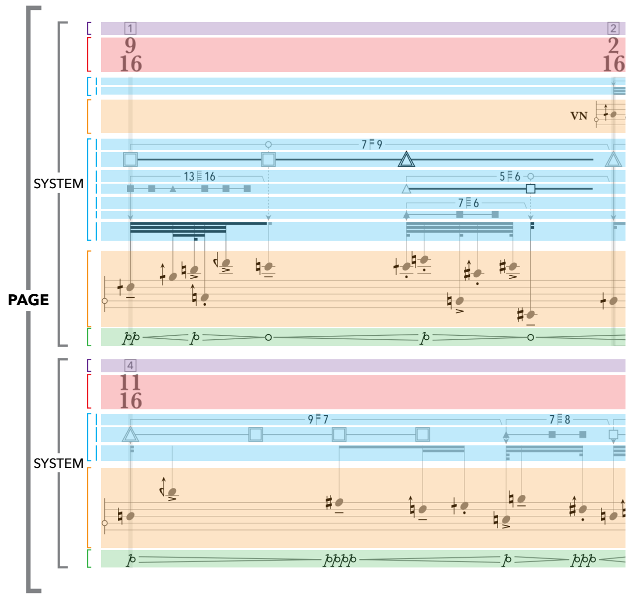


Figure 6. Layout Organization.

2.3 Rhythm Features

2.3.1 Metrical Grid

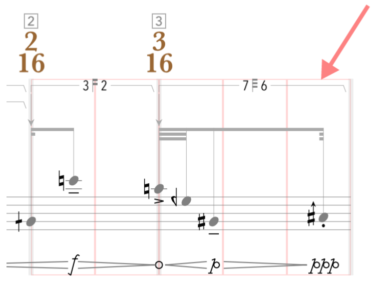


Figure 7. Screenshot of a Metrical Grid.

A performer can tap the time signature of any measure to reveal a grid showing the beats of that measure. This provides a quick reference for the relationship of complex rhythmic events to a global tactus. Quickly drawing lines at the point of each beat in a measure is often the first thing a performer does when receiving a new piece of rhythmically complex music [16], [20].

2.3.2 Metronome Graphics

When a user taps any point in a rhythm, graphics are displayed indicating the best way to subdivide a rhythm (reduced to sequences of duple- and triple-beats). Duple-beats

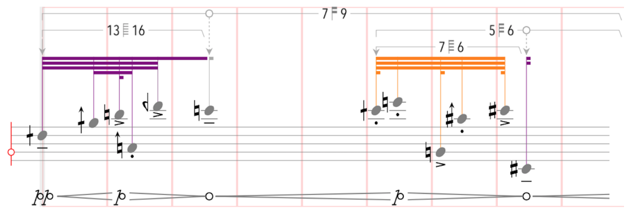


Figure 8. Screenshot of another Metrical Grid.

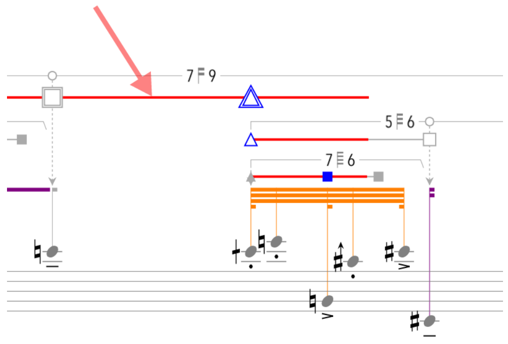


Figure 10. Screenshot of metronome playback.

are represented as rectangles and triple-beats are represented as triangles. Each subdivision-level (e.g. 8th, 16th, 32nd, etc.) has its own graphic, which is a uniquely styled version of the duple- and triple-beat primitives, making the subdivision-level of the metronome understandable at a glance. The process of generating these subdivision references can be seen in Sec. 3.1

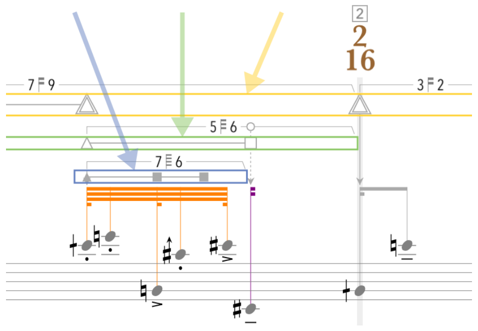


Figure 9. Screenshot of Metronome Graphics.

2.3.3 *Metronome Visual Playback*

Performers often create click-tracks for learning, rehearsing, and performing rhythmically complex music. Currently, the Metronome Graphic objects can be played-back when a performer taps on the time signature for a measure. The Metronome Graphics "click" by flashing a different color in time. An animated bar progresses from left to right at the speed prescribed by the current tempo of the music. This process has yet to be implemented with other objects in the system, though this will continue to be developed.

As development continues further, a performer will be able to extract any portion of the musical part and rehearse it with the visual click-track of the Metronome Graphics at any tempo. Ultimately, an audio element will be integrated into this metronome process, with sonic attributes mirroring those of the visual Metronome Graphics, to represent subdivision-level and placement in the Metrical Analysis hierarchy (as described in Sec. 3.1).

2.4 Other Players' Parts

Performers often notate aspects of the parts of the other players in an ensemble context. Because this information already exists in the musical model, it can be graphically represented immediately. This feature is currently implemented at a proof-of-concept level. Fig. 11 shows the process of verifying the automatic layout recalculation needed when inserting new musical material. In this case, hard-coded musical material is inserted into the layout when a measure number is tapped by a user.

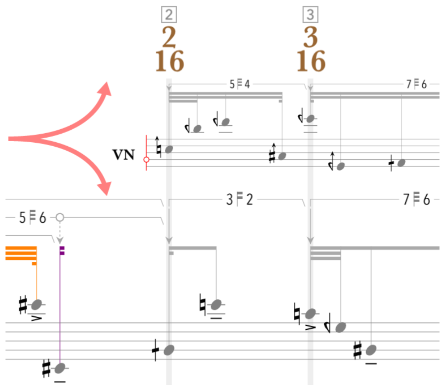


Figure 11. Screenshot of cue revelation.

3. MUSIC ANALYSIS ALGORITHMS

In order to provide performers with rehearsal tools in real-time, robust analysis tools must be developed.

3.1 Metrical Analysis

denm analyzes rhythms of any complexity. The result of this analysis is an optimal manner in which to subdivide the rhythm. Information like syncopation and agogic placement of events can be ascertained from this process. This process can be seen in Alg. 1.

Rhythm in **denm** is modeled hierarchically. The base object in this model is the DurationNode. Any DurationNode that contains children nodes (e.g. traditional single-depth rhythm, or any container in an embedded tuplet) can be analyzed rhythmically. The result of this analysis of a single container node is a MetricalAnalysisNode (a DurationNode itself with leaves strictly containing only duple- or triple-beat durations). MetricalAnalysisNodes are the model used by the Metronome Graphics, the graphical representation of which is described in Sec. 2.3.2.

Algorithm 1 Metrical Analysis

```
1: $durNodes$ ←DurationNode.children
2: $parent$ ←Root MetricalAnalysisNode
3: function ANALYZE($durNodes, parent$)
4:     $s \leftarrow durNodes.sum()$
5:     if $s = 1$ then
6:         $child$ ←MANode(beats: 2)
7:         ▷ subdivision level $* = 2$
8:         ▷ add $child$ to $parent$
9:     else if $s <= 3$ then
10:        $child$ ←MANode(beats: $s$)
11:        ▷ add $child$ to $parent$
12:    else if $4 <= s <= 7$ then
13:        $p \leftarrow prototypeWithLeastSyncopation$
14:        for $pp$ in $p$ do
15:            $child$ ←MANode(beats: $pp$)
16:            ▷ add $child$ to $parent$
17:        end for
18:    else if $8 <= s <= 9$ then
19:        $p \leftarrow prototypeWithLeastSyncopation$
20:        if $p$ contains values $> 3$ then
21:            for $pp$ in $p$ do
22:                $part \leftarrow durNodes$ partitioned at $pp$
23:                $newParent$ ←MANode(beats: $cc$)
24:                analyze($part$, $newParent$)
25:            end for
26:        end if
27:    else
28:        ▷ create array of all combinations
29:        ▷ of values $4 <= v <= 7$ with sum of $s$
30:        $c \leftarrow combinationWithLeastSyncopation$
31:        for $cc$ in $c$ do
32:            $part \leftarrow durNodes$ partitioned at $cc$
33:            $newParent$ ←MANode(beats: $cc$)
34:            analyze($part$, $newParent$)
35:        end for
36:    end if
37: end function
```

A MetricalAnalysisPrototype is a sequence of two or three elements, each of which having a duple- or triple-beat duration. Values between and including 4 and 7 have a dedicated list of prototypes [3], each of which can be compared against any rhythm with a relative durational sum of the same value, the process of which can be seen in Alg. 2.

Algorithm 2 Syncopation

```
1: $d \leftarrow durationNodes.cumulative()$
2: ▷ e.g. $[4, 5, 7] \leftarrow [4, 1, 2].cumulative()$
3: $p \leftarrow prototype.cumulative()$
4: ▷ e.g. $[2, 4, 7] \leftarrow [2, 2, 3].cumulative()$
5: $syncopation \leftarrow 0$
6: function GETSYNCOPATION($d, p$)
7:     if $d[0] = p[0]$ then
8:         ▷ Rhythm beat falls on prototype beat
9:         ▷ No syncopation penalty added
10:        ▷ Adjust $d$ and $s$ accordingly
11:        $getSyncopation(d, s)$
12:    else if $d[0] < p[0]$ then
13:        ▷ Rhythm beat falls before prototype beat
14:        ▷ Check if next duration falls on prototype beat
15:        if $delayedMatch$ then
16:            ▷ No syncopation penalty added
17:        else
18:            $syncopation \leftarrow syncopation + penalty$
19:        end if
20:        ▷ Adjust $d$ and $s$ accordingly
21:        $getSyncopation(d, s)$
22:    else
23:        ▷ Rhythm beat falls after prototype beat
24:        ▷ Check if next prototype value falls on duration
25:        if $delayedMatch$ then
26:            ▷ No syncopation penalty added
27:        else
28:            $syncopation \leftarrow syncopation + penalty$
29:        end if
30:        ▷ Adjust $d$ and $s$ accordingly
31:        $getSyncopation(d, s)$
32:    end if
33: end function
```

Cuthbert and Ariza [12] apply their Metrical Analysis process to beaming in the score representation of rhythms. This strategy will be the model for continued development in **denm**, extending to cases of arbitrarily-deep nested-tuplet rhythms.

[3] List of prototype sequences by relative durational sum:
$4 : (2, 2)$
$5 : (3, 2), (2, 3)$
$6 : (2, 2, 2), (3, 3)$
$7 : (2, 2, 3), (3, 2, 2), (2, 3, 2)$
$8 : (4, 4), (3, 3, 2), (2, 3, 3), (3, 2, 3)\ *$
$9 : (3, 3, 3), (4, 5), (5, 4)\ *$
$*$ Rhythms with relative durational sums of 8 and 9 are compared against all of these combinations shown here. If the combination with least syncopation contains only values of 2 or 3, a MetricalAnalysisPrototype is generated with children containing Durations of duple- and triple-values. In the case that the combination with least syncopation contains values > 3, internal MetricalAnalysisNodes are created with the Durations of these values. The original DurationNode array is partitioned at points determined by the combination. The MetricalAnalysis process is then applied for each partition, and MetricalAnalysisNode leaves with duple- and triple-value Durations are added to each of these internal MetricalAnalysisNodes.

3.2 Pitch Spelling

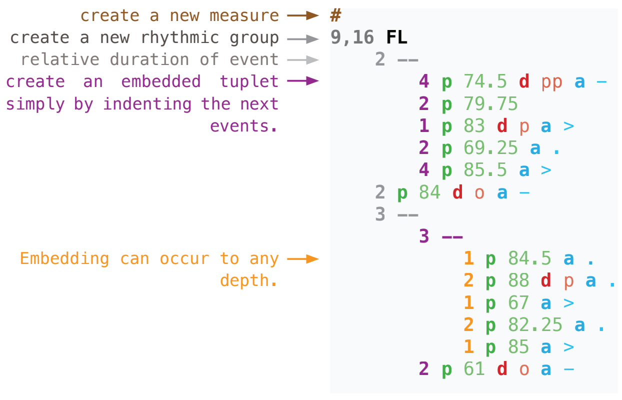
Effective pitch spelling is critical in the process of generating staff notation for music that is algorithmically composed or extracted from spectral analyses. Algorithms for pitch spelling within tonal musical contexts have been compared by Meredith [23] and Kilian [24]. The musical contexts that **denm** is most immediately supporting are rarely tonal. More often these musical contexts are microtonal. The preferences in the current tonally-based pitch spelling algorithms, however, are defined by establishing tonal center.

The benefits of tonal-center-based pitch spelling are lost in atonal musical contexts. Rather, primitive intervallic relationships are preserved objectively, rather than being subjected to the requirements of a tonal center. When spelling pitches in microtonal contexts, other pragmatics arise that influence the decision-making process.

The short-term goal of the pitch spelling process in **denm** is to spell microtonal polyphonic music up to the resolution of an 1/8-tone (48 equal divisions of the octave). In time, this process may be extended to accommodate other tuning systems. The development of this microtonal pitch spelling procedure is in process. Currently, dyads of any combination of resolutions (1/2-tone, 1/4-tone, 1/8-tone) are spelled correctly, though more rigorous testing is underway to verify this. Further development of this pitch spelling algorithm into homophonic and polyphonic microtonal contexts will incorporate aspects of Cambouropoulos' shifting overlapping windowing technique [25].

4. INPUT FORMATS

At this point in development, there is a working prototype input text format. All figures in this paper have been created with this input text format. A long-term priority for development is to build conversion from common music interchange formats, such as musicXML [26], into the native **denm** text input format. This conversion will enable composers to generate and input music in the style that best serves them, while they benefit from a performer-facing interactive graphical user interface.


create a new measure
create a new rhythmic group
relative duration of event
create an embedded tuplet simply by indenting the next events.
Embedding can occur to any depth.

```
#
9,16 FL
    2 --
        4 p 74.5 d pp a -
        2 p 79.75
        1 p 83 d p a >
        2 p 69.25 a .
        4 p 85.5 a >
    2 p 84 d o a -
    3 --
        3 --
            1 p 84.5 a .
            2 p 88 d p a .
            1 p 67 a >
            2 p 82.25 a .
            1 p 85 a >
        2 p 61 d o a -
```

Figure 12. Demonstration of text input syntax.

5. FUTURE WORK

The current development of **denm** is focused on building robust musical operations within the musical model. The extension of the microtonal pitch spelling procedure into homophonic and polyphonic contexts is in development now. Once the pitch spelling procedure is completed and tested, the accidental collision avoidance procedure will be of primary focus. In the longer-term, development will center around the extension of the musical model into multi-voiced and multi-part musical contexts. When these steps are completed, this more fully-featured musical model will be hooked into the graphical realm. User interface strategies will be developed in accordance with the advancement in the musical model and graphical capabilities.

6. REFERENCES

[1] H. Wulfson, G. D. Barrett, and M. Winter, "Automatic notation generators," in *Proceedings of the 7th International Conference on New Interfaces for Musical Expression*, ser. NIME '07, 2007, pp. 346–351.

[2] H.-W. Nienhuys and N. Jan, "Lilypond, a system for automated music engraving," *Colloquium on Musical Informatics (XIV CIM 2003)*, May 2003.

[3] K. Renz, "Algorithms and data structures for a music notation system based on guido music notation," Ph.D. dissertation.

[4] T. Baca, J. Oberholtzer, and V. Adan. In conversation. [Online]. Available: `http://abjad.mbrsi.org/in_conversation/from_trevor_josiah_and_victor/index.html`

[5] D. Psenicka, "Fomus, a music notation software package for computer music composers," in *International Computer Music Conference*. San Francisco: International Computer Music Association., 2007, pp. 75–78. San Francisco.

[6] J. Bresson, C. Agon, and G. Assayag, “Openmusic: Visual programming environment for music composition, analysis and research,” in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 743–746.

[7] A. Agostini and D. Ghisi. What is bach? [Online]. Available: `http://www.bachproject.net/what-is-bach`

[8] M. Laurson, M. Kuuskankare, and V. Norilo, “An overview of pwgl, a visual programming environment for music,” *Computer Music Journal*, vol. 33, no. 1, pp. 19–31, March 2009.

[9] M. Kuuskankare and M. Laurson, “Expressive notation package-an overview.” in *ISMIR*, 2004.

[10] N. Didkovsky and P. Burk, “Java music specification language, an introduction and overview,” in *Proceedings of the International Computer Music Conference*, 2001, pp. 123–126.

[11] W. Burnson, *Introducing Belle, Bonne, Sage*. Ann Arbor, MI: MPublishing, University of Michigan Library, 2010.

[12] M. S. Cuthbert and C. Ariza, “music21: A toolkit for computer-aided musicology and symbolic music data,” J. S. Downie and R. C. Veltkamp, Eds., vol. 11. Utrecht, Netherlands: International Society for Music Information Retrieval, August 2010, pp. 637–642.

[13] D. Fober, Y. Orlarey, and S. Letz, “Inscore: An environment for the design of live music scores,” *Proceedings of the Linux Audio Conference - LAC 2012*, 2014.

[14] G. D. Barrett and M. Winter, “Livescore: Real-time notation in the music of harris wulfson,” *Contemporary Music Review*, vol. 29, no. 1, pp. 55–62, 2010.

[15] C. Hope and L. Vickery, “Visualising the score: Screening scores in realtime performance,” 2011.

[16] S. Schick, “Developing an interpretive context: Learning brian ferneyhough’s bone alphabet,” *Perspectives of new music*, pp. 132–153, 1994.

[17] I. Pace, *Notation, Time and the Performer’s Relationship to the Score in Contemporary Music*. Leuven University Press, 2009.

[18] S. E. Kanach, *Performing Xenakis*. Pendragon, 2010.

[19] K. Schulmeister, “The interpretation of new complexity: Reflections on the learning process of [d(ks)b] by joan arnau pàmies,” April 2012.

[20] P. Archbold, “Performing complexity.”

[21] D. Spreadbury, “Accidental stacking.” [Online]. Available: `http://blog.steinberg.net/2014/03/development-diary-part-six/`

[22] W. A. Burnson, H. G. Kaper, and S. Tipei, *Automatic Notation Of Computer-Generated Scores For Instruments, Voices And Electro-Acoustic Sounds*. Citeseer.

[23] D. Meredith, “Comparing pitch spelling algorithms on a large corpus of tonal music,” in *Computer Music Modeling and Retrieval*. Springer, 2005, pp. 173–192.

[24] J. Kilian, “Inferring score level musical information from low-level musical data,” Ph.D. dissertation, TU Darmstadt, 2005.

[25] E. Cambouropoulos, “Pitch spelling: A computational model,” *Music Perception*, vol. 20, no. 4, pp. 411–429, 2003.

[26] S. Cunningham, “Suitability of MusicXML as a Format for Computer Music Notation and Interchange,” in *Proceedings of IADIS Applied Computing 2004 International Conference, Lisbon, Portugal*, 2004.

OSSIA: TOWARDS A UNIFIED INTERFACE FOR SCORING TIME AND INTERACTION

Jean-Michaël Celerier
Blue Yeti
jeanmichael@blueyeti.fr

Pascal Baltazar
L'Arboretum
pascal@baltazars.org

Clément Bossut
LaBRI
bossut.clement@gmail.com

Nicolas Vuaille
LaBRI
nicolas.vuaille@gmail.com

Jean-Michel Couturier
Blue Yeti
jmc@blueyeti.fr

Myriam Desainte-Catherine
LaBRI
myriam@labri.fr

ABSTRACT

The theory of interactive scores addresses the writing and execution of temporal constraints between musical objects, with the ability to describe the use of interactivity in the scores. In this paper, a notation for the use of conditional branching in interactive scores will be introduced. It is based on a high level formalism for the authoring of interactive scores developed during the course of the OSSIA research project. This formalism is meant to be at the same time easily manipulated by composers, and translatable to multiple formal methods used in interactive scores like Petri nets and timed automaton. An application programming interface that allows the interactive scores to be embedded in other software and the authoring software, I-SCORE, will be presented.

1. INTRODUCTION

This article will focus on a novel approach to represent and execute conditional branching in interactive scores. Interactive scores, as presented in [1], allow a composer to write musical scores in a hierarchical fashion and introduce interactivity by setting interaction points. This enables different executions of the same score to be performed, while maintaining a global consistency by the use of constraints on either the values of the controlled parameters, or the time when they must occur. This is notably achieved in the current version of the I-SCORE[1] software, presented in [2].

Previously, interactive scores did not offer the possibility to make elaborate choices in case of multiple distinct events occurring at the same time; the work presented here removes this limitation. Here, we will initially present the use cases for conditional branching, as well as several existing works of art which involve conditions. Then, we will introduce new graphical and formal semantics, researched during the course of the OSSIA project. Their goal is to allow composers to easily make use of conditional branching during the authoring of interactive scores. We will show the compliance with previous research on the same field, which allows for strong verification capabilities. We will conclude by presenting the software implementation of these formalisms in the upcoming version 0.3 of I-SCORE, which will be able to edit and play such scenarios in a collaborative way.

[1] `http://i-score.org/`

2. A CASE FOR CONDITIONAL INTERACTIVE SCORES

Even before the advent of computing, there was already a need to write scores containing informations of transport : in western sheet music, manifestations of this are the *D. S. Al Coda*, *D. S. Al Fine*, *Da Capo*, and repetition sign. There is however no choice left at the interpretation.

A case with more freedom for the performer is the *fermata*, which allows for the duration of a musical note to be chosen during the interpretation of the musical piece : the score moves from purely static to interactive, since there can be multiple interpretations of the lengths written in the sheet.

There is also the different case of improvisational parts where each musician has the freedom of his own choice during a few bars – or even a whole piece. In our case, the choices might involve multiple people at the same time (for instance multiple dancers each with his position mapped and used as a parameter), and lead to completely different results.

2.1 Conditional works of art

Some of the most interesting cases happen in more recent times, with the advent of composers trying to push the boundaries of the composition techniques. John Cage's *Two* (1987), is a suite of phrases augmented with flexible timing : "Each part has ten time brackets, nine which are flexible with respect to beginning and ending, and one, the eight, which is fixed. No sound is to be repeated within a bracket.". The brackets are of the form : $2'00'' \leftrightarrow 2'45''$ and are indicated at the top of each sequence.

Branching scores can be found in Boulez's *Third sonata for Piano* (1955–57) or in Boucourechliev's *Archipels* (1967-70) where the interpreter is left to decide which paths to follow at several points of bifurcation along the score. This principle is pushed even further in the polyvalent forms found in Stockhausen's *Klavierstücke XI* (1957) where different parts can be linked to each other to create a unique combination at each interpretation. Some of these compositions have already been implemented in computers, however it was generally done in a case-by-case basis, for instance using specific Max/MSP patches that are only suitable for a single composition. The use of patches to record and preserve complex interactive musical pieces is described in [3].

The scripting of interactive pieces can also be extended towards full audio-visual experiences, in the case of artistic installations, exhibitions and experimental video games. Multiple case studies of interactive installations involving conditional constraints (*Concert Prolongé*, *Mariona*, *The Priest*, *Le promeneur écoutant*) were conducted in the OSSIA project. *Concert Prolongé* (i.e. extended concert) offers an individual listening experience, controllable on a touchscreen where the user can choose between different "virtual rooms" and listen to a different musical piece in each room, while continuously moving his virtual listening point – thus making him aware of the (generally unnoticed) importance of the room acoustics in the listening experience. *Mariona* [4, section 7.5.3] is an interactive pedagogic installation relying on automatic choices made by the computer, in response to the users behaviours. This installation relies on a hierarchical scenarization, in order to coordinate its several competing subroutines. *The Priest* is an interactive system where a mapping occurs between the position of a person in a room, and the gaze of a virtual priest. *Le promeneur écoutant*[2] (i.e. the wandering listener) is a stand-alone interactive sound installation designed as a video game with different levels of exploration, mainly by auditory means.

In closing, interactive applications for exhibitions offer various situations in which conditional constraints are required, from touchscreen applications to full-fledged interactive installations. Several projects have been studied in the scope of OSSIA, in order to make the creation of new complex interactive applications more efficient by using the tools that are developed in this research project.

[2] http://goo.gl/et4yPd

2.2 Existing notations for conditional and interactive scores

We chose to compare the existing notations in a scale that goes from purely textual like most programming environments, to purely graphic like traditional sheet music. Similarly, there are multiple ways to define interactivity and, consequently, multiple definitions of what is an interactive score.

The programmatic environments generally take a preexisting programming language, like LISP, and extend it with constructs useful for the description of music. This is the case with for instance Abjad [5], based on Python and Lilypond, a famous music typesetting software based on a $\TeX$-like syntax. There are also programming languages more axed towards interpretation and execution of a given score, which can take the form of the program itself. This is the case with Csound and CommonMusic [6]. In general, programming languages of this kind offer a tremendous amount of flexibility in term of flow-control. However, they require additional knowledge for the composer to write scores with it.

The purely graphic environments allow compositions of scores without the need to type commands, and are much closer to traditional scores. For instance, multiple Max/MSP externals, Bach for Max/MSP [7], note~[3], rs.delos[4] and MaxScore [8] allow to write notes in a piano roll, timeline, or sheet music from within Max. But they are geared towards traditional, linear music-making, even if one could build a non-linear interactive song by combining multiple instances, or sending messages to the externals from the outside.

Finally, there is a whole class of paradigms that sit between the two, with the well-known "patcher"-like languages: PureData, Max/MSP, OpenMusic [9], PWGL [10]. These software work in term of data-flow : the patch represents an invariant computation which processes control and/or audio data. In each case, it is possible to work purely graphically, and flow control is generally implemented as a block that acts on this data (`[expr]` in Pd/-Max or `[conditional]` and `[omif]` in OpenMusic, for instance). These software all allow to use a textual programming language to extend the capabilites or express some ideas more easily.

[3] http://www.noteformax.net

[4] http://arts.lu/roby/index.php/site/maxmsp/rs_delos

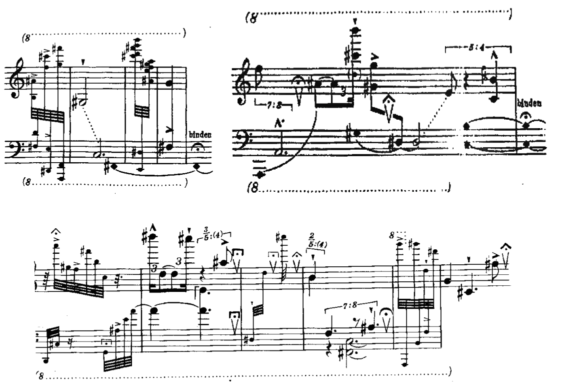


Figure 1. Fragments B2, C1, C3 of *Klavierstücke XI*, Karlheinz Stockhausen. [5]

2.3 Goals

The examples presented in section 2.1 allow us to devise the specification for the kind of conditional capabilities that we want to allow composers to express in our notation system. Most examples studied here operate at a macroscopic level : the choices of the performer generally concerns sections, but at the phrase level, these are often traditional scores, as can be seen from an excerpt from *Klavierstücke XI* in fig. 1. However, the case of a single note which would last longer depending on a given condition can also happen.

The main problem is that there is generally no specific symbol to indicate the conditional execution; instead, the explanation is part of the description of the musical piece. Hence, we have to devise a graphical notation simple enough and yet able to convey easily these different levels of conditions.

These conditions operate on a span of time, which can range from instantaneous, like in the Stockhausen piece, where the performer has to choose his next phrase at the end of the one he is currently playing, to indeterminate, in the case of a perpetual artistic installation waiting for the next visitor. A single symbol might then not be enough to convey in a readable fashion the whole meaning, and multiple symbols would be necessary to explain the articulation of the time in the musical piece.

Finally, an important requirement is to be able to study formal properties on the written score. The presence of conditional expressions means that there is some kind of flow control in the song. Like in a traditional computer software, we want to be able to verify that some properties will remain true for the score : for instance, again in the case of *Klavierstücke XI*, we would like to be able to specify : *at a given time, there cannot be two overlapping fragments*, and be informed if there might be a possible execution of the score that would lead to this case. This has practical implications especially when working with hardware, which can have hard requirements on the input data. This means that the notation will have to be grounded with solid formal semantics.

[5] ©Copyright 1957 by Universal Edition (London) Ltd., London/UE 12654. Retrieved from [11]

2.4 Formal semantics

The current work is based on previous work at the LaBRI by Jaime Arias, Mauricio Toro and Antoine Allombert, that attempt both to formalize the composition semantics and to provide ways for real-time performance of interactive scores. Our work is threefold: finding formal semantics adapted to complex conditional constraints; studying their execution; devising a consistent and simple graphical representation, in order to make the creation of interactive conditional scores as intuitive as possible.

The four following interactive scores formalisms were researched, in order to give a solid foundation, and enforce strong provability properties:

2.4.1 Petri nets

One of the most prominent ideas in the research, on which the current implementation of I-SCORE is based, is the use of Petri nets in order to model interactive scores, by focusing on agogic variations.

The followed methodology was to define basic nets for each Allen relationship [12], and then to apply a transformation algorithm, described in [13, section 9.2].

When a score is played, it is compiled by I-SCORE 0.2 (shown in fig. 2) into a Hierarchical Time Stream Petri Net (HTSPN) which is in turn executed using its well-known semantics.

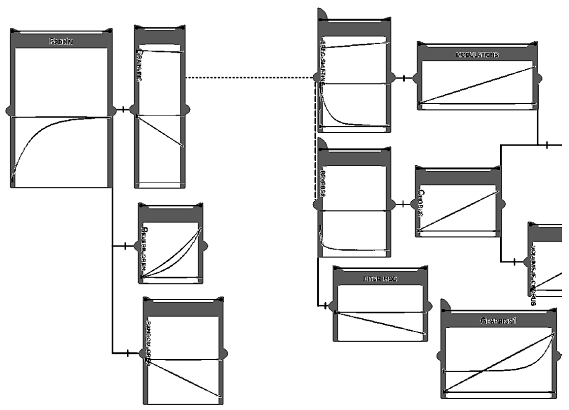


Figure 2. i-score 0.2. The model is based on a succession of boxes (containing curves and other boxes), and relationships between these boxes. A box can have trigger points at the beginning and the end; this introduces an unnecessary graphical coupling between the data and the conditions.

Coloured Petri nets were also used to model complex data processing in interactive scores [14], in order to allow the description and execution of sound processes to occur directly in the score.

2.4.2 *Temporal Concurrent Constraint Programming*

Since the interactive scores can be expressed in terms of constraints (A *is after* B), one of the recurrent ideas for their formalisation was to use Non-deterministic Temporal Concurrent Constraint Programming (NTCC), since it allows constraint solving. This approach was studied by Antoine Allombert [15] and Mauricio Toro [4, 16].

However, there are multiple problems, notably the impossibility to compute easily the duration of a rigid constraint, and the exponential growth of the computation time of constraint solving, which led to some latency in the implementation, making real-time operations impossible.

2.4.3 *Reactive programming*

Due to the static nature of models involving Petri nets and temporal constraints, a domain-specific language, REACTIVEIS [17], was conceived in order to give dynamic properties to interactive scores. An operational semantic is defined using the synchronous paradigm, to allow both static and dynamic analysis of the interactive scores. This also allows composers to easily describe parts of their score that cannot be efficiently represented in a visual manner.

2.4.4 *Timed Automata*

The current focus of the research is put upon the investigation of models for the formal semantics of conditional constraints in interactive scores.

This has been achieved using the extendend timed automata of UPPAAL. Timed Automata allow to describe both logical and temporal properties of interactive scores. Moreover, the shared variables provided by UPPAAL allows to model the conditionals. They are also used for hardware synthesis, in order to target Field-Programmable Gate Arrays (FPGAs) [18]. Real-time execution semantics is implemented with this method.

The problem of the implementation of loops is however still unresolved : it makes static analysis on the score harder, since we hurt the reachability problem.

3. THE OSSIA PARADIGM

3.1 Presentation

OSSIA (Open Scenario System for Interactive Applications) is a research project, presented in [19] and funded by the french agency for research (ANR). Its goal is to devise methods and tools to write and execute interactive scenarios. The two main objectives are to provide a formalisation for interactive scenarisation and seamless interoperability with the existing software and hardware. This paper will focus on the interactive scoring part, the interoperability being provided by the Jamoma Modular framework [20], which allows the use of multiple protocols, such as OSC or MIDI.

When comparing with the previous approaches for interactive scores (Acousmoscribe, Virage, i-score 0.2), the OSSIA project tries to follow a “users first” philosophy : the research work is shared and discussed with artists, developers, and scenographers from the musical and theater fields, and their use case serve as a basis for the focus of the research. They are in turn asked to try the software and discuss about the implementation.

For instance, in the previous studies of interactive scores, a mapping had to be done between the theoretical foundation (Petri nets, temporal constraints. . .) and the domain objects with which the composer had to interact. This has led to mismatches between the representation and the execution [17] of the score. The most prominent problem was the inability to express cleanly multiple synchronized conditions, and to route the time flow according to these conditions. The formalism also did not allow for boxes directly following each other in a continuous manner, and always required the existence of a relationship between them. Instead, in the OSSIA project, we tried to conceive high-level concepts that would allow a composer to easily write an interactive score, build a software over these concepts, and then implement them on the basis of the formalisms presented in part. 2.4.

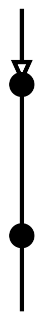
The main concepts of interactive scores can be grouped in two categories: temporal elements and contents. The temporal elements (scenarios, instantaneous events, temporal constraints, conditional branching and hierarchy) allow to create the temporal and logical structure of the scenario, and the contents (states and processes) allow to give actual control over several kind of processes.

3.2 Temporal elements

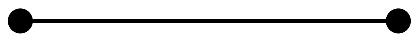
In order to allow the composer to write interactive conditional scores, it is necessary to provide temporal constraints, to allow at least a partial ordering between the different parts of the score. This is done using four base elements : Node, Event, Constraint and Scenario. A *Node* (Time Node) represents a single point in time. An *Event* describes an instantaneous action. A *Constraint* describes the span of time between two given Events. Finally, the *Scenario* structures the other elements and checks that the temporal constraints are valid and meaningful.

3.2.1 *Scenario*

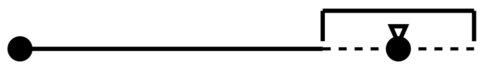
A Scenario is defined as an union of directed acyclic graphs. The vertices are Events and the edges are Constraints. The



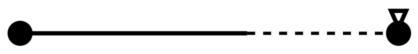
(a) A Node with two Events, one with a trigger, and one without



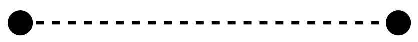
(b) A rigid constraint between two events. Minimum and maximum duration of the constraint are equal ; the date of the end event is fixed with regards to the date of the start event.



(c) A constraint with a non-null minimum and a different, non-infinite maximum



(d) A constraint with a non-null minimum and an infinite maximum



(e) A constraint with a null minimum and an infinite maximum. Instead of making the representation heavier by having the dashes of the constraint continue indefinitely, we chose to remove the rake to symbolize infinity.

Figure 3. The OSSIA Graphical Formalism

direction is the flow of time. It allows to organize the other base elements in time.

Scenarios follow these basic rules:

- A Scenario begins with a Node
- There can be multiple Events explicitly synchronized by a single Node
- A Constraint is always started by an Event and finished by another, distinct Event

Events and Constraints are chained sequentially. Multiple Constraints can span from a single Event and finish on a single Event, as shown in fig. 7. The operational semantics of these cases will be described later. This allows different processes to start and/or stop in a synchronized manner.

3.2.2 Events and Nodes

An Event allows to describe precisely a part of what will happen in a specific, instantaneous point in the execution of the score. It is the basic element, to which are further attached Constraints.

Events can be explicitly synchronized using Nodes. This means that when an Event is triggered, all the other Events

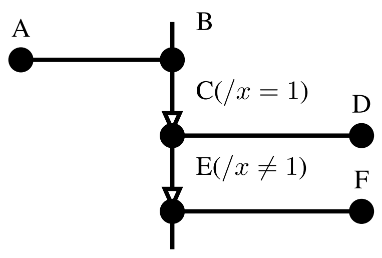


Figure 4. Implementation of temporal branching. A, B, C, D, E, F are Events. B, C, E are on the same Node. C contains the condition $/x = 1$ and E contains !($/x = 1$) in order to have an *if - then - else* mechanism. C and E are evaluated when the constraint between A and B has ended.

on the same Node are also evaluated and instantaneously triggered (or discarded if their Condition is not met, see section 3.3.1).

3.2.3 Constraints

A Constraint represents a span of time. Due to the interactive nature of the proposed paradigm, the span can change at execution time, like a *fermata*. When the author wants to allow a Constraint to have a variable duration, he renders it flexible. This means that the end of the Constraint depends on the Condition of its final Event.

A Constraint can be activated or deactivated: if it is deactivated, it will not count for the determination of the execution span of its end event.

The graphical representation of a Constraint can change according to its minimum and maximum duration. The minimum m's range is $[0; +\infty]$, and the maximum M's range is $[m; +\infty] \setminus \{0\}$. In the user interface (introduced in section 4), the duration is directly linked to the horizontal size and is visible on a ruler.

3.2.4 Graphical formalism

The graphical formalism for these elements is presented in fig. 3.

The Node is a vertical line. An Event is a dot on a Node. If there is a trigger on the Event, a small arrow indicates it. The colour of the arrow can change at run-time to indicate the current state of the trigger.

The Constraint is an horizontal line that represents a span of time, like a timeline. If the constraint is flexible, the flexible part is indicated by dashes and a rake. When there is no maximum to the constraint, there is no rake.

3.3 Operational semantics

3.3.1 Conditions

Each Event carries a condition of execution, and a maximum range of time for its evaluation. The effective range of time for execution is computed by constraint-solving algorithms. For the Event to enter its execution range, all the

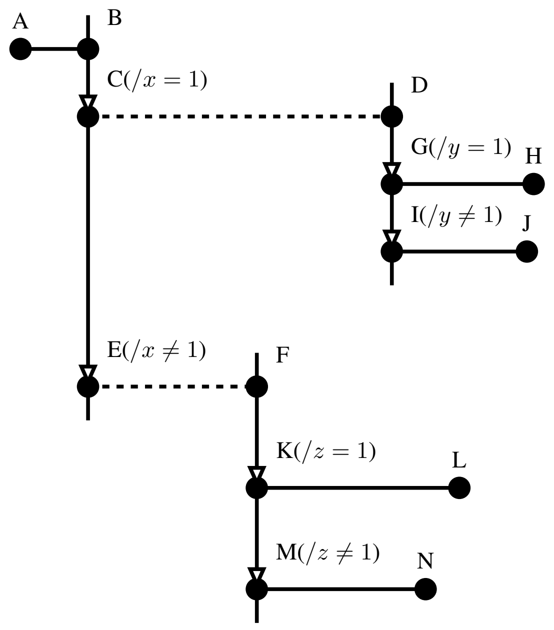


Figure 5. Nested *if - then - else* using flexible constraints

Constraints that finish on this Event must be between their minimal and maximal duration.

An Event is executed as soon as its condition evaluates to True in its execution range. As a result, the Constraints that follow this Event are started, and the messages that might be stored in the Event are sent. Otherwise, the Event is discarded and all the following Constraints are deactivated.

There is usually a default condition which is “all the constraints that explicitly finish on the event have ended”. This default condition can be replaced or extended. For instance, there can be checks on the arrival of a specific network message, or checks on a remote or local address’s value with a specific expression, with the following syntax:

- For parameters that can have a value, there can be comparisons between the values. For instance:

  ```
  /some/parameter > 35 &&
  (   /other/parameter != "a string"
   || /last/parameter == true)
  ```

- Value-less parameters (akin to *bangs* in PureData) can also be used as triggers for the evaluation of expressions. In this mode, logical operators have a different meaning. For instance:
  `/some/bang && !/another/bang`
  will trigger if :

  - `/some/bang` is received, and
  - `/another/bang` is not received within the synchronization interval.

- This is not to be confused with the comparison with boolean values :

  ```
  /a/val == true &&
  /another/val == false
  ```

  which will trigger when the parameters will both be set (not necessarily at the same time) to the required values.

Higher-level operations, like a mouse click on a GUI can then be translated in conditions on Events, in order to bring rich interaction capabilities to the software dedicated to the execution of the scores.

3.3.2 Conditional branching

Branching occurs when, at a single point in the score, two different executions can happen, which leads the scenario to distinct states. For example, the classic *if - then - else* construct can be implemented by having two Events with opposite conditions, as shown in fig. 4. It is also possible to have other cases : for instance, there could be a set of conditions that would lead to either both constraint, or no constraint executed. It is also possible to have multiple constraints with the same condition. Figure 5 presents a method to instantaneously nest conditions, using a flexible constraint with a minimal duration of zero. These patterns can be used as is. However, the authoring software should provide graphical ways to simplify these common cases, for instance by not showing the duplicated conditions. This is not yet done in the current development version of i-score, which allows its users to author scores using the raw formalism presented here.

Convergence occurs when we want to synchronize parts of a scenario that branched previously. The constraint solver ensures that the durations are coherent during the authoring of the score. The execution date of the Node for which a convergence happens will necessarily be after the minimum of each converging Constraint.

3.3.3 Execution of a Node

In this part, we will first present a high-level algorithm that explains the general order of execution of a Node. Then, we will study a particular Node and see how the algorithm translates into a Petri net that can be executed for this particular Node.

As we said before in section 3.2.2, Events are attached to a Node. They are ordered and will be evaluated one after another according to the algorithm given in fig. 6.

The software guarantees that at a high level, if the composer sets an order, the messages will be sent in this order. The only delay will be the one induced by the ordering of instructions in the CPU.

However, it is necessary to be aware that the protocols used underneath, like UDP which is commonly used in OSC protocol implementations, might not always have such

```
Require: We enter the evaluation range for a Node $n$
  if all($n$.eventList(), Event::emptyCondition) then
    repeat
      wait()
    until $n$.DefaultDate
    for Event $e$ in $n$.eventList() do
      $e$.run()
    end for
  else
    repeat
      if any($n$.eventList(), Event::isReady) then
        nodeWasRun $\leftarrow$ true
        for Event $e$ in TimeNode.eventList() do
          if validate($e$.condition()) then
            $e$.run()
          else
            $e$.disable()
          end if
        end for
      end if
    until nodeWasRun
  end if
```

Figure 6. Execution algorithm for a Node

strong guarantees. For this reason, and also because it is not possible to expect network messages to arrive exactly at the same time, the author can specify a synchronization time on a Node: it can wait for a brief time after the triggering of another Event in the same Node, in order to let some time for messages to be received and change the result of conditional choices. However we don't have yet a way to represent this graphically; the idea of a bolder time node was proposed. The synchronization time can also express the will to synchronize events at a lower rate, like two people clapping hands at the same time for example.

Another possibility for the ordering would be to run the Event whose condition did trigger the time node first, and then run the others. This should be a choice left at the discretion of the score writer.

Figure 7 presents the different branching and converging cases that may occur, all mixed in a single Node.

3.4 Contents

An Event may contain a State, which contains messages, and can itself hierarchically contain other States. At a conceptual level, for the composer, a State generally represents a change of state in a remote or local device, i.e. a discontinuity.

A Constraint also acts as a container: during its execution, several Processes can be executed in parallel.

The two main Processes are:

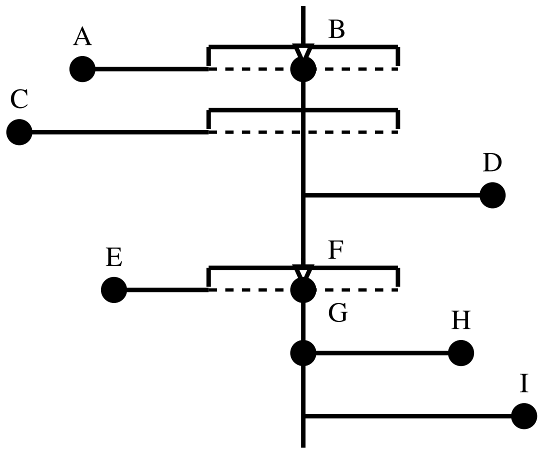


Figure 7. A complete example of Node in the OSSIA graphical formalism. Two constraints converge on the Event B, and three constraints branch from the Node that synchronizes B, F, G. The durations are already processed by the constraint solver.

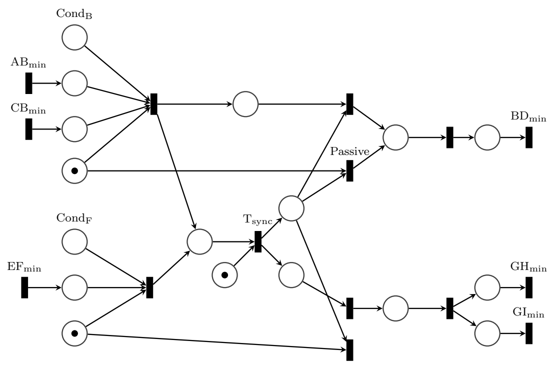


Figure 8. Petri net for the Node of fig. 7.
The previous Constraints's minimum duration have to elapse in $\mathrm{AB_{min}}$, $\mathrm{CB_{min}}$, $\mathrm{EF_{min}}$. When a Condition is validated, a token is put in the respective place ($\mathrm{Cond_B}$ or $\mathrm{Cond_F}$); this triggers the evaluation of the whole Node, unless it was already triggered. The central transition $\mathrm{T_{sync}}$ introduces a small synchronization delay to allow other conditions to validate if necessary. When an Event's condition is not validated, a passive token is instead sent to the following Constraints, which will not trigger the execution of any process or state, and will simply allow the scenario to keep going after a failed condition. This is achieved by the Passive transition.

- The Curve, which allows to send interpolated data in any protocol available to the software.
- The Scenario, which allows hierarchy to happen seamlessly: a constraint can contain a scenario, which can in turn contain other Constraints.

There can be two possible executions for processes: they can do something on each tick of a scheduler; or they can send start and stop signals and behave however they want in-between.

Processes can share data with the Events at the beginning and the end of the parent Constraint, by putting them in specific States.

The API provides ways for somebody to implement his own processes and use them afterwards in scores.

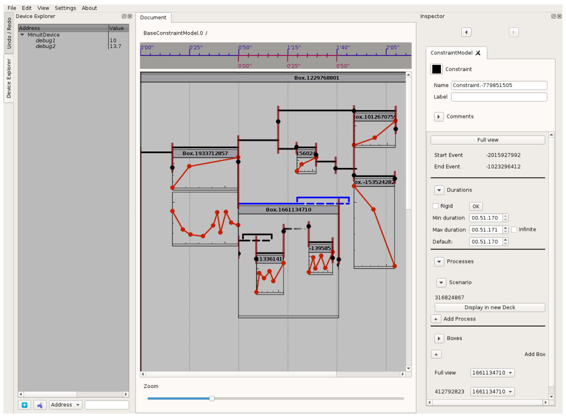


Figure 9. An example of score in i-score 0.3. Not all the graphical features presented here are already implemented.

3.5 Usage as an API

One of the goals of this high-level paradigm is to allow at the same time, a simple mapping with graphical elements, clear semantics of execution, and simple translations with the proven semantics that were studied before, like Petri nets and Timed Automata. In this way, a composer could describe his score which could then be translated into a variety of formats that allow for static and dynamic analysis.

This is achieved by writing a C++ API, that allows for multiple implementations and the use generic programming. It is accessible in `https://github.com/OSSIA/API`. It consitutes a kind of domain-specific language, with the elements that were talked about earlier.

3.6 Implementation in terms of Petri nets

In order to maintain cohesiveness with the previous works on the field, we chose to represent the temporal logic of our formalism in terms of Petri nets. The span of time of the constraints is represented as a transition, and states are emitted when a token enters a state-containing place. These places are not represented here because there can be multiple cases according to the requirements of the writer of the score: it would for instance be possible to send the last state either after the default duration of the constraint, or as soon or late as the condition is validated. It is also possible to add places and transitions in order to have a specific behaviour occurs when the maximum duration of a constraint elapses without triggering.

Figure 8 represents the translation of the Node shown in fig. 7 into Petri nets. We refer to Constraints by the name of their start and end Events.

4. I-SCORE: TEMPORAL AND LOGICAL USER INTERFACES

The API talked about in section 3.5 is being used as the basis of different projects tied to the interactive scores. The dependency graph is shown in fig. 10.

The different sub-projects are:

- I-SCORE, a graphical editor and player for the interactive scores. It solves the problem of the display, edition and interaction with simultaneous elements of the interactive scores. Its current development version (0.3) is available at (`https://github.com/OSSIA/i-score`).
- J.SCORE and I-SCORE-CMD: two players for the scores produced in i-score. The first is an external for Max, the second is a standalone command-line executable. Their current version is available in the repository of the Jamoma project (`https://github.com/Jamoma/Jamoma`).

I-SCORE also exposes its own dynamic device in order to provide some kind of external control at run-time. For example, the conditions could be changed prior to their evaluation, in order to set them in advance at true or false according to events that might have occurred previously during the execution of the score.

The current version (0.2) of I-SCORE, shown in fig. 2, relies on the idea that relations are used to separate boxes. At the time of writing, the upcoming version (0.3, shown in fig. 9) is able to create constraints, events, and nodes using the graphical formalism that was presented in this article, and can play and export these scores so that they can be played on the other software of the suite. It also allows a primitive form of collaborative edition of scores on a local network, and the authoring and execution of distributed scores is currently being studied. Distributed scores would allow for instance to have a part of a score run on a computer, and another part run on another computer.

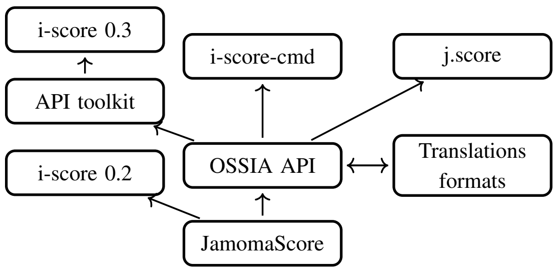


Figure 10. Diagram of the different components of the OSSIA project

5. CONCLUSION

We presented in this article a new interactive score semantic that allows the conception and execution of conditional scores. This semantic is thought of as a mapping into well-known formal models, such as Petri nets, Timed Automata, and Reactive languages: it is meant to be easily understandable and usable for the composer.

However, in order to achieve more expressive power, we still need to find a way to implement loops. Two approaches are currently being studied: one using a Loop process, and another using a concept of `goto`; once one is chosen, we will try to find a relevant graphical element to present it.

Furthermore, there could be some interest in the specification and implementation of variables, which could alleviate the need for an adjacent software like Max/MSP to perform complex logical computations. This would maybe pave the way towards a time-oriented Turing complete programming language, with a simple graphical representation which would allow composers to write complex scores in an understandable way. Another track is the implementation of an audio engine, for instance by embedding FaUST[6], in order to be able to produce sound directly from i-score. The relevant parameters would then be exposed and controlled within i-score.

The next step for the graphical formalism is to make usability studies in order to find the most convincing interactions in the authoring software for the composers.

Acknowledgements

The authors would like to thank Théo de la Hogue as well as Jaime Arias for their contributions and insightful comments during the writing of this article. The OSSIA project was funded by the french agency for research (ANR) under the reference ANR-12-CORD-0024 for the 2012-2015 period.

6. REFERENCES

[1] A. Allombert, G. Assayag, M. Desainte-Catherine *et al.*, "A system of interactive scores based on Petri nets," in *Proc. of the 4th Sound and Music Conference*, 2007, pp. 158–165.

[2] P. Baltazar, T. de la Hogue, and M. Desainte-Catherine, "i-score, an interactive sequencer for the intermedia arts," in *Submitted to the joint ICMC/SMC conference*, Athens, Greece, 2014.

[3] A. Bonardi and J. Barthélemy, "Le patch comme document numérique: support de création et de constitution de connaissances pour les arts de la performance," *CIDE'10 Proceedings*, 2007.

[4] T. Mauricio, "Structured Interactive Scores," Ph.D. dissertation, Université Sciences et Technologies-Bordeaux I, 2012.

[5] V. Adán and T. Bača, "Abjad documentation," 2011. [Online]. Available: `http://abjad.mbrsi.org/`

[6] `http://faust.grame.fr/`

[6] H. Taube, "An Introduction to Common Music," *Computer Music Journal*, pp. 29–34, 1997.

[7] A. Agostini, D. Ghisi, and C.-C. de Velázquez, "Gestures, Events, and Symbols in the Bach Environment," *Actes des Journées d'Informatique Musicale*, pp. 247–255, 2012.

[8] N. Didkovsky and G. Hajdu, "MaxScore: Music notation in Max/MSP," in *Proceedings of the International Computer Music Conference*, 2008, pp. 483–486.

[9] J. Bresson, C. Agon, and G. Assayag, "OpenMusic: visual programming environment for music composition, analysis and research," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 743–746.

[10] M. Laurson, M. Kuuskankare, and V. Norilo, "An overview of PWGL, a visual programming environment for music," *Computer Music Journal*, vol. 33, no. 1, pp. 19–31, 2009.

[11] D. Guigue and E. Trajano de Lima. La répartition spectrale comme vecteur formel dans la Klavierstück n.11 de Stockhausen. [Online]. Available: `http://www.musimediane.com/numero4/Guigue-Trajano/`

[12] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[13] A. Allombert, "Aspects temporels d'un système de partitions musicales interactives pour la composition et l'interprétation," Ph.D. dissertation, PhD thesis, Université de Bordeaux, 2009.

[14] J. Arias, M. Desainte-Catherine, and C. Rueda, "Modelling Data Processing for Interactive Scores Using Coloured Petri Nets," in *Proceedings of the14th International Conference On Applications Of Concurrency To System Design (ACSD'14)*, 2014, pp. 186–195.

[15] A. Allombert, G. Assayag, M. Desainte-Catherine, C. Rueda *et al.*, "Concurrent constraints models for interactive scores," in *Proc. of the 3rd Sound and Music Computing Conference*, 2006.

[16] M. Toro-Bermúdez, M. Desainte-Catherine *et al.*, "Concurrent constraints conditional-branching timed interactive scores," in *Sound and Music Computing conference. Barcelona, Spain.* Citeseer, 2010.

[17] J. Arias, M. Desainte-Catherine, S. Salvati, and C. Rueda, "Executing Hierarchical Interactive Scores in ReactiveML."

[18] J. Arias, M. Desainte-Catherine, and C. Rueda, “Exploiting Parallelism in FPGAs for the Real-Time Interpretation of Interactive Multimedia Scores.” submitted to Journées de l’Informatique Musicale, 2015.

[19] T. De la Hogue, P. Baltazar, M. Desainte-Catherine, J. Chao, and C. Bossut, “OSSIA : Open Scenario System for Interactive Applications.” Journées de l’Informatique Musicale, 2014.

[20] T. De La Hogue, J. Rabin, and L. Garnier, “Jamoma Modular: une librairie C++ dediee au developpement d’applications modulaires pour la création,” in *Proc. of the 17es Journées d’Informatique Musicale*, Saint-Etienne, France, 2011.

A SIGN TO WRITE ACOUSMATIC SCORES

Jean-Louis Di Santo
SCRIME
Jean-Louis.Di-Santo@wanadoo.fr

ABSTRACT

This paper aims at describing an approach meant to build a sign adapted to acousmatic music and based on reduced listening. The sign, to be efficient, must obey to a certain number of requisits: precision, ergonomics, relevance... It must be both easy to use and able to create relations between sounds. A simple description of their qualities is not enough: it must be able to create or analyse sound compositions and structures, such as instrumental scores. To fulfill this purpose, it must be able to give each sound a value, in a saussurian meaning of the word. I will try to show the genealogy of my sign, how I took elements of reflexion from musical knowledge, linguistics, semiotics and aesthetics. From there I deduced the concept of minimal unit of sound applied to electroacoustic music and I created a sign combining symbols to describe its features. I'll show how I have reorganized sound parameters described by Schaeffer and how this sign works. At last, I will show the possibilities of writing scores sound by sound and I'll show two kinds of analysis: the analysis of a pure acousmatic work from a formal point of view and the analysis of a work for tape and instruments both from a formal and a symbolic point of view.

INTRODUCTION

In the middle of the twentieth century, the concept of reduced listening by Pierre Schaeffer and his description of sound based on perception parameters was a real revolution. But if this description, which is to be found in the TARSOM, was really new and interesting, it could not be used for composition or analysis because of its complexity and because there was no system to create relations between sounds. The classification he done in the TARTYP did not really describe sounds: it only established a typology. Lasse Thoresen proposed a "spectromorphological analysis of sound objects" by "the introduction of graphic symbols as opposed to letters or verbal designations to represent the analysis". In other terms, he created a notation corresponding to Schaeffer sound objects. After Schaeffer, Denis Smalley built a new approach of the sound based on sound perception that he called spectromorphology that mainly aimed at describing a relation between sounds and archetype of gestures, and at being a help to composition. Manuella Blackburn translated these concepts in graphic symbols.. Yet in all these approaches, in my opinion, the matter parameters of sound were not precise enough. It was thus necessary, looking at the TARSOM, both to simplify some sound parameters and to complexify others. This is what I did to build my sign. In this paper, I'll call sign what represents all the features of a sound, and symbol what represents only one feature. As a sign is the result of the combination of several symbols, I'll also call sign each virtual result of all the results of these combinations and system of sign both the possibility of these combinations and the fact that each sign can make sense with another. Once the sign elaborated, I separated matter parameters from shape parameters to write scores. I call it score and not sound representation, for example, because it is possible to create sounds corresponding to the signs, like the software Acousmoscribe did (on Mac OS 10.6), in the same way than instrumental scores. This sign aims at writing acousmatic scores as tools both for composition and musical analysis, from a phenomenological point of view.

ORIGINS OF THE SIGN

Theoretical origins

TARSOM

Historically, electroacoustic music is linked to P. Schaeffer's work and his phenomenological approach of the sound. It is based on the concept of reduced listening, and the description of the sound that is in the TARSOM. The TARSOM describes seven criterions of musical perception (mass, dynamic, harmonic timbre, melodic profile, mass profile, grain and gait) and nine criterions of qualification/evaluation distributed in 6 categories (types, classes, kind, pitch, intensity and duration). These criterions describe the sound as perceived from a phenomenological point of view. The TARSOM works with the TARTYP which aims at fixing acceptable sound objects. These sound objects are considered from their beginning to their end.

Linguistics

My sign is based on the concept of minimal unit, or discreet unit, that comes from linguistics (Benveniste, Jakobson). It refers to the smaller sonic element that cannot be divided. For instance, a word can be divided into syllables, a syllable can be divided into phonemes, but a phoneme cannot be divided: it is a minimal unit. This minimal unit is the result of the association of different distinctive features. As linguistics and music are dealing with sounds, I applied this method to electroacoustic music. This way I obtained smaller units than TARTYP units that can be combined to decribe bigger units like phonemes can be combined to create syllables and syllables can be combined to create words. I called electroacoustic sound minimal unit phase and bigger units entity or group. The distinctive features of a phase are the sound features that are described in the TARSOM, and that I reorganised.

Another idea I took from linguistics to build my sign is to use a small amount of elements to create a great number of combinations. One can write several thousands words with only twenty six letters. This way, it prevents from having to memorize a great number of elements and it is easy to use them.

Ch. S. Peirce Sign Theory

Peirce defined a sign as a triadic relationship between the object, the *representamen* and an its *interpretant*. Only considering the relation between the representamen and the object, he established three kinds of relation: icon, index or symbol. I wanted my sign to be easy to read: on the one hand, I used iconic representation every time I could because it is very easy to understand it: for example, concerning dynamic profile, pitch increasing or decreasing and gait. On the other hand, in its symbolic part that needs an interpretant and that is more complex for this reason, I used the same symbols applied to different parameters of the sound to represent the same indications: dot means little, dash an dot middle and dash big, a broken line means random. This way I reduced the number of symbols one has to remind.

Nelson Goodman's Theory

In his book *Languages of art*, Nelson Goodman was comparing notation and art work. According to him, the characteristics of notation are semantic and syntactic non ambiguity. In other words, each sign must not be confused with another, and its interpretant must be clear. Other conditions are syntactical and semantic disjuncture. It means that a sign or a meaning must not have something in common with another. To aim these goals, it is necessary to avoid analogic representation. This is the reason why, for example, a small gait is represented by one curve and not by a small curve, a meddle gait by two curves and not by a meddle curve, and a big gait by three curves and not by a big curve.

Temporal Semiotics Units

The morphological description of Temporal Semiotics Units often describes a certain number of “phases”. That means that a big sonic unit can be constituted by several small units and that each small unit has a value, like it has in linguistics. These phases can be a process concerning one sound parameter, or concerning several sound parameters at the same time. In fact, the idea of musical minimal unit was born from a research that was aiming at transforming these analytic tools in compositional tools[1].

Music Theory Notation

Of course, the music theory notation offers an excellent example of music notation: it is clear and as simple as it can be. It also uses minimal units and tries to indicate the most important features to realise sounds or analyse score. In a certain way, it is an open system because it allows to add any sort of indication. The simplification of the sound reduced to pitch and rhythmic parameters do not prevent any other precision. The keys and key signatures allow to avoid the repetition of what does not change, and they are very ergonomic.

Simplification Of TARSOM's criterions

Criterions of Musical Perception

In order to create a sign quite simple to read, I reduced the seven criterions to four profiles: concerning criterions of form, I established the dynamic profile and the rhythmic profile. Concerning criterions of matter, I established the melodic profile and the harmonic profile. One finds here the four traditional dimensions of sound. The term of profile refers to three kinds of processes: augmentation, diminution, or stability which is a particular kind of process but not the only one. Why and how has the modification of Schaeffer criterions been done? In TARSOM or TARTYP, Schaeffer is describing sound objects constituted by several phases. Yet, I was interested only in one phase sound objects, but I took into account all of the possible variations. Schaeffer had already described melodic profile and mass profile. He put them in the category of criterions of sound variations. The TARSOM establishes seven categories: three

[1] Di Santo, Jean-Louis, “Composer avec les UST”, *Vers une sémiotique générale du temps dans les arts*, Actes du colloque "Les Unités Sémiotiques Temporelles (UST), nouvel outil d'analyse musicale : théories et applications", Sampzon, Delatour, 2008

categories regarding the sound itself (mass, dynamic, harmonic timbre), two categories regarding variations (melodic profile and mass profile) and two categories regarding maintenance (grain and gait). If one looks at the column two of the TARSOM, one can read on the “*timbre harmonique*” line: “*lié aux masses*” (linked to masses), and comparing the *masse* line to the *timbre harmonique* line, one can almost see the same classifications. Thus I merged these two criterions into harmonic profile in order to simplify them. In the same way, always considering the column two, the “*dynamique*” and “*profil de masse*” lines are very similar and I merged them into “dynamic profile”. However Schaeffer was mostly describing two phase profiles: following the concept of minimal unit which is based on a unity of process, I only considered one phase processes. Here too, still taking into account one phase processes and adding the stability that was missing, I kept the “*profil mélodique*” line. I also transformed the column 8 (impact) into “rhythmic profile”, with the caracteristics of slow, moderate and fast that are in the TARSOM, and I added the processes of accelerando, of rallentando and irregular. The rhythmic profile refers both to the internal speed of a sound and to iterative processes of the same sound. This way, the four traditional musical criterions were redefined. At last, I respectively linked maintenace criterions (grain and gait) to dynamic profile and melodic profile. However some criterions can be linked to some others: the rhythmic profile, that describes speed variations, can be as well applied to iteration or gait. The gait also often refers to melodic profile that contains the idea of pitch, thus also the caliber, which is the difference between the lower and the higher frequency of the sound. Grain is a particular variation that is applied to the dynamic of sound. This way, some criterions that disappear from the seven Schaeffer criterions reappear applied to the four profiles.[2]

Criterions of Qualification/ Evaluation

As shown above, some of them are integrated to the four profiles. The categories of species are integrated as quantities: small, middle, big or random. The concept of random or irregular is very useful when some processes are changing quickly in different ways: for example to describe the sound of creaking wood which is sometimes fast and sometimes slow.

Number Of Phases Of The Sound

In the TARSOM or The TARTYP, sounds can have several phases. For the reasons I explained above my sign describes sounds phase by phase. Phase refers to any kind of sound, whatever its duration is, featuring the same process (this process commands the same modification or non-modification of the sound and can be applied to intensity, pitch, timbre or rhythm). Phase is the name I gave to the minimal electroacoustic sound unit. This way, one obtains 4 profiles that will be described later. Profile is here the name of distinctive feature.

Complexification Of Mass/Harmonic Timbre

I merged the schaefferian Mass and Harmonic Timbre into the term of harmonic profile (in my sign, the species of mass are mainly linked to melodic profile). The harmonic profile concerns the very matter of sound, which does not depend on pitch, dynamic or other criterions about form. Schaeffer determined seven categories of sound considering this parameter (*son pur*, *son tonique*, *groupe tonique*, *son cannelé*, *groupe nodal*, *nœud* and *bruit blanc*). “*Son pur*” is sine curve, and “*bruit blanc*” is white noise. They will not be taken into account here, since they do not vary (except sine curve which pitch can vary depending on its height, which is not our purpose here). Thus five categories of sound remain. Their description, being very large, is very imprecise, even if the number of categories is increased by the distinction between “simple” sounds and groups. According to Schaeffer, these five categories can be rich or poor. In EMS 11, in New York, I suggested to increase these categories[3]. I determined three categories of homogeneous sounds, that can be rich or poor, and three categories of hybrid sound that can be rich or poor too. Combining these categories in groups or *sons cannelés* (distonic sounds using Thoresen translation), and adding stable or filtered colours (bright, dark, hollow...), one can have 40 000 descriptions of harmonic profile

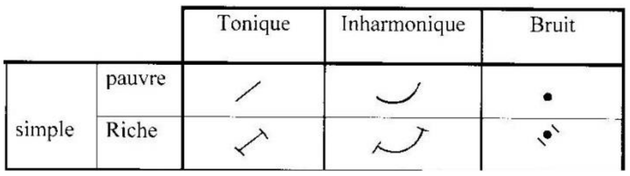

Tonique
Inharmonique
Bruit
pauvre
simple
Riche

Figure 1 : Homogeneous sounds. The dash on each side of the symbol always means rich. It will be the same, of course, concerning hybrid sounds.

[2] http://www.ems-network.org/IMG/EMS06-JLDSanto.pdf, p. 4-5

[3] http://www.ems-network.org/IMG/pdf_EMS11_di_santo.pdf

Hybrid sounds will be represented as below:

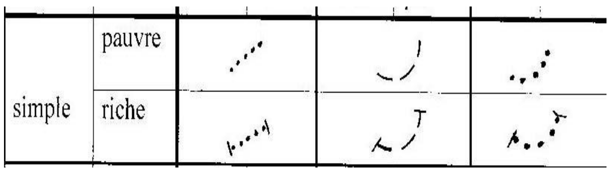


Figure 2 : Hybrid sounds. A hybrid sound is a sound that haves features from two homogeneous categories. For example a fly sound has features of tonic sound and features of noise. Thus it is represented by a line (tonic sound) made of dots (noise).

The twelve simple signs described above will be used to build all the other signs, and particularly what one will call "group" and "*son cannelé*". One will call "group" sounds of the same category combined between them. A group made of homogeneous sounds will be called homogeneous group and a group made of one or two hybrid sounds will be called hybrid group. The sign that represents a group is made of two symbols. The lower one represents the sound one hears the most (called fundamental), and the higher one represents the sound one hears the less or as much as the other (called harmonic).

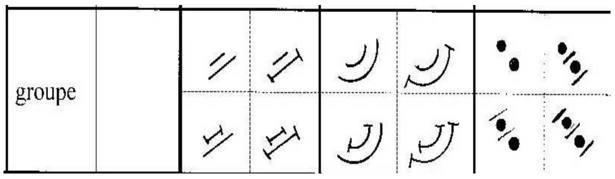


Figure 3 : Homogeneous groups. Hybrid groups will be represented either by a symbol of homogeneous sound and a symbol of hybrid sound, either by two hybrid sounds symbols.

Tonic	Inharmonic	Noise
Tonic fundamental/ Homogeneous harmonic Tonic fundamental / Hybrid harmonic Hybrid fundamental/ Homogeneous harmonic Hybrid fundamental/ Hybrid harmonic	Inharmonic fundamental/ homogeneous harmonic Inharmonic fundamental/ hybrid harmonic Hybrid fundamental/ Homogeneous harmonic Hybrid fundamental/ Hybrid harmonic	Noise fundamental/ homogeneous harmonic Noise fundamental / hybrid harmonic Hybrid fundamental/ Homogeneous harmonic Hybrid fundamental/ Hybrid harmonic

Figure 4 : Categories of dystonic sounds.

If the sounds of the group belong to two different categories, one will call it *son cannelé* (for example a bell sound is made by a first audible tonic sound and a thin inharmonic halo. Now, tonic sound and inharmonic sound belong to two different categories, so a bell produces a *son cannelé*. This sound will be represented by a tonic symbol under an inharmonic symbol). In order to have clearer signs, one will limit the number of symbols to two by group and three for *son cannelé*. To build all the possibilities of *son cannelé*, one will use the table from Figure 4.

There are also symbols to describe the "colour" of the sound, if it is more or less dark or bright. The symbol is a dot put on symbols of harmonic profile, except for noise that can't have a colour because of its very rich spectrum.

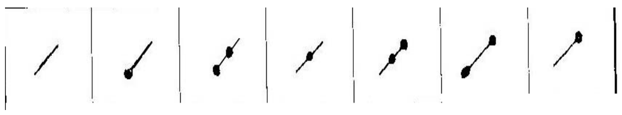


Figure 5 : Seven different "colours" of sound put on the symbol of tonic sound (the same thing can be done for inharmonic sounds). From left to rignt: equilibrated sound, strong low frequencies, weak high frequencies, strong medium frequencies, weak low frequencies, weak medium frequencies and strong high frequencies. These colours can be filtered and can change but I don't reproduce the symbols here.

PRESENTATION OF THE SIGN

The all sign is built assembling symbols to describe minimal untit of sound. Of course, different minimal units can be assembled to create a higher level of unit, like in linguistics. These symbols represent the different profiles of the sound. The concept of profile is very useful to create a link between the continuity of reality and the categories without which it is impossible to think. Basically, the sign represent four profiles. These profiles correspond to distinctive features in linguistics[4].

Dynamic Profile

It concerns the features of intensity variations of sound (crescendo, decrescendo or stable). It is represented by a quadrangular or a triangle. The bottom of this figure indicates speed variations of sound and the top indicates grain. The sign offers five possibilities of dynamic profile:

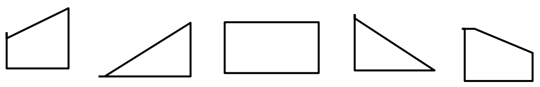


Figure 6 : Dynamic profiles. From left to right: soft, support, flat, straight attack, straight truncated attack.

[4] http://www.ems-network.org/ems09/papers/disanto.pdf

If the dynamic profile irregularly varies, a broken line is added at the top of one of these figures.

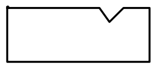


Figure 7 : A flat dynamic profile varying irregularly.

Rhythmic Profile

It concerns the internal speed variation of sound or its speed iteration (acceleration, deceleration or rhythm, allure or grain's stability). It is notated by dots, dashes and dots or dashes at the bottom of the rhythmic profile, as explained above. If the rhythmic profile irregularly varies, a broken line is added at the bottom of the figure. A vertical dash at the beginning or the end of the figure means rallentando or accelerando.

Melodic Profile

It concerns tessitura (pitch becoming higher, lower or stable). It is represented by five dots on the left or rignt side of the figure that represents dynamic profile. The lower one indicates very low tessitura, the one above, low tessitura and so on until very high tessitura. A line is attached to these dots to represent the tessitura of the sound. This line can be straight and horizontal if the tessitura is always the same, or can also come up or down if the pitch increases or decreases. This line is curved if the sound has gait. At last, this line indicates the caliber of the sound: a line made with dots if the caliber is thin, dash and dot if it is meddle, and a dash if it is large. The same symbols can be applied to a curve. If the melodic profile irregularly and quickly varies, a broken line is added at the end of this symbol.

Harmonic Profile

The term harmonic profile replaces the terms of Mass and Harmonic Timbre in the TARSOM. It concerns harmonic timbre: richer, poorer or stable. The harmonic profile is represented by symbols inside the geometrical figure: a line for a tonic sound, a curve for inharmonic sounds and a dot for noise. The sound can be homogeneous or hybrid if it has the features of two different sorts of sound (see above). Each category, homogeneous or hybrid, can be rich or poor.

Tonic and inharmonic sounds can have a colour (see above, EMS11, New York). The sign allows to represent seven stable colours and fourty two filtered colours.

The combination of two symbols belonging the same category represent a group (tonic, inharmonic or noise) that can be homogeneous or hybrid. The combination of two symbols belonging to different categories represent dystonic sounds.

Number of combinations

A complete sign is made assembling symbols on the different sides of the dynamic profile or putting them inside.

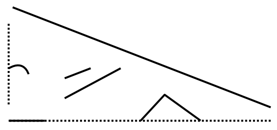


Figure 8 : An example of complete sign. The triangle shows the dynamic profile and means *straight attack*. The line at the top of this figure shows that there is no grain. At the left one can see the melodic profile in a medium tessitura. This full line shows a large caliber ; this line is curve, that means that there is a gait. This gait is small because there is only one curve. At the bottom of the figure, dots indicate that the speed of this gait is fast (rhythmic profile). The broken line in the rhythmic profile means that this rhythm is irregular. At last, the two lines inside the figure represent the harmonic profile: they mean t*onic group*.

A sign, to be efficient, must be precise. This precision depends on the number of possibilities it offers. The Acousmoscribe's sign offers five symbols for dynamic profile, three symbols for grain and three symbols for rhythmic profile. Dynamic or rhythmic profiles can be regular or irregular, thus the number of possibilities is doubled.

There are five possibilities of stable melodic profile, ten possibilities describing increasing pitches (from very low to low, from very low to medium, from very low to high, from very low to very high, from low to high and so on...), and ten possibilities describing decreasing pitches. Of course, all these possibilities can offer irregular processes. The symbol supporting melodic profile also represents caliber. There are three possibilities of caliber and, for each of it, the possibility of being irregular. If the sound has a gait, the line representing melodic profile is replaced by one, two or three curves, depending on the amplitude of the gait.

At last there are three basic symbols for harmonic profiles becoming six merging different features, and becoming twelve adding a symbol meaning "rich". As already said above, adding the different colours and the possibilities of groups and dystonic sounds, there are 40.000possibilities of harmonic profile.

The different combinations of all these different symbols allow approximately five billions possibilities to build a sign always easy to read.

FROM SIGN TO SCORE

The Acousmoscribe

The first step was the Acousmoscribe, an experimental software created in 2009 that works on Mac OS 10.6: it allows the creation of a sign assembling different symbols that one can choose to represent different profiles and parameters. The windows have two parts: the left part represents tracks on which one can put the signs. The right part is a palette where one can create the sign assembling the symbols of the different profiles. One can assemble the symbols to create a sign and then put it in the tracks. This software was able to generate some sounds corresponding to the sign.[5]

Score of *Incidences/résonances*

The following step was the analysis of *Incidences, résonances* by Bernard Parmegiani. A poietic analysis of this piece has already been made by Ph. Mion, J. J. Nattiez and J. C. Thomas, and I wanted to compare this kind of analysis with the transcription I did using my sign. I called it score because one can create an interpretation of this work following the indications of the signs, because each sign can generate a sound with the same features, even if it is not exactly the same. In this score each sound has its own track. Matter parameters that don't change are used as keys, on two columns at the beginning of each page of the score (harmonic and melodic keys), and shape parameters are written on the tracks. I choosed to put sounds in order of appearance, from the bottom to the top. The analysis of the harmonic key generated the concepts of soundality and soundulation, which are the equivalent of tonality and modulation, in instrumental scores, applied to the very matter of the sound. This analysis is a purely formal analysis and the use of the sign allows to unterstand some processes of composition that are impossible to understand otherwise, specially the relationship between the different harmonic profiles of this work. The concepts of soundality and soundulation were born from this kind of analysis. What is a soundality? What I call soundality is a sonic configuration where a majority of sounds, or the main sounds, belongs to the same category of sound, referring to the paper I presented at the EMS 11 conference (see fig.. 1 and 2 above). Of course, a soundulation is a change of soundality.[6]

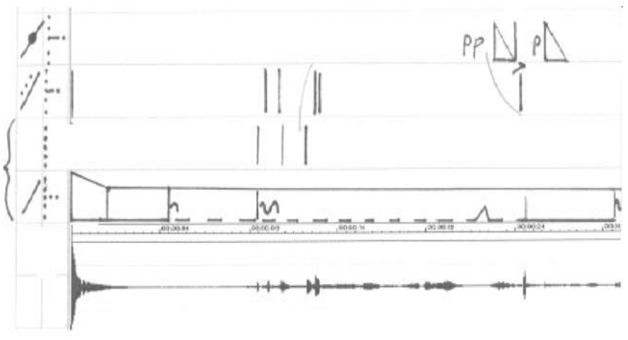


Figure 9 : Beginning of *Incidences/résonances* by B. Parmegiani. Matter parameters are put on the two columns on the left: first column describes harmonic profile and second column describes melodic profile. Shape parameters, dynamic profile and rhythmic profile are put on tracks corresponding to each sound.

Score of S*ix japanese gardens*

The analysis of *Six japanese gardens*, first movement, by Kaija Saariaho, realised for her nomination doctor honoris causa of the university of Bordeaux Michel de Montaigne, enables the analysis of the relationships between instruments and tape with the signs. Tape and instruments are considered from a phenomenological point of view, using reduced listening, and can be compared: what is different and what is the same, and the relations between the different sounds. But not only: this work is obviously a symbolic work and is speaking about time. The analysis of this work with my sign allows to study the semiosis, the way the plane of contents works with the plane of expression. At last, this score also shows the descriptive goal of this work. Its complete title is *Tenju-an Garden of Nanzen-ji Temple.* Looking both at a photograph of this temple and at the score, it is possible to see some isomorphisms.

CONCLUSION

The sign I have elaborated for ten years is based on the reduced listening and describes the sounds from a phenomenological point of view. It is an open system, and it is possible to add any sort of annotation. Not only does this sign system aim at describing sounds, but it also aims at creating structures where each sound can have a value, in a saussurian meaning of this term, i.e. where sound parameters create a relation between each sound to make sense. It is now precise enough to write scores but, of course, still can and must be ameliorated.

REFERENCES

[1] F. de Saussure, *Cours de linguistique générale*, Payot, 1972.

[2] P. Schaeffer, *Traité des objets musicaux*, Seuil, 1966.

[5] *Ibid.*
[6] `http://www.ems-network.org/spip.php?article377`

[3] L. Thoresen, “Spectromorphogical Analysis of Sound Objects, An adaptation of Pierre Schaeffer’s Typomorphology”, EMS 2006.

`http://www.ems-network.org/IMG/EMS06-LThoresen.pdf`, p. 3

[4] D. Smalley, “La spectromorphologie, une explication des formes du son”, http://www.ars-sonora.org/html/numeros/numero08/08d.htm

[5] BLACKBURN, Manuella, “Composing from spectromorphological vocabulary: proposed application, pedagogy and metadata”, http://www.ems-network.org/ems09/papers/blackburn.pdf

[6] BENVENISTE, Emile, *Problèmes de linguistique générale*, Gallimard, 1966.

[7] JAKOBSON, Roman, *Essai de linguistique générale, T 2,* Paris, Minuit, 1973.

[8] PEIRCE, Charles Sanders, *Ecrits sur le signe*, Paris, Seuil, 1978, p.p. 148,149.

[9] GOODMAN, Nelson, *Langages de l’art*, Paris, Hachette littératures, 2005 (édition originale : Jacqueline Chambon, 1990), pp.168-192.

[10] MIM, *les Unités Sémiotiques Temporelles*, Documents Musurgia, 1996.

[11] DI SANTO, Jean-Louis, “Composer avec les UST”, "*Vers une sémiotique générale du temps dans les arts, Acte*s du colloque "Les Unités Sémiotiques Temporelles (UST), nouvel outil d'analyse musicale : théories et applications", Paris, Delatour, 2008, pp.257-270

[12] NATTIEZ, Jean-Jacques, *Fondements d’une sémiologie de la musique*, Paris, 10/18 Esthétique, 1975

[13] DI SANTO, Jean-Louis, "Harmonic profile: typology and notation", http://www.ems.network.org/IMG/pdf_EMS11_di_santo.pdf , 2011.

[14] DI SANTO, Jean-Louis, "L'acousmoscribe, un éditeur de patitions acousmatiques", http://www.ems-network.org/ems09/papers/disanto.pdf , 2009.

[15] DI SANTO, Jean-Louis, “Proposition d’une terminologie structurée et de notation symbolique de la musique électroacoustique”, http://www.ems-network.org/article.php3?id_article=239, 2006.

[16] MION Philippe, NATTIEZ Jean-Jacques, THOMAS Jean-Christophe, *L'envers d'une œuvre, De natura sonorum de Bernard Parmegiani*, Paris, Buchet/Chastel, 1982, p.p. 35-43.

A PARADIGM FOR SCORING SPATIALIZATION NOTATION

Emile Ellberger
Institute for Computer Music and Sound Technology, Zurich University of the Arts
emile.ellberger@zhdk.ch

Germán Toro Perez
Institute for Computer Music and Sound Technology, Zurich University of the Arts

Johannes Schuett
Institute for Computer Music and Sound Technology, Zurich University of the Arts

Giorgio Zoia
Institute for Computer Music and Sound Technology, Zurich University of the Arts

Linda Cavaliero
Institute for Computer Music and Sound Technology, Zurich University of the Arts

ABSTRACT

The present paper is a shortened version of the one presented at the ICMC/SMC2014 [1] where it was demonstrated that SSMN (Spatialization Symbolic Music Notation) research seeks to establish a paradigm wherein OSC (Open Sound Control) [2] and a Rendering Engine allow a musical score to be heard in divers Surround formats.

The research team consists of composers, spatialization experts, IT specialists and a graphic designer. After having established a taxonomy identifying and classifying spatiality of sound with associated parameters, open source software is being developed and tested by practitioners in the field. Composers, utilizing dedicated graphic symbols integrated into a score editor, have full control over spatialization characteristics. They can audition the results and communicate their intentions to performers (i.e. conductors, musicians, dancers, actors) as well as to all participants in the chain from rehearsal to performance.

SSMN capitalizes on time-based phenomena: choreographers can combine and synchronize sound and body movement; installation artists can program interactively visuals with audio manipulation; film and video can be enhanced with 3D sound effects and spatialized scores. SSMN focuses not only on musical composition, other performing and media arts or even game interaction design, but is useful in academic contexts such as professional training in conservatories and in musicological research addressing the perennity of spatialization in early electroacoustic music.

1. INTRODUCTION

Research on spatialization in music dates from practices in early civilizations through today's contemporary output. SSMN investigations concentrate on composers' means of expressing placement and/or motion in space (e.g. Stockhausen, Boulez, Brant), and of more recent methods of graphic representation proposed in various research centers (i.e. Ircam's OpenMusic [3] & Antescofo [4], MIM's UST [5], Grame's 'inScore' [6]). During the past decade certain composers using WFS [7] and Ambisonics [8] pointed to the need of musical notation wherein graphic symbols and CWMN (Common Western Music Notation) could coexist on a time line along with audio rendering.

2. DEFINING A SPATIAL TAXONOMY

The SSMN Spatial Taxonomy is an open-ended systematic representation of all musical relevant features of sound spatiality. It is organized as follows: basic units of the SSMN Spatial Taxonomy are called descriptors, i.e. room descriptors and descriptors of sound sources. Descriptors can be simple or compound and are assumed to be perceptually relevant. Simple descriptors denote all single primary features relevant to sound spatiality and can be represented as symbols. Compound descriptors are arrays of simple descriptors used to represent more complex spatial configurations and processes. Structural operations and behavioral interactions can be used to transform elements previously defined using descriptors or to generate new elements. Descriptors are progressively being implemented in the project when proven to be of general user interest. Although the taxonomy is classifying and describing sound in a three-dimensional space, some objects and symbols are, for practical reasons represented in two dimensions. As this taxonomy contains a very systematic vocabulary it proves to be useful for other research projects related to 3D Audio currently under development at the ICST. To assure the validity of concepts within this taxonomy, the SSMN team has undertaken the task of testing perception of sound spatiality elements both in 2D and 3D mode, with key questions being what can be perceived or not, and under which conditions.

3. CREATING GRAPHIC SYMBOLS

In accordance with the SSMN Spatial Taxonomy requirements, a basic set of symbols was researched and designed with the primary criteria requiring clarity, legibility and rapid recognition. Equally, the choice between symbolic or descriptive designs becomes particularly relevant. Thus, the SSMN Symbol Set synthesizes both approaches. Depending on the requirements of a musical composition, spatialization information can be very complex; configurations consisting of simultaneous trajectories with varied types and durations require transmitting elaborate I/O data that must be readily understood and communicated to all in the chain from creator to performer to sound engineer. Communication between the target users is simplified with SSMN: the symbols could be common to various types of outputs (score, cue sheet, sound design, video editors) and the associated rendering parameters can be freely edited in available and future tools. They can also be used in remastering situations, preparation of audio tracks for video games, 3D cinema, surround radio broadcasting, theater productions, choreography and installations.

The symbol set

The SSMN Symbol set and subsets are organized so as to be easily inserted in a GUI (Figure 1). In order to facilitate the use of the SSMN symbols and their introduction into the musical score five categories of symbols related to the following aspects are defined:

- Physical performance space characteristics (geometrical form, size, reverberance, inside/outside)
- Initial physical placements of performers, microphones, loud speakers and objects
- Localization and quality of sound sources (acoustic and projected audio[1])
- Trajectories and/or displacement of sound sources, microphones, loud speakers, and objects whether individually, in groups or more complex configurations (sound clouds, planes, surfaces)
- Inter-application communication possibilities and protocols (OSC, MIDI) as well as integration with external programming environments.

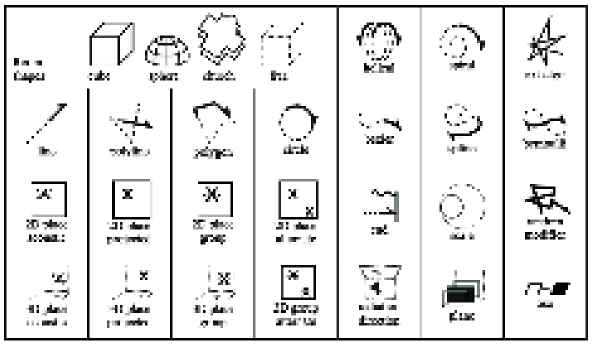
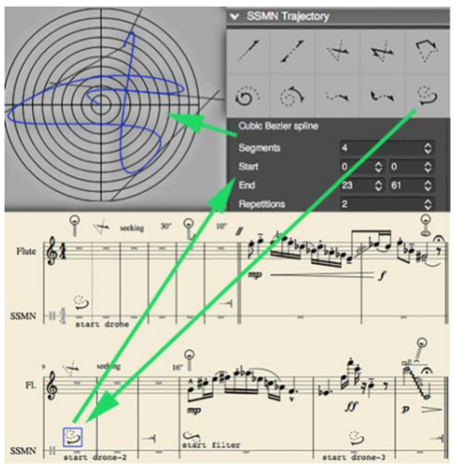


Figure 1. Extract of SSMN symbols set.

4. IMPLEMENTATION OF MUSESCORE_SSMN

The notation editor MuseScore was chosen due to its Open Source characteristics and its OSC communication possibilities, i.e. on/off/play/pause/next/. The SSMN implementation now allows all parameters and values of the symbols to be transmitted to target software within the tool set and equally receive data for control. Symbols are organized into palettes and menus according to SSMN categories, classes and functions. Once placed in the score, an Inspector window displays user-defined rendering parameters and flags specific to each type of symbol. A 2D/3D radar view displays the activity of the spatial movements from a selected note to another, or over a section of the score. Clicking on a symbol in the score allows seeing the entire trajectory in the radar. Several templates have been designed to facilitate formatting various score-types. The user commonly places SSMN symbols on any instrumental staff; nonetheless, a dedicated *SSMN Staff* can be utilized to transmit spatialization data as well as OSC messages, independently of notation, to any software with OSC functionality (Figure 2).



Cubic Bezier spline
Segments 4
Start 0 0
End 23 61
Repetitions 2

Figure 2. MuseScoreSSMN example:
symbol → score → parameters → radar

1 Acoustic audio refers to the natural sound of instruments whereas projected audio refers to sounds coming from loudspeakers.

5. INTER-APPLICATION COMMUNICATION

The use of OSC (Open Sound Control) possibilities allows messages to be directed to various target software modules. Typically, spatialization data from MuseScoreSSMN flows to an audio renderer-engine capable of spatializing in various output formats, e.g. Ambisonic B-Format, WFS, multi-channel encoded audio files. OSC messages and RAW data are also routed to DAW (Digital WorkStation) or to programming environments (e.g. SuperCollider, C-Sound, MaxMSP.) At this time exporting possibilities include MusicXML and SVG.

6. DEVELOPING THE RENDERING ENGINE

Compatible with the Open Source Initiative for standardized Max/MSP Module, the SSMN Rendering Engine has been engineered to allow real-time spatialized audio rendering and visual feedback for all SSMN activity. Functionalities include OSC routing over UDP ports, and user control of encoding and decoding in various formats; the user determines speaker configuration, designs the distance characteristics and is able to select effects such as reverb, air absorption, and Doppler. All audio activity can be saved and reopened in common audio file formats. Real time visual feedback allows the user to monitor single or multiple trajectories and sound placements in 2D/3D. An AUAmbi plug-in allows communication with audio software that have AU implementation. In order to facilitate overall OSC control, a set of descriptions were created that would allow multiple cross-application communication, also adaptable to other protocol context such as SpatDIF and MusicXMuse-SoreML (Figure 3).

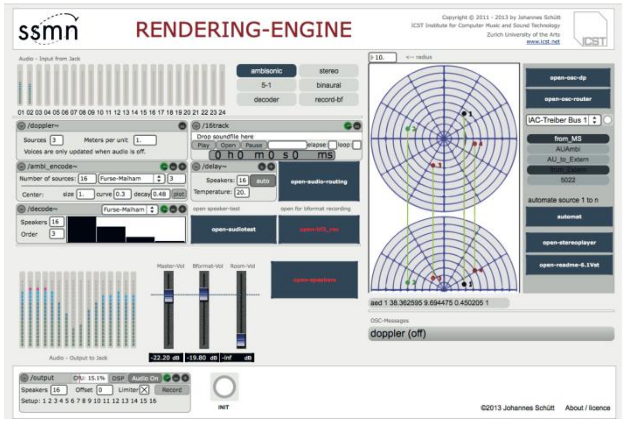


Figure 3. SSMN Rendering Engine main screen.

7. TWO CASE STUDIES

Urwerk by Vincent Gillioz

A first SSMN case study consisted of a film score, which revealed the combining of instrumental notation with 3D spatialization effects to be integrated into 3D cinema. Here a score for 9 instruments and electronics was originally notated in a popular score editor. Initially the composer created his personal symbols and spatialization annotations, but was limited to hearing the results in a stereo version. He now exported his score in MusicXML format (notation only), and imported it into MuseScoreSSMN utilizing the SSMN spatialization symbols. Then, the composition with accompanying audio files was rendered in B-Format onto an Ambisonic speaker system. Having been able to audition the impact of the sound motion, he could consequently edit and modify various parameters of SSMN symbols to his taste and allow for more coherent musical effects. Interestingly, Gillioz had little experience in spatialization at first and began by creating erratic sound movements – skips and wide jumps at 8thnote-120BPM rate. His esthetics obliged him to modify the displacement rate (speed and distance). Having mastered the process, he modified the score as necessary and gave us precious feedback.[2]

CHoreo by Melissa Ellberger, choreographer

***CH**oreo* was a simple case study demonstrating advantages in using SSMN within a rehearsal context. A choreographer trained performers wearing portable loudspeakers to move along trajectories in a hall. Sound files projected from the portable loudspeakers accompanied the body movements. In play mode, MuseScoreSSMN triggered sound files transmitted to the SSMN Rendering Engine, all the while sending streams of OSC data controlling the 3D spatialization process. The performers could execute their roles by following the printed MuseScoreSSMN; the learning process prior to an actual public presentation was greatly facilitated (Figure 4).

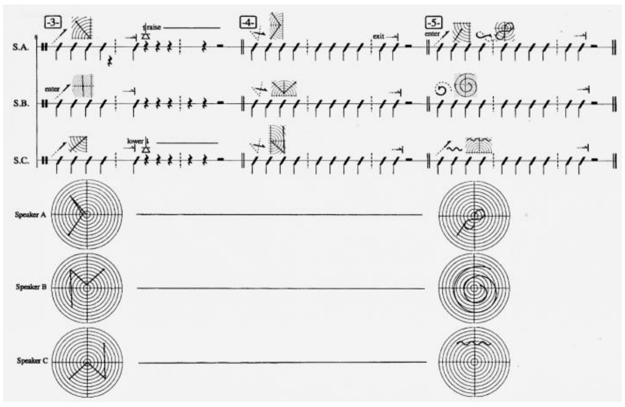


Figure 4. *CHoreo* trajectory score.

[2] *Urwerk* score/renderer/qtmovie (binaural version) can be accessed at `http://blog.zhdk.ch/ssmn/movies/`

8. CONCLUSION

At this stage of the “work-in-progress” of SSMN, its basic workflow is optimized for the user case in which notation for instrumental music (often incorporating live electronics) is introduced into a music editor and spatialized audio rendering is a requirement. Other user cases include the additional use of audio files managed within DAW software. SSMN equally targets state of the art venues, namely 3D cinema (with a great need for encapsulating height information into surround systems), 5.1 radio and web-based broadcasting (video, music and radio theater productions), choreography notation, artistic multi-media and interactive installations, surround CD, DVD and Blu-Ray market, as well as game design.

An SSMN user group provides inestimable feedback. Questions that are continuously taken into account concern the type of strategies adopted, their usefulness, the choice of symbols, the clarity and speed of recognition, the flexibility offered by the tool set and overall user friendliness. Performers and audio engineers note that they find useful features that allow them to consult both a printed version of the score containing the SSMN symbols as well as its electronic version allowing rendering the symbols in an active timeline.

The potential of the prototype was also tested with several choreographers and their composers at Tanzhaus Zurich. Results of the SSMN project have been incorporated into the composition curriculum at the Zurich University of the Arts and have been presented at the Haute École de Musique of Geneva. The actual experience with the composers, interpreters and composition students has shown that they have experienced increased awareness of spatialization possibilities within their own creation process and developed an augmented spatial listening acuity. A future SSMN goal addresses developing awareness of spatialization through pedagogical interactive software for all school ages as well as for pre-professional music education. There also appears to be a need within musicological research for archiving and assuring the perennity of electroacoustic music, transcribed with symbols for study purposes. It is also expected that the SSMN project will contribute to generating a sustainable impact on creative processes involving three-dimensional spatialization.

Further aspects are also being investigated such as the integration within the MusicXML protocol and SpatDIF compatibility (Peters, Lossius and Schacher 2013). The SSMN tools set and documentation are available to the scientific and artistic communities via a website that has been setup to document project results, distribute the software, and receive user input.[3] The SSMN workflow is shown below (Figure 5).

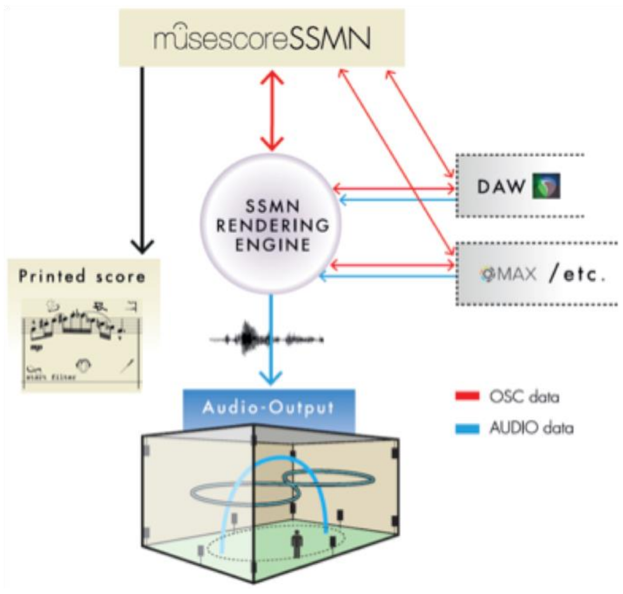


Figure 5. Basic MuseScoreSSMN I/O workflow.

Acknowledgements

The SSMN research team is grateful for the assistance and support offered by the Swiss National Foundation for Scientific Research, the members of Institute for Computer music and Sound Technology – University of the Arts of Zurich, the Computer Music division of the Haute École de Musique of Geneva, composers Vincent Gillioz, Mathias Aubert, Adam Maor and participants of SSMN courses and Tanzhaus choreographers.

REFERENCES

[1] Ellberger, Toro-Perez, Schuett, et al., “Spatialization Symbolic Music Notation at ICST”, *Proceedings ICMC|SMC*, Athens, 2014, pp. 1120-25.

[2] Wright, Freed, and Momeni, “OpenSound Control: State of the art 2003”, *Proceedings of the 2003 Conference on New Interfaces for Musical Expression* (NIME-03), Montreal, Canada, 2003.

[3] Agon, Assayag, and Bresson (Eds.), *The OM Composer's Book Vol. 1.*, Collection Musique/Sciences, Editions Delatour France/IRCAM, 2006.

[4] Cont, Giavitto, and Jacquemard, “From Authored to Produced Time in Computer-Musician Interactions”, *CHI 2013*, Workshop on Avec le Temps! Time, Tempo, and Turns in Human-Computer Interaction, Paris, France, ACM, 2013.

[5] Favori, Jean, “Les Unités Sémiotiques Temporelles”, *Mathematics and Social Sciences*, 45e année, n°178, pp. 51-55.

[3] http://blog.zhdk.ch/ssmn/

[6] Fober, Orlarey, and Letz, “Inscore - Un environnement pour la conception de Live Partitions”, *Actes de la Conférence Audio Linux*, BAC 2012,

[7] Theile, Günther, “Wave Field Synthesis – A Promising Spatial Audio Rendering Concept”, *Proceedings of 7th ICDAE*, DAF’04, Naples, 2004.

[8] Daniel, Jerome (2000). *Représentation de champs acoustiques, application à la transmission et à la reproduction de scènes sonores complexes dans un contexte multimédia.* Thèse de doctorat de l’Université, Paris 6.

ACCRETION: FLEXIBLE, NETWORKED ANIMATED MUSIC NOTATION FOR ORCHESTRA WITH THE RASPBERRY PI

K. Michael Fox
Rensselaer Polytechnic Institute
`kmichaelfox.contact@gmail.com`

ABSTRACT

In 2014, the author set out to expand the notational potential of their generative music systems to be performed by the Rensselaer Orchestra in Troy, NY. The experiments resulted in the use of several networked Raspberry Pi devices delivering a realtime, generative Animated Music Notation to subsections of the live orchestra during performance. This paper outlines the structure of the piece, *Accretion*; the technical details of its implementation; and the possibilities presented by using the Raspberry Pi to deliver scored materials to performers. Ultimately, the paper seeks to make a case for adopting the Raspberry Pi as a powerful device and method of distribution/performance of Animated Music Notation.

1. INTRODUCTION

This paper describes the author's composition for orchestra, *Accretion*, and the technological developments that enabled the creation and performance from an Animated Music Notation (AMN). *Accretion* is structured around techniques and structural schemata derived from granular synthesis. The term accretion refers to the formation of a thing by means of some attraction, perhaps gravitational as in the case of the formation of celestial bodies like planets, stars, and nebulae. This process works as a sufficient metaphor for the kinds of interactions, including granular synthesis, that the author has previously implemented in electronic and electroacoustic compositions. With granular synthesis, sounds that are near imperceptibly short can be overlapped in very high densities to produce particular sonic textures to emerge. The piece *Accretion* is a translation of these electronic idioms into the context of the acoustic orchestra.

The realtime generative systems comprising my previous work have diverse inspirations, origins, and implementations. However, *Accretion* explicitly seeks to incorporate a previously unused feature: rendering the internal interactions of the generative system into scored material that is performed in realtime by an ensemble. The opportunity to write for an orchestra posed that significant challenge in realtime notation. In order to translate these techniques from the realm of microsounds to the timescales and context of the orchestra, a custom AMN was designed. I set about to design a system that used the kinds of compositional methods I enjoy in the digital realm and rendered them legible (and with the utmost specificity) to the orchestra.

Gerhard Winkler, describing his own work towards realtime generative notation for ensembles, has emphasized the benefit of removing as much of the simulation hardware from the stage as possible [1]. However, it was clear that using a single screen was neither ideal nor practical. The projected image, for an ensemble the size of an orchestra, would have have to be impractically large to legibly display all necessary player parts. Additionally, the projection would either have to be positioned behind the audience, or the orchestra would have to face away from the audience. The solution I developed divided the score into four different screens; networked and synchronized these screens with a master simulation; and, positioned these screens in such a way that was unobtrusive to the audience while remaining legible to players up to 10 meters away. With these challenges in mind, I consulted other works using networked devices, screens or processes, including those described by Winkler [1], Jason Freeman's LOLC [2], and Decibel Ensemble's "Score Player" [3]. The Raspberry Pi (RPi) was affordable and flexible enough as a platform to realize these goals, and I adopted the device as a way to enable the compositional goals I had established. The rest of this paper describes the process of moving from compositional intent to functional Animated Music Notation design, its technological implementation, and future trajectories for the system.

2. *ACCRETION*

My compositional intentions with *Accretion* required each section of the orchestra to act independently of the others, facilitating the coordination of instrumental articulations into clouds where each event had a seemingly arbitrary timing. The components and structure of the piece, being derived from granular synthesis, relied on events happening in absolute time, as opposed to subdivisions of metrical time based on tempo. Coordination of these events, then, form clouds or clusters that are partially identified by their densities. However, since the instruments playing these sound grain-like notes are resonating bodies activated by humans, there are three additional components introduced: pitch, playing technique and articulations, and dynamics. Together, these components formed the main design considerations of the simulation system, programmed in C++ with openFrameworks.

2.1 Time & Pitch

The generative software system created "events" of two types: singular or durational. The singular events were realized as the shortest possible articulation of a note the instrument. Durational events, on the other hand, could be long sustained notes or collections of staccato notes occurring in a strictly defined time duration. Both event types consisted of a single pitch assigned at the time of generation. These pitch assignments are determined by an active "global" pitch class, constraining all instrumental parts to a pre-defined harmonic space, specifically the octatonic (WH) scale and the whole-tone scale.

Durational notes had an additional property when comprised of collections of short notes. These staccato notes were to be articulated as fast as possible at the prescribed dynamic. The resulting effect is a slightly asynchronous timing for the events resulting from the mechanical nature of the action and the limitations of the human body. This led to subtle emergent variation on the overlapping patterns of coexistent events, which was further amplified by the use of different playing techniques.

2.2 Technique

For each player part, divided by instrument sections, the events were assigned articulation techniques idiomatic to the instruments. Because the orchestra was comprised of student players, I limited the techniques to those that they would feel comfortable and confident performing (i.e., not extended). A string instrument could perform events as one of: arco, pizzicato, col legno tratto, col legno battuto, staccato, or tremolo; while winds would more uniformly play events legato.

These techniques would be assigned to each event as that event was generated and could apply to different event types in different ways. For example, col legno battuto for a singular event would be realized as a single strike of the bow, while the durational event using col legno battuto would be realized as multiple strikes of the bow over the course of the specified time duration and articulated as fast as possible.

2.3 Dynamics

In the case of singular events, a single discrete dynamic is generated (as there is only one short note played). Durational events, however, are assigned continuous dynamics that vary over the time duration of its articulation. Whether the durational event is a sustained note or a collection of short notes, these are contained within a the continuous dynamic envelope that makes each moment of that event vary in volume, intensity, and timbral quality.

Dynamic envelopes for these durational events were based on the Attack, Decay, Sustain, and Release (ADSR) envelopes of electronic sound synthesis. However, abstracted from the synthesis function, each phase of the envelope can increase or decrease an Attack phase of an envelope need not start at zero and can decrease before encountering the beginning of the Decay phase. The one exception to this freedom is the Release phase, which will always approach zero at the end of its duration.

These envelopes are applied to the durational events to vary the dynamics at any given moment between *dal niente* (when possible on the instrument) and *fff* (available, but seldom reached). Since dynamic envelope is given to each durational event, each section of the orchestra is completely decoupled from the others with respect to crescendi or descrescendi. This allows the ebbing and flowing of different timbres over and under each other in graceful coordination (or, in reality, lack thereof).

2.4 Notational Framework

David Kim-Boyle has noted that "computer-generated scores, particularly those that employ real-time animation, create a heightened sense of anticipation amongst performers, especially given that performance directive can change from moment to moment" [4]. Similarly, Pedro Rebelo notes that there is a delicate balance in animated notations between representing gestures too literally and too abstractly [5]. Given these two considerations and the goals of the piece, I believed that it was important to render the notation in a form that was reasonably approachable by any of the performers. Like the use of idiomatic over extended techniques, I wanted to present the notation for my piece which functionally achieved the specific timings and re-

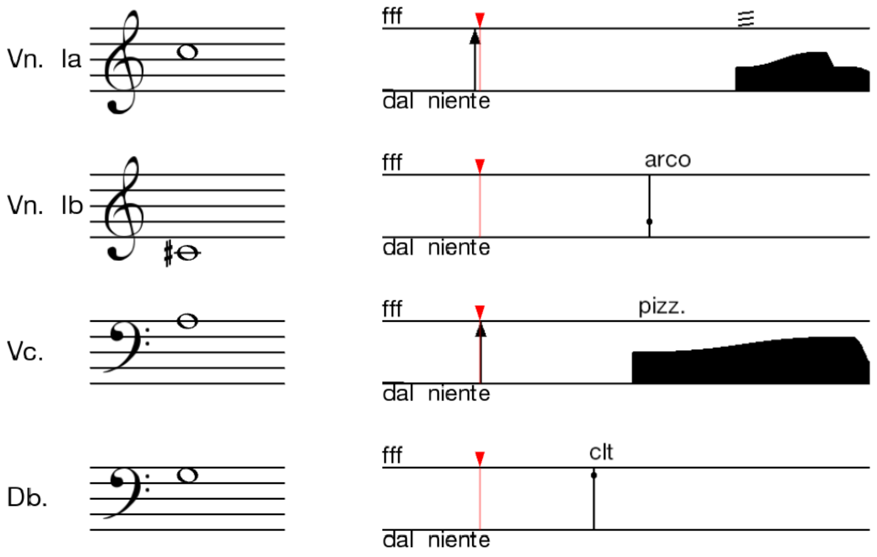


Figure 1. Reduction of concert score.

sults that I desired while taking the form of notation that the student players would feel confident reading and performing from. This led to an extension of conventional notation with meaningfully animated gesture-figures. As seen in Fig. 1, on the left side of the score, notes for each instrument's *next* or *current* event were displayed on staves with accidentals. To the right, the articulation for each instrument's gestures would slide from right to left down a pipeline.

Using the "playhead" style indicator described by Hope and Vickery, the articulation point occurs when a gesture crosses a red line with a downward facing arrow [6] (See Fig. 3). For singular events, the dynamic of the articulation is featured as a small circle that is vertically bisected by a black line (See Fig. 4). The performers were coached to regard the center of this circle both as the "dynamic value" of the note and as the point of attack as it passed through the playhead indicator. Thus, as the circle passes the playhead a note at the pitch specified on the staff is articulated with the specified technique (described below) at the dynamic corresponding to the height of the circle's center point. For durational events, the dynamic is read as the height of the top of the envelope at the point where it is currently intersecting the playhead. In either case, the vertical range of the envelope pipeline is listed on the score as *dal niente* at the bottom and *fff* at the top. Most instruments were expected to perform *dal niente* as the quietest dynamic they could possibly play.

Since all durational events approach *dal niente* as they

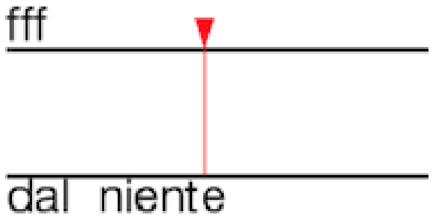


Figure 2. Playhead notational indicator.

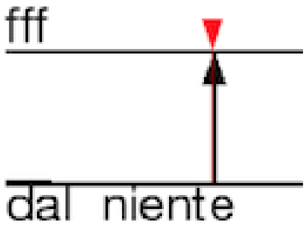


Figure 3. Termination of durational event.

conclude, the final moment of these types of events was initially very ambiguous. Because of the specificity that I was seeking with the score, I added a vertical line to clearly demarcate the termination point of these envelopes. To distinguish the function of this line from the line that is connected to singular events, the termination point featured an upward-pointing triangle at its top that is clearly distinguishable from the circle (of the singular event). Similar to the singular event, the performers were coached to interpret the top-most point of the triangle as the point of

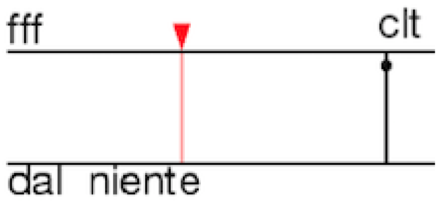


Figure 4. Singular Event.

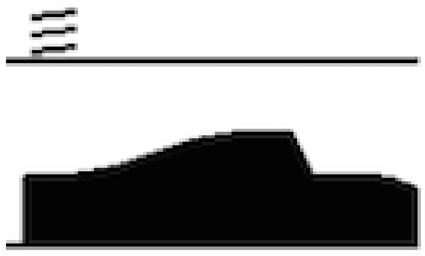


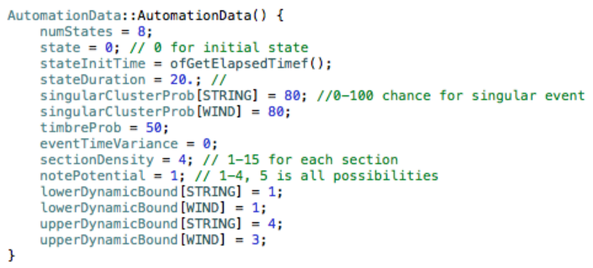
Figure 5. Durational event envelope.

contact as it passes the playhead. This helped to cue events such as that pictured in the Fig. 3, where a part stays at *dal niente* for a long period of time (seemingly absent).

Initially, the technique was placed above the notes on the stave, as would be expected on a more traditional paper score. However, rehearsals demonstrated that much more consistent attention was required for the dynamic envelopes, and the techniques were more effectively executed when they were rendered with the envelope to which they referred. This resulted in envelopes that are rendered with their respective technique also moving down the pipeline directly above their leading edge.

3. RASPBERRY PI AS NOTATION RENDERING CLIENT

The most critical point in understanding the notation rendering system is the distinction that the notation itself is a front-facing, or front end notational representation of events generated within a simulation system, or back end. To understand how the client renders these events, I will first describe how the simulation system generates events within *Accretion*. At a macro-level, the piece is organized around clusters of events with particular attention to densities of players, the types of events they are performing (and in what distribution the types of events appear amongst these active players), and the dynamic levels of these events. Within these events, or clusters, the collection of events are generated much like random grains are spawned within granular synthesis clouds. Fig. 6 shows a C struct called automationData that holds the distribution parameters of the event generation. As the piece progresses, these distribution parameters are updated to vary the density, number



```
AutomationData::AutomationData() {
    numStates = 8;
    state = 0; // 0 for initial state
    stateInitTime = ofGetElapsedTimef();
    stateDuration = 20.; //
    singularClusterProb[STRING] = 80; //0-100 chance for singular event
    singularClusterProb[WIND] = 80;
    timbreProb = 50;
    eventTimeVariance = 0;
    sectionDensity = 4; // 1-15 for each section
    notePotential = 1; // 1-4, 5 is all possibilities
    lowerDynamicBound[STRING] = 1;
    lowerDynamicBound[WIND] = 1;
    upperDynamicBound[STRING] = 4;
    upperDynamicBound[WIND] = 3;
}
```

Figure 6. "automationData" struct.

of active players, the probability of each instrument family generating events, and the probability of which event types will be active at any given point.

Following from the notion that the score is a front-facing representation of the system's event generation, the networked RPi scores are simply the interface elements of the simulation without the simulation backend running locally on the devices. Instead, the RPi's listen for OSC messages that encapsulate the parameters of events as they are generated. At the moment when an event is generated, its parameters are packaged as this OSC message and broadcast to the clients via a LAN connection. The address of these messages takes the format "\number_of_RPi_device\number_of_part" and these address numbers are hardcoded locally. These messages take different forms depending on the type of event they represent, so each OSC message starts with an identifier of which event type it refers to, 0 for singular events and 1 for durational events. When the messages represent a singular event, the rest of the message's arguments includes, in order: an integer midi pitch value, an integer delay in milliseconds from the start of the cluster, an enumerated integer value for the dynamic level, an enumerated integer value for the instrument family (to identify the technique behavior), and an enumerated integer value for the technique type (Winds only use legato, so 0 is used as a placeholder).

When a durational event is created, the OSC message must describe the parameters of the ADSR envelope. The arguments that appear after the durational events first identifier argument are as follows: an integer midi pitch value, an integer delay in milliseconds from the start of the cluster, an integer value for Attack time in milliseconds, an enumerated integer value for initial Attack dynamic, an enumerated integer value for the final Attack dynamic, an integer value for Decay time in milliseconds, an enumerated integer value for the final Decay dynamic, an integer Sustain time in milliseconds, an enumerated integer value for the final Sustain dynamic, an integer value for Release time in milliseconds, an enumerated integer value for instrument family, and an enumerated integer value for technique type. The instrument identifier and technique type behave as described for the singular events.

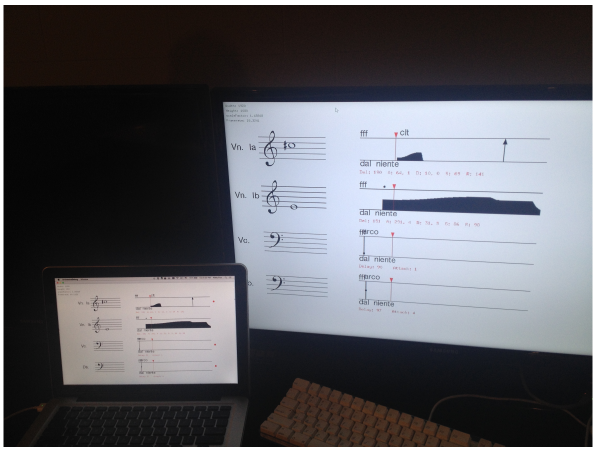


Figure 7. Score rendering on linked Server (laptop, left) and Client (large monitor, right) applications.

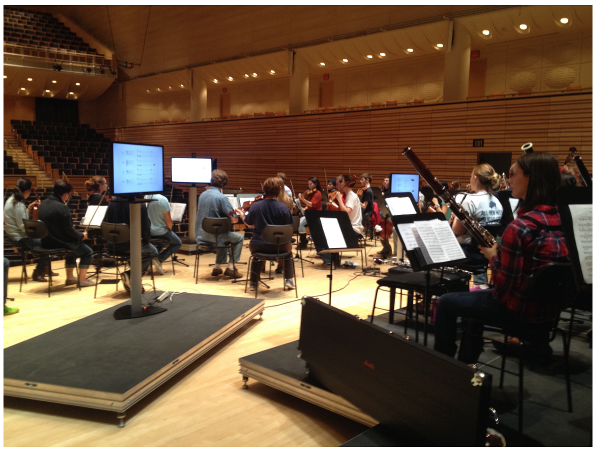


Figure 8. Stage setup in the Concert Hall at Rensselaer Polytechnic Institute's Experimental Media and Performing Arts Center. Monitors 2 (middle), 3 (right), and 4 (left) are visible.

Fortunately, openFrameworks is becoming increasingly well-supported for the Raspberry Pi. The code that rendered the notation could be written initially on an Apple OS X computer and redeployed later by recompiling the application for ARM devices (RPi). Minor performance optimizations were then be implemented to assure smooth performance on the different architectures. One such improvement was using a time-based tick system for progressing the system. This allowed for slight fluctuations in the framerate of the rendering system while maintaining the temporal specificity necessary for the coordination of events across multiple RPi's.

The HDMI support that is natively included with the RPi allows easy connection to most monitors or projectors on hand. In the case of *Accretion*, I used four 32 inch video monitors. Each of the RPi devices are small enough that they could be located on stage with the monitors displaying the rendered material and supplied only with power and ethernet cables. Thus, each monitor worked as one client in a vaguely server-client relationship with the simulation. Because the RPi works just like most other Unix systems, the rendering client on each device could be activated using SSL from the simulation machine off-stage at the beginning of the performance. Once the performance was over, each device was sent kill message also using the SSL connection, allowing a quick strike from the stage.

4. CONCLUSION & FUTURE WORK

The first major benefit that has emerged since completing *Accretion* is the portability of the score. These inexpensive RPi devices are small enough that they can easily be shipped to new ensembles, plugged in, and ready for rehearsal or performance in a relatively short amount of time. By supplying the devices themselves, instead of the application or source files for compilation, this setup has incidentally avoided the potential headaches stemming from incompatible versioning of libraries or system architectures.

Furthermore, the RPi features a number of standard GPIO ports that allow a variety of sensors to be attached. Each Raspberry Pi client could simultaneously collect sensor data, communicate that data back to the server system via OSC, and create responsive feedback loops for interactive generative music systems with human performers. Though *Accretion* was not interactive or responsive to outside elements in the performance, this is fertile ground of future work and significant precedent exists for adaptive realtime scores [Nick Didkovsky's *Zero Waste*, Harris Wulfson's *LiveScore* system [7], Paul Turowski's *Hyperions*, Jason Freeman's LOLC system [2]].

Likewise, the devices are easily extended with other microcontroller hardware, such as Arduino boards. By attaching external Arduino devices to the device, it could be possible to do sophisticated distributions of sensor and data processing in a highly networked setting.

There is also flexibility in the spatially-distributed networking potential on these devices, enabling work reminiscent of Arthur Clay's *China-Gates* or *Going Public*. RPi's are equipped out of the box with ethernet capabilities, and are extendable with usb wifi adapters, a tremendous advantage in distributed or spatialized performance situations. For instance, it could be possible to synchronize devices in remote parts of a building via a network without any significant demand to setup or infrastructure.

All of these considerations, however, require extensions to hardware, software, notational practices, or all of the above. This seems to be the nature of much of the prominent practices in AMN already, though. Each composition's needs can be contextually defined to the extent that entirely new notational schemata might be created for individual pieces. Indeed, the system I designed was motivated by the need to realize the compositional ideas that formed *Accretion*. However, I see many potentials for generalizing the tool along several of the trajectories outlined above to further enable the compositional ideas I am interested in.

The main addition that I see for the system is inspired by another common usage of the RPi: the networked media browser. Common media browser applications include KODI (formerly XBMC). In the case of networked scores, this type of score access would require the development and implementation of configuration flat-files to aid in the networking functionality. The configuration file schema would theoretically allow scores to specify their role in the network at runtime (such as the particular instruments they display parts for, in the context of *Accretion* for example), or support preset values that can be implemented at device startup.

Similar benefits are represented by the ScorePlayer app, developed by the Decibel Ensemble. This implementation simply provides an alternative platform that enables a more diverse configurability and extended low-level control over notational structures, but at the expense of a less widely distributed device (relative to iOS devices and their ostensible ubiquity). However, as stated above, the packaging of scores on a standalone, pre-configured device, but requiring little to no setup by performers, may increase the potential for wider distribution.

Acknowledgments

Special thanks to Ryan Ross Smith and Nicholas DeMaison.

5. REFERENCES

[1] G. Winkler, "The realtime-score. a missing-link in computer-music performance," in *Sound and Music Computing*, Paris, 2004.

[2] S. W. Lee, J. Freeman, and A. Collela, "Real-Time Music Notation , Collaborative Improvisation , and Laptop Ensembles," in *NIME 2012 Proceedings of the International Conference on New Interfaces for Musical Expression*, 2012, pp. 361–364.

[3] C. Hope and A. Wyatt, "Animated music notation on the ipad." in *PROCEEDINGS OF THE INTERNATIONAL COMPUTER MUSIC CONFERENCE*, ser. International Computer Music Conference, 2013, pp. 201 – 207. [Online]. Available: `http://quod.lib.umich.edu/cgi/p/pod/dod-idx/animated-music-notation-on-the-ipad.pdf?c=icmc;idno=bbp2372.2013.025`

[4] D. Kim-Boyle, "Real-time score generation for extensible open forms," *Contemporary Music Review*, vol. 29, no. 1, pp. 3–15, February 2010.

[5] P. Rebelo, "Notating the unpredictable," *Contemporary Music Review*, vol. 29, no. 1, pp. 17–27, February 2010.

[6] C. Hope and L. Vickery, "Screen scores: New media music manuscripts." in *PROCEEDINGS OF THE INTERNATIONAL COMPUTER MUSIC CONFERENCE*, ser. International Computer Music Conference, 2011, pp. 224 – 230. [Online]. Available: `http://libproxy.rpi.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsbl&AN=CN080981465&site=eds-live&scope=site`

[7] G. D. Barrett and M. Winter, "Livescore: Real-time notation in the music of harris wulfson." *Contemporary Music Review*, vol. 29, no. 1, pp. 55 – 62, 2010. [Online]. Available: `http://libproxy.rpi.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=53772450&site=eds-live&scope=site`

SINGLE INTERFACE FOR MUSIC SCORE SEARCHING AND ANALYSIS (SIMSSA)

Ichiro Fujinaga
Schulich School of Music
CIRMMT, McGill University
ich@music.mcgill.ca

Andrew Hankinson
Schulich School of Music
CIRMMT, McGill University
andrew.hankinson@mail.mcgill.ca

ABSTRACT

Single Interface for Music Score Searching and Analysis (SIMSSA) project seeks to design tools and techniques for searching and analyzing digitized music scores. Specifically, we seek to provide researchers, musicians, and others to access the contents and metadata of a large number of scores in a searchable, digital format. In this project, we are developing prototypes for processing and accessing the scores by consulting closely music researchers, musicians, and librarians.

1. INTRODUCTION

While a growing number of digitized images of music scores are being made available on-line to a global audience, these digital images are only the beginning of true accessibility since the musical content of these images cannot be searched by computer. The goal of the Single Interface for Music Score Searching and Analysis (SIMSSA) project is to teach computers to recognize the musical symbols in these images and assemble the data on a single website, making it a comprehensive search and analysis system for online musical scores. SIMSSA is creating an infrastructure for processing music documents, transforming vast music collections into symbolic representations that can be searched, studied, analyzed, and performed anywhere in the world.

SIMSSA is made up of two research axes. The first axis, *Content*, is developing large-scale optical music recognition (OMR) systems for digital images to transforming them into searchable symbolic notation. The second axis, *Analysis*, is developing tools and techniques for large-scale search and analysis. We have assembled a diverse team of researchers and partners to accomplish this mission: music scholars, composers, and performers will ensure that we build tools to address real-world problems, librarians will provide expertise in collection management, metadata, and information-seeking behaviour, and music technologists will develop OMR systems, accessible web-based interfaces, and search and analysis software. Partner institutions including museums, national and research libraries, and universities will provide both digital images and expertise. Central to SIMSSA is the use of collaborative computing, which has been shown to reduce costs and increase accuracy. Musicians, students, and scholars from around the world will be provided with tools to correct and improve the results of the recognition process. They will correct the OMR output for music sources they care about, resulting in searchable music for their own work as well as for other musicians. The SIMSSA network will be a global network of digital music libraries and participant-users: anyone with a web browser will be able to search through vast amounts of music from anywhere in the world.

2. OBJECTIVES

Our objective is to develop a new approach to building globally-accessible digital music scores with a public online digital document analysis and retrieval system. Using OMR technology, we are working with partner institutions to automatically transcribe the contents of their large digital collections, and allow users to search music notation in millions of music scores.

The searchable symbolic content will make it possible to easily compare, analyze, study, arrange, and transpose musical material in new ways. Our tools will provide new kinds of access and exposure to the collections of our partner institutions, from document viewing technology to search engines. New access to large amounts of music and new tools will provide important fundamental materials for future scholarship, creation, and performance. The complexity and variety of musical styles and music notations will lead to important advances in information retrieval and digital document analysis with multiple uses beyond music. As the first project of its kind, we hope that SIMSSA will establish common standards and best practices for these types of music information retrieval and serve as a baseline for future work in this field.

3. BACKGROUND

OMR research began in the late 1960s and has seen limited but continuous interest with several commercial software packages available (e.g., SmartScore and SharpEye). Development of this technology has been slow, and most of the research on OMR has concentrated on Common Western Notation, the most widely used music notation system today (for a recent review, see [1]). In the development stage of this project, we have created a site for the *Liber usualis* (`http://liber.simssa.ca`), to experiment with the methods and procedures of performing OMR on entire books containing older music notation. The website allows a user to search a digitized edition of this book using pitch names, neume names, and OCR-transcribed text [2] (Figure 1). Our experience with the Liber project has reinforced the need for a robust and efficient workflow system for OMR.

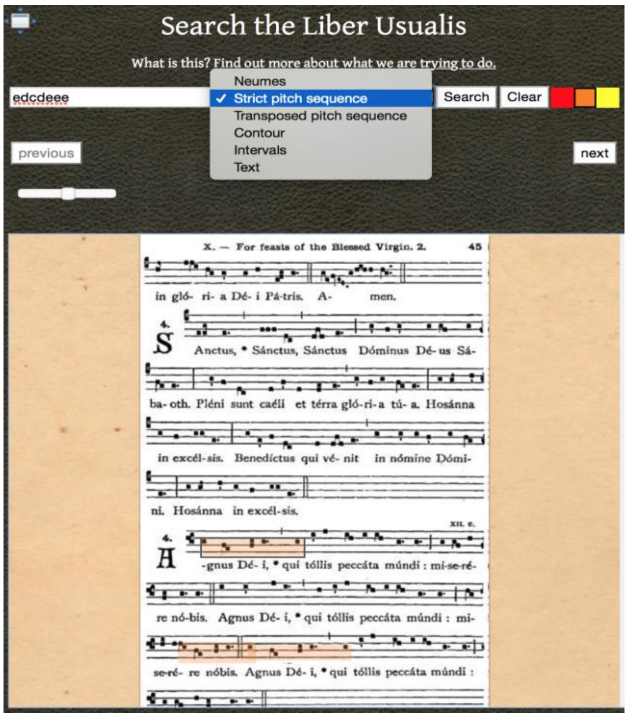

Search the Liber Usualis
What is this? Find out more about what we are trying to do.
edcdeee
Neumes
✓ Strict pitch sequence
Transposed pitch sequence
Contour
Intervals
Text
Search
Clear
previous
next
X. — For feasts of the Blessed Virgin. 2.
45
in gló- ri- a Dé- i Pá-tris. A- men.
Anctus, * Sánctus, Sánctus Dóminus Dé- us Sá-
ba- oth. Pléni sunt caéli et térra gló-ri- a tú- a. Hosánna
in excél- sis. Benedíctus qui vé- nit in nómine Dómi-
ni. Hosánna in excél- sis.
XII. c.
-gnus Dé- i, * qui tóllis peccáta múndi : mi-se-ré-
re nó-bis. Agnus Dé- i, * qui tóllis peccáta múndi : mi-
se-ré- re nóbis. Agnus Dé- i, * qui tóllis peccáta múndi :

Figure 1. Search the *Liber usualis* website (`http://liber.simssa.ca`) with the *Mary had a little lamb* tune search.

4. THEORETICAL FRAMEWORK

The SIMSSA project is developing a new cloud-based OMR system that introduces a completely new paradigm for this class of software. Typically, OMR software is installed and operated on a single computer or workstation. Advanced techniques used to perform image restoration and automatic music transcription, however, are computationally intensive, sometimes requiring hours or even days to run on personal computers. Instead, we are developing systems where computationally intensive procedures can be distributed across many powerful server machines attached to the Internet to perform processing in parallel, meaning any computer or mobile device with a modern web browser and access to the Internet may act as a document recognition station, offloading the computationally-intensive recognition tasks to large clusters of computers in data centers. We see this as our most significant technological contribution: these techniques, known as distributed computing, are currently being explored in text-recognition research but have not yet been explored for music recognition systems.

There is a successful precedent for projects of this scope and scale. The IMPACT project [3] was a project funded by the European Union (€15.5 million, 2008–2012) that focused on digitization, transcription, and policy development for historical text documents. This project brought together national and specialized libraries, archives, universities, and corporate interests to advance the state of the art in automatic text document transcription, explicitly for the purposes of preserving and providing access to unique or rare historical documents. They have published significant advances in historical text recognition, tool development, policies, and best practices [4], [7].

At the core of the IMPACT project was a networked and distributed document recognition suite, providing a common document recognition platform for all their partners across Europe. As the computer vision and software engineering teams developed new tools and algorithms to improve recognition, these were made available immediately to all partners simply by updating the online suite of tools. All partners could then supply realtime feedback and evaluation on these updates, comparing them to previous techniques “in the field” and reporting their findings. The development teams then incorporated the feedback into further developments and refinements. This project has become self-sustaining and is now known as the IMPACT Centre of Competence, a not-for-profit organization that continues to build the technologies and best practices of the formally funded project. This represents a model that we hope to reproduce in the domain of music.

5. METHODOLOGY

5.1 Content and Analysis Axis

We are building a robust infrastructure with workflow management and document recognition systems, crowdcorrection mechanisms, networked databases, and tools for analyzing, searching, retrieving, and data-mining symbolic music notation. Responsibility for developing these tools within the project is shared between the *Content* and *Analysis* axes.

The Content axis is divided into three sub-axes: *Recognition*, *Discovery*, and *Workflow*. The Recognition sub-axis is responsible for developing the underlying technologies in machine learning and computer vision. The Discovery sub-axis is responsible for large-scale web crawling, finding, and identifying images of books that contain musical content. Finally, the Workflow sub-axis is responsible for developing user-friendly web-based tools that harness the technologies developed by the other two sub-axes.

The Analysis axis is divided into two sub-axes: *Search and Retrieval*, and *Usability*. Searching music is complex since, unlike text, it is not simply a string of characters: there are pitches, rhythms, text, multiple voices sounding simultaneously, chords, and changing instrumentation. The Search and Retrieval axis is responsible for developing ways of mining the notation data generated by the Content axis in all its complexity, building on the work done in the ELVIS Digging into Data Challenge project (http://elvisproject.ca). This axis is also developing techniques for computer-aided analysis of musical scores. The Usability sub-axis is responsible for studying retrieval systems and user behavior within the context of a symbolic music retrieval system, identifying potential areas where the tools may be improved to suit real-world retrieval needs.

5.2 Content: Discovery sub-axis

Mass digitization projects have been indiscriminately digitizing entire libraries' worth of documents—both text and musical scores—and making them available on individual libraries' websites. The Discovery sub-axis is developing a system that will automatically crawl millions of page images looking for digitized books with musical examples [8]. When it finds a document containing printed music it will use the OMR software to transcribe and index the music content for these documents.

5.3 Content: Recognition sub-axis

One of the major tasks of the Recognition sub-axis is the integration of two desktop open-source OMR software platforms: Gamera, a document analysis toolkit [9], and Aruspix, an advanced OMR system developed by Laurent Pugin [10]. These systems are unique for their ability to "learn" from their mistakes by using human corrections of misrecognized symbols to improve their recognition abilities over time. We have shown this to be cost-effective in digitization and recognition workflows [11]. The next logical step is to bring these systems to our cloud-based OMR platform. This will allow us to distribute the correction tasks to potentially thousands of users around the globe, thereby providing the means to collect large amounts of human correction data. This crowd-sourced adaptive recognition system will be the first of its kind [12].

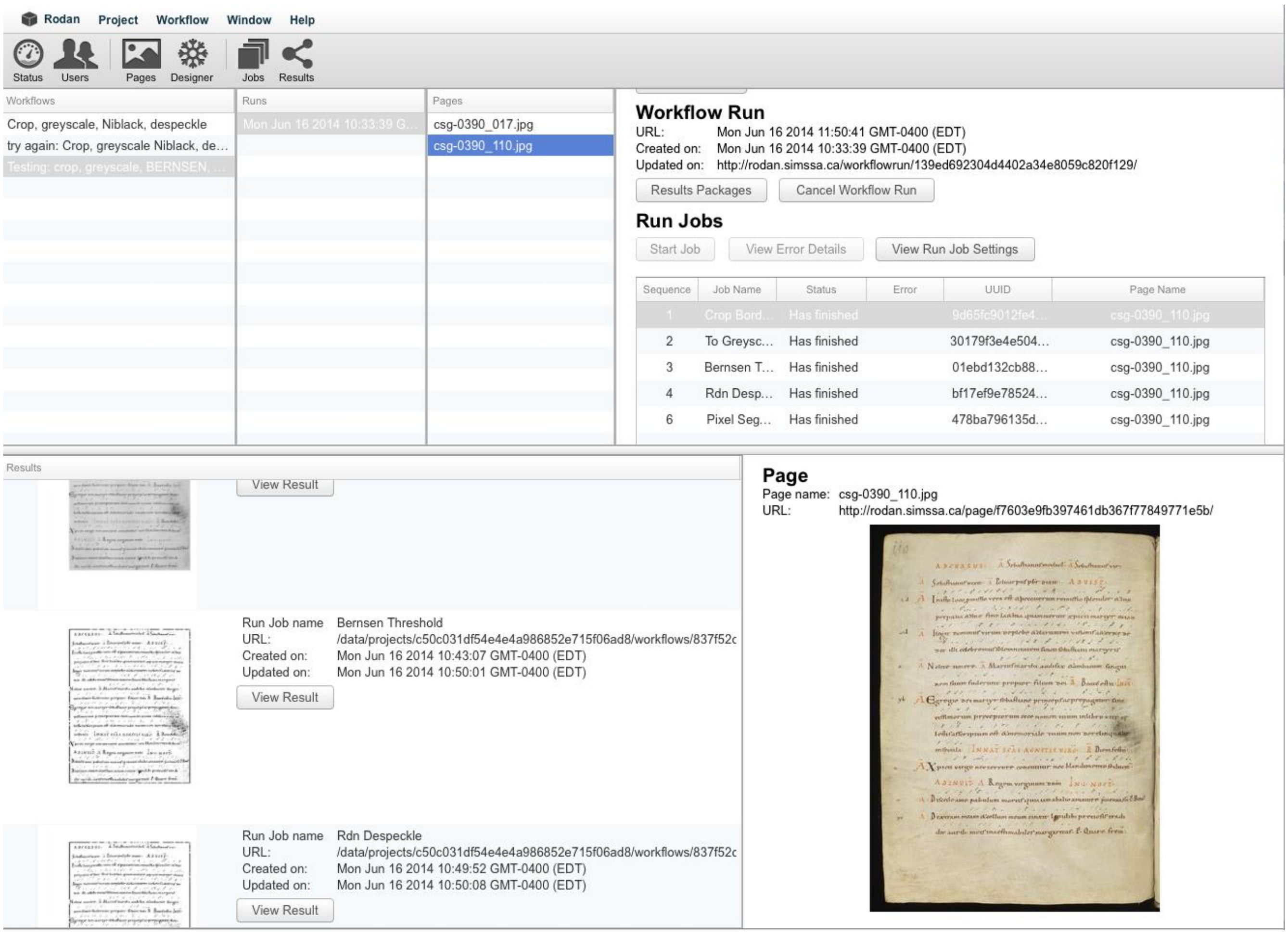
5.4 Content: Workflow sub-axis

The Workflow sub-axis is primarily responsible for developing Rodan, the core platform for managing cloud-based recognition. Rodan is an automatic document recognition workflow platform. Its primary function is to allow users to build custom document recognition workflows containing document recognition tasks, such as, image pre-processing and symbol recognition (Figure 2). Rodan is capable of integrating many different recognition systems, such as Aruspix and Gamera, with other systems (e.g., integrating text recognition tasks for performing automatic lyric extraction) (see Figure 3). Once a workflow has been created, Rodan manages digital document images' progression through these tasks. Users interact with their workflows through a web application, allowing them to manage their document recognition on any Internet-connected device, but all tasks are actually run on the server-side. Storage and processing capabilities can be expanded dynamically, and new tasks can be seamlessly integrated into the system with no need for the users to update their hardware or software.

Moreover, as a web-based system, Rodan can incorporate many different methods for distributed correction or "crowd-sourcing" to provide human-assisted quality control and recognition feedback for training and improving recognition accuracy. This follows a similar model to that proposed by the IMPACT project where distributed proof-readers provide feedback. These proof-readers correct any misrecognized symbols, and their corrections will then be fed back into the recognition system, thereby improving the recognition for subsequent pages and documents. This type of crowd-sourced correction system is employed in many text-recognition projects [13], [14], but there are no such systems in development for musical applications. The success of crowd-sourcing as a viable means of collecting correction and verification data has been demonstrated by a number of projects, most notably the Australian newspaper (TROVE) [15], Zooniverse [16] and reCAPTCHA [17]. Along with developing the technical mechanisms for crowd-sourced musical corrections, the Workflow team is also working with the Usability sub-axis on creating new ways to entice users to participate. Some ways of doing this would be to create a game that rewards users with points or community credibility in exchange for performing work [18], or reframing musical tasks as simple non-musical tasks (e.g., shape or colour recognition) so that they become solvable by an untrained audience. By diversifying the number of approaches to collecting crowd-sourced correction data, we expect to appeal to a wide number of communities, from specialists to the general public.

Later in this project, we will experiment with optical character recognition (OCR) for print and manuscript sources of music. By this point in the project we will have collected a large number of written texts with human-transcribed ground-truth data. We will use this to train machine-learning algorithms to automatically recognize the various text scripts present in these sources. Our goal here is to automatically align text with the music above it, an important step that represents a significant challenge, and an avenue of research that has never before been explored. This will allow users to perform searches for recurring patterns that include music and text — to identify whether, for example, a particular musical idiom is frequently used when the text refers to "God" or "love" — a type of search that is not possible with current systems. When the text-alignment task is complete, the Recognition team will work with the Analysis team to design and implement a search interface so that the users can search music and text simultaneously.

Many musical documents, especially those that are hundreds of years old, pose difficulties for computer recognition due to faded inks, bleed-through, water, or insect damage. Each of these problems is a potential source of transcription errors. The Recognition team is working on integrating the latest document-imaging enhancement technologies, such as adaptive binarization, bleed-through reduction, colour adjustment, and distortion analysis and correction.



Workflows
Crop, greyscale, Niblack, despeckle
try again: Crop, greyscale Niblack, de...
Testing: crop, greyscale, BERNSEN, ...

Runs
Mon Jun 16 2014 10:33:39 G...

Pages
csg-0390_017.jpg
csg-0390_110.jpg

Workflow Run
URL: Mon Jun 16 2014 11:50:41 GMT-0400 (EDT)
Created on: Mon Jun 16 2014 10:33:39 GMT-0400 (EDT)
Updated on: http://rodan.simssa.ca/workflowrun/139ed692304d4402a34e8059c820f129/

Results Packages | Cancel Workflow Run

Run Jobs

Start Job | View Error Details | View Run Job Settings

Sequence	Job Name	Status	Error	UUID	Page Name
1	Crop Bord...	Has finished		9d65fc9012fe4...	csg-0390_110.jpg
2	To Greysc...	Has finished		30179f3e4e504...	csg-0390_110.jpg
3	Bernsen T...	Has finished		01ebd132cb88...	csg-0390_110.jpg
4	Rdn Desp...	Has finished		bf17ef9e78524...	csg-0390_110.jpg
6	Pixel Seg...	Has finished		478ba796135d...	csg-0390_110.jpg

Results

View Result

Run Job name: Bernsen Threshold
URL: /data/projects/c50c031df54e4e4a986852e715f06ad8/workflows/837f52c
Created on: Mon Jun 16 2014 10:43:07 GMT-0400 (EDT)
Updated on: Mon Jun 16 2014 10:50:01 GMT-0400 (EDT)

View Result

Run Job name: Rdn Despeckle
URL: /data/projects/c50c031df54e4e4a986852e715f06ad8/workflows/837f52c
Created on: Mon Jun 16 2014 10:49:52 GMT-0400 (EDT)
Updated on: Mon Jun 16 2014 10:50:08 GMT-0400 (EDT)

View Result

Page
Page name: csg-0390_110.jpg
URL: http://rodan.simssa.ca/page/f7603e9fb397461db367f77849771e5b/

Figure 2. Rodan client interface.

It is also important to have a robust modern file format to store all of the symbolic data representations of these musical documents to meet our needs. Based on previous work we have chosen the MEI (Music Encoding Initiative) format [19]. As part of the SIMSSA project we will be forming a workgroup to enhance MEI support for digital encoding of early notation systems for chant and polyphonic music.

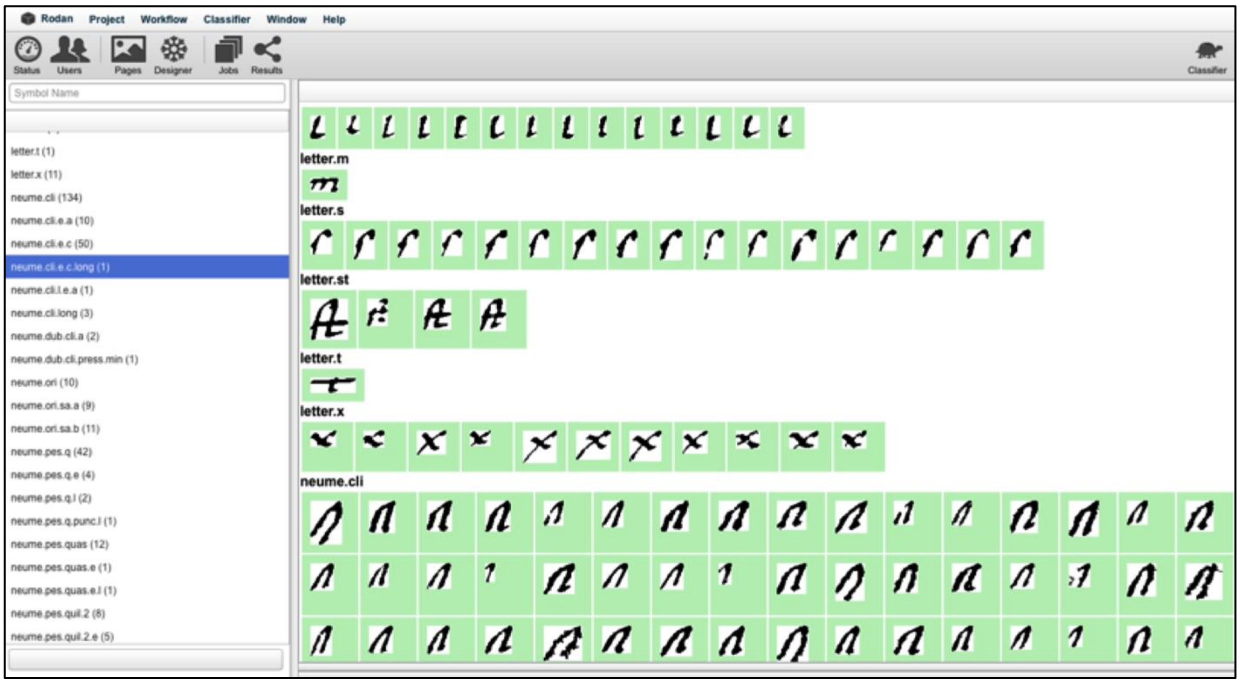
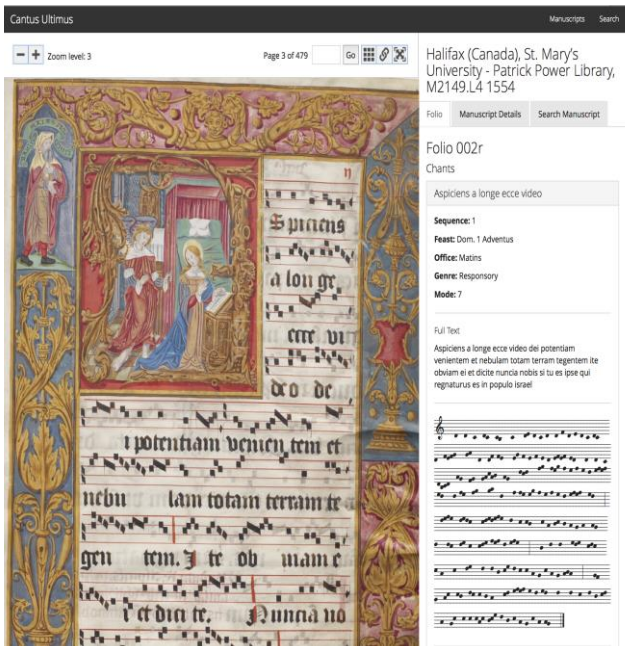


Figure 3 : Rodan's Gamera symbol classifier interface. The symbols are from a 10th-century St. Gallen music manuscript.

To evaluate Rodan and the accuracy of our OMR systems, we have selected several manuscripts and early printed scores that will be processed in order of increasing difficulty for our tools. We have started with a selection of Renaissance prints and late chant manuscripts, some of which are already available online (Figure 4).



Halifax (Canada), St. Mary's University - Patrick Power Library, M2149.L4 1554

Folio 002r

Chants

Aspiciens a longe ecce video

Sequence: 1

Feast: Dom. 1 Adventus

Office: Matins

Genre: Responsory

Mode: 7

Full Text

Aspiciens a longe ecce video dei potentiam venientem et nebulam totam terram tegentem ite obviam ei et dicite nuncia nobis si tu es ipse qui regnaturus es in populo israel

Figure 4 : Salzinnes manuscript website, `http://cantus.simssa.ca`.

As we proceed, we will evaluate the strengths and weaknesses of the workflow system, constantly adjusting our methods before moving on to the next source. For each document, we will create a human-created transcription of the music notation. This data will be the "ground truth" against which we will evaluate the performance and accuracy of the OMR system (and later for the OCR system). This will allow us to quantify any improvements we make in our OMR systems as we develop new recognition methods. By making incremental modifications using different types of sources, we hope to build a robust system capable of processing a wide range of musical documents.

5.5 Analysis: Search and Retrieval sub-axis

The Search and Retrieval component of the Analysis axis will involve music historians and music theorists to investigate how computerized searches of large collections of digital music can fundamentally change music history, analysis, and performance. They will develop new techniques for searching and analyzing digitized symbolic music. Searching music poses special challenges. A search interface must be able to search for strings of pitches, rhythms, or pitches and rhythms combined, search polyphonic music for multiple simultaneous melodies or chords, and search vocal music for both text and music. Searching and retrieving are only the beginning, however; members of the Analysis axis are developing software for many different types of computerized analysis of large amounts of music. This will allow scholars to describe style change over time, discovering which features of style stay the same and which change, or to describe what makes one composer's music unlike that of his or her contemporaries. Musicians and students will be able to find all the different ways composers have harmonized a specific melody from the Middle Ages to the present. Representation of search and analysis findings will be another focus of this axis, investigating new methods for searching and retrieving millions of digitized music documents.

Recent projects such as the Josquin Research Project (`http://jrp.ccarh.org`), Music Ngram Viewer (`http://www. peachnote.com`) [20], and ELVIS project (now part of the SIMSSA project) are already searching millions of notes. All these projects, however, have mostly depended on centralized, labour-intensive, manual processes to transcribe the sources into symbolic notation, append metadata to the resulting files, and arrange them in structured databases. SIMSSA will greatly streamline this process through automation and distributed labour, and enable the sophisticated automatic music analysis of very large corpora begun through ELVIS.

5.6 Analysis: Usability sub-axis

Librarians and information scientists are leading the Usability sub-axis. They are continually review the usability of our tools: Rodan, search interfaces, crowdsourcing interfaces, and analysis and visualization interfaces, considering the needs and skillsets of many different types of users, from senior music scholars with little technical expertise, to computer-savvy amateur musicians, to choral directors and guitarists searching for sheet music.

6. CONCLUSIONS

The most important outcome of this project is to allow users—scholars, performers, composers, and the general public—to search and discover music held in archives and libraries around the world. We expect that this will fundamentally transform the study of music and allow a global audience of musicians and artists to discover previously unknown or overlooked pieces for performance, making undiscovered repertoires that extend beyond the classics available to the general public. We also expect the public availability of large amounts of musical data to lead to significant advances in the field of music theory and the birth of the long-awaited field of computational musicology. Lastly, we expect that the free and open-source tools we are developing will help lead significant advances in the following areas, all of which are either completely new or novel applications of existing technologies:

- Public, web-based tools for historical image restoration;
- Public, web-based distributed (“cloud”) processing tools for OMR and OCR;
- A large database of automatically transcribed music;
- Prototypes for a web-based editor for making corrections or comparative editions of digital sources;
- A music exploration interface allowing quick and efficient content-based search and retrieval across a large-scale notation database; and
- Advanced public, web-based music analytical tools.

Acknowledgments

This research is, in part, supported by the Social Sciences and Humanities Council of Canada and le Fonds de Recherche du Québec sur la Société et la Culture, and McGill University.

7. REFERENCES

[1] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marcal, C. Guedes, and J. S. Cardoso, “Optical music recognition: State-of-the-art and open issues.” *International Journal of Multimedia Information Retrieval* (March 2012), pp. 1–18, 2012.

[2] A. Hankinson, J. A. Burgoyne, G. Vigliensoni, A. Porter, J. Thompson, W. Liu, R. Chiu, and I. Fujinga. “Digital document image retrieval using optical music recognition,” in *Proceedings of the International Society for Music Information Retrieval Conference*, Porto, Portugal, 2012, pp. 577–582.

[3] H. Balk and L. Ploeger, “IMPACT: Working together to address the challenges involving mass digitization of historical printed text,” *OCLC Systems & Services*, vol. 25, no. 4, pp. 233–248, 2009.

[4] V. Kluzner, A. Tzadok, Y. Shimony, E. Walach, and A. Antonacopoulos, “Word-based adaptive OCR for historical books,” in *Proceedings of the 10th International Conference on Document Analysis and Recognition*, Barcelona, Spain, 2009, pp. 501–505.

[5] C. Neudecker, S. Schlarb, Z. Dogan, P. Missier, S. Sufi, A. Williams, and K. Wolstencroft, “An experimental workflow development platform for historical document digitisation and analysis,” in *Proceedings of the Workshop on Historical Document Imaging and Processing,* Singapore, 2011, pp. 161–168.

[6] Z. M. Dogan, C. Neudecker, S. Schlarb, and G. Zechmeister, “Experimental workflow development in digitisation,” in *Proceedings of the International Conference on Qualitative and Quantitative Methods in Libraries,* Chania, Greece, 2010, pp. 377–384.

[7] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher, “A realistic dataset for performance evaluation of document layout analysis,” in *Proceedings of the International*

Conference on Document Analysis and Recognition, Barcelona, Spain, 2009, pp. 296–300.

[8] D. Bainbridge and T. Bell, “Identifying music documents in a collection of images,” in *Proceedings of the International Conference on Music Information Retrieval*, Victoria, BC, 2006, pp. 47–52.

[9] K. MacMillan, M. Droettboom, and I. Fujinaga, “Gamera: A structured document recognition application development environment,” in *Proceedings of the International Conference on Music Information Retrieval*, Bloomington, IN, 2001, pp. 15–16.

[10] L. Pugin, J. Hockman, J. A. Burgoyne, and I. Fujinaga, “Gamera versus Aruspix: Two optical music recognition approaches,” in *Proceedings of the International Conference on Music Information Retrieval*, Philadelphia, PA, 2008, pp. 419–424.

[11] L. Pugin, J. A. Burgoyne, and I. Fujinaga, “Reducing costs for digitising early music with dynamic adaptation,” in *Proceedings of the European Conference on Digital Libraries*, Budapest, Hungary, 2007, pp. 471–474.

[12] S. Charalampos, A. Hankinson, and I. Fujinaga, “Correcting large-scale OMR data with crowdsourcing,” in *Proceedings of the International Workshop on Digital Libraries for Musicology*, London, UK, 2014, pp. 88–90.

[13] H. Goto, “OCRGrid: A platform for distributed and cooperative OCR systems,” in *Proceedings of the International Conference on Pattern Recognition*, Hong Kong, 2006, pp. 982–985.

[14] G. Newby and C. Franks, “Distributed proofreading,” in *Proceedings of the Joint Conference on Digital Libraries*, Houston, TX, 2003, pp. 361–363.

[15] R. Holley, “Extending the scope of Trove: Addition of E-resources subscribed to by Australian libraries.” *D-Lib Magazine*, vol. 7, no. 11/12, 2011.

[16] K. D. Borne and Zooniverse Team, “The Zooniverse: A framework for knowledge discovery from citizen science data,” in *The Proceedings of the American Geophysical Union Fall Meeting*, 2011.

[17] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, “reCAPTCHA: Human-based character recognition via web security measures,” *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008.

[18] L. Von Ahn, “Games with a purpose,” *Computer*, vol. 39, no. 6, pp. 92–94, 2006.

[19] A. Hankinson, P. Roland, and I. Fujinaga, “MEI as a document encoding framework,” in *Proceedings of the International Society for Music Information Retrieval Conference*, Miami, FL, 2011, pp. 293–298.

[20] V. Viro, “Peachnote: Music score search and analysis platform,” in *Proceedings of the International Society for Music Information Retrieval Conference*, Miami, FL, 2011, pp. 359–362.

BROWSING SOUNDSCAPES

Patrice Guyot
SAMoVA team - IRIT
University of Toulouse - France
guyot.patrice@gmail.com

Julien Pinquier
SAMoVA team - IRIT
University of Toulouse - France
julien.pinquier@irit.fr

ABSTRACT

Browsing soundscapes and sound databases generally relies on signal waveform representations, or on more or less informative textual metadata. The TM-chart representation is an efficient alternative designed to preview and compare soundscapes. However, its use is constrained and limited by the need for human annotation. In this paper, we describe a new approach to compute charts from sounds, that we call *SamoCharts*. SamoCharts are inspired by TM-charts, but can be computed without a human annotation. We present two methods for SamoChart computation. The first one is based on a segmentation of the signal from a set of predefined sound events. The second one is based on the confidence score of the detection algorithms. SamoCharts provide a compact and efficient representation of sounds and soundscapes, which can be used in different kinds of applications. We describe two application cases based on field recording corpora.

1. INTRODUCTION

Compact graphical representations of sounds facilitate their characterization. Indeed, images provide instantaneous visual feedback while listening sounds is constrained by their temporal dimension. As a trivial example, record covers allow the user to quickly identify an item in a collection. Such compact representation is an efficient means for sound identification, classification and selection.

In the case of online databases, the choice of a sound file in a corpus can be assimilated to the action of *browsing*. As proposed by Hjørland, “Browsing is a quick examination of the relevance of a number of objects which may or may not lead to a closer examination or acquisition/selection of (some of) these objects” [1].

Numerous websites propose free or charged sound file downloads. These files generally contain sound effects, isolated sound events, or field recordings. Applications are numerous, for instance for music, movie soundtracks, video games and software production. In the context of the CIESS project [1] , our work focuses on an urban sound database used for experimental psychology research.

Most of the times, on-line access to sound files and databases is based on *tags* and textual metadata. These metadata are generally composed of a few words description of the recording, to which may be added the name of its author, a picture of the waveform, and other technical properties. They inform about the sound sources, recording conditions or abstract concepts related to the sound contents (for example “Halloween”).

Natural sonic environments, also called *field recordings* or *soundscapes* [2], are typically composed of multiples sound sources. Such audio files are longer than isolated sound events, usually lasting more than one minute. Therefore, short textual descriptions are very hard to produce, which makes it difficult to browse and select sounds in a corpus.

The analysis and characterization of urban sound events has been reported in different studies. Notably, they can be merged in identified categories [3], which leads to a taxonomical categorization of environmental sounds (see [4] for an exhaustive review). Outdoor recordings are often composed of the same kinds of sound sources, for instance *birds, human voices, vehicles, footstep, alarm*, etc. Therefore, the differences between two urban soundscapes (for example, a park and a street) mostly concern the time of presence and the intensity of such identified sources. As a consequence, browsing field recordings based on the known characteristics of a set of predetermined sound events can be an effective solution for their description.

Music is also made of repeated sound events. In instrumental music, these events can be the notes, chords, clusters, themes and melodies played by the performers. When electroacoustic effects or tape music parts come into play, they can be of a more abstract nature. In the case of *musique concrete*, the notion of “sound object” (which in practice is generally a real sound recording) has its full meaning and

[1] `http://www.irit.fr/recherches/SAMOVA/pageciess.html`

a central position in the music formalization itself [5]. As long as events are identified though, we can assume that the previous soundscape-oriented considerations hold for musical audio files as well.

The *TM-chart* [6] is a tool recently developed to provide compact soundscape representations starting from a set of sound events. This representation constitutes a bridge between physical measures and categorization, including acoustic and semantic information. Nevertheless, the creation of a TM-chart relies on manual annotation, which is a tedious and time-consuming task. Hence, the use of TM-charts in the context of big data sets or for online browsing applications seems unthinkable.

Besides sound visualization, automatic annotation of audio recordings recently made significant progress. The general public has recently witnessed the generalization of speech recognition system. Significant results and efficient tools have also been developed in the fields of Music Information Retrieval (MIR) and Acoustic Event Detection (AED) in environmental sounds [7], which leads us to reckon with sustainable AED in the coming years.

In this paper, we propose a new paradigm for soundscape representation and browsing based on the automatic identification of predefined sounds events. We present a new approach to create compact representations of sounds and soundscapes that we call *SamoCharts*. Inspired by TM-Charts and recent AED techniques, these representations can be efficiently applied for browsing sound databases. In the next section we present a state of the art of online sound representations. The TM-chart tool is then described in Section 3, and Section 4 proposes a quick review of Audio Event Detection algorithms. Then we present in Section 5 the process of SamoCharts creation, and some applications with field recordings in Section 6.

2. SOUND REPRESENTATION

2.1 Temporal Representations

From the acoustic point of view, the simplest and predominant representation of a sound is the temporal waveform, which describes the evolution of sound energy over time. Another widely used tool in sound analysis and representation is the spectrogram, which shows more precisely the evolution of the amplitude of frequencies over time. However, spectrograms remain little used by the general public.

While music notation for instrumental music has focused on the traditional score representation, the contemporary and electro-acoustic music communities have introduced alternative symbolic representation tools for sounds such as the Acousmograph [8], and the use of multimodal information has allowed developing novel user interfaces [9].

All these temporal representations are more or less informative depending on the evolution of the sound upon the considered duration. In particular, in the case of field recordings, they are often barely informative.

2.2 Browsing Sound Databases

On a majority of specialized websites, browsing sounds is based on textual metadata. For instance, freeSFX [2] classifies the sounds by categories and subcategories, such as *public places* and *town/city ambience*. In a given subcategory, each sound is only described with a few words text. Therefore, listening is still required to select a particular recording.

Other websites, such as the Freesound project, [3] add a waveform display to the sound description. In the case of short sound events, this waveform can be very informative. On this website it is colored according to the spectral centroid of the sound, which adds some spectral information to the image. However, this mapping is not precisely described, and remains more aesthetic than useful.

The possibility of browsing sounds with audio thumbnailing has been discussed in [10]. In this study, the authors present a method for searching structural redundancy like the chorus in popular music. However, to our knowledge, this kind of representation has not been used in online systems so far.

More specific user needs have been recently observed through the DIADEMS project [4] in the context of audio archives indexing. Through the online platform Telemeta [5], this project allows ethnomusicologists to visualize specific acoustic information besides waveform and recording metadata, such as audio descriptors and semantic labels. This information aims at supporting the exploration of a corpus as well as the analysis of the recording. This website illustrates how automatic annotation can help to index and organize audio files. Improving its visualization could help to assess the similarity of a set of songs, or to underline the structural form of the singing turns by displaying homogeneous segments.

Nevertheless, texts and waveforms remain the most used and widespread tools on websites. In the next sections, we present novel alternative tools, that have been specially designed for field recording representation.

[2] `http://www.freesfx.co.uk/`
[3] `https://www.freesound.org/`
[4] `http://www.irit.fr/recherches/SAMOVA/DIADEMS/`
[5] `http://telemeta.org/`

3. TM-CHART

3.1 Overview

The Time-component Matrix Chart (abbreviated TM-chart) was introduced by Kozo Hiramatsu and al. [6]. Based on a <Sound Source × Sound level> representation, this chart provides a simple visual illustration of a sonic environment recording, highlighting the temporal and energetic presence of sound sources. Starting from a predetermined set of sound events (e.g. *vehicles*, etc.), and after preliminary annotation of the recording, the TM-chart displays percentages of time of audibility and percentages of time of level ranges for the different sound sources. They constitute effective tools to compare sonic environment (for instance daytime versus nighttime recordings).

3.2 Method

Despite a growing bibliography [11, 12], the processing steps involved in the creation of TM-charts as not been precisely explained. We describe in this part our understanding of these steps and our approach to create a TM-chart.

3.2.1 Estimation of the Predominant Sound

TM-charts rely on a preliminary manual annotation, which estimates the predominant sound source at each time. To perform this task, the signal can be divided in short segments, for example segments of one second. For each segment, the annotator indicates the predominant sound source. This indication is a judgment that relies on both the loudness and the number of occurrences of the sources. An example of annotation can be seen on Figure 1.

Afterwards, each segment label is associated to a category of sound event, which can be for instance one of *car*, *voice*, *birds*, or *miscellaneous*.

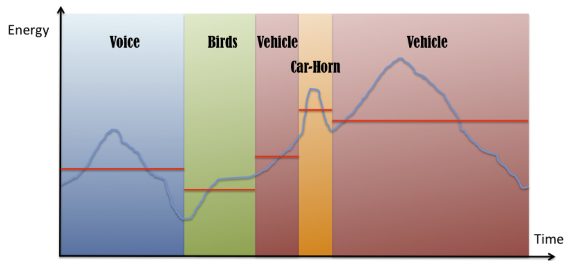


Figure 1. Preliminary annotation of a sound recording for the creation for the creation of a TM-chart.

3.2.2 Computation of the Energy Level

An automatic process is applied to compute the energy of the signal and the mean energy of each segment (respectively in blue and red curves on Figure 1). We assume that the sound pressure level can be calculated from the recording conditions with a calibrated dB meter.

In this process, we can notice that the sound level of a segment is not exactly the sound level of its predominant source. Indeed the sound level of an excerpt depends upon the level of each sound sources, and not only the predominant one. However, we assume that these two measures are fairly correlated.

3.2.3 Creation of the TM-chart

We can now calculate the total duration in the recording (in terms of predominance) and the main sound levels for each category of sound. From this information, a TM-chart can be created.

Figure 2 shows a TM-chart based on the example from Figure 1. It represents, for each category of sound, the percentage of time and energy in the soundscape. The abscissa axis shows the percentage of predominance for each source in the recording. For one source, the ordinate axis shows the duration of its different sound levels. For example, the car-horn is audibly dominant for over 5 % of time. Over this duration, the sound level of this event exceeds 60 dB for over 80 % of time.

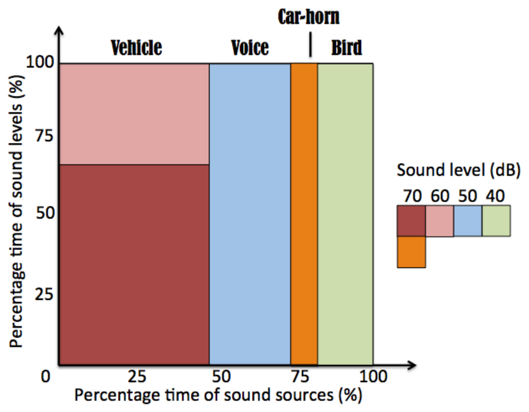


Figure 2. Example of a TM-chart.

3.2.4 Interpretation of the TM-chart

Charts like the one on Figure 2 permit quick interpretations of the nature of the sound events that compose a soundscape. We could infer for instance that the soundscape has been recorded close to a little traffic road, with distant conversations (low energy levels). From such interpretation, one can clearly distinguish and compare sonic environments recorded in different places [6].

The main issue in the TM-chart approach is the need for manual annotation, a time-consuming operation which cannot be applied to big data sets. Therefore, the use of TM-charts seems currently restricted to specific scientific research on soundscapes. In the next sections we will show how recent researches and works on sound analysis can be leveraged to overcome this drawback.

4. AUDIO EVENT DETECTION

Various methods have been proposed for the Audio Event Detection (AED) from continuous audio sequences recorded in real life. These methods can be divided in two categories.

The first category of methods aims at detecting a large set of possible sound events in various contexts. For instance, the detection of 61 types of sound, such as *bus door, footsteps* or *applause*, has been reported in [7]. In this work the author modeled each sound class by a Hidden Markov Model (HMM) with 3 states, and Mel-Frequency Cepstral Coefficients (MFCC) features. Evaluation campaigns, such as CLEAR [13] or AASP [14], propose the evaluation of various detection methods on a large set of audio recordings from real life.

The second category of methods aims at detecting fewer specific types of sound events. This approach privileges accuracy over the number of sounds that can be detected. It generally relies on a specific modeling of the "target sounds" to detect, based on acoustic observations. For example, some studies propose to detect gunshots [15] or water sounds [16], or the presence of speech [17].

These different methods output a segmentation of the signal informed by predetermined sound events. They can also provide further information that may be useful for the representation, particularly in the cases where they are not reliable enough. Indeed, the detection algorithms are generally based on a confidence score, that allows to tune the decisions. For instance, Hidden Markov Model, Gaussian mixture models (GMM) or Support Vector Machine (SVM), all rely on confidence or "likelihood" values. Since temporal confidence values can be computed by each method of detection, it is possible to output at each time the probability that a given sound event is present in the audio signal.

Based on these observations, we propose a new tool for soundscape visualization, the *SamoChart*, which can rely either on automatic sound event segmentation, or on confidence scores by sound events.

5. SAMOCHART

The SamoChart provides a visualization sound recordings close to that of a TM-chart. At the difference of a TM-chart, it can be computed automatically from a segmentation or from temporal confidence values.

In comparison with TM-charts, the use of the automatic method overcomes a costly human annotation and avoids subjective decision-making.

5.1 Samochart based on Event Segmentation

5.1.1 Audio Event Segmentation

SamoCharts can be created from Audio Event Detection annotations. This automatic annotation is an independent process that can be performed following different approaches, as mentioned in Section 4. We will suppose in the next part that an automatic annotation has been computed from a set of potential sound events ("targets"). For each target sound event, this annotation provides time markers related to the presence or absence of this sound in the overall recording. In addition to the initial set of target sounds, we add a sound *unknown* that corresponds to the segments that have not been labeled by the algorithms.

5.1.2 Energy Computation

As in the TM-chart creation process, we compute the energy of the signal. However, if the recording conditions of the audio signal are unknown, we cannot retrieve the sound pressure level. In this case, we use the RMS energy of each segment, following the equation:

$$RMS(w) = 20 \times log_{10} \sqrt{\sum_{i=0}^{N} w^2(i)} \qquad (1)$$

where w is an audio segment of N samples, and $w(i)$ the value of the i^{th} sample.

5.1.3 SamoChart Creation

From the information of duration and energy, we are able to create a SamoChart. Figure 3 shows an example of a SamoChart based on event segmentation considering two possible sound events.

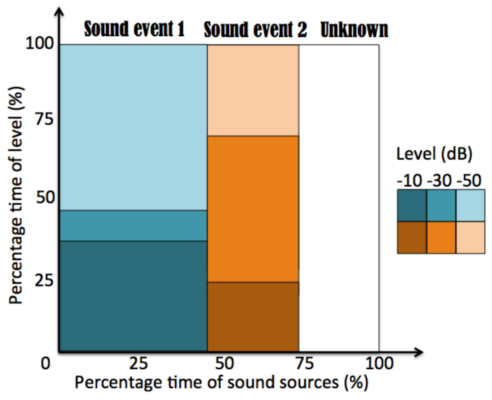


Figure 3. SamoChart based on event segmentation.

Unlike TM-charts, we can notice from this method that the total percentage of sound sources can be higher than 100% if the sources overlap.

5.2 Samochart based on Confidence Values

Most Audio Event Detection algorithms actually provide more information than the output segmentation. In the following approach, we propose to compute SamoCharts from the confidence scores of these algorithms.

We use for each target sound the temporal confidence values outputted by the method, which can be considered as probabilities of presence (between 0 and 1). The curve on Figure 4 shows the evolution of the confidence for the presence of a given sound event during the analyzed recording. We use a threshold on this curve, to decide if the sound event is considered detected or not. This threshold is fixed depending on the detection method and on the target sound. To obtain different confidence measures, we divide the upper threshold portion in different parts.

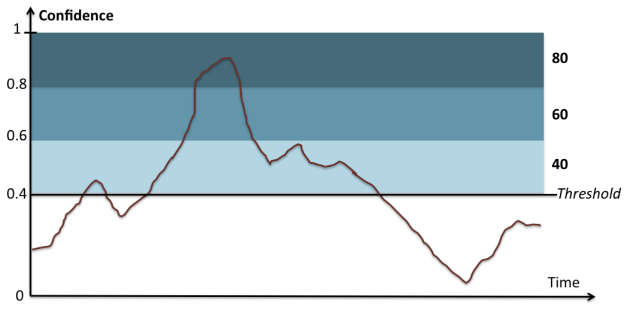


Figure 4. Confidence measures for a sound event.

With this approach, we infer the probability of presence for each sound event according to a confidence score. Figure 5 shows the SamoChart associated to a unique sound event. In this new chart, the sound level is replaced by the confidence score.

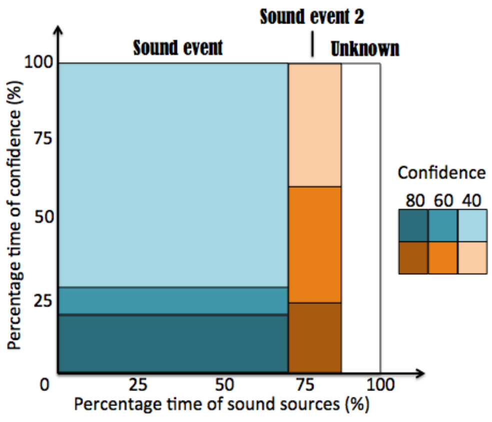


Figure 5. SamoChart based on confidence value.

5.3 Implementation

We made a JavaScript implementation to create and display SamoCharts, which performs a fast and "on the fly" computation of the SamoChart. The code is downloadable from the SAMoVA web site[6]. It uses an object-oriented paradigm to facilitate future development.

In order to facilitate browsing applications, we also chose to modify the size of the chart according to the duration of the corresponding sound excerpt. We use the equation 2 to calculate the height h of the Samochart from a duration d in seconds.

$$h = \begin{cases} 1 & \text{if } d < 1 \\ 2 & \text{if } 1 \leq d < 10 \\ 2 \times \log_{10}(d) & \text{if } d \geq 10 \end{cases} \qquad (2)$$

We also implemented a *magnifying glass* function that provides a global view on the corpus with the possibility of zooming in into a set of SamoCharts. Furthermore, the user can hear each audio file by clinking on the plotted charts.

6. APPLICATIONS

6.1 Comparison of soundscapes (CIESS project)

Through the CIESS project, we have recorded several urban soundscapes at different places and times. The sound events of these recordings are globally the same, for instance *vehicle* and *footstep*. However, their numbers of occurrences are very different according to the time and place of recording. As an application case, we computed several representations of two soundscapes. Figure 6 shows the colored waveforms of these extracts as they could have been displayed on the Freesound website.

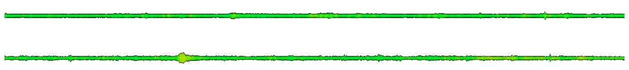


Figure 6. Colored waveforms of two soundscapes.

As we can see, these waveforms do not show great differences between the two recordings.

We used AED algorithms to detect motor vehicle, footstep and car-horn sounds on these two example recordings [18]. Then, we computed SamoCharts based on the confidence score of these algorithms (see Figure 7).

The SamoCharts of Figure 7 are obviously different. They provide a semantic interpretation of the soundscapes, which reveals important dissimilarities. For instance, the vehicles are much more present in the first recording than in the second one. Indeed, the first recording was recorded on an important street, while the second one was recorded on a pedestrian street.

[6] http://www.irit.fr/recherches/SAMOVA/pageciess.html

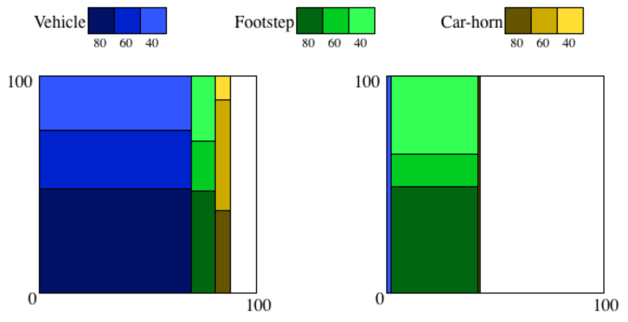


Figure 7. SamoCharts of the two recordings of Figure 6, based on confidence values.

6.2 Browsing a corpus from the UrbanSound project

If the differences between two soundscapes can easily be seen by comparing two charts, the main interest of the SamoChart is their computation on bigger sound databases.

UrbanSound dataset[7] has been created specifically for soundscapes research. It provides a corpus of sounds that are labeled with the start and end times of sound events of ten classes: *air conditioner, car horn, children playing, dog bark, drilling, enginge idling, gun shot, jackhammer, siren* and *street music*. The SamoCharts created from these annotations allow to figure out the sources of each file, as well as their duration and their sound level. They give an overview of this corpus. Figure 8 shows the SamoCharts of nine files which all contain the source *car horn*. The duration of these files range form 0.75 to 144 seconds.

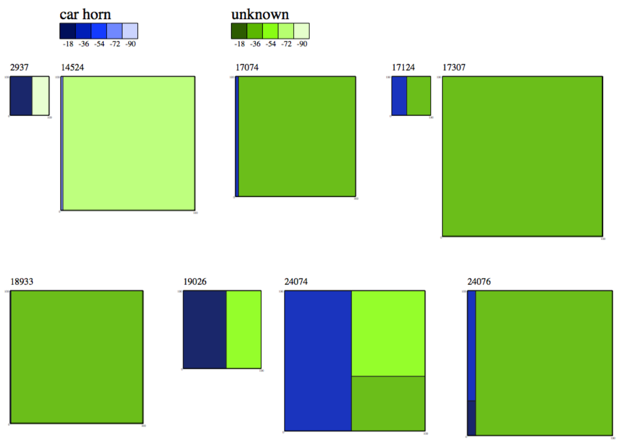


Figure 8. Browsing recordings of the UrbanSound corpus.

6.3 SoundMaps

Other applications can be found from the iconic chart of a soundscape. Soundmaps, for example, are digital geographical maps that put emphasis on the soundscape of every specific location. Various projects of sound maps have been proposed in the last decade (see [19] for a review). Their goals are various, from giving people a new way to look at the world around, to preserving the soundscape of specific places. However, as in general with sound databases, the way sounds are displayed on the map is usually not informative. The use of SamoCharts on soundmaps can facilitate browsing and make the map more instructive.

6.4 Music Representations

If the process we described to make charts from sounds was originally set up to display soundscapes, it could certainly be extended to other contexts. Indeed, Samocharts give an instantaneous feedback on the material that compose the sonic environment. Handled with the appropriate sound categories, they could provide a new approach to overview and analyze a set of musical pieces composed with the same material.

For example, Samocharts could be used on a set of concrete music pieces. The charts could reveal the global utilization of defined categories of sounds (such as *bell* or *birds songs*). In the context of instrumental music analysis, they could reflect the utilization of the different families of instrument (e.g. *brass*, etc.), representing the duration and musical nuances. Applied on a set of musical pieces or extracts, they could emphasize orchestration characteristics.

Figure 9 shows an analysis of the first melody (Theme A) of Ravel's Boléro, which is repeated nine times with different orchestrations. The SamoCharts on the figure display orchestration differences, as well as the rising of a crescendo. The main chart (Theme A-whole) shows how each family of instrument is used during the whole extract.

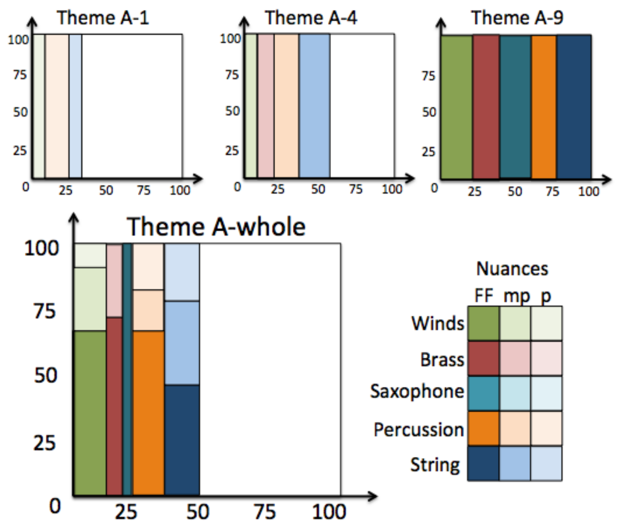


Figure 9. Analysis of the first melody of Ravel's Bolérol (repetitions number 1, 4 and 9, and global analysis). The horizontal axis corresponds to the percentage of time where a family of instrument is present. This percentage is divided by the number of instruments: the total reaches 100% only if all instruments play all the time. The vertical axis displays the percentage of time an instrument is played in the different nuances.

[7] `https://serv.cusp.nyu.edu/projects/urbansounddataset/`

7. CONCLUSION AND FUTURE WORKS

In this paper, we presented a new approach to create charts for sound visualization. This representation, that we name SamoChart, is based on the TM-chart representation. Unlike TM-charts, the computation of SamoCharts does not rely on human annotation. SamoCharts can be created from Audio Event Detection algorithms and computed on big sound databases.

A first kind of SamoChart simply uses the automatic segmentation of the signal from a set of predefined sound sources. To prevent eventual inaccuracies in the segmentation, we proposed a second approach based on the confidence scores of the previous methods.

We tested the SamoCharts with two different sound databases. In comparison with other representations, SamoCharts provide great facility of browsing. On the one hand, they constitute a precise comparison tool for soundscapes. On the other hand, they allow to figure out what kinds of soundscapes compose a corpus.

We also assume that the wide availability of SamoCharts would make them even more efficient for accustomed users. In this regard, we could define a fixed set of color which would correspond to each target sound.

The concepts behind TM-charts and Samocharts can finally be generalized to other kind of sonic environments, for example with music analysis and browsing.

Acknowledgments

This work is supported by a grant from Agence Nationale de la Recherche with reference ANR-12-CORP-0013, within the CIESS project.

8. REFERENCES

[1] B. Hjørland, "The importance of theories of knowledge: Browsing as an example," *Journal of the American Society for Information Science and Technology*, vol. 62, no. 3, pp. 594–603, 2011.

[2] R. Schafer and R. Murray, *The tuning of the world*. Knopf New York, 1977.

[3] C. Guastavino, "Categorization of environmental sounds." *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, vol. 61, no. 1, p. 54, 2007.

[4] S. Payne, W. Davies, and M. Adams, "Research into the practical and policy applications of soundscape concepts and techniques in urban areas," *University of Salford*, 2009.

[5] P. Schaeffer, *Traité des objets musicaux*. Paris: Editions du Seuil, 1966.

[6] K. Hiramatsu, T. Matsui, S. Furukawa, and I. Uchiyama, "The physcial expression of soundscape: An investigation by means of time component matrix chart," in *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, vol. 2008, no. 5. Institute of Noise Control Engineering, 2008, pp. 4231–4236.

[7] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, "Acoustic event detection in real life recordings," in *Proceedings of the 18th European Signal Processing Conference*, 2010, pp. 1267–1271.

[8] Y. Geslin and A. Lefevre, "Sound and musical representation: the acousmographe software," in *Proceedings of the International Computer Music Conference*, 2004.

[9] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen, "Multimodal presentation and browsing of music," in *Proceedings of the 10th international conference on Multimodal interfaces*. ACM, 2008, pp. 205–208.

[10] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *Multimedia, IEEE Transactions on*, vol. 7, no. 1, pp. 96–104, 2005.

[11] T. Matsui, S. Furukawa, T. Takashima, I. Uchiyama, and K. Hiramatsu, "Timecomponent matrix chart as a tool for designing sonic environment having a diversity of sound sources," in *Proceedings of Euronoise*, 2009, pp. 3658–3667.

[12] E. Bild, M. Coler, and H. Wörtche, "Habitats assen pilot: Testing methods for exploring the correlation between sound, morphology and behavior," in *Proceedings of Measuring Behavior*, M. W.-F. A.J. Spink, L.W.S. Loijens and L. Noldus, Eds., 2014.

[13] R. Stiefelhagen, K. Bernardin, R. Bowers, J. Garofolo, D. Mostefa, and P. Soundararajan, "The CLEAR 2006 evaluation," in *Multimodal Technologies for Perception of Humans*. Springer, 2007, pp. 1–44.

[14] D. Giannoulis, E. Benetos, D. Stowell, M. Rossignol, M. Lagrange, and M. Plumbley, "Detection and classification of acoustic scenes and events," *An IEEE AASP Challenge*, 2013.

[15] C. Clavel, T. Ehrette, and G. Richard, "Events detection for an audio-based surveillance system," in *Proceedings of the International Conference on Multimedia and Expo, ICME*. IEEE, 2005, pp. 1306–1309.

[16] P. Guyot, J. Pinquier, and R. André-Obrecht, "Water sound recognition based on physical models," in *Proceedings of the 38th International Conference on Acoustics, Speech, and Signal Processing, ICASSP*. IEEE, 2013.

[17] J. Pinquier, J.-L. Rouas, and R. André-Obrect, "Robust speech/music classification in audio documents," *Entropy*, vol. 1, no. 2, p. 3, 2002.

[18] P. Guyot and J. Pinquier, "Soundscape visualization: a new approach based on automatic annotation and samocharts," in *Proceedings of the 10th European Congress and Exposition on Noise Control Engineering, EURONOISE*, 2015.

[19] J. Waldock, "Soundmapping. critiques and reflections on this new publicly engaging medium," *Journal of Sonic Studies*, vol. 1, no. 1, 2011.

GRAPHIC TO SYMBOLIC REPRESENTATIONS OF MUSICAL NOTATION

Craig Stuart Sapp
CCARH/Stanford University
craig@ccrma.stanford.edu

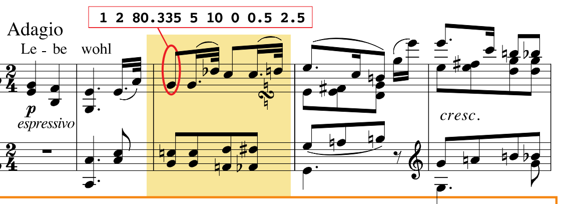
ABSTRACT

This paper discusses the SCORE data format, a graphically oriented music representation developed in the early 1970's, and how such a representation can be converted into sequential descriptions of music notation. The graphical representation system for the SCORE editor is presented along with case studies for parsing and converting the data into other symbolic music formats such as Dox, Humdrum, MusicXML, MuseData, MEI, and MIDI using *scorelib*, an open-source code library for parsing SCORE data. Knowledge and understanding of the SCORE format is also useful for OMR (Optical Music Recognition) projects, as it can be used as an intermediate layer between raw image scans and higher-level digital music representation systems.

1. INTRODUCTION

The SCORE notation editor is the oldest music-typesetting program in continual use. It was created at Stanford University in the early 1970's by Leland Smith and initially was developed on mainframe computers with output to pen plotters that was then photo-reduced for publication. In the 1980's SCORE was ported to IBM PCs running MS-DOS with output to Adobe PostScript, and later ported to Microsoft Windows. Due to the program's long-term stability and excellent graphical output, many critical editions have been created over the years using SCORE, such as the complete works of Boulez, Verdi, Wagner, C.P.E. Bach, Josquin and Dufay.

Throughout its history the SCORE editor has used a simple and compact data format that allows forwards and backwards compatibility between different versions of the SCORE editor. The music representation system is symbolic, but highly graphical in nature. Each notational element is represented by a list of numbers that derive their meanings based on their positions in the list. This format was adapted from the one used in Music V software for computer-generated sound developed by Max Mathews in the late 1950's at Bell Labs. In both cases, the list of numbers serves as a set of parameters describing an object—either to generate a sound in Music V or to place a graphical element on the page in SCORE. This organization of the data is also parsimonious, due largely to memory limitations of computers on which these systems were developed.



```
14 1 77.022  2
1  2 80.335  5 10 0 0.5 2.5
6  2 80.335  7.5 8.88 95.35 11 0 0 11 88.88 95.35 21 0 95.35
1  1 80.335  5 10 0 0.5 1.5
1  1 80.335  8 3 0 0.5
6  1 80.335  6.5 7.5 107.87 11
1  1 88.169  8 0 0 0.5
1  1 88.169  5 10 0 0.5 1.78
1  2 88.169  5 10 0 0.375 3.31 10.95 1
5  2 90.169 15.88 16.5 96.25 1.0133 -1 0 0.55 0 0 0.8
1  2 95.353  9 11 0 0.125 -0.13
1  1 99.301  4 13 0 0.5 3.19
1  1 99.301  9 0 0 0.5
6  2 99.301  9 10 115.39 11 0 0 11 107.87 115.39 21 0 115.39
1  2 99.301  8 10 0 0.5 1
1  1 107.87  4 11.18 0 0.5 3.5
1  1 107.87  9 2 0 0.5
1  2 107.87  8 10 0 0.375 1.53 10
5  2 109.16 17.38 17.88 116.18 1.0633 -1
9  2 111.86  2.88 75 0.65
9  2 113.77 -0.25 8 0.18
9  2 113.89  4.75 8 0.2
1  2 115.39  9 13 0 0.125 1
```

Figure 1. SCORE data for bar 3 of Beethoven *Op. 81a*.

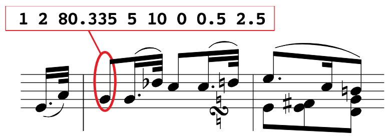
Figure 1 illustrates music typeset in the SCORE editor along with data describing the third measure. Each line of numbers represents a particular graphical element, such as the circled first note of the third measure measure that is represented on the second line in the data excerpt.

The first four numbers on each line have a consistent meaning across all notational items:

P1: Item type (note, rest, clef, barline, etc.).
P2: Item staff number on the page.
P3: Item horizontal position on the page.
P4: Item vertical position on the staff.

Parameter one (P1) indicates the element type—in this example 1=note, 5=slur, 6=beam, and 14=barline. The second number is the staff onto which the element is placed, with P2=1 for the bottom staff and P2=2 for the next higher staff on the page. The third parameter is the horizontal position of the item on the page, typically a number from 0.0 representing the page's left margin, to 200.0 for the right margin. In Figure 1, items are sorted by horizontal position (P3) from left to right on the page; however, SCORE items may occur with any ordering, which typically indicates drawing sequence (z-order) when printing the items. P4 indicates the diatonic vertical position on a staff, with positions 3, 5, 7, 9, and 11 being the lines of a five-lined staff from bottom to top.

These first four numbers on a line give each item an explicit location on the page. The horizontal position is an absolute value dependent on the printing area, while the vertical axis is a hierarchical system based on the staff to which an item belongs: an item's vertical position is an offset from the staff's position on the page, and the staff may have an additional offset from its default position on the page.[1]



P1:	1	= Item type (note).
P2:	2	= Staff no. (second staff).
P3:	80.335	= Horizontal position.
P4:	5	= Vertical position (2nd line from staff bottom).
P5:	10	= Up stem, no accidentals.
P6:	0	= Solid black note.
P7:	0.5	= Duration (eighth-note).
P8:	2.5	= Stem length (2.5 diatonic steps higher than an octave).
P9:	[0]	= No flags on stem; no augmentation dots.

Figure 2. Parameter values and meanings for a note.

The meaning of parameters greater than P4 depends on the type of graphical element being described. Objects with left and right endpoints (beams, slurs, lines) will use P5 as the right vertical position and P6 as the right horizontal position. Figure 2 illustrates some of the higher parameter positions for a note. In this example, P5 describes the stem and accidental display type for the note, with "10" in this case meaning the note has a stem pointing upwards and that there are no accidentals displayed in front of the note. P6 describes the notehead shape, with 0 meaning the default shape of a solid black notehead. P7 indicates the musical duration of the note in terms of quarter notes, such as 0.5 representing an eighth-note. P8 indicates the length of the stem with respect to the default height of an octave. All other unspecified parameters after the last number in the list are implied to be zero. This means either a literal 0, or it may mean to use the default value for that parameter. For this example the implied 0 of P9 indicates that the note has no flags on the stem, nor are there any augmentation dots following the notehead.

Multiple attributes may be packed into a single parameter value, such as P5 and P9 in the above example. This parameter compression was due to memory limitations in computers during the 1970's and 1980's. All values in SCORE data files use 4-byte floating-point numbers. When a parameter can be represented by ten or fewer states, they are typically stored as a decimal digit within these numbers. For example stem directions of notes are given in the 10's digit of P5, while the accidental type is given in the 1's digit. In addition, the 100's digit of P5 indicates whether parentheses are to be placed around the accidental, and the fractional portion of P5 indicates a horizontal offset for the accidental in front of the note. The Windows version of the SCORE editor retains this attribute packing system, primarily for backwards compatibility with the MS-DOS version of the program, since many professional users of SCORE still use the MS-DOS version of the program. This minimal data footprint could also be taken advantage of in low memory situations such as mobile devices or over slow network connections.

SCORE parameters have an interpreted meaning based on the item type and parameter number. With the advent of greater and cheaper memory in computers, the general trend as seen in XML data formats is to provide a key description along with the parameter data. Note that this is a trivial difference between data formats in terms of functionality, but is more convenient for readability and error checking. Below is a hypothetical translation of the SCORE note element discussed in Figure 2 that has been converted into an XML-style element, providing explicit key/value pairs for parameters rather than the fixed-position compressed parameter sequence :

```
<note>
        <staff>2</staff>
        <hpos>80.335</hpos>
        <vpos>5</vpos>
        <stem>up</stem>
        <accidental>none</accidental>
        <shape>solid </shape>
        <duration>0.5</duration>
        <stem-length>2.5</stem-length>
        <flags>0</flags>
        <aug-dots>0</aug-dots>
</note>
```

[1] For a detailed description of the layout axes, see pp. 7–10 of `http://scorelib.sapp.org/doc/coordinates/StaffPositions.pdf`

A translation of these note parameters into MusicXML syntax might look like this:

```
<note default-x="13">
        <pitch>
                        <step>G</step>
                        <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <voice>1</voice>
        <type>eighth</type>
        <stem default-y="25">up</stem>
        <beam number="1">begin</beam>
</note>
```

The primary difference is that SCORE data does not encode explicit pitch information. The pitch "G4" can be inferred from the context of the current clef and key signature as well as any preceding accidentals on G4's in the measure. Extracting pitch information from SCORE data requires non-trivial but straightforward parsing of the data (excluding slur/tie analysis). A second important structural difference is encoding of beams. In SCORE beams are independent notational items, and linking of notes to beams is inferred within the editor by their spatial proximity and orientation.

MusicXML 3.0 includes a relatively complete layout description, which is more hierarchical than SCORE's layout description. For example the attribute default-x="13" of the <note> element describes the distance from the left barline of the measure to the notehead, while in SCORE the P3=80.335 describes the distance from the left margin to the notehead. The stem length is indicated in SCORE and MusicXML in an equivalent fashion, with SCORE setting P8=2.5, which means that the stems should be 2.5 diatonic steps longer than an octave, while MusicXML indicates the same information with the default-y attribute on the <stem> element. Staff assignment in MusicXML is inferred from the part to which the note belongs, while SCORE encodes an explicit staff assignment.

SCORE data is not purely a graphical description of music notation as demonstrated in the above conversion example into MusicXML. It also contains some symbolic information necessary for manipulating graphical items in musically intelligent ways. Within the SCORE editor the musical data can be played, transposed, moved between systems, reformatted, and processed in other musically intelligent ways.

For notes and rests, P7 indicates the duration of the item. This means that there are two horizontal axes present in the data: a spatial axis quantified in P3, and a temporal axis in P7 that describes time in quarter-note units. Figure 3 illustrates these two spatial/time axes present in SCORE data. The SCORE editor can manipulate the data based on either of these descriptions of the music. For example, data entry on each staff can be done independently, in which case the notes on each staff are not aligned vertically. The SCORE program's LJ command aligns the notes across system staves based on the P7 durations, and this will cause the P3 values of notes to match their rhythmic partners on other staves.

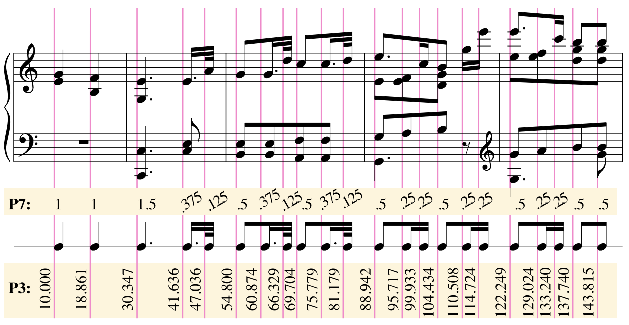


Figure 3. Duration and horizontal position information.

In Figure 3, the vertical lines (in red) are located at the P3 positions of notes in both both staves. In the cases of chords containing intervals of a second, the notes offset to the opposite side of the stem have the same P3 horizontal position of the other notes in the chord, but have a non-zero horizontal offset value (P10). Thus all notes sounding at the same time on a staff must all have the same P3 horizontal position; otherwise, the SCORE editor will misinterpret the notes in a chord as a melodic sequence. Notes on the offbeat of the first beat in measure three have been given an intentional P10 offset from the default spacing, so they do not visually align with the red guide line although their P3 values match the position of the line.

The P7 duration values of notes and rests can be used to calculate the composite rhythm of polyphonic music as illustrated by the rhythm on the single-lined staff below the main musical excerpt in Figure 3. Calculating this rhythmic pattern is necessary for horizontal spatial layout in music notation. In SCORE, horizontal music spacing is calculated on a logarithmic scale, using a spacing factor of approximately the Golden ratio for every power-of-two rhythmic level.

2. SIMILARITY TO OMR PROCESSING

Extracting symbolic musical data in optical music recognition (OMR) can be divided into two basic steps: (1) recognizing graphical elements in a scan, and (2) interpreting their functions and interrelations. In practice there is feedback between these two steps for interpreting the meanings of the elements: if a graphical symbol is ambiguous or incorrect, the context of other symbols around it may clarify the meaning of that item. For musicians this interaction mostly occurs at a subconscious level that can often be difficult to describe within a computer program in order to generate a correct interpretation of the notation. As an example of the inter-dependency of these two steps, the OMR program SharpEye[2] is quite sensitive to visual breaks in note stems. Finding stemless noteheads often leads it to identifying the noteheads as double whole rests which roughly have the same shape as a stemless black notehead. This is clearly a nonsensical interpretation when occurring in meters such as 4/4 or against notes on other staves that do not have the same duration as a double whole note. In such cases where interpretation stage yields such strange results, the identification stage of a graphical element should be reconsidered.

SCORE's data format can be considered a perfect representation of the first stage in OMR processing where all graphical elements have been correctly identified. Converting between a basic OMR representation of graphical elements and SCORE data is relatively easy. For example Christian Fremerey of the University of Bonn/ViFaMusik was able to write a Java program, called *mro2score*, within a few days that converts the SharpEye's graphical representation format into SCORE data.[3]

The *mro2score* program essentially transcodes the identification-stage of musical data from OMR identification and adds minimal markup to convert into SCORE data. In order to convert such symbols into musically meaning syntaxes, more work is necessary. Most OMR programs have built-in editors used to assist the correction of graphic symbol identification as well as their final interpretation. Such editors function in a manner similar to the SCORE editor, which can display graphical elements containing syntactic errors such as missing notes, or incorrect rhythms. Most graphical notation editors such as MuseScore, Sibelius or Finale require syntactically correct data, so they are not as well suited to interactive correction of OMR data.

In order to convert from SCORE data into more symbolic music formats, an open-source parsing library and related programs called *scorelib* has been developed by the author.[4] This library provides automatic analysis of the relations between notational elements in the data, linking music across pages, grouping music into systems and parts, linking notes to slurs and beams, as well as interpreting the pitches of notes. This library is designed to handle the second stage in OMR conversions of scanned music into symbolically manipulable musical data. Conversion from SCORE, and by extension low-level OMR recognition data, into other more symbolic data formats becomes much simpler once these relationships between graphical items have been analyzed using *scorelib*. Currently the *scorelib* codebase can convert SCORE data into MIDI, Humdrum, Dox, MuseData, MusicXML and MEI data formats.[5]

The following sub-sections describe the basic order of analyzing SCORE data in order to extract higher-level musical information needed for conversion into other musical data formats.

2.1 Staves to Systems

SCORE data does not include any explicit grouping of staves into musical systems (a set of staves representing different parts playing simultaneously). So when extracting symbolic information from SCORE data, the first step is to group staves on a page into systems. Errors are unlikely to occur in this grouping process, since staves linked together by barlines are the standard graphical representation for systems. In orchestral scores, parts may temporarily drop out on systems where they do not have notes. In SCORE data, staves are give a part number so that printed parts can be generated from such scores by inserting additional rests for systems on which the part is not present.

2.2 Systems to Movement

Once musical systems have been identified on a page in SCORE (or with any raw OMR graphical elements), the identification of the sequence of systems across multiple pages forming a full movement is necessary in order to interpret items such as slurs and ties. These may be broken graphically by system line breaks. If a set of pages describes a single work, this process is generally as trivial as the staves to systems identification; however, automatic identification of new movements/works will be dependent on the graphical style of the music layout. Typically indenting the first system indicates a new movement/work, but this assumption is not always true. When interpreting SCORE or OMR data, manual intervention may sometimes be needed to handle non-standard or unanticipated cases in movement segmentation.

2.3 Pitch Identification

Pitch identification takes extensive processing of the data. The previous two steps linking staves into systems and systems across pages into movements must first be done before identifying pitch. The data must then be read temporally system by system throughout the movement, keeping track of the current key and resetting the spelling of pitches at each barline for each part/staff. Figure 4 illustrates the result of automatic identification of the

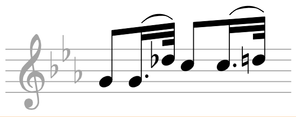
[2] `http://www.visiv.co.uk`

[3] `http://www.ccarh.org/courses/253/lab/mro2score`

[4] `http://scorelib.sapp.org`

[5] See `http://scorelib.sapp.org/program` for a list of available conversion and processing programs.

pitch sequence (g, g, d-flat, c, c, d-natural) for the top staff of music in measure three of Figure 1.



```
6 2 80.335 7.5 8.88 95.35 11 0 0 11 89.17 95.35 21 0 95.35
1 2 80.335 5 10 0 0.5 2.5
@auto@base40Pitch:    185    (g)
5 2 90.169 15.88 16.5 96.25 1.0133 -1 0 0.55 0 0 0.8
1 2 88.169 5 10 0 0.375 3.31 10.95 1
@auto@base40Pitch:    185    (g)
1 2 95.353 9 11 0 0.125 -0.13
@auto@base40Pitch:    207    (dd-)
6 2 99.301 9 10 115.39 11 0 0 11 107.87 115.39 21 0 115.39
1 2 99.301 8 10 0 0.5 1
@auto@base40Pitch:    202    (cc)
1 2 107.866 8 10 0 0.375 1.53 10
@auto@base40Pitch:    202    (cc)
5 2 109.16 17.38 17.88 116.18 1.0633 -1
1 2 115.386 9 13 0 0.125 1
@auto@base40Pitch:    208    (dd)
```

Figure 4 : Automatic pitch labeling of SCORE data.

The *scorelib* library extends the basic SCORE data format to include a list of key/value pairs following the initial line of parameters for a graphical item. In Figure 3, the lines starting "@auto@base40Pitch" are examples of this additional key/value parameter system. In this case the namespace "auto" indicates automatic identification for the pitch of the note. This can be overridden by a manual setting for the pitch with the "@base40Pitch" key.

2.4 Beam grouping

Grouping notes connected to a common beam is a step that can be done either before or after pitch identification, since these two components of notation are independent. In SCORE data this can be done deterministically with little error. Since SCORE data is not organized into measures like many symbolic music data formats, beams crossing barlines are not a difficulty in SCORE, although expressing such barline-crossing beams in translated formats can be difficult.

2.5 Layer Identification

After beaming identification, the most appropriate analysis is to interpret the number of independent monophonic rhythmic streams of notes/rests in each measure. For music with one or two rhythmic streams on a staff, the assignment is relatively straightforward. Three or more rhythmic layers in the music can be difficult to automatically interpret. Graphical music editors typically have four independent layers that can be overlaid on a single staff. SCORE does not have a formalized system for keeping track of rhythmic layers (although there is an informal system in the Windows version of the SCORE editor), so occasionally manual intervention is necessary to assign music to different layers. Figure 5 illustrates the layer interpretation of the music from Figure 1. Since there are no more than two layers on any staff, automatic recognition of the layers is unambiguous. The first layer (as defined in most graphical music editors) is the highest pitched music in the measure with stems pointing upwards if there is a second layer below it. In Figure 5, the second layers in measures 5 and 6 are highlighted in red (or gray in black-and-white prints). The circled rest on the bottom staff of measure 4 presents an interpretational ambiguity: either the bottom layer can be considered to drop out at the rest, or the rest can be interpreted as shared between the two layers on the bottom staff. When extracting orchestral parts in such situations, both parts would share the rest, and the extracted parts would both display the rest.

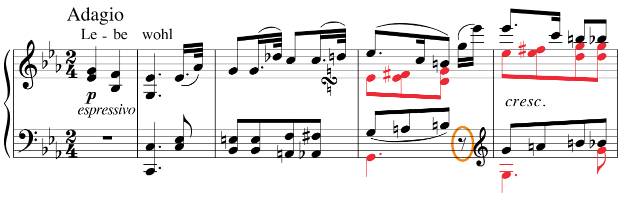


Figure 5. Automatic layer identification, 2nd layer in red.

2.6 Slur/Tie differentiation

After layers have been identified, the final complex step is to distinguish between slurs and ties. For monophonic parts this is straightforward, but in polyphonic parts there are many corner cases to deal with, making 100% correct distinctions difficult to achieve. SCORE has a weak implicit labeling system to differentiate between ties and slurs, but this cannot be depended upon on since the system is primarily intended for graphical offsets of slurs rather than differentiation between slurs and ties. After ties have been identified, pitch identifications need to be reconsidered since tied notes without accidentals will take their accidental from the starting note of a tied group.

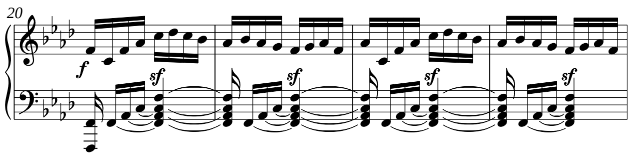


Figure 6. Disjunct ties in Beethoven op. 57, Presto, mm 20-24.

Additionally difficulties arise in both identifying and representing ties that do not connect rhythmically adjacent notes. In particular notated arpeggios such as shown in Figure 6 bypass notating intermediate notes in a slur group, and instead have a single tie connecting the first and last notes in the tie group. Music editors such as MuseScore/Sibelius/Finale cannot handle such cases, and it is also difficult to automatically identify such cases in OMR or SCORE data.

3. DATA CONVERSION FROM SCORE

SCORE uses a two-dimensional description of musical notation, and its data can be serialized into any order since items' positions on the page are independent of each other. Nearly all other music-notation formats impose a sequential structure onto their data, typically chopping up the score into parts, measures, and then layers, which form monophonic chunks that are serialized in different ways. This section presents some of the conversions available with sample programs accompanying the *scorelib* library.

Figure 7 illustrcates three serialization methods within measures that are commonly found in music-notation data formats. In Humdrum data, notes are always serialized by note-attacks times—in other words, all notes from each part/layer played at the same time are found adjacent to each other in the data. This configuration is also true of Standard MIDI Files in type-0 arrangement, where all notes are presented in strict note-attack order. Most other data formats will organize music into horizontal/monophonic sequences by measure rather than by vertical/harmonic slices. MEI chops up a score into a sequence of measures/parts/staves, and finally the staves are segmented into a parallel sequence of monophonic layers. MuseData and MusicXML use the same serialization technique within a measure, but layer segmentations are not as hierarchical as MEI. MusicXML has two ways of serializing measures in a score (*partwise* and *timewise*), but these methods do not affect serialization within a measure.

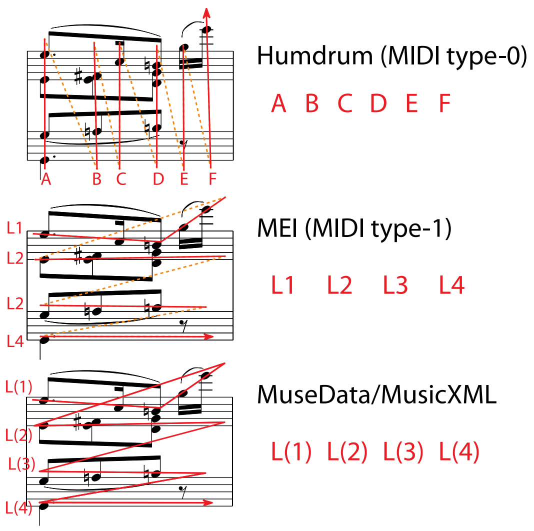


Figure 7. Measure-level serialization schemes in sequential data formats.

In addition to serialization, an important distinction between data formats is the presence or lack of layout information. SCORE data always contains explicit and complete layout information for displaying musical notation, while other data formats have a range of layout description capabilities. The complexity of the notation will determine the necessity of preserving layout information when translating to other file formats. Simple music can automatically be re-typeset without problems; however, complex music is difficult to automatically typeset with a suitable readability quality, and usually human intervention is required to maximize readability in complex notational situations. Many music-notation editing programs focus on ease of manipulation for the musical layout and try to minimize the need for manual control. Likewise, they internally hide the layout information that would be necessary to convert into layout explicit representations such as SCORE data.

Automatic layout will always fail at some point, since the purpose of music notation is to convey performance data to a musician in the most efficient means necessary. Typesetting involves lots of rules and standards, but frequently the rules will need to be broken, or conflicting rules will override each other. Any confusion in the layout decreases the effectiveness of the notation, which a professional typesetter can deal with on a cognitive level much higher than a computer program. Being able to preserve the precise musical layout of SCORE (or OMR) data is very useful, since this can retain human-based layout decisions.

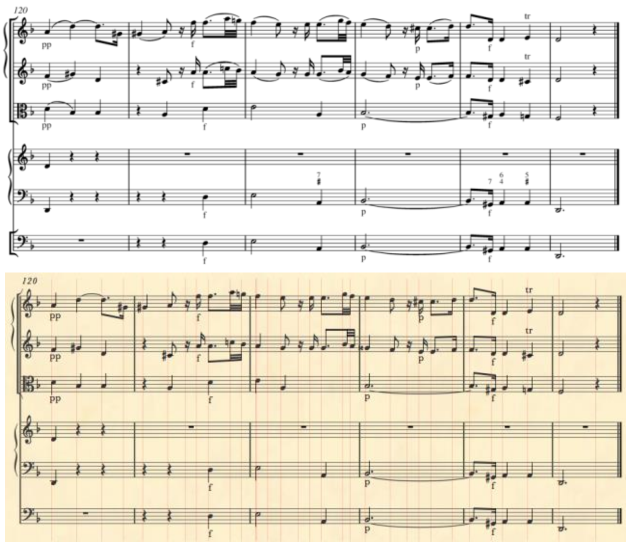


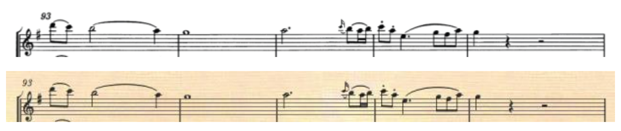
Figure 8 : SCORE PostScript output (top) and SCORE data converted into Dox data in a screen-shot of the Dox editor (bottom).

3.1 SCORE to Dox

Figure 8 shows graphical output from a SCORE PostScript file above a conversion displayed in the Dox music editor written by David Packard. The Dox data format encodes explicit layout information in a header for each system, followed by a listing of symbolic data for each part in the system. For each system measure, a *grid* instruction specifies a spatial distance between times in the composite rhythm for the system. These grid points can be displayed as red vertical lines within the editor as show in the screen capture at the bottom of Figure 8. These gridlines are calculated directly from the horizontal placement (P3) of notes when converting from SCORE data. Within Dox data, the absolute horizontal positions are converted into incremental distances from the previous composite rhythm time in the measure.

Unlike SCORE data, the Dox format separates layout information from symbolic musical elements. Figure 9 shows some sample Dox data illustrating this property. At the start of the data for each system, a header gives layout information. The *bars* directive controls the absolute positions of the measures within the system, and each *grid* directive controls the spacing between composite rhythm positions within each measure. For example "147x13" at the start of the grid for the first measure means that the first beat is 147 spatial units from the start of the measure (relatively wide, to allow for the system clef and key signature to be inserted), then the next position in the composite rhythm sequence is a sixteenth note later, and this is placed 13 units after the notes of on the first beat.

The Dox editor manipulates note spacing by adjusting these grid points, so notes across multiple staves in a system sounding at the same time are always vertically aligned. Vertical positioning of staves as well as the size of staves are also stored in Dox data, so page layout can be preserved when converting from SCORE data.



```
system {pg:10} {tiff:page10.tif} {pix:83,718,1019,1363}
bars {h:144,391,541,693,839,1049}
grid 147x13x13x13x25e14x12x23e15x12x23e14x12
grid 2 7x12x11x11x11e15e17e16e18e16
grid 3 11e15e18e17e17e17e17e17x11
grid 4 10e17e16e18e16e18e16e16
grid 5 10x12x12x12x16x13x13x12x13x12x12x12x14x12x12x12
staff treble k:1 p:1 {v:0 brk bc}
E D6 sa {b:3a} {s:5,6,3,7,-4,-6,1}
E C6 {e:3c} {z}
H B5 sa {s:5,6,2,6,-10,-14,1}
Q A5 {z}
/
W G5
/
```

```
H. A5
~X C6 sa {s:1,-7,-3,-5,5,2,1}
E B5 sa {b:3c} {s:5,6,3,7,-7,-7,1} {z}
X A5 sa
X B5 {e:3c} {z}
/
E C6 st {b:3a}
E A5 st {e:3c}
Q. E5 sa {s:5,7,3,6,-15,-11,1}
E G5 sa {b:2c}
E F#5 sa
E A5 {e:2c} {z}
/
Q G5
QR
HR
```

Figure 9 : Scanned notation (top staff) with matching layout of music in Dox editor (bottom staff). System layout is highlighted in gray below the staves, along with symbolic notation in Dox format for the staff.

3.2 SCORE to Humdrum

As a sample of a primarily symbolic data format, this section gives an example conversion result into the Humdrum data format, which is used in computational music analysis applications. This data format typically contains no layout information since the primary focus is on encoding pitch, rhythm and meter for analysis, and not on layout for printing. The Humdrum format is compact and allows the musical content to be read directly from the representation more so than any other symbolic digital representation of musical notation that encode parts serially rather than in the parallel fashion of Humdrum.

The following text lists a conversion from the SCORE data of Figure 1 into Humdrum syntax. Each staff is represented by column of data (*spines*), with staff layers causing splits of the spines into sub-columns. Each line of data represents notes sounding at the same time, so the rows represent the composite rhythm of all parts, which is similar to the rhythm sequence of *grid* directives in Dox.

```
**kern            **kern
*staff2           *staff1
*clefF4           *clefG2
*k[b-e-a-]        *k[b-e-a-]
*M2/4             *M2/4
=1-               =1-
2r                4e-/ 4g/
.                 4B-/ 4f/
=2                =2
4.CC/ 4.C/        4.G/ 4.e-/
8C/ 8E-/          (16.e-/LL
.                 32a-/JJk)
=3                =3
8BB-/ 8En/L       8g/L
8BB-/ 8E/         (16.g/L
.                 32dd-/JJk)
8AAn/ 8F/         8cc/L
8AA-/ 8F#/J       (16.cc/L
.                 32ddn/JJk)
=4                =4
*^                *^
(8G/L     4.GG\   (8.ee-/L    8e-\L
8An/      .       .           8e-\ 8f#\
.         .       16cc/k      .
8Bn/J)    .       8bn/J)      8d\ 8g\J
8r        8r      (16gg\LL    8r
.         .       16eee-\JJ)  .
*clefG2   *clefG2 *           *
*v        *v      *           *
=5        =5      =5
*^        *       *
8g/L      4.G\    8.eee-/L    8ee-\L
8an/      .       .           8ee-\ 8ff#\
.         .       16ccc/Jk    .
8bn/      .       8bbn/L      8dd\ 8gg\
8b-/J     8g\     8bb-/J      8dd\ 8gg\J
*v        *v      *           *
*                 *v          *v
*-                *-
```

Humdrum syntax is a generalized system, so if layout information needs to be preserved, an additional column of for horizontal positions could be added. This would duplicate the functionality of the grid directives in Dox files. Other formats that do not encode layout information would be converted in a similar manner as the conversion process from SCORE into Humdrum. Data formats in this category include MIDI, ABC, LilyPond, and Guido Music Notation.

3.3 SCORE to musicXML

MusicXML is primarily used as a symbolic music format, but has a mostly complete system for specifying layout in notation. In contrast to the Dox format, the layout parameters are interleaved within the data, typically being given as element attributes. Figures 10 and 11 illustrate conversions from SCORE into musicXML for a work by Dufay generated by the *score2musicxml* converter. These two figures highlight the page layout information that can be preserved when translating between SCORE and musicXML. Both figures have the same system break locations, staff scalings and system margins. While musicXML 3.0 has the capability to specify the horizontal layout of notes and measures, this information is currently stripped out of the data when importing into the most recent version of Finale (2014).

3.4 SCORE to MEI

From SCORE's point of view, conversion into musicXML and to MEI are similar, and the *score2mei* converter was initially adapted from the musicXML conversion program. MEI data is more hierarchical than musicXML data, with elements such as beams and chords stored in a tree structure, while musicXML attaches these features to a flat listing of the notes. Figure 12 demonstrates the different encoding methods for a chord in SCORE, MEI and MusicXML. MEI wraps individual notes within a <chord> element, while musicXML marks secondary notes of the chord with a Boolean <chord/> child element.

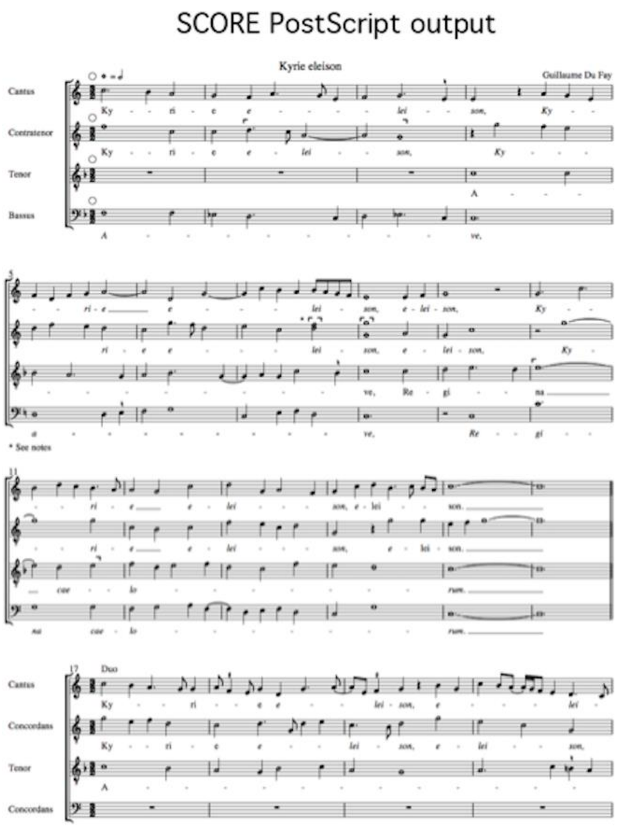


Figure 10 : SCORE PostScript output matching to musicXML translation shown in Figure 11.

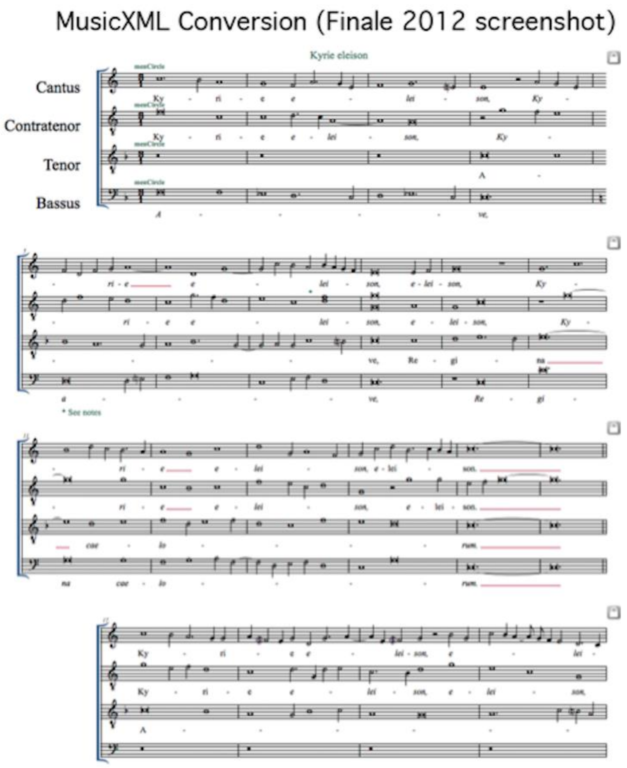
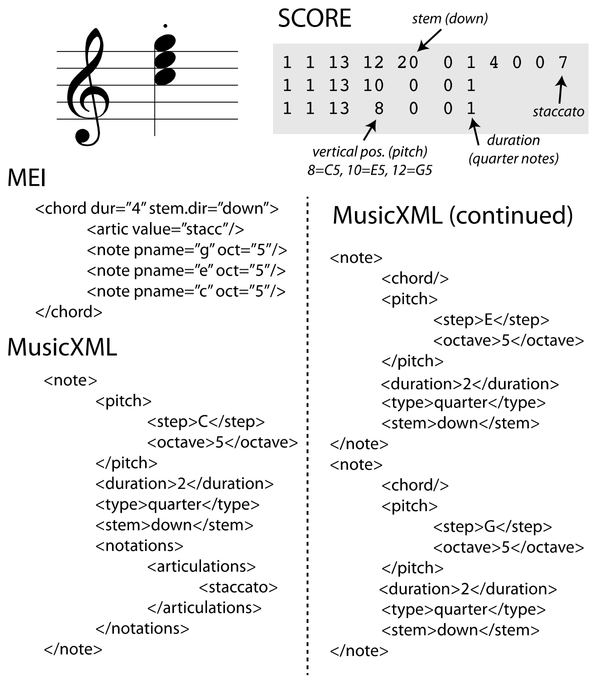


Figure 11 : Screen shot of a musicXML conversion in the Finale music editor. During conversion the rhythmic values of the converted score have been doubled to match the rhythmic values of the original 15th century score.



SCORE

```
1  1  13  12  20   0  1  4  0  0  7
1  1  13  10   0   0  1
1  1  13   8   0   0  1
```

MEI

```
<chord dur="4" stem.dir="down">
        <artic value="stacc"/>
        <note pname="g" oct="5"/>
        <note pname="e" oct="5"/>
        <note pname="c" oct="5"/>
</chord>
```

MusicXML

```
<note>
        <pitch>
                <step>C</step>
                <octave>5</octave>
        </pitch>
        <duration>2</duration>
        <type>quarter</type>
        <stem>down</stem>
        <notations>
                <articulations>
                        <staccato>
                </articulations>
        </notations>
</note>
```

MusicXML (continued)

```
<note>
        <chord/>
        <pitch>
                <step>E</step>
                <octave>5</octave>
        </pitch>
        <duration>2</duration>
        <type>quarter</type>
        <stem>down</stem>
</note>
<note>
        <chord/>
        <pitch>
                <step>G</step>
                <octave>5</octave>
        </pitch>
        <duration>2</duration>
        <type>quarter</type>
        <stem>down</stem>
</note>
```

Figure 12. A chord in SCORE format with translations into MEI and MusicXML below.

3.5 SCORE and MuseData

The MuseData printing system uses two data formats: one for symbolic data encoding, and another for explicit layout. Typically music is encoded in the symbolic format that is then compiled into the format with specific layout for interactive editing.[6] MusicXML is structurally based on the symbolic for of MuseData. The compiled layout-specific format is analogous to SCORE data. A useful property of the MuseData printing system is access to both the high-level symbolic representation as well as the low-level graphical representation.

3.6 SCORE and SVG

Due to SCORE data's graphical nature, converting it into images is less complex than generating images from purely symbolic representations (outside of the intended software for a representation, of course). While each graphical element in SCORE can be placed independently at a pre-determined position in an image, software processing symbolic formats must first calculate a graphical layout, and unlike MuseData this layout representation is typically inaccessible as an independent data format. While SCORE software does not have native export to SVG images, minimal processing of its EPS output can produce SVG images.[7] Analytic overlays on the notation image can be aligned to the image using the layout information from the original SCORE data.

Since SCORE data is compact, it can be stored within an XML files. For example the complete SCORE data for the music of Figure 1 can be found in an SVG image of the incipit used on the Wikipedia page for Beethoven's 26th piano sonata.[8] At the bottom of the SVG image's source code, the SCORE data used to create the SVG image is embedded within a processor instruction using this syntax:

```
<?SCORE version="4"
     SCORE data placed here
   ?>
```

Embedding the source code for creating the image allows the data to be used to regenerate an SVG image to fix notational errors or to prepare a new layout, and the embedded data can also be used to generate additional analytic markup.

Further samples of SCORE data can be found in the GitHub repository for *scorelib*.[9] Additional SCORE data samples can be found on IMSLP as attachments to PDFs of music that the author has typeset in SCORE.[10]

[6] The batch-processing version of the MuseData printing system (`http://musedata.ccarh.org`) can be used to generate both PostScript output and the intermediate layout representation called *Music Page Files* (MPG).

[7] Using the open-source converter `https://github.com/-thwe/seps2svg`

[8] `http://en.wikipedia.org/wiki/Piano_Sonata-_No._26_(Beethoven)`

[9] `https://github.com/craigsapp/scorelib/tree/-master/data`

[10] `http://imslp.org/wiki/User:Craig`

4. CONCLUSIONS

SCORE is an important historical data format for computer-based music typesetting. Understanding its graphical representation system is particularly useful for projects in OMR, where interpreted graphical symbols must be organized in a similar process as converting from SCORE into other data formats. In addition, the SCORE representation system should be studied by projects writing automatic music layout of purely symbolic data. SCORE is primarily used by professional typesetters due to its high-quality output and the degree of control afforded to the typesetter. Using the *scorelib* software allows SCORE data to be more easily converted into other musical formats, usually with minimal manual intervention and exactly preserving the original layout.

CODE SCORES IN LIVE CODING PRACTICE

Thor Magnusson
Department of Music
University of Sussex, Brighton
t.magnusson@sussex.ac.uk

ABSTRACT

This paper explores live coding environments in the context of notational systems. The improvisational practice of live coding as combining both composition and performance is introduced and selected systems are discussed. The author’s Threnoscope system is described, but this is a system that enables the performer to work with both descriptive and prescriptive scores that can be run and altered in an improvisational performance.

1. INTRODUCTION

The live coder sits on stage and writes software in front of a live audience. The desktop is projected on the wall in a gesture of sharing and audience engagement [1, 2]. In the past decade, live coding has become a popular performance practice, supported by the diverse interpreted and high level programming languages that suit the practice. Furthermore, the popular hacker and maker cultures are affecting general culture such that coding is now considered a creative activity on par with drawing or playing an instrument. The live coding community has played an important role here and been active in disseminating the practice by sharing code, organizing festivals and conferences, and establishing research networks.

Code is a form of notation that works extremely well in musical composition, especially when the aim is to write non-linear, interactive, or context aware music [3]. Although general-purpose languages can be used for musical live coding, many live coders have created their own mini-languages for a particular performance style, genre, or even a performance. The new language becomes an instrument, a framework for thinking, with strong considerations of notational design. Here, language designers have invented graphical interfaces like we find in Pure Data or Max/MSP; game interfaces, as in Dave Griffiths' *Al Jazaari*; functional notation, like McLean's *Tidal*; or Chris Kiefer’s physical controllers that encode genetic algorithms of sound synthesis [4].

2. NOTATION AND INTERPRETATION

Notation is a way of communicating abstract ideas to an interpreter, and in live coding that interpreter is typically a compiler called the “language interpreter.” Standard Music Notation is a system of notation that has developed from the general recognition that symbols can capture more information, coherent in meaning between composers, interpreters and cultures, in a smaller space than natural language or bespoke new symbolic languages. Standard Music Notation is a cognitive tool that has developed with requirements for a standard language and concerns about sight-reading and rapid understanding. Composers are able to rely on the performer’s expertise and creative interpretation skills when the piece is played. Conversely, in the symbolic notation for computer music composition and performance, we encounter an important difference in the human and the computer capacity for interpretation: the human can tolerate mistakes and ambiguity in the notation, whereas the computer cannot. Natural language programming of computers is clearly possible, for example:

```
produce a sine wave in A
name this synth "foo"
wrap this in an ADSR envelope
repeat foo four times per second
name this pattern "ping"
```

However, the problem here is one of syntax: what if the coder writes “Sine” instead of “sine,” “440” instead of “A,” or “every 0.25 second” instead of “four times per second?” The cognitive load of having to write natural language with the programming language’s unforgiving requirements for perfect syntax makes the natural language approach less appealing than writing in traditional programming languages, for example in functional or object orientated languages. Of course, semi-natural language programming languages have been invented, such as COBOL, Apple Script, or Lingo. The problems with those were often that they became quite verbose and the 'naturalness' of their syntax was never so clear. Consequently, in a more familiar object oriented dot-syntax, the above might look like:

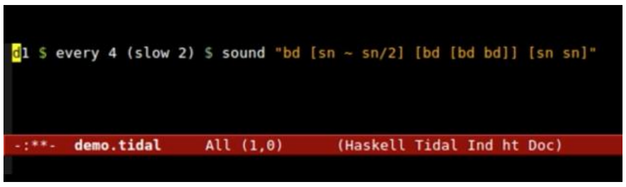
```
w = Sine("foo", [\freq, 440]);
w.addEnvelope(\adsr);
p = Pattern("ping");
p.seq(\foo, 0.25);
```

In both cases we have created a synthesizer and a pattern generator that plays the synth. The latter notation is less prone to mistakes, and for the trained eye the syntax actually becomes symbolic through the use of dots, camelCase, equals signs, syntax coloring, and brackets with arguments that are differently formatted according to type. This is called 'secondary notation' in computer science parlance, and addresses the cognitive scaffolding offered by techniques like colorization or indentation [5].

3. LIVE CODING AS NOTATION

Live coding is a real-time performance act and therefore requires languages that are relatively simple, forgiving in terms of syntax, and high level. Certain systems allow for both high and low level approach to musical making, for example SuperCollider, which enables the live coder to design instruments (or synths) whilst playing them at another level of instructions (using patterns). Perhaps the live coding language with the most vertical approach would be Extempore [6], which is a live coding environment where the programming language Scheme is used at the high level to perform and compose music, but another language – type sensitive and low level, yet keeping the functional programming principles of Scheme – can be used for low level, real-time compiled instructions (using the LLVM compiler). In Extempore, an oscillator, whether in use or not, can be redesigned and compiled into bytecode in real-time, hotswapping the code in place.

However, live performance is stressful and most live coders come up with their own systems for high-level control. The goals are typically fast composition cycle, understandability, novel interaction, but most importantly to design a system that suits the live coder's way of thinking. Below is an introduction of four systems that all explore a particular way of musical thinking, language design, and novel visual representation.



```
d1 $ every 4 (slow 2) $ sound "bd [sn ~ sn/2] [bd [bd bd]] [sn sn]"

-:**-  demo.tidal      All (1,0)      (Haskell Tidal Ind ht Doc)
```

Figure 1 : A screen shot of Tidal. We see the score written in the quotation marks, with functions applied.

Alex McLean's *Tidal* [7] is a high level mini-language built on top of Haskell. It offers the user a limited set of functionality; a system of constraints that presents a large space for exploration within the constraints presented [8]. The system focuses on representing musical pattern. The string score is of variable length, where items are events, but these items can be in the form of multi-dimensional arrays, representing sub-patterns. This particular design decision offers a fruitful logic of polyrhythmic and polymetric temporal exploration. The system explicitly *affords* this type of musical thinking, which consequently limits other types of musical expression. The designers of the live coding languages discussed in this section are not trying to create a universal solution to musical expression, but rather define limited sets of methods that explore certain musical themes and constraints.

Dave Griffiths' *Scheme Bricks* is a graphical coding system of a functional paradigm, and, like Tidal, it offers a way of creating recursive patterns. Inspired by the MIT Scratch [9] programming system, a graphical visualization is built on top of the functional Scheme programming language. The user can move blocks around and redefine programs through visual and textual interactions that are clear to the audience. The colored code blocks are highlighted when the particular location of the code runs, giving an additional representational aspect to the code.

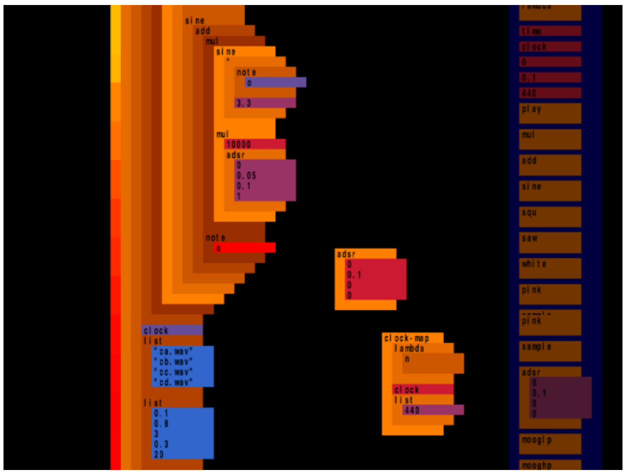
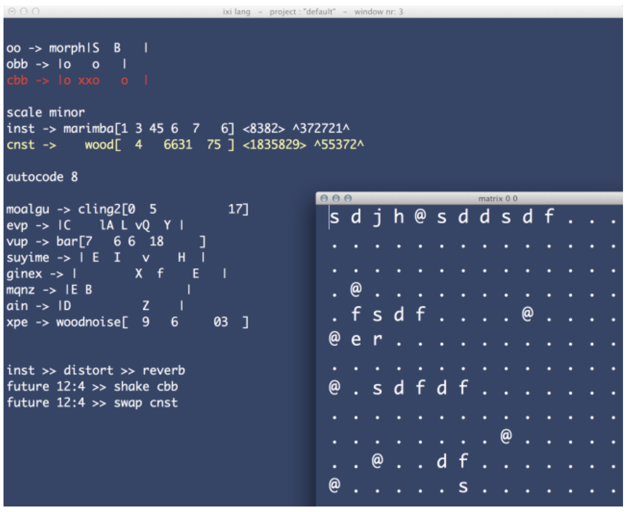


Figure 2 : Scheme Bricks. In simple terms, what we see is the bracket syntax of Scheme represented as blocks.

Scheme Bricks are fruitful for live musical performance as patterns can be quickly built up, rearranged, muted, paused, etc. The modularity of the system makes it suitable for performances where themes are reintroduced (a muted block can be plugged into the running graph).

This author created *ixi lang* in order to explore code as musical notation [10]. The system is a high level language built on top of SuperCollider and has access to all the functionality of its host. By presenting a coherent set of bespoke '*ixi lang*' instructions in the form of a notational interface, the system can be used by novices and experienced SuperCollider users alike. The system removes many of SuperCollider's complex requirements for correct syntax, whilst using its synth definitions and patterns; the original contribution of *ixi lang* is that it creates a mini-language for quick prototyping and thinking.

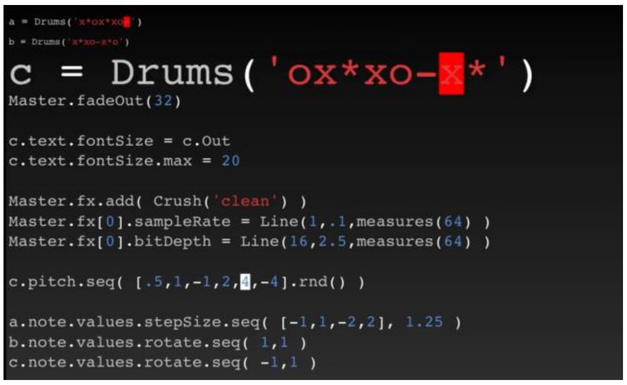


```
oo -> morph|S  B   |
obb -> |o    o   |
cbb -> |o xxo    o  |

scale minor
inst -> marimba[1 3 45 6  7   6] <8382> ^372721^
cnst ->    wood[  4   6631  75 ] <1835829> ^55372^

autocode 8

moalgu -> cling2[0  5           17]
evp -> |C    1A L vQ  Y |
vup -> bar[7   6 6  18      ]
suyime -> | E  I   v    H  |
ginex -> |        X  f    E   |
mqnz -> |E B               |
ain -> |D           Z    |
xpe -> woodnoise[  9   6     03  ]


inst >> distort >> reverb
future 12:4 >> shake cbb
future 12:4 >> swap cnst
```

```
s d j h @ s d d s d f . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. @ . . . . . . . . . . . .
. f s d f . . . . @ . . . .
@ e r . . . . . . . . . . .
. . . . . . . . . . . . . .
@ . s d f d f . . . . . . .
. . . . . . . . . . . . . .
. . . . . . . . @ . . . . .
. . @ . . d f . . . . . . .
@ . . . . . s . . . . . . .
```

Figure 3 : *ixi lang* Agents are given scores that can be manipulated, and changed from other code.

In *ixi lang* the user creates agents that are assigned percussive, melodic, or concrete scores. The agents can be controlled from other locations in the code and during that process the textual document is automatically rewritten to reflect what is happening to the agents. This makes it possible for the coder and the audience to follow how the code is changing itself and the resulting music. As code can be rewritten by the system, it also offers the possibility of storing the state of the code at any given time in the performance. This is done by writing a snapshot with a name: the snapshot can then be recalled at any time, where running new code is subsequently muted (and changes color), and agents whose score has changed return to their state when the snapshot was taken.



```
a = Drums('x*ox*xo ')
b = Drums('x*xo-x*o')
c = Drums('ox*xo-x*')
Master.fadeOut(32)

c.text.fontSize = c.Out
c.text.fontSize.max = 20

Master.fx.add( Crush('clean') )
Master.fx[0].sampleRate = Line(1,.1,measures(64) )
Master.fx[0].bitDepth = Line(16,2.5,measures(64) )

c.pitch.seq( [.5,1,-1,2,4,-4].rnd() )

a.note.values.stepSize.seq( [-1,1,-2,2], 1.25 )
b.note.values.rotate.seq( 1,1 )
c.note.values.rotate.seq( -1,1 )
```

Figure 4 : Gibber. Here textual code can change size, color or font responding to the music. All user-defined.

A recent live coding environment by Charlie Roberts called *Gibber* [11] takes this secondary notation and visual representation of music further than ixi lang: here we can see elements in the code highlighted when they are played: the text flashes, colors change, and font sizes can be changed according to what is happening in the music. Gibber allows for any textual element to be mapped to any element in the music. The code becomes a visualization of its own functionality: equally a prescription and description of the musical processes.

Gibber is created in the recent Web Audio API, which is a JavaScript system for browser-based musical composition. As such it offers diverse ways of sharing code, collaborating over networks in real-time or not.

All of the systems above use visual elements as both primary and secondary notation for musical control. The notation is prescriptive – aimed at instructing computers – although elements of secondary notation can represent information that could be said to be of a descriptive purpose [12]. The four systems have in common the constrained set of functions aimed to explore particular musical ideas. None of them – bar Gibber perhaps – are aimed at being general audio programming systems, as the goals are concerned with live coding: real-time composition, quick expression, and audience understanding.

4. THE THRENOSCOPE

In the recent development of the Threnoscope system, the author has explored representational notation of live coding. This pertains to the visualization of sound where audible musical parameters are represented graphically. The system is designed to explore microtonality, tunings, and scales; and in particular how those can be represented in visual scores aimed at projection for the audience.

The Threnoscope departs from linear, pattern-based thinking in music and tries to engender the conditions of musical stasis through a representation of sound in a circular interface where space (both physical space and pitch space) is emphasized, possibly becoming more important than concerns of time.

The system is object oriented where the sound object – the 'drone' – gets a graphical representation of its state. This continuous sound can be interacted with through code, the graphical user interface, MIDI controllers, and OSC commands, and changes visually depending upon which parameters are being controlled. The user can also create 'machines' that improvise over a period of time on specific sets of notes, as defined by the performer. These machines can be live coded, such that their behavior changes during their execution. Unlike the code score, discussed below, the machines are algorithmic: they are not intended to be fully defined, but rather to serve as an unpredictable accompaniment to the live coder.

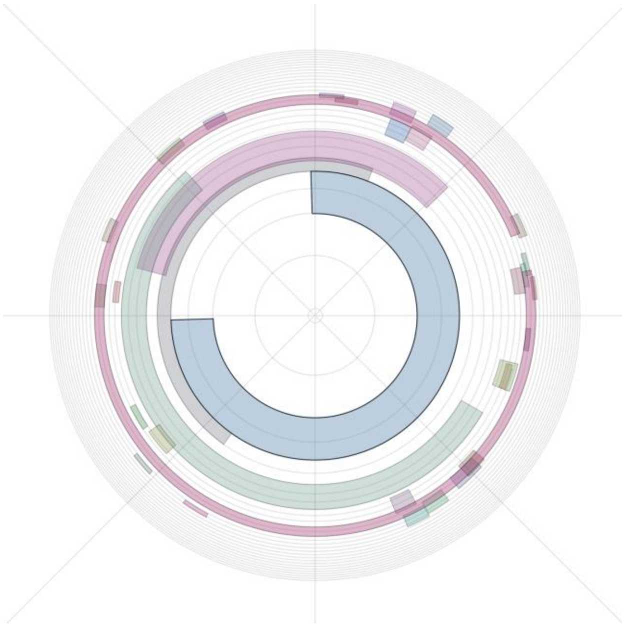
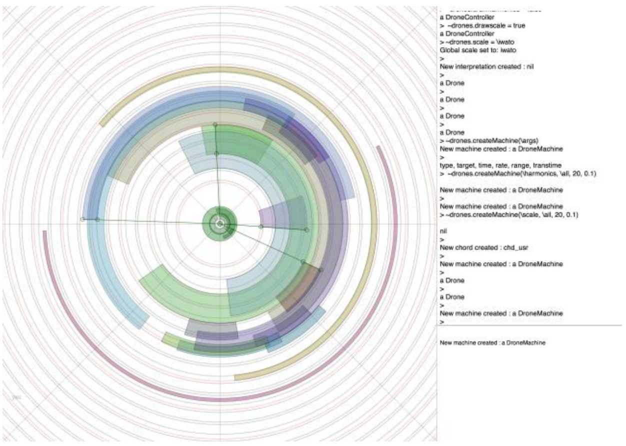


Figure 5. The Threnoscope in 8-channel resolution. The straight crossing lines are speakers. The circles are harmonics, and the colored wedges are (moving) drones.



```
a DroneController
> ~drones.drawscale = true
a DroneController
> ~drones.scale = \wato
Global scale set to: iwato
>
New interpretation created : nil
>
a Drone
>
a Drone
>
a Drone
>
a Drone
> ~drones.createMachine(\args)
New machine created : a DroneMachine
>
type, target, time, rate, range, transtime
> ~drones.createMachine(\harmonics, \all, 20, 0.1)

New machine created : a DroneMachine
>
New machine created : a DroneMachine
> ~drones.createMachine(\scale, \all, 20, 0.1)

nil
>
New chord created : chd_usr
>
New machine created : a DroneMachine
>
a Drone
>
a Drone
>
New machine created : a DroneMachine
>

New machine created : a DroneMachine
```

Figure 6. The Threnoscope's code interface on the right. The system is here in a scale-mode, with scale degrees rather than harmonics. A machine is running in the middle, affecting a selection of the running drones.

Figure 5 shows the circular interface where the innermost circle is the fundamental frequency, with the harmonics repeated outwardly. The lines crossing the interface represent the audio channels or speakers (the system can be set from 2 to 8 channels). The sound/drone can have a length extending up to 360 degrees, but it can also be short and move fast around the space. Figure 6 depicts the system with the command line prompt on the right, and a console underneath that reports on the state of the engine, errors in code, or events being played in a running score. By clicking on a particular drone, its sonic information appears in the console in a format that gives the coder quick entry to manipulate the parameters.

Musical events in the Threnoscope system are created through code instructions. Since the default envelope of the drone is an ASR (Attack, Sustain, Release) envelope, a note duration can range from a few milliseconds to an infinite length. Each of the speaker lines could be seen as a static playhead, where notes either cross it during their movement or linger above it with continuous sound. A compositional aspect of the Threnoscope is to treat notes as continuous objects with states that can be changed (spatial location, pitch, timbre, amplitude, envelope, etc.) during its lifetime.

The Threnoscope has been described before both in terms of musical notation [11] and as a system for improvisation [12]. This paper explores further the notational aspects of the system, and the design of the code score.

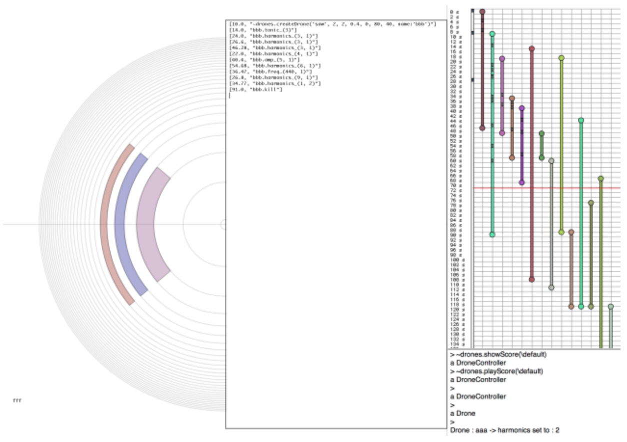
5. THE CODE SCORE

Code is rarely represented on a timeline, although certain systems have enabled programmers to organize code linearly over time, although in Macromedia's Director and Flash multimedia production software this becomes a key feature. This general lack of a timeline can pose a problem when working with code as a creative material in time-based media such as music, games or film. The lack of timeline makes navigating the piece for compositional purposes cumbersome and often impossible.

The Threnoscope's code score is a two dimensional textual array where the first item is the scheduled time and the second contains the code to be executed. This makes it possible to jump to any temporal location in the piece, either directly or through running the code that is scheduled to happen before (with some limitations though, as this code could be of a temporal nature as well).

Scores in textual code format, like that of the Threnoscope, can make it difficult to gain an overview of the musical form, as multiple events can be scheduled to take place at the same moment with subsequent lack of activity for long periods. This skews the isomorphism between notational space (the lines of code) and time if working with the mindset of a linear timeline. For this reason the Threnoscope offers an alternative chronographic visualization to represent the code in the spatial dimension. This is demonstrated in Figure 7.

The code score timeline is vertically laid out as is common in tracker interfaces. The code can be moved around in time, deleted, or new elements added. By clicking on relevant 'code tracks' the user can call up code into a text field and edit the code there. The drones are created on the vertical tracks on the timeline. They have a beginning and an end, with code affecting the drones represented as square blocks on the drone track. The drone itself and the events within it can be moved around with the mouse or through code. The score can therefore be manipulated in real-time, much like we are used to with MIDI sequencers or digital audio workstations.



```
[10.0, "~drones.createDrone('saw', 2, 2, 0.4, 0, 80, 40, name:'bbb')"]
[14.0, "bbb.tonic_(3)"]
[24.0, "bbb.harmonics_(3, 1)"]
[26.6, "bbb.harmonics_(3, 1)"]
[46.28, "bbb.harmonics_(3, 1)"]
[22.0, "bbb.harmonics_(4, 1)"]
[60.4, "bbb.amp_(5, 1)"]
[54.68, "bbb.harmonics_(6, 1)"]
[36.47, "bbb.freq_(440, 1)"]
[26.8, "bbb.harmonics_(9, 1)"]
[34.77, "bbb.harmonics_(1, 2)"]
[91.0, "bbb.kill"]
```

```
> ~drones.showScore(\default)
a DroneController
> ~drones.playScore(\default)
a DroneController
>
a DroneController
>
a Drone
>
Drone : aaa -> harmonics set to : 2
```

Figure 7. A graphical visualization of the code score. When a vertically lined drone is clicked on, a page with its code appears above the circular interface.

Most timelines in music software run horizontally from left to right, but in the Threnoscope the score vertical and runs from top down. This is for various reasons: firstly, the available screen space left on most display resolutions when the circular score has taken up the main space on the left is a rectangular shape with the length on the vertical axis; secondly, when a user clicks on the visual representation of the drone, its score pops up in the textual form of code, and this text runs from top to bottom. It would be difficult to design a system where code relates to events on a horizontal timeline.

6. PERFORMING WITH SCORES

The code score was implemented for two purposes: to enable small designed temporal patterns to be started at any point in a performance: just like jazz improvisers often memorize certain musical phrases or licks, the code score would enable the live coder to pre-compose musical phrases. The second reason for designing the code score system is to provide a format for composers to write longer pieces for the system, both linear and generative.

The score duration can therefore range from being a short single event to hours of activity; it can be started and stopped at any point in a performance, and the performer can improvise on top of it. Scores can include other scores. As an example, a performer in the middle of a performance might choose to run a 3-second score that builds up a certain tonal structure. The code below shows the code required to start a score.

```
~drones.playScore(\myScore, 1) // name of score & time scale
~drones.showScore(\myScore) // visual display of the score
```

The first method simply plays the score without a graphical representation. This is very flexible, as multiple scores can be played simultaneously, or the same score started at different points in time. Scores can be stopped at will. Whilst the scores are typically played without any visual representation, it can be useful to observe the score graphically. The second method creates the above-mentioned graphical representation of the score shown in Figure 7. For a live performance, this can be helpful as it allows the performer to interact with the score during execution. The visual representation of the score can also assist in gaining an overview of a complex piece.

For this author, the code score has been a fruitful and interesting feature of the system. Using scores for digital systems aimed at improvisation becomes equivalent to how instrumentalists incorporate patterns into their motor memory. The use of code scores question the much broken unwritten 'rule' that a live coding performance should be coded from scratch. It enables the live coder work at a higher level, to listen more attentively to the music (which, in this author's experience, can be difficult when writing a complex algorithm), and generally focus more on the compositional aspects of the performance.

7. CONCLUSION

This short paper has discussed domain specific programming languages as notational systems. Live coding systems are defined as often being idiosyncratic and bespoke to their authors' thought processes. The Threnoscope and its code score was presented as a solution to certain problems of performance and composition in live coding, namely of delegating activities to actors such as machines or code scores.

8. REFERENCES

[1] N. Collins, A. McLean, J. Rohrhuber, and A. Ward. "Live coding in laptop performance." *Organised Sound*, vol 8. n° 3, 2003, pp. 321-330.

[2] T. Magnusson. "Herding Cats: Observing Live Coding in the Wild" in *Computer Music Journal*, vol. 38 n° 1, 2014, pp. 8-16.

[3] T. Magnusson. "Algorithms as Scores: Coding Live Music" in *Leonardo Music Journal*. vol 21. n° 1, 2011, pp. 19-23.

[4] C. Kiefer. "Interacting with text and music: exploring tangible augmentations to the live-coding interface" in *Proceedings of the International Conference for Life Interfaces*, 2014.

[5] A. F. Blackwell, and T. R. G. Green. "Notational systems - the Cognitive Dimensions of Notations framework" in J.M. Carroll (Ed.) *HCI Models, Theories and Frameworks: Toward a multidisciplinary science*. San Francisco: Morgan Kaufmann, 2003, pp. 103-134.

[6] A. Sorensen, B. Swift, and A. Riddell. “The Many Meanings of Live Coding” in *Computer Music Journal*. vol. 38. n° 1, 2014, pp. 65-76.

[7] A. McLean. “Making programming languages to dance to: Live coding with Tidal” in *Proceedings of the 2nd ACM SIGPLAN International Workshop on Functional Art, Music, Modelling and Design*, 2014.

[8] T. Magnusson. “Designing constraints: composing and performing with digital musical systems” in *Computer Music Journal*, vol. 34. n^{o} 4., 2010, pp. 62-73.

[9] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. S. Silverman, and Y. Kafai. “Scratch: Programming for All” in *Communications of the ACM*, vol. 52. n^{o} 11, 2009.

[10] T. Magnusson. “ixi lang: a SuperCollider parasite for live coding” in *International Computer Music Conference*, 2011.

[11] C. Roberts, and J. Kuchera-Morin. “Gibber: Live Coding Audio in the Browser.” in *Proceedings of the International Computer Music Conference*, 2012, pp. 64-69.

[12] A.F. Blackwell, and T.R.G. Green. “Notational systems - the Cognitive Dimensions of Notations framework” in J.M. Carroll (Ed.) *HCI Models, Theories and Frameworks: Toward a multidisciplinary science*. San Francisco: Morgan Kaufmann, 2003, pp. 103-134.

[13] T. Magnusson. “Scoring with code: composing with algorithmic notation” in *Organised Sound*, vol. 19. no. 3, 2014, pp. 268-275.

[14] T. Magnusson. “Improvising with the threnoscope: integrating code, hardware, GUI, network, and graphic scores” in *Proceedings of the New Interfaces for Musical Expression Conference*, 2014.

THEMA: A MUSIC NOTATION SOFTWARE PACKAGE WITH INTEGRATED AND AUTOMATIC DATA COLLECTION

Peter McCulloch
New York University
peter.mcculloch@nyu.edu

ABSTRACT

This paper introduces Thema, a custom music notation software environment designed to automatically and transparently capture quantitative data into a relational database. The majority of research into musical creativity is qualitative in nature, and this software addresses several areas, such as search and improvisational data, which are difficult to study with current qualitative methods. Thema's database provides advantages over ad hoc file collection mechanisms by providing integrated search; the software also is able to consistently identify musical material via automatically assigned identification codes, and this provides a useful supplement to content-based search. In 2013, a study was conducted of ten graduate-level composers using Thema, and the dataset from this study was used to develop new analytical tools for examining compositional data.

1. INTRODUCTION

Until recently, most research into the compositional process of adult composers has been conducted using qualitative methodologies. Creativity is complex and researchers have rightly appreciated the role that composers play in illuminating their creative process. A variety of techniques have been used by researchers and composers including interviews, verbal protocol, sketch studies, and journals. With the commodification of recording technology, so-called real-time studies of compositional process have become more common. These typically involve audio or video recordings of the composer. To provide insight into the composer's thought process, verbal protocol techniques are often used, either concurrently with composition or retrospectively. This data may then be triangulated with versioned musical sketches or computer files and supplemented by journals and other documentation. While combining multiple sources gives relatively good coverage of activity, utilizes the composers' personal insights, and allows composers to work with familiar tools, it also requires effort on the part of the composer and the researcher to collect and organize data. [1, p. 246-7]. In the short run, this is certainly manageable, but it is difficult to scale these techniques up to larger studies, and, as might be expected, longitudinal studies are rare in the qualitative literature concerning adult composers, as are studies featuring large sample sizes. [1]

Musical informatics have proved useful for addressing questions that concern a large amount of music, and it stands to reason that they could be of some aid in the study of compositional process. While computerized analysis is unlikely to bring the same type of insight that a qualitative study can, it has strengths in complementary areas: namely, it can be pursued over time and at scale because the data can be automatically analyzed. Anecdotal accounts from composers in the process literature suggest that composers benefit from participating in these studies but most composers will never have the opportunity to do so. Quantitative analytical tools could allow composers to systematically examine their own music and behavior on an ongoing basis without a dedicated researcher.

It is increasingly common for composers to use software in the act of composing music rather than purely for typesetting. Though the impacts of this trend are subject to debate, it seems unlikely to change in the near future. The software that composers use to create music is uniquely suited to observing quantitative data about the creative process since it can see not only the document, but also the composer's interactions with the document. Several studies in the literature have used dedicated software to observe compositional behavior [4–6]; these have not, however, utilized Common Music Notation. Other studies have observed composers at work via automatic screen captures or screen recording [7–9]. Peterson's 2008 study presents quantitative results, but the data was acquired through the manual analysis of automatically collected screen captures.

[1] Collins [2] and Donin [3] provide useful reviews of the literature.

2. PREVIOUS AND RELATED WORK

A few projects have used custom software to study compositional process. Otto Laske and Barry Truax used the Observer I program to observe children making music with a synthesizer [4]. Maude Hickey wrote a program to study creativity in children [5]. The QSketcher research project at IBM developed an environment for film music composition that automatically recorded and organized improvisational material, and maintained a persistent view of its environment. [10]. Recently, Chris Nash created reViSiT, a free "tracker"-style audio plugin that captured usage data [6]. He used the software to collect data from over 1,000 musicians; his is by far the largest sample size in this area, but his data set does not include the music that his subjects created.

In considering the use of quantitative data for studying compositional process, it is helpful to consider existing infrastructure for storing data. Standardized open file formats have been a boon to music informatics researchers as they allow musical data to be examined independently of the program that created the data. This is helpful since designing music notation software is a time-intensive task, and standard file formats provide some degree of interoperability between programs. At present, however, there is no standard for how a music notation program should operate, and that also means that standard music file formats such as MusicXML include very little, if any, information outside that already present in the score such as how the composer is using the program or improvisational MIDI data. Additionally, our ability to understand how multiple versions of a score are related depends on effective comparison algorithms. Though comparison works for simple additions and deletions, it becomes less useful as the composer's actions become more dispersed across time.

As an alternative to an ad hoc approach to data collection which combines multiple data formats such as MusicXML and SMF and exists separately from the music software, there is a good argument to be made for an integrated data collection system which operates from within the software used to create the music. Such a system would have a greater understanding of how its constituent elements relate and would be able to integrate the information it collects into its operation. This capability could also be materially useful to composers, particularly in its ability to bridge the gap between improvisational development and transcription. There are certainly drawbacks to such a system, most notably, in that it ties the composer's work to a particular software. Nevertheless, it allows access to data which is not well-served by existing methods; this data may provide insight outside of that available in score and MIDI data.

3. THEMA

Thema is a music notation software environment, written using the Java Music Specification Language's JScore package [11], that has been purpose-built for automatic data collection. On the surface, Thema is designed to operate in a manner similar to existing music notation software. Note entry occurs in step-entry mode, optionally with MIDI input, and most functions in the interface are available via keyboard shortcuts, including score playback. Selections may be made with the mouse, and a variety of commands such as transposition and clipboard operations are available. Thema has two different playback modes: cursor and selection; cursor mode plays the score from the current cursor location, while selection mode plays back only the selected material. Like ENP [12], Thema can make use of non-contiguous selections in the score, and this includes playback operations. Thema's musical representation is relatively limited relative to other software in the field such as Bach [13] and ENP. It cannot, for instance, represent nested tuplets, and it does not support breakpoint function notation or proportional notation in editing.

Thema, like JScore, allows the user to operate on rhythm in relative units via doubling and halving operations, and this works for both selections as well as for setting the cursor's duration. This latter aspect is useful in that it does not require the composer to remember specific key mappings for durations, though those are also available. In step-entry mode, the user may also select a rhythmic motive and then use its durations and ties as a source sequence for a new stream of notes, as shown in figure 1. This simplifies the entry of repetitive figures such as dotted eighth and sixteenth note pairs. Additionally, the composer may advance to an arbitrary position within the sequence or reset to the beginning, and this allows for greater flexibility in applying the current sequence without requiring changes to its content; this can be useful, for example, in creating rhythmic ostinato figures in fluctuating meters.

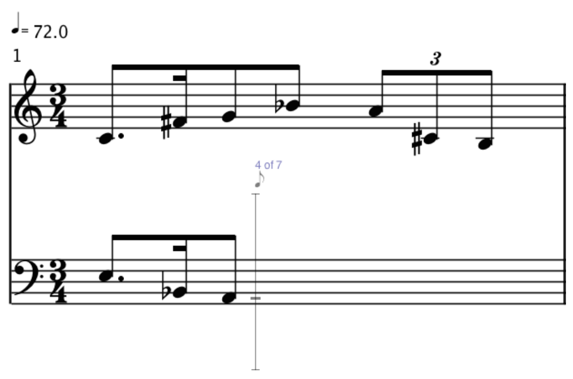


Figure 1. Step entry with rhythmic sequence

3.1 Storage

By design, Thema focuses on automatically and transparently capturing low-level data at a fine time granularity. Where possible, data is stored as it is captured, and all entries are stored with time stamps at millisecond resolution. In order to enable frequent storage, Thema models the score, as well as the state of the program, as a collection of tables within a relational database, and only stores material that has changed over subsequent states. The database maintains previous versions of edited items so that all past states of an item are reachable. In order to easily identify multiple versions of an item across time, each item is tagged with a unique, automatically generated permanent identifier which is consistent across all versions of the item.

Storage in Thema is tightly integrated into to the workings of the program, and it occurs as the result of actions by the composer, instead of at an arbitrary time interval. This makes it easier to discern the composer's actions in the data stream, since any entry in the database is present as the direct result of the composer's actions; it also prevents multiple actions from being condensed into a single entry, as might occur when saving at a particular time interval. A command identifies elements which may have been intentionally changed, and the program proceeds to discern whether those elements were actually changed, as well as any changes to surrounding elements which may have occurred as a result. For example, changing the duration of the first note in the measure has the effect of changing the starting times of subsequent notes within the measure.

3.2 Data Representation

Thema represents the structure of the score in a straightforward, hierarchical fashion: a score contains measures, measures contain staves, staves contain tracks, and tracks contain notes. Following normalized database practices, objects within that hierarchy only reference the class of objects immediately above them in the hierarchy, with the exception that all objects maintain a reference to their containing score.[2] For example, when storing a note in the database, a reference is stored to its track's identifier, but not to its containing staff or measure. This insulates lower level objects from being affected by changes in higher-level objects such as the insertion of a measure earlier in the piece.

With a score changing over time, the amount of data that could be stored is potentially large. It would be inefficient to store the entire document for a small change, so Thema stores only objects that have changed. On load, it reassembles the score from the desired version. This is different than a `diff`-based approach because the state of the entire score across time is visible to search functions without the need for derivation. Accessing earlier versions of a score is as efficient as loading the current score, and it is simple to make comparisons across versions. When undoing commands or reverting to a previous state of the score, the state of the score at the desired moment is loaded from the database and then made current. A detailed description of this mechanism is provided in [14, p.85-109]; one interesting feature of this design is that though it appears to operate like a conventional undo-redo stack, all past states are accessible.

3.3 Processes

Processes are the primary unit of work within Thema. Any action that the composer initiates within the program creates a timestamped entry in the process table, and each entry represents a state in the score or the environment, with the exception of MIDI data, which is stored independently of process information. This distinction is made because the volume of MIDI data is considerable, and may happen in parallel with other actions. MIDI data is recorded with a time stamp, and may be combined with the process table data via an SQL join.

Thema separates processes into two categories: score and environment. Score processes alter the content of the score whereas environment actions, such as adjusting the viewable area, playing back the score, and making drag selections with the mouse, do not. With score processes, the program also records whether or not a command had any effect, e.g., the user attempting to transpose a rest or make a deletion when the cursor is in a blank measure. This allows the program to skip over commands which had no impact when undoing, and also makes it possible to remove empty operations from consideration for analysis.

3.4 Separation of Identity from State

Whether using qualitative or quantitative data, working with multiple versions of a single score often can lead to reference problems. The score is not static over time in these situations, but most of our musical terminology for labeling material depends on it being so. For example, a label such as "the first A5 in measure 7" is tied inextricably to its current state. If the note is transformed, or measures are added in front of it, it is no longer an accurate descriptor. Adding timing information makes it easier to find the particular material as it existed at the moment, but it does not provide a sense of its identity across time. Labeling systems may be used where the software supports it, but these depend on the composer or the researcher to main-

[2] A database can contain multiple scores, and by explicitly filtering based on the score, queries run an order of magnitude faster.

tain consistency, as indicated by David Collins notes in his long-term study of compositional process [1].

Rather than identifying material in the score based on mutable state, it is a better approach to use a permanent identifier that is decoupled from state, e.g., "note #1273". In relational database design, synthetic keys are preferred over natural keys for this same reason in that they preserve the unique identity of a row even when its values change. By using a synthetic key to identify notes and other objects, continuity across time is preserved. It also makes it simple to compare different versions of the same passage which are separated by a large amount of time. This does not preclude searches based on content or comparison, but it reduces dependency on comparison. It is worth noting, however, that this approach is only practical in a situation where the program automatically manages this information.

Thema identifies material by using unique identification numbers. Each object in the score has a unique, permanent identifier as well as a version number. The first value is static and identifies an object across time; it is also guaranteed to be unique across object classes, e.g., for a note with the identifier #1273, no other objects in the database will share the same identifier, even across scores. The second value indicates the specific version of the object within the database table; each version occupies a row in the table. The permanent identifier decouples an object's identity from its state which allows searches to be conducted on the basis of identity rather than musical content. Search based on comparison is certainly still possible, but it is not necessary in order to locate material across time. This is useful because identity-based searches will typically run several orders of magnitude more rapidly than comparison-based ones; for example, it is much simpler to find the history of a particular group of notes by searching for rows with corresponding identifiers than it is to compare thousands of iterations of a score.

3.5 Attribution and Context

When storing data, it is useful not only to be able to identify changed material but also to know how the changes were effected. For example, a "cut" operation is identical to a "clear" operation in terms of the difference between successive states in the score; in the case of a "cut" command, however, it is likely that the material may reemerge as the result of a "paste" command, and it is helpful to identify this relationship as it indicates a larger cognitive process. In this example, Thema will not only indicate the command type during storage, but it will also store a set of parent-child relationships between the source and destination materials so that the link between the two states of the score—however distant—is maintained. The software also records the context of the program, including the location of the cursor, any selections in the score, the current state of playback, and so forth.

Additionally, in storing objects, Thema makes a distinction between directly edited objects and objects that have changed state as a result of edits to other objects. For example, a change in the duration of a note appearing at the beginning of the measure would affect the state of subsequent notes; the first note would be considered to be directly edited, while the other notes would be marked as indirectly edited in the database. Similarly, deleting a measure causes the measure numbers of all subsequent measures to be decremented. By indicating the target of the operation, Thema reduces ripple effects in the data and provides a more accurate picture of the composer's actions.

3.6 Model Objects

Latency can be a challenge for object-oriented applications which use databases for storage. If an object changes before it is correctly stored, the values in the database could become inconsistent with the program values. At the same time, it is important that the user interface for the program is responsive to its user, so it is also impractical to delay processing user input while storage is occuring. To address this problem, Thema uses immutable data objects between the application logic and the database. For object in the score or the environment, the program maintains an immutable data model of each object's current state, e.g., a `Note` object contains a reference to a `NoteModel` object. [3] When an object may have changed as the result of a command, a new model is constructed and compared against the previous state; if different, the new model is added to the storage queue and replaces the previous model. Though the construction of models increases memory requirements, it also allows storage to safely proceed in a separate thread from the user interface, ensuring responsiveness while maintaining data integrity. Because the model objects cannot change once created, they may also be safely cached in memory in order to accelerate loading when undoing or redoing actions in the score. Each model has two fields titled `process` and `kill_process`. These fields serve to mark the lifespan of the specific model. Figure 3 and table 1 show an example of the lifecycle of models in Thema.

4. DATASET

In order to establish a dataset for studying compositional process data collected in Thema, a study was conducted at New York University with ten graduate-level composers creating piano pieces using the software. Subjects were

[3] For a useful discussion of the virtues of immutability, see [15].

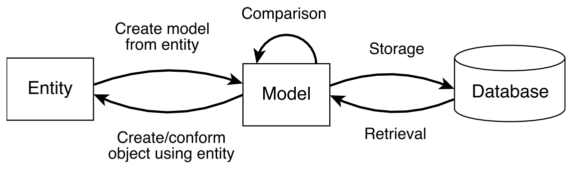


Figure 2. Entities, models, and the database

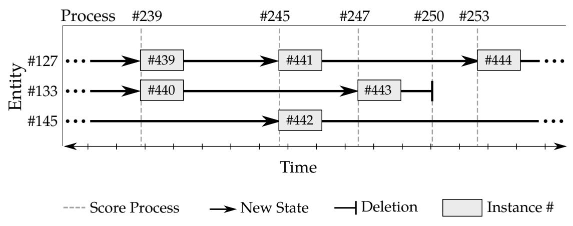


Figure 3. Timeline

introduced to the software via a tutorial session, and were then asked to notate a brief excerpt from Bartok's *Mikrokosmos*; the excerpt was selected because it would require users to perform a variety of tasks, handling time-signature changes, articulations, and multiple voices within a staff. Following this, the remainder of the four-hour session was spent composing a short piano piece for an intermediate-level performer in a style of the composer's choosing. This controlled setting ensured that participants had access to a full-size MIDI keyboard and were able to receive technical support if they had questions about the software. While this arrangement is not ideal from the standpoint of naturalism, it ensured that the data was captured reliably and that composers were able to use the program, and the knowledge gained will allow for future, less-restrictive studies to proceed. Though the composers had relatively little time to work with the software, all of the composers were able to complete the study. At the end of the session, the composers provided a segmentation of the work, indicating major sections, as well as any smaller subdivisions. The composers were compensated for their time and agreed to release their work and data under a Creative Commons license with the option of being attributed for their work or remaining anonymous. [4]

In addition to capturing quantitative data, Thema also recorded screen captures for every entry in the process table. The bounding-box coordinates for currently visible notes, dynamics, and articulations were also stored into a table in the database. This makes it possible to create graphical overlays on the score images without having to use the program itself, and provides a simple means of rapidly browsing through past states of the score. Non-linear browsing and montaging can be realized via database queries, e.g., "select all activity within a two-minute window of a clipboard operation" or "show all versions of the selected passage."

[4] The terms of the license are available at `https://creativecommons.org/licenses/by-nc-sa/4.0/`

entity	*id*	*process*	*kill_process*
127	439	239	245
133	440	239	247
127	441	245	253
145	442	245	null
133	443	247	250
127	444	253	null

Table 1. Entity states from fig. 3 as represented in the database.

5. VISUALIZATIONS

Thema contains a suite of visualizations in a variety of formats for examining compositional data. These visualizations can be synchronized together via a central time slider. For example, one window might contain the score at the time indicated by the slider, while another window displays the structure of the score over the course of a two hour sliding window, and a third window contains a score which displays incoming pitches from the MIDI keyboard in a two minute window. Each window can also contain multiple graphical overlays, such as, for example, showing the location in the score and duration of playback superimposed on the long-term structural view. Thema also features an API for developing user-defined graphical overlays.

The score overlay section of the API allows programmers to access the drawing subroutines for notes in the score. Figure 4 shows a heat map of the composer's edit activity superimposed on the score.

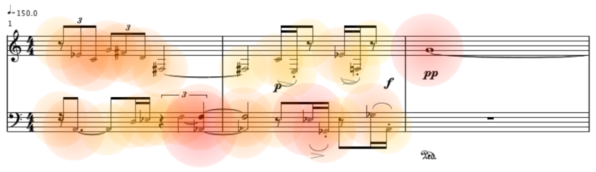


Figure 4. Heatmap Overlay

In figure 5, the arrows between pairs of notes indicate pairs of notes that were edited in close time proximity to each other. The weight of the line is proportional to the number of times they were edited as well as how closely in time they were edited, with simultaneous edits producing thicker lines. As can be seen, the notes in the first two measures are densely connected to each other; they are also connected to the previous (unseen) measure. The notes in the last measure, however, are not connected to the notes in the prior measures, indicating that they were never edited in close time proximity to the notes in the second measure.

This location is also one of the major boundaries in the score indicated by the composer.

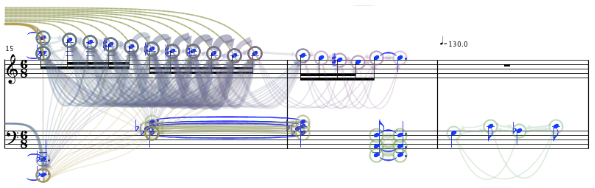


Figure 5. Edit Time Connections Overlay

Observations such as this inspired the development of a novel pitch-agnostic boundary detection algorithm, described in [14, p.183-190]. The algorithm operates on the premise that low-level musical boundaries parallel boundaries in the composer's process as represented by edit times; similarly, areas of musical continuity in the score are more likely to be closely related in terms of editing time. While the premise is naive, when tested against the boundaries indicated by the composers, the algorithm achieved surprisingly respectable results. Future work will address the algorithm and its parameters in depth, and compare the segments found via this method against those found by content-based algorithms.

6. CONCLUSIONS AND FUTURE WORK

The sample size from the NYU study is small, but it demonstrates that Thema can be an effective tool for research. More data in this area is needed, particularly in tandem with current qualitative methods. The planned public release of the software in the Fall of 2015 will allow composers to experiment with the program over time and in a naturalistic setting, with the option of sharing their data with researchers. Development is also underway on a wrapper written in Python to convert Thema's data into Music21 streams and this will provide access for other researchers in computational musicology.

Low-level musical behavior has not as yet received much attention in the compositional process literature, and it is hoped that this tool will provide a new means for studying it and, in so doing, allow compositional process research to connect to existing research in computational musicology.

Acknowledgments

Thanks to Robert Rowe, Juan Pablo Bello, Elizabeth Hoffman, Panayotis Mavromatis, and Sarah Marlowe for their help, as well as to the composers who gave of their time and talents in the study.

7. REFERENCES

[1] D. Collins, "Real-time tracking of the creative music composition process," *Digital Creativity*, vol. 18, no. 4, pp. 239–256, 2007.

[2] ——, "A synthesis process model of creative thinking in music composition," *Psychology of Music*, vol. 33, no. 2, pp. 193–216, 2005.

[3] N. Donin, "Empirical and historical musicologies of compositional processes: Towards a cross-fertilisation," in *The Act of Musical Composition : Studies in the Creative Process*, D. Collins, Ed. Ashgate Publishing, 2012, ch. 1, pp. 1–26.

[4] B. Truax, "For Otto Laske: A communicational approach to computer sound programs," *Journal of Music Theory*, vol. 20, no. 2, pp. 227–300, 1976.

[5] M. Hickey, "Qualitative and quantitative relationships between children's creative musical thinking processes and products," Ph.D. dissertation, Northwestern University, 1995.

[6] C. Nash, "Supporting virtuosity and flow in computer music," Ph.D. dissertation, University of Cambridge, 2012.

[7] J. Peterson and E. Schubert, "Music notation software: Some observations on its effects on composer creativity," *Proceedings of ICoMCS December*, p. 127, 2007.

[8] J. Peterson, "Computer notation-based music composition and the delayed introduction of musical expression markings," *Journal of Education, Informatics and Cybernetics*, vol. 1, no. 3, pp. 37–41, 2008.

[9] B. Eaglestone, N. Ford, G. J. Brown, and A. Moore, "Information systems and creativity: an empirical study," *Journal of Documentation*, vol. 63, no. 4, pp. 443–464, 2007.

[10] S. Abrams, R. Bellofatto, R. Fuhrer, D. Oppenheim, J. Wright, R. Boulanger, N. Leonard, D. Mash, M. Rendish, and J. Smith, "QSketcher: an environment for composing music for film," in *International Computer Music Conference*, 2001.

[11] N. Didkovsky and P. L. Burk, "Java music specification language, an introduction and overview," in *Proceedings of the International Computer Music Conference (ICMC)*. Havana: International Computer Music Association, 2001, pp. 123–126.

[12] M. Kuuskankare and M. Laurson, "Expressive notation package," *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[13] A. Agostini and D. Ghisi, "Bach: an environment for computer-aided composition in max," in

Proceedings of the International Computer Music Conference. Ljubljana, Slovenia: ICMC, 2012, pp. 373–378. [Online]. Available: http://quod.lib.umich.edu/cgi/p/pod/dod-idx/bach-an-environment-for-computer-aided-composition-in-max.pdf?c=icmc;idno=bbp2372.2012.068

[14] P. McCulloch, “Thema: A software framework for the quantitative study of compositional process,” Ph.D. dissertation, New York University, 2014.

[15] R. Hickey. (2009) Are we there yet? a deconstruction of object-oriented time. [Online]. Available: http://wiki.jvmlangsummit.com/images/a/ab/HickeyJVMSummit2009.pdf

STANDARD MUSIC FONT LAYOUT (SMuFL)

Daniel Spreadbury
Steinberg Media Technologies GmbH
d.spreadbury@steinberg.de

Robert Piéchaud
robert.piechaud@gmail.com

ABSTRACT

Digital typefaces containing the symbols used in Western common music notation have been in use for 30 years, but the development of the repertoire of symbols that are included, their assignment to code points, and design considerations such as glyph metrics and registration, have been rather *ad hoc*. The Standard Music Font Layout (SMuFL) establishes guidelines for all of these areas, and a reference implementation is available in the Bravura font family.

Software developers and font designers alike are beginning to develop implementations of SMuFL in their products, and benefits including easier data interchange, interoperability of fonts with a variety of software packages, are already being felt.

1. A BRIEF HISTORY OF MUSIC FONTS

Computer software has been displaying musical symbols of various kinds since the 1960s, but the first font for musical symbols did not arrive until 1985, when Cleo Huggins designed Sonata for Adobe.[1]

Sonata mapped the musical symbols onto keys on the standard QWERTY keyboard, using some simple mnemonics (the treble G clef, for example, was mapped onto the **&** key, and the sharp sign onto **#**). Most music fonts developed since then, including Steve Peha's Petrucci (the first music font for the commercial scoring application Finale, dating from 1988[2]) and Jonathan Finn's Opus (the first music font for the commercial scoring application Sibelius, dating from 1993), have followed Sonata's layout.

However, since Sonata includes fewer than 200 characters, and even conventional music notation[3] requires many more symbols than that, individual vendors have devised their own mappings for characters beyond Sonata's initial set.

By 2013, for example, the Opus font family that is still Sibelius's default font set contains no fewer than 18 fonts with more than 600 characters between them.

In 1998, Perry Roland of the University of Virginia drafted a proposal for a new range of musical symbols to be incorporated into the Unicode Standard.[4] This range of 220 characters was duly accepted into the Unicode Standard, and is found at code points U+1D100–U+1D1FF.[5] However, its repertoire of 220 characters does not extend dramatically beyond the scope of the original 1985 version of Sonata, though it does add some characters for mensural and Gregorian notation.

To date the only commercially available music font that uses the Unicode mapping is Adobe Sonata Std, and its repertoire is incomplete.

The designers of other music applications have developed their own approaches to laying out music fonts that are incompatible with both the Sonata-compatible approach, and the Unicode Musical Symbols range. In short, existing standards are either *ad hoc* or insufficient for the development of fonts for rich music notation applications.

2. GOALS FOR A NEW STANDARD

Steinberg began work on a new scoring application at the start of 2013, and quickly identified both the need for a new music font, and the lack of an adequate standard for the layout and design of such a font.

Surveying a range of commercial, open source and freeware music fonts from a variety of sources, and

[1] See `http://www.identifont.com/show?12A`

[2] See `http://blog.finalemusic.com/post/2010/02/18/Meet-Steve-Peha-creator-of-Petrucci-Finales-first-music-font.aspx`

[3] A term coined by Donald Byrd, Senior Scientist and Adjunct Associate Professor of Informatics at Indiana University.

[4] The original proposal is no longer available, but an archived version can be found at `http://archive.is/PzkaT`

[5] See `http://www.unicode.org/charts/PDF/-U1D100.pdf`

considering the needs of the in-development application, provided the impetus to create a new standard, with the following goals identified from the outset:

2.1 Extensible by design

The existing Unicode Musical Symbols range is a fixed set of 220 characters in a fixed range of 256 code points at U+1D100–U+1D1FF. This range is not easily extensible, though of course it would be possible for one or more non-contiguous supplemental ranges to be added to future versions of Unicode.

Sonata pre-dates the introduction of Unicode: in common with other PostScript Type 1 fonts of its age, it uses an 8-bit encoding that limits its repertoire of glyphs to a maximum of 256 within a single font. Fonts that broadly follow a Sonata-compatible layout are therefore likewise limited to a maximum of 256 glyphs, and as their developers have needed to further expand their repertoire of characters, they have unilaterally added separate fonts, with no agreement about which characters should be included at which code points.

A new standard should be extensible by design, such that even if the repertoire of characters needs to expand, there is both a procedure for ratifying the inclusion of new characters into the standard, and a means for individual font designers or software developers to add glyphs for their own private use in a way that does not break the standard for other users.

2.2 Take advantage of modern font technologies

The development of the Unicode standard and the OpenType font specification, and their adoption by operating system, software, and font developers, are both enormously important: Unicode provides categorization and structure to the world's language systems, while OpenType enables the development of more advanced fonts with effectively unlimited glyph repertoires and sophisticated glyph substitution and positioning features.

A new standard should enable software developers and font designers to build software that takes advantage of these features, without tying the standard to a specific set of technologies, so that it is as broadly applicable and resistant to future obsolescence as is practical to achieve.

2.3 Open license

In order to minimize the number of obstacles for software developers and font designers to adopt the new standard, it should be free of onerous software licensing terms.

A new standard should be released under a permissive, open license that both protects Steinberg's copyright in the standard, but makes it free for anybody to use in whole or in part in any project, whether that project itself is made available on a commercial basis or under a permissive or free software license.

Accordingly, Steinberg has released SMuFL under the MIT License,[6] which is a permissive free software license that allows reuse within both proprietary and open source software.

2.4 Practical and useful

Although it is impossible to say with certainty why the Unicode Musical Symbols range has failed to gain support among software developers and font designers, it is reasonable to assume that the range did not sufficiently solve the existing problems with the *ad hoc* Sonata-compatible approach, perhaps most crucially the lack of extensibility afforded by the limit of 220 characters, which represented only a very modest expansion of the 176 characters present in Sonata.

A new standard should not only be extensible, but should be developed with the practical needs of software developers and font designers as the top priority, including providing detailed technical guidelines on how to solve some of the issues inherent in representing music notation using a combination of glyphs drawn from music fonts and drawn primitive shapes (stroked lines, filled rectangles, curves, etc.).

2.5 Facilitate easier interchange

As existing music fonts have been developed in isolation by independent software developers and font developers, despite broad intent to make it possible for end users of scoring programs to use a variety of fonts, including those designed for other applications, in practice the level of compatibility between fonts and scoring programs is rather low.

A comparison of the repertoire of glyphs in Sonata, Petrucci, and Opus shows that only 69 of 176 glyphs in Sonata are also present in both Petrucci and Opus; a further 38 glyphs are present in Sonata and Petrucci, but not Opus; and a further 5 glyphs are present in Petrucci and Opus, but not Sonata; a further 59 glyphs in Sonata are present in neither Opus nor Petrucci.

Furthermore, there is no practical way for an end user to know in advance of attempting to use a different font whether or not a given range of characters is implemented in that font, and when transferring documents created in software between systems there is little guarantee that the software can translate the required glyph from one font to another.

A new standard should improve the compatibility of music fonts between different systems by providing not only an agreed mapping of characters to specific code

[6] See `http://opensource.org/licenses/MIT`

points, but also a means for font designers to describe programmatically the repertoire of characters implemented in a given font.

2.6 Build community support

The range of symbols used in Western music notation is so deep and broad that it is difficult for any individual person or small group to have sufficient knowledge to correctly identify and categorize the characters. Furthermore, without broad support among software developers and font designers, any new standard is destined to languish unused.

A new standard should be developed in the open, inviting interested parties to contribute ideas and discussion to the development of the repertoire of characters, their categorization, and technical recommendations about font design, glyph metrics, and glyph registration.

3. NON-GOALS FOR A NEW STANDARD

At the outset of the project, it was determined that, in the short- to medium-term at least, targeting ratification of the new standard by the Unicode Consortium in order to broaden the range of musical symbols encoded by Unicode was not a goal of the project. Developing the standard independently, away from the more rigorous requirements of the proposal and review process, gives greater agility and faster iteration as new requirements emerge.

Initially it was also determined that attempting to develop a set of recommendations for fonts to be used inline with text fonts in word processing or page layout software would be too much work to undertake right away, in addition to the core goal of developing recommendations for fonts to be used in specialized music notation software. However, after the launch of the new standard at the Music Encoding Conference in Mainz, Germany in May 2013, the members of the nascent community identified this as a high priority activity, and the development of guidelines for fonts to be used in text-based applications was added as a requirement for the first stable release of the new standard.

4. WHAT IS SMUFL?

The Standard Music Font Layout, or SMuFL (pronounced “smoofle”), provides both a standard way of mapping music symbols to the Private Use Area of Unicode’s Basic Multilingual Plane, and a detailed set of guidelines for how music fonts should be built.

As a consequence of the joint effort of the community that has arisen around the development of the standard, it also provides a useful categorization of thousands of symbols used in Western music notation.

4.1 Character repertoire and organization

The initial public release of SMuFL, version 0.4, included around 800 characters. By the time of the release of version 1.0, in June 2014, the total number of characters included had grown to nearly 2400, organized into 104 groups.

SMuFL makes use of the Private Use Area within Unicode’s Basic Multilingual Plane (code points from U+E000–U+F8FF). The Unicode standard includes three distinct Private Use Areas, which are not assigned characters by the Unicode Consortium so that they may be used by third parties to define their own characters without conflicting with Unicode Consortium assignments.

SMuFL is a superset of the Unicode Musical Symbols range, and it is recommended that common characters are included both at code points in the Private Use Area as defined in SMuFL and in the Unicode Musical Symbols range.

The groups of characters within SMuFL are based on the groupings defined by Perry Roland in the Unicode Musical Symbols range, but with finer granularity. There are currently 108 groups, proceeding roughly in order from least to most idiomatic, i.e. specific to particular instruments, types of music, or historical periods. The grouping has no significance other than acting as an attempt to provide an overview of the included characters.

Groups are assigned code points in multiples of 16. Room for future expansion has, where possible, been left in each group, so code points are not contiguous. The code point of each character in SMuFL 1.0 is intended to be immutable, and likewise every character has a canonical name, also intended to be immutable. Since the release of SMuFL 1.0, a few additional characters have already been identified that should be added to groups that were already fully populated, and, in common with the approach taken by the Unicode Consortium, new supplemental groups have been added at the end of the list of existing groups to accommodate these additions.

4.2 Inclusion criteria

No formal criteria have been developed for whether or not a given character is suitable for inclusion in SMuFL. Members of the community make proposals for changes and additions to the repertoire of characters, giving rise to public discussion, and once consensus is reached, those changes are made in the next suitable revision.

In general a character is accepted if it is already in widespread use: although composers and scholars invent new symbols all the time, such a symbol can only be included in SMuFL if there is broad community support.

4.3 Recommended and optional glyphs

One of the aims of SMuFL is to make it as simple as possible for developers both of fonts and of scoring software to implement support for a wide range of musical symbols. Although modern font technologies such as OpenType enable a great deal of sophistication in automatic substitution features, applications that wish to use SMuFL-compliant fonts are not obliged to support advanced OpenType features.

The basic requirements for the use of SMuFL-compliant fonts are the ability to access characters by their Unicode code point, to measure glyphs, and to scale them (e.g. by drawing the font at different point sizes). If applications are able to access OpenType features such as stylistic sets and ligatures, then additional functionality may be enabled.

However, all glyphs that can be accessed via OpenType features are also accessible via an explicit code point. For example, a stylistic alternate for the sharp accidental designed to have a clearer appearance when reproduced at a small size can be accessed as a stylistic alternate for the character **accidentalSharp**, but also by way of its explicit code point, which will be in the range U+F400–U+F8FF.

Because optional glyphs for ligatures, stylistic alternates, etc. are not required, and different font developers may choose to provide different sets (e.g. different sets of glyphs whose designs are optimized for drawing at different optical sizes), SMuFL does not make any specific recommendations for how these glyphs should be assigned explicit code points, except that they must be within the range U+F400–U+F8FF, which is reserved for this purpose and for any other private use required by font or application developers.

In summary, recommended glyphs are encoded from U+E000, with a nominal upper limit of U+F3FF (a total of 5120 possible glyphs), while optional glyphs (ligatures, stylistic alternates, etc.) are encoded from U+F400, with a nominal upper limit of U+F8FF (a total of 1280 possible glyphs).

In order for a font to be considered SMuFL-compliant, it should implement as many of the recommended glyphs as are appropriate for the intended use of the font, at the specified code points. Fonts need not implement every recommended glyph, and need not implement any optional glyphs, in order to be considered SMuFL-compliant.

4.4 SMuFL metadata

To aid software developers in implementing SMuFL-compliant fonts, three support files in JSON format [1] are available.

glyphnames.json maps code points to canonical glyph names, which by convention use lower camel case, a convenient format for most programming languages. The file is keyed using the glyph names, with the SMuFL code point provided as the value for the **codepoint** key, and the Unicode Musical Symbols range code point (if applicable) provided as the value for the **alternateCodepoint** key. The **description** key contains the glyph's description.

classes.json groups glyphs together into classes, so that software developers can handle similar glyphs (e.g. noteheads, clefs, flags, etc.) in a similar fashion. Glyphs are listed within their classes using the names specified in **glyphnames.json**. Not all glyphs are contained within classes, and the same glyph can appear in multiple classes.

ranges.json provides information about the way glyphs are presented in discrete groups in this specification. This file uses a unique identifier for each group as the primary key, and within each structure the **description** specifies the human-readable range name, **glyphs** is an array listing the canonical names of the glyphs contained within the group, and the **range_start** and **range_end** key/value pairs specify the first and last code point allocated to this range respectively.

4.5 Font-specific metadata

It is further recommended that SMuFL-compliant fonts also contain font-specific metadata JSON files. The metadata file allows the designer to provide information that cannot easily (or in some cases at all) be encoded within or retrieved from the font software itself, including recommendations for how to draw the elements of music notation not provided directly by the font itself (such as staff lines, barlines, hairpins, etc.) in a manner complementary to the design of the font, and important glyph-specific metrics, such as the precise coordinates at which a stem should connect to a notehead.

Glyph names may be supplied either using their Unicode code point or their canonical glyph name (as defined in the **glyphnames.json** file). Measurements are specified in staff spaces, using floating point numbers to any desired level of precision.

The only mandatory values are the font's name and version number. All other key/value pairs are optional.

The **engravingDefaults** structure contains key/value pairs defining recommended defaults for line widths etc.

The **glyphsWithAnchors** structure contains a structure for each glyph for which metadata is supplied, with the canonical glyph name or its Unicode code point as the key, and is discussed in more detail below.

The **glyphsWithAlternates** structure contains a list of the glyphs in the font for which stylistic alternates are provided, together with their name and code point. Applications that cannot access advanced font features like OpenType stylistic alternates can instead determine the presence of an alternate for a given glyph, and its code point, using this data.

The **glyphBBoxes** structure contains information about the actual bounding box for each glyph. The glyph bounding box is defined as the smallest rectangle that encloses every part of the glyph's path, and is described as a pair of coordinates for the bottom-left (or southwest) and top-right (or northeast) corners of the rectangle, expressed staff spaces to any required degree of precision, relative to the glyph origin.

The **ligatures** structure contains a list of ligatures defined in the font. Applications that cannot access advanced font features like OpenType ligatures can instead determine the presence of a ligature that joins together a number of recommended glyphs, and its code point, using this data.

The **sets** structure contains a list of stylistic sets defined in the font. Applications that cannot access advanced font features like OpenType stylistic sets can instead determine the presence of sets in a font, the purpose of each set, and the name and code point of each glyph in each set, using this data.

4.5.1 Example of how font-specific metadata is used

Figure 1 shows how font-specific metadata may be used in conjunction with the conventions of glyph registration

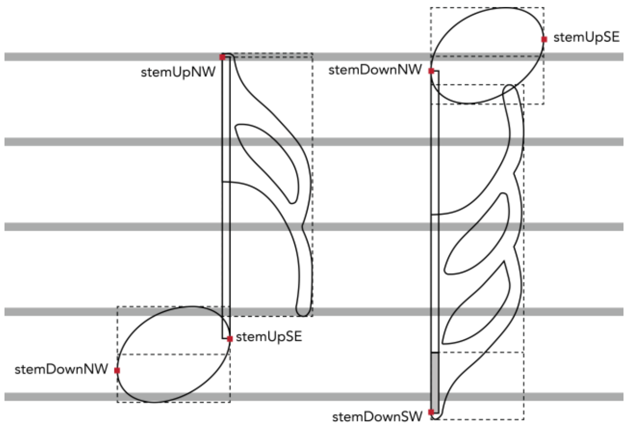


Figure 1 : Diagram illustrating how points defined in font-specific metadata can be used by scoring software.

to construct two notes: an up-stem 16th note (semiquaver), and a down-stem 32nd (demisemiquaver).

- The horizontal grey lines denote staff lines, for scale.
- The dashed boxes show glyph bounding boxes, with the left-hand side of the box corresponding to x=0, while the horizontal lines bisecting the blue boxes show the origin for each glyph, i.e. y=0.
- The shaded red boxes show the locations of the glyph attachment points, as specified in the font metadata JSON file.
- The shaded area on the down-stem note shows the amount by which a stem of standard length (i.e. the unfilled portion of the stem) should be extended in order to ensure good on-screen appearance at all zoom levels.

Note that the **stemUpSE** attachment point corresponds to the bottom right-hand (or south-east) corner of the stem, while **stemDownNW** corresponds to the top left-hand (or north-west) corner of the stem. Likewise, for correct alignment, the flag glyphs must always be aligned precisely to the left-hand side of the stem, with the glyph origin positioned vertically at the end of the normal stem length.

4.6 Glyph registration and metrics recommendations

In addition to providing a standard approach to how musical symbols should be assigned to Unicode code points, SMuFL also aims to provide two sets of guidelines for the metrics and glyph registration, addressing the two most common use cases for fonts that contain musical symbols, i.e. use within dedicated scoring applications, and use within text-based applications (such as a word processors, desktop publishers, web pages, etc.).

Since it is helpful for scoring applications that all symbols in a font be scaled relative to each other as if drawn on a staff of a particular size, and conversely it is helpful for musical symbols to be drawn in-line with text to be scaled relative to the letterforms with which the musical symbols are paired, in general a single font cannot address these two use cases: the required metrics and relative scaling of glyphs are incompatible.

Therefore, it is recommended that font developers make clear whether a given font is intended for use by scoring applications or by text-based applications by appending "Text" to the name of the font intended for text-based applications; for example, "Bravura" is intended for use by scoring applications, and "Bravura Text" is intended for use by text-based applications (or indeed for mixing musical symbols with free text within a scoring application).

The complete guidelines for key font metrics and glyph registration are too detailed to reproduce here, so they can be read in full in the SMuFL specification.[7] Those guidelines that apply to the font as a whole, rather than specific groups of glyphs, are reproduced below.

[7] See `http://www.smufl.org/download`

4.6.1 Guidelines for fonts for scoring applications

Dividing the em in four provides an analogue for a five-line staff: if a font uses 1000 upm (design units per em), as is conventional for a PostScript font, one staff space is equal to 250 design units; if a font uses 2048 upm, as is conventional for a TrueType font, one staff space is equal to 512 design units.

The origin (bottom left corner of the em square, i.e. x = 0 and y = 0 in font design space) therefore represents the middle of the bottom staff line of a nominal five-line staff, and y = 1 em represents the middle of the top staff line of that same five-line staff.

All glyphs should be drawn at a scale consistent with the key measurement that one staff space = 0.25 em.

Unless otherwise stated, all glyphs shall be horizontally registered so that their leftmost point coincides with x = 0.

Unless otherwise stated, all glyphs shall have zero-width side bearings, i.e. no blank space to the left or right of the glyph.

4.6.2 Guidelines for fonts for text-based applications

Upper case letters in a text font do not typically occupy the whole height of the em square: instead, they typically occupy around 75–80% of the height of the em square, with the key metrics for ascender and caps height both falling within this range. In order for the line spacing of a font containing music characters to be equivalent to that of a text font, its key metrics must match, i.e. the ascender, caps height and descender must be very similar. Glyphs with unusually large ascenders and descenders (such as notes of short duration with multiple flags) should not be scaled individually in order to fit within the ascender height, as they will not then fit with the other glyphs at the same point size; however, the behavior of glyphs that extend beyond the font’s ascender and descender metrics is highly variable between different applications.

Leading on from the premise that a SMuFL-compliant font for text-based applications should use metrics compatible with regular text fonts, specific guidelines are as follows:

Dividing 80% of the height of the em in four provides an analogue for a five-line staff. If a font uses 1000 upm (design units per em), as is conventional for a PostScript font, the height of a five-line staff is 800 design units, or 0.8em; therefore, one staff space height is 200 design units, or 0.2 em. If a font uses 2048 upm, as is conventional for a TrueType font, the height of a five-line staff is 1640 design units, and one staff space is 410 design units.

The origin (bottom left corner of the em square, i.e. x = 0 and y = 0 in font design space) therefore represents the middle of the bottom staff line of a nominal five-line staff, and y = 0.8 em represents the middle of the top staff line of that same five-line staff.

Unless otherwise stated, all glyphs should be drawn at a scale consistent with the key measurement that one staff space = 0.2 em.

Unless otherwise stated, all glyphs shall be horizontally registered so that their leftmost point coincides with x = 0.

Unless otherwise stated, all glyphs shall have zero-width side bearings, i.e. no blank space to the left or right of the glyph.

Staff line and leger line glyphs should have an advance width of zero, so that other glyphs can be drawn on top of them easily.

5. REFERENCE FONT

To demonstrate all of the key concepts of SMuFL, a reference font has been developed. The font family is called Bravura, and consists of two fonts: Bravura, which is intended for use in scoring applications; and Bravura Text, which is intended for use in text-based applications.

The word *Bravura* comes from the Italian word for “cleverness”, and also, of course, has a meaning in music, referring to a virtuosic passage or performance; both of these associations are quite apt for the font. From an aesthetic perspective, Bravura is somewhat bolder than most other music fonts, with few sharp corners on any of the glyphs, mimicking the appearance of traditionally-printed music, where ink fills in slightly around the edges of symbols, and the metal punches used in plate engraving lose their sharp edges after many uses. A short musical example set in Bravura is shown below (Figure 2).

Steinberg has released the Bravura fonts under the SIL Open Font License [2]. Bravura is free to download, and can be used for any purpose, including bundling it with other software, embedding it in documents, or even using it as the basis for a new font. The only limitations placed on its use are that: it cannot be sold on its own; any derivative font cannot be called “Bravura” or contain “Bravura” in its name; and any derivative font must be released under the same permissive license as Bravura itself.

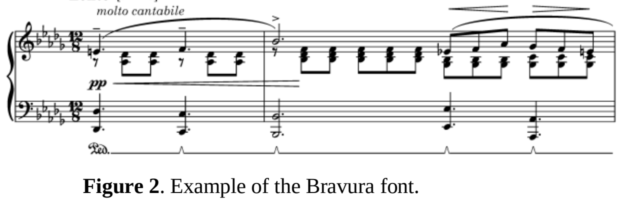


Figure 2. Example of the Bravura font.

6. IMPLEMENTATION CASE STUDY: THE NOVEMBER FONT

Unlike designers of text fonts, music font designers have historically had great freedom, which has been both a blessing and a curse. Before SMuFL, while there was some common sense about what the kernel of music symbols should be (clefs, noteheads, accidentals, etc.), the actual position of characters in the font, their naming (though there was generally none provided), and the addition of rarer symbols beyond the basic set was left up to the designer's imagination and to some specific requirements of the target music notation software.

Things are changing for the font designer with SMuFL as its main goal is to address the issues of symbol position, naming and repertoire in a universal way. SMuFL is a great source of inspiration for the designer – surely one of its benefits – but it also imposes new constraints and requirements, and leads to a more demanding design workflow.

6.1 The November Font – Summary

The November music font was designed in 1999 specifically for the software Finale, and its repertoire of 330 characters, spread over two font files, ranging through historical periods spanning the Renaissance to the 20th century *avant garde*, was considered large at that time. Before SMuFL, the extension of November's repertoire had often been considered, but it would have most likely led to the multiplication of font files, as had occurred with, for example, Opus or Maestro, which the designer was reluctant to do, and consequently only small updates had been made over the years.

6.2 Moving to SMuFL

The emergence of SMuFL back in 2013 was a great opportunity for November to make a bigger jump: one single font file with a greatly extended range of characters, wrapped in OpenType, and complying with a new standard.

By switching to SMuFL, the font designer, who generally is a single individual, must be ready to face the temptation of adding more and more symbols, making the development process potentially much longer.[8] And not only must the designer deal with thousands of vectors and points, but also to some extent he or she must turn into a programmer. Python scripting, for instance, can be a great ally for generating the required metadata automatically; this was used extensively for the November 2.0 project.[9] For SMuFL-scaled font projects, it is impractical to create those metadata manually, and,

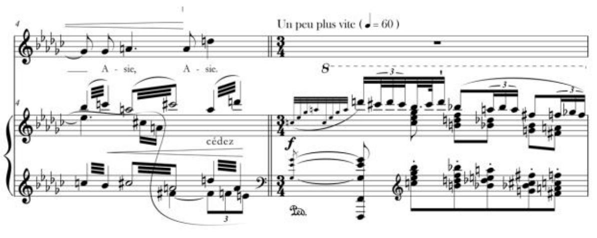


Figure 3. Example of the November 2.0 font.

to make the design workflow even better, one can invent sophisticated tools, for instance to compare the font being crafted with the reference font, Bravura. All of these considerations change the font development workflow deeply.

November 2.0, released in February 2015, now has over 1100 characters, with about 80% of them coming from the SMuFL specifications, and is the first commercially-released font to comply with SMuFL. A short musical example set in November 2.0 is shown below (Figure 3).

6.3 Compatibility with existing scoring software

Unlike the font Bravura, which for now has largely served as a reference font for SMuFL, commercial SMuFL-compliant music fonts are intended to be used in existing music notation programs.

At the present time, no currently available notation software officially directly supports SMuFL, though such support is likely forthcoming in the future. In the short- to medium-term, therefore, a SMuFL-compliant font like November 2.0 must still be packaged specifically for each notation program. The SMuFL metadata, for instance, is currently not consumed at all by any of the major existing applications (including Finale, Sibelius, and LilyPond), and idiosyncratic component files[10] must be supplied along with the font in order to ensure a smooth user experience.

But in a positive way, the claim of SMuFL-compliance for a popular music font like November can potentially help serve as an impetus for the developers of music notation software to support SMuFL more quickly.

7. SUPPORT FOR SMUFL

SMuFL 1.0 was released in June 2014. The standard remains under active development, and it is hoped that an increasing number of software developers and font designers will adopt it for their products. Below is a

[8] Somehow the designer could not resist this temptation with November 2.0 in any case!

[9] November 2.0 was made with the open source program FontForge (`http://fontforge.github.io/`), which has a powerful Python interface.

[10] Finale's Font Annotations and Libraries, Sibelius's House Styles, LilyPond's snippets…

summary of the projects that have been publicly announced that are making use of SMuFL.

7.1 Software with SMuFL support

Steinberg's forthcoming scoring application will support SMuFL-compliant fonts.

The open source scoring application MuseScore supports SMuFL-compliant fonts in version 2.0, which is currently in beta testing.[11]

The web browser-based interactive sheet music and guitar tablature software Soundslice uses SMuFL and Bravura for its music notation display.[12]

The open source Music Encoding Initiative (MEI) rendering software, Verovio, also uses SMuFL for its music notation display.[13]

The commercial scoring application Finale, from MakeMusic Inc., will support SMuFL in a future version[14]. MakeMusic's MusicXML import/export plug-in for Finale, Dolet, supports SMuFL as of version 6.5.[15]

The commercial digital audio workstation application Logic Pro X, from Apple Inc., supports SMuFL and is compatible with Bravura from version 10.1.[16]

7.2 Fonts with SMuFL support

In addition to the reference font Bravura, other SMuFL-compliant music fonts are beginning to be available.

MuseScore 2.0 includes SMuFL-compliant versions of Emmentaler and Gootville, based respectively on the Emmentaler and Gonville fonts designed for use with LilyPond.

Verovio includes a SMuFL-compliant font called Leipzig.

Robert Piéchaud has designed an updated version of his November font family that is SMuFL-compliant[17].

8. FUTURE DIRECTIONS

Although SMuFL has reached version 1.0 and contains an enormous range of characters, it remains under active development, and further minor revisions are expected for the indefinite future as new characters are identified, proposed, and accepted for inclusion, and as the need for new or improved metadata is identified.

It is also expected that MusicXML, a widely-used format for the interchange of music notation data between software of various kinds, will develop closer ties to SMuFL in its next major revision, version 4.0, which may necessitate some changes to SMuFL.

9. CONCLUSIONS

In this paper, a new standard for the layout of musical symbols into digital fonts has been outlined. The new standard, called the Standard Music Font Layout (SMuFL) is appropriate for modern technologies such as Unicode and OpenType. Through community-driven development, the standard has reached version 1.0 and includes nearly 2400 characters, categorized into 104 groups, and is poised for future expansion as necessary. A reference font family, Bravura, has been developed to promote the adoption of the new standard. Both SMuFL and Bravura are available under permissive free software licenses, and are already being adopted by software developers and font designers.

Acknowledgements

SMuFL is developed in the open by a community of music software developers, academics, font designers, and other interested parties. Too many people to list here have contributed to the development of the standard to date, and their contributions have been of great value to the project.[18]

10. REFERENCES

[1] ECMA-404, *The JSON Data Interchange Format, 1st edition*, 2013.

[2] N. Spalinger and V. Gaultney, *SIL Open Font License (OFL)*, 2007.

[11] See http://musescore.org/en/node/30866

[12] See http://www.soundslice.com

[13] See https://rism-ch.github.io/verovio/smufl.xhtml?font=Leipzig

[14] See http://www.sibeliusblog.com/news/finale-2014d-and-beyond-a-discussion-with-makemusic/

[15] See http://www.musicxml.com/dolet-6-5-finale-plugin-now-available/

[16] See http://support.apple.com/en-us/HT203718

[17] See http://www.klemm-music.de/notation/november/index.php

[18] A full list of contributors is printed in the SMuFL specification, which can be found at http://www.smufl.org/download

SVG TO OSC TRANSCODING: TOWARDS A PLATFORM FOR NOTATIONAL PRAXIS AND ELECTRONIC PERFORMANCE

Rama Gottfried
Center for New Music and Audio Technolgies (CNMAT)
University of California, Berkeley
rama.gottfried@berkeley.edu

ABSTRACT

In this paper we present a case study for the creation of an open system for graphically developing symbolic notation which can function both as professional quality print or online documentation, as well as a computer performable score in electro-acoustic music and other computer aided contexts. Leveraging Adobe Illustrator's graphic design tools and support for the Scalable Vector Graphics (SVG) file format, the study shows that SVG, being based on Extensible Markup Language (XML), can be similarly used as a tree-based container for score information. In the study, OpenSoundControl (OSC) serves as middleware used to interpret the SVG representation and finally realize this interpretation in the intended media context (electronic music, spatial audio, sound art, kinetic art, video, etc.). The paper discusses how this interpretive layer is made possible through the separation of visual representation from the act of rendering, and describes details of the current implementation, and outlines future developments for the project.

1. BACKGROUND

The twentieth century was a time full of notational experimentation and development. Due to the explosion of performance technologies, new materials entered the scope of composition that previously were considered outside the realm of "music." [1] Works dealing with complex rhythmic and microtonal inflections, chance, and stochastic processes each exposed new compositional parameters that where not easily addressed by traditional music notation. Developments in purely mechanical music production such as piano rolls, automata, optical synthesizers, and eventually digital audio workstations allowed composers to precisely sculpt the resulting sounds by manipulating the many parameters of sound creation. [2, 3]

In some works, such as Stockhausen's *Plus-Minus*, Feldman's *Projection* series, Riley's *In C*, and others [1], the score is designed as a description of structural processes which require the performer to interpret the instructions and create their own performance score. In other works, such as Cardew's *Treatise*, the score contains no instruction, but only graphic symbols to be interpreted by the performer. Some, like Lachenmann, began to draw from tablature notation where the score describes the actions of the performers on their instruments rather than the results, while others like Kagel began to compose physical spatial movements. Similarly, in the dance world choreographers were composing movements based on the work of Laban and others [4].

Key to all of the above works is that they were all conceived and printed on paper and were designed to be read and performed by humans. Thus, the symbolic information contained in them is expressive to a human intelligence, who then performs their interpretation with an instrument or physical action. In the fields of electronic music, kinetic, video, and other types of mechanical and digital arts, the output of the instrument is via an electronically mediated system, or *rendering*, which often is unnotated [5]. With the ubiquity of powerful personal computers we now have immense rendering capabilities at our fingertips, along with many specialized tools to control these various media. These systems provide new ways for artists to incorporate many new types of media into their practice that were not previously available, however there remains a dearth of symbolic notation tools to compose with these new medias while still providing the richness of symbolic representation designed to be interpreted by a human intelligence.

The following study looks at what a more open framework for notation might look like for composing and performing scores designed for new and existing types of rendering contexts.

[1] Including Mozart's *Musikalisches Würfelspiel*

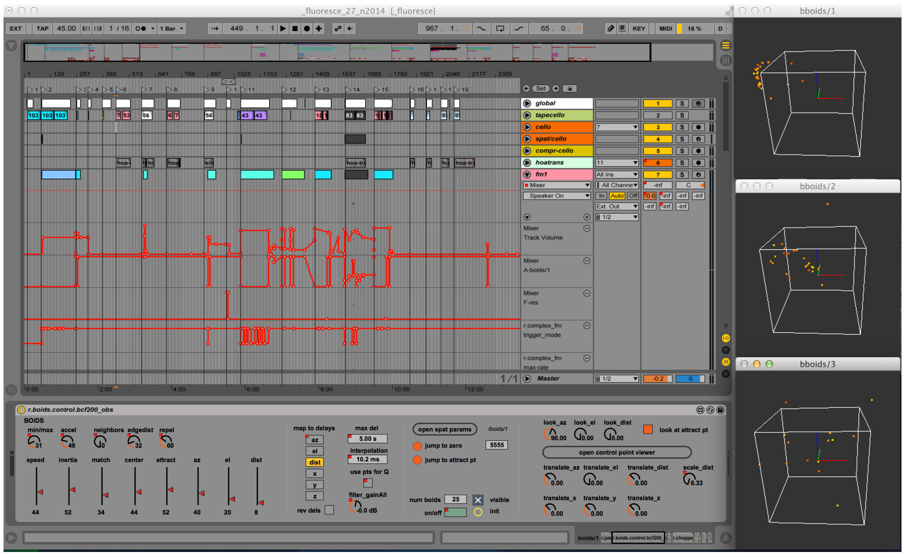


Figure 1. Screenshot from Max for Live environment showing breakpoint functions used to control flocking behaviors of virtual sources in a piece using spatial audio. On the right side of the screen are OpenGL visualization of point locations generated by the many parameters contained within the algorithm. In a more symbolic notation environment, the parameters of the resulting rendering would be representational of their function.

2. CONTEXTUAL EXAMPLE: COMPOSING FOR SPATIAL AUDIO SYSTEMS

Mixed works for live acoustic instruments and real-time spatial audio rendering systems present significant challenges for relating the graphic visualizations of spatial processing with the traditional musical notations used in a score.

For example, Ircam's "Spat" [2] MaxMSP [3] library for spatial audio processing comes equipped with several useful forms of representation for visualizing spatial processing. The simplest visualization tools in Spat display the placement of a point source in two dimensions, viewed from either "top" (XY) or "front" (XZ) vantage points. The 2D representation makes the location of points very clear and minimizes occlusion issues. For 3D visualization Spat's viewer may be easily integrated with OpenGL tools in Max's Jitter library. These are valuable methods for visualizing and developing a conceptual basis for spatial design, but as interactive graphic user interfaces they are time-variant, and so do not provide a clear mechanism for relating spatial processing with score notated actions to be performed by the instrumentalist.

Part of this problem is that there is no widely adopted notation system for incorporating spatial movement within a musical score. Interactive UIs are an intuitive way to experiment and learn the expressive capabilities in new media contexts, however when seeking to compose temporal structures, time must be represented as well. Traditional music notation symbolically represents the articulation of sound over metric time, and composers trained in this tradition learn to silently hear through the spatial organization of symbols on paper. Similarly, we might develop inner spatial perception by drawing from a long history of dance notation to describe spatial movement [4], which could be used to control a rendering system like Spat. What is needed is a symbolic graphic environment to explore ways of composing for these new types of media contexts, we have many new tools for controlling media, but very few ways of utilizing symbolic notation practice in these contexts.

2.1 Perceptual representations and breakpoint functions

Composition for spatial processing systems typically occurs in media programming environments or digital audio workstations, where the compositional approach must be contextualized within the types of controls provided by the software tools. As with interactive UIs, real-time processing is time-variant, and so the control parameters of a given process need to be contextualized in time if a score is desired [4]. This type of system has a natural affordance towards the triggers and breakpoint functions, which are extremely useful for fine control over the movement of one value (Y) over time (X).

This 2D representation, however, requires our spatial perception to be fragmented into three separate parameters (X-Y-Z or azimuth-elevation-distance). Working in this perforated situation the user must compose each parame-

[2] `http://forumnet.ircam.fr/product/spat/`
[3] `https://cycling74.com`

[4] Keeping in mind that the score does not necessarily need to describe all events as "fixed" in time.

ter individually, so there is a natural tendency to focus on a smaller number of dimensions (e.g. a tendency to focus on azimuth over distance). This computationally friendly representation comes at the expense of a more intuitive data manipulation, which is always one step removed in univariate control over multi-dimensional spaces. Figure 1 shows an example of this, where many lines of automation are composed to describe the spatial behavior of the three dimensional space shown on the right.

The strength of a well-developed notation system is in the way layers of contextual meanings are signaled by a combination of symbols. For example, a staccato dot above a note head is immediately heard and physically felt in the mind of a musician. There is an interpretive act that accompanies a notation symbol that is bound up in a cultural history of practice and experience. This interpretive act may also be present for electronic musicians who have worked with breakpoint functions for many years, however, the breakpoint function is fundamentally a concrete control over a *single* parameter as a function of time, where a symbolic representation is an aggregate of *many* parameters, functioning through abstract, contextual implications for how it should be interpreted.

In order to take advantage of the expressivity contained in symbolic notation into other media contexts, we need a way to experiment with different notational systems and strategies outside of music notation.

3. WHY NOT USE MUSIC NOTATION SOFTWARE?

Software has built-in affordances that simplify certain uses, while making other approaches more difficult [6]. As music notation has become increasingly digitalized over the last 20 years, software applications designed for music notation have also become increasingly specialized tools focusing on a specific context at the exclusion of others. Music notation software tools expose the author(s)'s idea of what "music" is, through the types of functions they provide their users.

The most used music notation programs, Finale [5], LilyPond [6], NoteAbility [7], and Sibelius [8], all target the production of traditional music scores, and also provide mechanisms for MIDI playback of these scores. Many of these applications provide APIs for manipulating the musical parameters computationally, in particular LilyPond's Scheme based scripting language.

As we have been discussing, traditional notation was not designed to handle sound's relationship to advanced instrumental techniques, spatial audio, dance, installation art, and so on. Most traditional notation software has predefined interpretations of the symbolic information contained in the score, a prime example being the ubiquitous MusicXML [9] format, which is designed specifically for "music," and so does not provide an optimal encoding for symbolic notation for contexts traditionally thought of as outside "music." While most of the above musical notation programs *do* allow the user to create custom symbols, the user is bound to an underlying assumption that the score is either to be read only by humans or to be performed as MIDI within a specifically musical context.

Computer aided composition tools such as Abjad [10], Bach [11], INScore [12], MaxScore [13], OpenMusic [14], and PWGL [15], provide environments for algorithmically generating musical scores, as well as providing connectivity to the types of new media outputs mentioned above. However, these tools rely on text input or visual programming, requiring the artist to formalize their thought process to function within the confines of a computational structure. In some cases basic drawing tools are available, however they are limited in flexibility.

Other experimental notation programs (e.g. GRM's Acousmographe [7], IanniX [16], MaxScore's Picster [17], Pure Data's Data Structures [8], etc.) provide new ways of performing graphic information, but they also contain symbolic limitations, which are not found in a graphic design environment, either through a forced method of playback, or through a limitation of graphic flexibility. Thus, at the moment, purely graphic design tools seem to provide a more flexible option for developing – and composing with – appropriate notation systems. For this reason, many contemporary composers use Adobe Illustrator for creating their scores.

The UPIC (Unit Polyagogique Informatique CEMAMu) project was one of the first to connect the act of human drafting with digital sound resources [9]. This integration of the drawing gesture is related to the working method in discussion here, the design of the UPIC was however very much tied to specific rendering contexts (amplitude

[5] `http://www.finalemusic.com`

[6] `http://www.lilypond.org`

[7] `http://debussy.music.ubc.ca/NoteAbility` – note: NoteAbility provides some support for communication with MaxMSP through the use of breakpoint functions and qlist max messages, as well as an option to export to Antescofo (`http://repmus.ircam.fr/antescofo`) score following format. This is useful for traditional music, but does not solve the problem of symbolic notation of the processes occurring in Max.

[8] `http://www.sibelius.com`

[9] `http://www.musicxml.com`

[10] `http://abjad.mbrsi.org`

[11] `http://www.bachproject.net`

[12] `http://inscore.sourceforge.net`

[13] `http://www.computermusicnotation.com`

[14] `http://repmus.ircam.fr/openmusic/home`

[15] `http://www2.siba.fi/PWGL`

[16] `http://www.iannix.org`

[17] `http://www.computermusicnotation.com/?page_id=314`

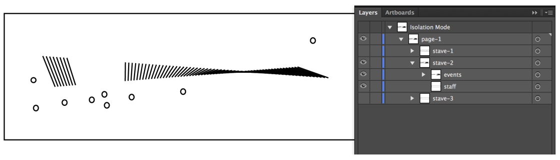


Figure 2. An example of using Adobe Illustrator's grouping mechanism to create a hierarchy of graphic data

envelopes, waveforms, etc.) and so required specific kinds of symbolic composition, which make it not extendable to other types of interpretation.

3.1 Sketching and babbling

The recent MusInk [10] and InkSplorer [11] projects have shown that Livescribe [18] pen technology may also be a way to connect symbolic thinking on paper with digital rendering capabilities. The MusInk project also provides the capability to assign a type to an arbitrary symbol, which is closely related to the present study, however since these Livescribe projects are designed for paper, they forgo some of the possibilities offered by graphic design environments. These studies point to the importance of sketching in developing new graphic ideas.

Sketches are by definition incomplete, and provide the mind with an image to reflect on and continually refine through iteration [12]. David Wessel describes this type of *enactive* engagement in his discussion of *babbling* as a method for free experimentation in sensory-motor development in language and instrument learning, leading towards the development of a "human-instrument symbiosis" [13]. Such a symbiosis should also be possible with symbolic thought and computer controlled rendering systems.

3.2 Performing digital graphic scores

Graphic design applications like Adobe Illustrator, InkScape, OmniGraffle are created to have the basic affordances of a drafting table: a piece of paper, pen, stencils, and ruler – with the end goal of creating publication ready documents. There are no built-in musical functions, no button for transposition, no MIDI playback, etc. What these applications provide are the basic tools for visually creating whatever it is that you want to draw, generally in two dimensions. The user is left to decide what the meaning of the graphics might be.

Composers who choose to work in graphic design programs rather than music notation programs are silently stating that they do not expect to be able to render their score with the computer in the way that a typical musical notation program will have built in MIDI playback. Rather, it is implied that they accept that due to software constraints their work is graphic, and either meant to be performed only by humans who will be able to interpret the score, or that the score is a descriptive notation of electronic results rather than proscriptive notation of how to perform the material. However this does not need to be the case. As a preliminary study implementation, the SVG output of Illustrator was used as a container for performable graphic information, leveraging Illustrator's layer panel as a control for hierarchical grouping.

4. IMPLEMENTATION

Scalable Vector Graphic (SVG) [19] is an XML-based open standard developed by the World Wide Web Consortium (W3C) for two dimensional graphics. In addition to being widely supported in software applications, the SVG Standard provides several key features that make it an attractive solution for digital graphic notation: (1) it is human readable which makes it easy to open an SVG file in a text editor and understand how the data is structured; (2) the SVG format provides a palette of primitive vector shapes that are the raw building blocks for most notations (and also provides tags for adding further types); (3) inheriting XML's tree structure model, SVG provides standardized grouping and layer tags allowing users to create custom hierarchies of graphic objects; and (4) the header for SVG files includes the canvas information for contextualizing the content of the file.

In this paper, we propose replacing the graphics renderer with a new type of rendering interpretation, be it sonic, spatial, kinetic, or any other possible output type. Thought of this way, the SVG file functions as hierarchical input data, to be rendered, or performed by an electronic instrument. In our implementation, OpenSoundControl (OSC) [14] serves as a transcoding layer used for processing an interpretation of the SVG file structure.

4.1 SVG $\rightarrow$ Odot $\rightarrow$ Performance

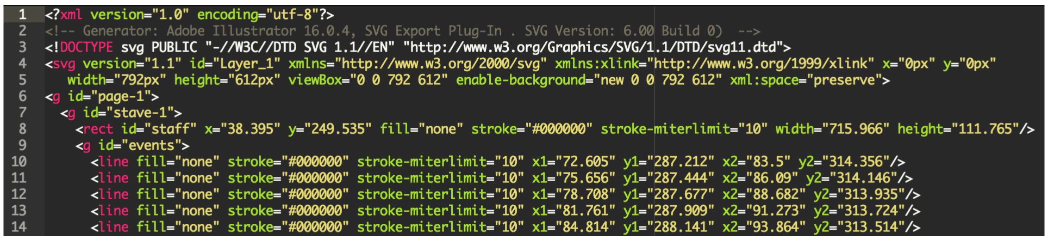
As a first test to interpret and perform the SVG score within the Max environment, the LibXML2 [20] library was used to parse the SVG file created in Adobe Illustrator (figure 2), and convert the SVG tree (figure 3), into an OSC bundle (figure 4). For convenience, this was implemented in C and put in a Max object called "o.svg". The SVG graphic data was then reformatted, and interpreted for performance utilizing the "odot" OSC expression language developed at CNMAT over the last few years [15].

Based on the OSC protocol, CNMAT's new odot library provides tools for handing OSC bundles as time-synchronized

[18] `http://www.livescribe.com`

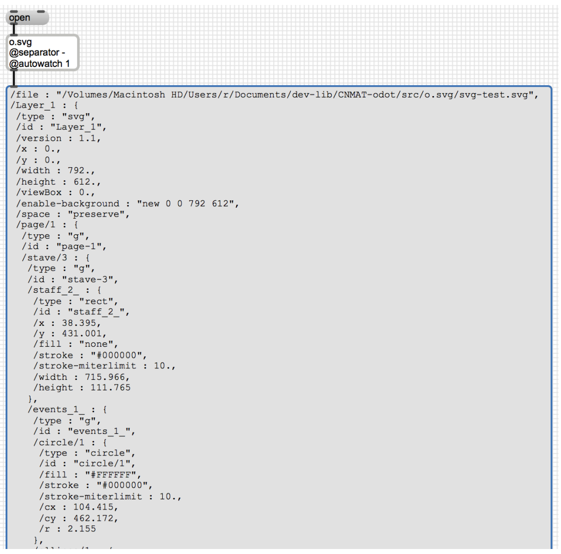
[19] SVG Standard: `http://www.w3.org/TR/SVG`

[20] `http://xmlsoft.org`



```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator: Adobe Illustrator 16.0.4, SVG Export Plug-In . SVG Version: 6.00 Build 0)  -->
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
   width="792px" height="612px" viewBox="0 0 792 612" enable-background="new 0 0 792 612" xml:space="preserve">
<g id="page-1">
  <g id="stave-1">
    <rect id="staff" x="38.395" y="249.535" fill="none" stroke="#000000" stroke-miterlimit="10" width="715.966" height="111.765"/>
    <g id="events">
      <line fill="none" stroke="#000000" stroke-miterlimit="10" x1="72.605" y1="287.212" x2="83.5" y2="314.356"/>
      <line fill="none" stroke="#000000" stroke-miterlimit="10" x1="75.656" y1="287.444" x2="86.09" y2="314.146"/>
      <line fill="none" stroke="#000000" stroke-miterlimit="10" x1="78.708" y1="287.677" x2="88.682" y2="313.935"/>
      <line fill="none" stroke="#000000" stroke-miterlimit="10" x1="81.761" y1="287.909" x2="91.273" y2="313.724"/>
      <line fill="none" stroke="#000000" stroke-miterlimit="10" x1="84.814" y1="288.141" x2="93.864" y2="313.514"/>
```

Figure 3. The contents of the SVG file, showing the hierarchical graphic information designed in Figure 2.



```
/file : "/Volumes/Macintosh HD/Users/r/Documents/dev-lib/CNMAT-odot/src/o.svg/svg-test.svg",
/Layer_1 : {
 /type : "svg",
 /id : "Layer_1",
 /version : 1.1,
 /x : 0.,
 /y : 0.,
 /width : 792.,
 /height : 612.,
 /viewBox : 0.,
 /enable-background : "new 0 0 792 612",
 /space : "preserve",
 /page/1 : {
  /type : "g",
  /id : "page-1",
  /stave/3 : {
   /type : "g",
   /id : "stave-3",
   /staff_2_ : {
    /type : "rect",
    /id : "staff_2_",
    /x : 38.395,
    /y : 431.001,
    /fill : "none",
    /stroke : "#000000",
    /stroke-miterlimit : 10.,
    /width : 715.966,
    /height : 111.765
   },
   /events_1_ : {
    /type : "g",
    /id : "events_1_",
    /circle/1 : {
     /type : "circle",
     /id : "circle/1",
     /fill : "#FFFFFF",
     /stroke : "#000000",
     /stroke-miterlimit : 10.,
     /cx : 104.415,
     /cy : 462.172,
     /r : 2.155
    },
```

Figure 4. The contents of the SVG file designed in Figure 2, transcoded to Odot in the MaxMSP programming environment.

hierarchical data structures within data-flow programming environments like Max, Pure Data (Pd) [21] , and NodeRed [22] . Through odot's expression language, the OSC bundle becomes a local scope for time-synchronized variables in which functions may be applied referencing the named contents of the bundle [16]. Thus, by transcoding the SVG file contents into an OSC bundle, it is possible to process the data in Max/Pd, interpreting the values intended for graphic rendering as control parameters for synthesis, spatialization, and any other parameter controllable with the computer.

The graphic content and grouping relations within an SVG file are described by the organization of XML elements, and graphic primitives as specified by the SVG Standard. For example, a group of SVG elements might look like this:

```
<g id="note-duration-event">
  <circle id="notehead" cx="100" cy="300" r="3.76"/>
  <line id="duration" fill="none" stroke="#000000" stroke-width=
       "3" stroke-miterlimit="10" x1="100" y1="300" x2="200" y2=
       "300"/>
</g>
```

Each SVG element follows a similar structure, the element tag name is followed by a list of attributes to the element. The `<g>` tag indicates a group which is closed by the `</g>` tag, and has an `id` attribute with the value `"note-duration-event"`. The o.svg object creates an OSC bundle, mirroring the structure of the SVG file, and creating OSC addresses for each attribute name of a given SVG element, using the `id` attribute as the element name for example:

```
/note_duration_event/notehead/type : "circle",
/note_duration_event/notehead/cx : 100,
/note_duration_event/notehead/cy : 300,
/note_duration_event/notehead/r : 3.76,
...
```

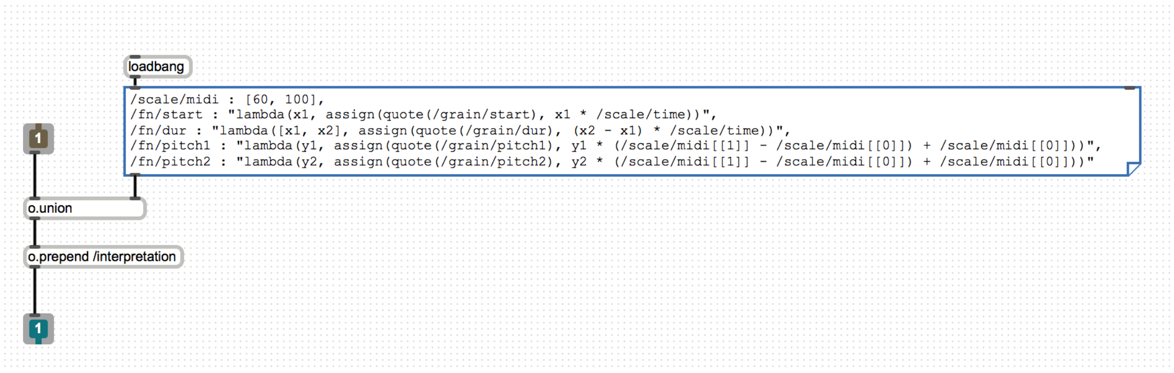
After transcoding the SVG file into OSC, the SVG data may be interpreted, and performed in Max through the odot library, allowing us to sort and iterate over the items, and to apply interpretive functions (figure 5).

4.2 Grouping strategies

With the transcoding from SVG to OSC in place it becomes possible to begin composing within a graphic design program in a way that facilitates the interpretation and performance downstream in OSC. Using the `id` attribute to identify groups and graphic objects, it is then possible to use the SVG tree structure as a framework for developing grammars which can be used later to interpret the graphic information for the generation of control parameters.

Taking traditional musical notation as a starting point, a logical structural design to facilitate rendering might be something like the one illustrated in figure 6. With the root `<svg>` tag understood as the global container for a full score, the next largest container would be the *page*, followed by a *system* which might contain many instrument *staves* (or other output types), each with their own *staff* and *clef*. Within the individual staff group, there might be graphic information providing the *bounds* of the staff (e.g. lines marking different qualities within the vertical range of the staff as described by the clef; where the X dimension usually (but not necessarily) representing time). Within this grammar structure, the bounds of the staff provide a context for interpreting *event objects* contained within the staff group. Further, each event object grouping may contain any number of graphic objects. For example a *note-duration-event* might contain a shape identified as a *note-head*, with other graphic objects representing the event's

[21] `http://puredata.info`
[22] `http://nodered.org`



```
/scale/midi : [60, 100],
/fn/start : "lambda(x1, assign(quote(/grain/start), x1 * /scale/time))",
/fn/dur : "lambda([x1, x2], assign(quote(/grain/dur), (x2 - x1) * /scale/time))",
/fn/pitch1 : "lambda(y1, assign(quote(/grain/pitch1), y1 * (/scale/midi[[1]] - /scale/midi[[0]]) + /scale/midi[[0]]))",
/fn/pitch2 : "lambda(y2, assign(quote(/grain/pitch2), y2 * (/scale/midi[[1]] - /scale/midi[[0]]) + /scale/midi[[0]]))"
```

Figure 5. Example of storing interpretations of graphic information contained in the SVG file in Odot lambda functions. Here, the function describes a process of interpretation where the x1 value indicates the start time, scaled to a given time constant, and the duration is the horizontal span of the object.

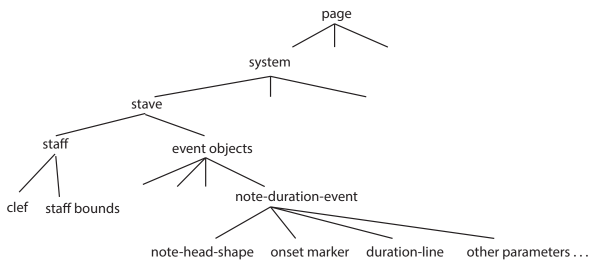


Figure 6. Example namespace hierarchy for identifying score elements. Event object could be any user defined graphic symbol (e.g. a circle, line, paths, gradients, etc.).

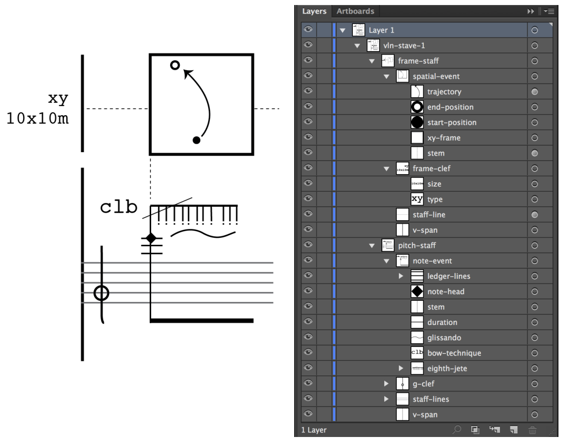


Figure 7. A more developed example showing a 2D trajectory in frame notation in Adobe Illustrator. The hierarchy structure is shown on the right side.

onset, *duration*, and any other parameters.

Figure 7 shows a potential expansion of the note-duration-event object to include with a second *frame-staff* placed above the *pitch-staff* used to notate the spatial trajectory of the sound source in a 2D frame. In this example, a dotted vertical line identified as a *stem* is used to coordinate the beginning of the trajectory with the beginning of a *col legno battuto jeté glissando*, with a duration indicated by the length of the beam line identified as *duration*. This type of trajectory is the simplest type of spatial processing, in cases with more complex treatments, such as spherical harmonic manipulation, other types of notation would be more appropriate.

5. CRITIQUE AND FUTURE WORK

The initial results of the work are encouraging, however, there are many areas that could be developed further. The study shows that it is possible to increase the rendering context flexibility by separating the score editing environment from pre-conceived ideas about how the score might be interpreted or rendered. The flip-side to the current implementation which uses only the odot/OSC expression language for parsing the assigned meaning of the notation, is that while the possibilities of the system are extremely large, this flexibility comes at the price of the parsing programming that must be written to interpret and perform the score. For example, if a user wants to also use traditional musical notation formatting rules, the entire mechanism for traditional score interpretation must be built using the odot expression language. As daunting as this might sound, the symbolic flexibility of the graphic design environment plus the many rendering media accessible through OSC may make the development of multiple rendering systems worth the effort.

Illustrator's editing environment has been very well developed for many years to become the standard for graphic design, which in addition to extensive support for print output, provides a large number graphic functions that can be leveraged for temporal media composition. However, since Illustrator is designed for graphic art not "renderable scores," there are noticeable limitations on the amount of data that can be contained in a graphics file before it begins to effect the application's responsiveness. This eventually points to the fact that a *specialized* tool for notation might indeed be useful, providing a database and Model View Controller (MVC) architecture for interacting with score data.

Adobe's recently announced support for Node.js[23] opens up several new options for working within the Illustrator application. For example, with odot's SWIG[24] based JS bindings, a Node plug-in could be created to stream OSC score data directly from Illustrator without the need to save the file to disk and reload it with the o.svg object. With the addition of Node as a plug-in backend this means that Paper.js[25] could be used to create custom interactive GUIs for handling data. Paper.js is developed by the same team who wrote Scriptographer[26] which was a powerful JS based drawing tool building suite for Illustrator, and was one of the initial tools for an earlier version of our study. Unfortunately, Adobe drastically changed their plugin design in Illustrator CS6, which broke Scriptographer. This is an important point to be considered for any future work in the present study, and is one indicator that possibly an independent design environment might be a more reliable long-term solution. A future iteration of the study might be in the form of a Node and Paper.js based editor with a stripped-down toolkit for symbolic graphic notation. The Paper.js front-end would allow users to easily create their own interactive tools, and either export a rendering of the score to SVG for printing with a program like Illustrator, or the score could be streamed via odot/SWIG. There is some possibility that INScore's V8[27] integration might provide a suitable platform for these developments, this would also allow the editor to take advantage of INScore's MVC design, and traditional notation tools.

Other improvements might include a more intuitive system for defining meaning for symbols. In the process of sketching and developing a notation, it was time consuming to constantly keep objects nicely grouped and labeled using Illustrator's layer and grouping tools. This issue can be mitigated thought the use of search algorithms to auto-detect symbol patterns (i.e. containing similar types of graphic objects, gestures, etc.) which would allow the artist to later apply semantic structuring rule to different members of these symbolic groupings.

6. CONCLUSION

The authoring of data in computer music systems is predominately done through graphical representations of univariate functions, whereas symbolic notation systems like music notation are aggregate and contextual. A symbol in a notation system is given meaning through the interpretation of a human or computerized intelligence based on contextual understanding, for example the nature of a staccato string articulation is different for different dynamic ranges. Complex rendering systems incorporating digital signal processing and/or other electronic media often have a large number of parameters that artists wish to control expressively. Due to the affordances of the programming environments in which these pieces are created, there is typically a focus on control of many *single* parameters. However, the symbolic representation of information such as spatial location becomes fragmented in these systems, forcing a point in space to be represented with three separate coordinates, which in many ways obscures its perceptual simplicity.

The SVG format provides a useful method for defining meanings of symbols leveraging Illustrator's grouping and layering tools, while the graphic editing environment provided by graphic design programs like Illustrator provide a flexible vector graphic drafting environment for symbolic experimentation. Since Illustrator was designed without musical applications in mind, there are no pre-conceived playback limitations based on the application developer's idea of what "music" is, or how graphic symbols on a page should be organized. This lack of meaning leaves room for the user to sketch and experiment, as well as requiring extra effort to create meaning through an interpretive algorithm if the score is meant to be performed by the computer. Transcoding SVG format into OSC facilitates the interpretation of notation through the use of the odot expression language in the Max media programming environment, providing digital artists a mechanism to perform graphic symbolic notation with any electronic media accessible with Max, Pd, or any other application that can interpret OSC.

Preliminary work on developing an interpretation and performance system for notation stored in SVG format has proven feasible, however there is still significant work needed to bring the system to a point where it would be competitive with existing rendering systems that are specifically designed for a given medium. On the other hand the openness of the SVG format, combined with its compatibility with OSC points towards a myriad of new ways to expressively controlling new media formats with symbolic notation. Looking towards the future, the above plans for a new symbolic graphic notation editor discussed in section 5 seem to be a promising direction for the creation of notation software that is capable of being used to render new media forms that have proven difficult to notate (such as spatial audio), as well as those that have yet to be thought of.

[23] http://www.adobe.com/devnet/cs-extension-builder/articles/extend-adobe-cc-2014-apps.html
[24] http://www.swig.org
[25] http://paperjs.org
[26] http://scriptographer.org
[27] https://code.google.com/p/v8

Acknowledgments

I would like to thank John MacCallum and Adrian Freed for their valuable feedback in developing this study, and in developing the odot system without which this work would not exist. I would also like to thank Olivier Warusfel, Markus Noisternig, and Thibaut Carpentier for their mentorship during my spatial composition residency at Ircam where many of these ideas were developed, and to Jean Bresson for his continued feedback and great work in the field of musical representation.

7. REFERENCES

[1] K. Stone, "Problems and methods of notation," *Perspectives of New Music*, vol. 1, no. 2, pp. 9–31, 1963.

[2] P. Manning, "The oramics machine: From vision to reality," *Organised Sound*, vol. 17, no. 2, pp. 137–147, August 2012.

[3] P. L. Smirnov A., "1917-1939. son z / sound in z." *PALAIS / Palais de Tokyo Magazine, Paris*, no. 7, pp. pp. 66–77, 2008.

[4] B. Farnell, "Movement notation systems," in *The world's writing systems*, O. U. Press, Ed. New York, NY: Weidenfeld and Nicholson, 1996, pp. 855–879.

[5] M. Battier, "Describe, transcribe, notate: Prospects and problems facing electroacoustic music," *Organised Sound*, vol. 20, no. Special Issue 01, pp. 60–67, April 2015.

[6] J. Greeno, "Gibon's affordances," *Psychological Review*, vol. 101, no. 2, pp. 336–342, 1994.

[7] Y. Geslin and A. Lefevre, "Sound and musical representation: the acousmographe software," in *ICMC*. Miami, USA: International Computer Music Conference, 2004.

[8] F. Barknecht, "128 is not enough - data structures in pure data," in *Linux Audio Conference*, Karlsruhe, Germany, 2006.

[9] H. Lohner, "The upic system: A user's report," *Computer Music Journal*, vol. 10, no. 4, pp. 42–49, 1986.

[10] T. Tsandilas, C. Letondal, and W. E. Mackay, "Musink: Composing music through augmented drawing," in *CHI*. Boston, Massachusetts, USA: Conference on Human Factors in Computing Systems, 2009.

[11] J. Garcia, T. Tsandilas, C. Agon, and W. Mackay, "Inksplorer: Exploring musical ideas on paper and computer," in *NIME*. Oslo, Norway: New Interfaces for Musical Expression, 2011.

[12] M. Tohidi, W. Buxton, R. Baecker, and A. Sellen, "User sketches: A quick, inexpensive, and effective way to elicit more reflective user feedback," in *NordiCHI*. Oslo, Norway: Nordic conference on Human-computer interaction, 2006.

[13] D. Wessel, "An enactive approach to computer music performance," in *Le Feedback dans la Creation Musical*, Y. Orlarey, Ed. Lyon, France: Studio Gramme, 2006, pp. 93–98.

[14] A. Freed and A. Schmeder, "Features and future of open sound control version 1.1 for nime," in *NIME*. New Interfaces for Musical Expression, 2009.

[15] A. Freed, J. MacCallum, and A. Schmeder, "Composability for musical gesture signal processing using new osc-based object and functional programming extensions to max/msp," in *NIME*. Oslo, Norway: New Interfaces for Musical Expression, 2011.

[16] J. MacCallum, A. Freed, and D. Wessel, "Agile interface development using osc expressions and process migration," in *NIME*, Daejeon Korea, 2013.

ABJAD: AN OPEN-SOURCE SOFTWARE SYSTEM FOR FORMALIZED SCORE CONTROL

Trevor Bača
Harvard University
`trevor.baca@gmail.com`

Josiah Wolf Oberholtzer
Harvard University
`josiah.oberholtzer@gmail.com`

Jeffrey Treviño
Carleton College
`jeffrey.trevino@gmail.com`

Víctor Adán
Columbia University
`vctradn@gmail.com`

ABSTRACT

The Abjad API for Formalized Score Control extends the Python programming language with an open-source, object-oriented model of common-practice music notation that enables composers to build scores through the aggregation of elemental notation objects. A summary of widely used notation systems' intended uses motivates a discussion of system design priorities via examples of system use.

1. INTRODUCTION

Abjad[1] is an open-source software system designed to help composers build scores in an iterative and incremental way. Abjad is implemented in the Python[2] programming language as an object-oriented collection of packages, classes and functions. Composers can visualize their work as publication-quality notation at all stages of the compositional process using Abjad's interface to the LilyPond[3] music notation package. The first versions of Abjad were implemented in 1997 and the project website is now visited thousands of times each month. This paper details some of the most important principles guiding the development of Abjad and illustrates these with examples of the system in use. The priorities outlined here arise in answer to domain-specific questions of music modeling (What are the fundamental elements of music notation? Which elements of music notation should be modeled hierarchically?) as well as in consideration of the ways in which best practices taken from software engineering can apply to the development of a music software system (How can programming concepts like iteration, aggregation and encapsulation help composers as they work?). A background taxonomy motivates a discussion of design priorities via examples of system use.

[1] `http://www.projectabjad.org`
[2] `http://www.python.org`
[3] `http://www.lilypond.org`

2. A TAXONOMY

Many software systems implement models of music but few of these implement a model of notation. Many music software systems model higher-level musical entities apparent in the acts of listening and analysis while omitting any model of the symbols of music notation. Researchers and musical artists have modeled many such extrasymbolic musical entities, such as large-scale form and transition [1–5], texture [6], contrapuntal relationships [7–13], harmonic tension and resolution [14–16], melody [17, 18], meter [19], rhythm [20–22], timbre [23–25], temperament [26, 27] and ornamentation [28, 29]. This work overlaps fruitfully with analysis tasks because models of listening and cognition can enable novel methods of high-level musical structures and transformations, like dramatic direction, tension, and transition between sections [30].

Software production exists as an organizationally designed feedback loop between production values and implementation [31]. It is possible to understand a system by understanding the purpose for which it was initially designed. This purpose can be termed a software system's generative task. In the classfication of systems created for use by artists, this priority yields a dilemma instantly, as analyses that explain a system's affordances with reference to intended purpose must contend with the creative use of technology by artists: a system's intended uses might have little or nothing in common with the way in which the artist finally uses the technology. For this reason, the notion of generative task is best understood as an explanation for a system's affordances, with the caveat that a user can nonetheless work against those affordances to use the system in novel ways.

While composers working traditionally may allow intuition to substitute for formally defined principles, a computer demands the composer to think formally about music [32]. Keeping in mind generative task as an analytical framework, it is broadly useful to bifurcate a notation system's development into the modeling of composition, on the one hand, and the modeling of musical notation, on the other. All systems model both, to greater or lesser degrees, often engaging in the ambiguous or implicit modeling of composition while focusing more ostensibly on a model of notation, or focusing on the abstract modeling of composition without a considered link to a model of notation. Generative task explains a given system's balance between computational models of composition and notation by assuming a link between intended use and system development.

Many notation systems — such as Finale, Sibelius, SCORE [33], Igor, Berlioz, Lilypond [34], GUIDO [35] NoteAbility [36], FOMUS [37, 38] and Nightingale — exist to help people engrave and format music documents; because these systems provide functions that operate on notational elements (i.e., transposition, spacing and playback), hidden models of common-practice music notation must underly all of these systems, and each system's interface constrains and directs the ways in which users interact with this underlying model of notation. These systems enable users to engrave and format music without exposing any particular underlying model of composition, and without requiring, or even inviting the user to computationally model composition. Such systems might go so far as to enable scripting, as in the case of Sibelius's ManuScript [39] scripting language or Lilypond's embedded Scheme code; although these systems enable the automation of notational elements, it remains difficult to model compositional processes and relationships.

Other systems provide environments specifically for the modeling of higher-level processes and relationships. OpenMusic [40], PWGL [41] and BACH [42] supply an interface to a model of common-practice notation, as well as a set of non-common-practice visual interfaces that enables the user to model composition, in the context of a standalone application and with the aid of the above notation editors for final engraving and layout via intermediate file formats. Similarly purposed systems extend text-based programming languages rather than existing as stand-alone applications, such as HMSL's extension of Forth [43], JMSL's extension of Java [44] and Common Music's extension of Lisp [45]. Other composition modeling systems, such as athenaCL [46] and PILE/AC Toolbox [47] provide a visual interface for the creation of compositional structures without providing a model of common-practice notation.

Some composers make scores with software systems that provide neither a model of notation nor a model of composition. Graphic layout programs, such as AutoCAD and Adobe Illustrator, have been designed broadly for the placement and design of graphic elements. While scripting enables automation, composers must model both notation and composition from scratch, and the symbolic scope of potential automations described in the course of modeling ensures that composers spend as much time addressing elemental typographical concerns (e.g., accidental collisions) as would be spent modeling compositional processes and relationships.

Many models of musical notation have been designed for purposes of corpus-based computational musicology. Formats such as DARM, SMDL, HumDrum and MuseData model notation with the generative task of searching through a large amount of data [48]. Commercial notation software developers attempted to establish a data interchange standard for optical score recognition (NIFF) [49]. Since its release in 2004, MusicXML has become a valid interchange format for over 160 applications and maintains a relatively application-agnostic status, as it was designed with the generative task of acting as an interchange format between variously-tasked systems [50].

An attempt to survey more comprehensively the history of object-oriented notation systems for composition, in the context of the broader history of object-oriented programming, lies beyond the scope of this paper but has recently been undertaken elsewhere [51].

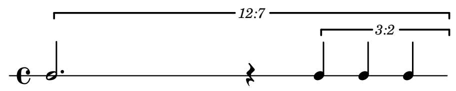
3. ABJAD BASICS

Abjad is not a stand-alone application. Nor is Abjad a programming language. Abjad instead adds a computational model of music notation to Python, one of the most widely used programming languages currently available. Abjad's design as a standard extension to Python makes hundreds of print and Web programming resources relevant to composers and further helps to make the global communities of software developers and composers available to each other.[4] [5] Composers work with Abjad exactly the same as with any other Python package. In the most common case this means opening a file, writing code and saving the file:

```
from abjad import *

def make_nested_tuplet(
    tuplet_duration,
    outer_tuplet_proportions,
    inner_tuplet_subdivision_count,
    ):
    outer_tuplet = Tuplet.from_duration_and_ratio(
        tuplet_duration, outer_tuplet_proportions)
    inner_tuplet_proportions = \
        inner_tuplet_subdivision_count * [1]
    last_leaf = outer_tuplet.select_leaves()[-1]
    inspector = inspect_(last_leaf)
    right_logical_tie = inspector.get_logical_tie()
    right_logical_tie.to_tuplet(inner_tuplet_proportions)
    return outer_tuplet
```

The classes, functions and other identifiers defined in the file can then be used in other Python files or in an interactive session:

```
>>> rhythmic_staff = Staff(context_name='RhythmicStaff')
>>> tuplet = make_nested_tuplet((7, 8), (3, -1, 2), 3)
>>> rhythmic_staff.append(tuplet)
>>> show(rhythmic_staff)
```



This paper demonstrates examples in Python's interactive console because the console helps distinguish input from output. Lines preceded by the >>> prompt are passed to Python for interpretation and any output generated by the line of code appears immediately after. The example above creates a tuplet with the tuplet-making function defined earlier and calls Abjad's top-level `show()` function to generate

[4] See the Python Package Index for extensions to Python for purposes as diverse as creative writing and aeronautical engineering. The Python Package Index contains 54,306 packages at the time or writing and is available at `https://pypi.python.org`.

[5] Abjad is an importable Python library. It can be used in whole or in part as a component of any Python-compatible system. For example, Abjad supports IPython Notebook, a Web-based interactive computational environment combining code execution, text, mathematics, plots and rich media into a single document. Notational output from Abjad can be transparently captured and embedded into an IPython Notebook that has loaded Abjad's IPython Notebook extension. See `http://ipython.org/notebook.html`.

a PDF of the result. But note that composers work with Abjad primarily by typing notationally-enabled Python code into a collection of interrelated files and managing those files as a project grows to encompass the composition of an entire score.

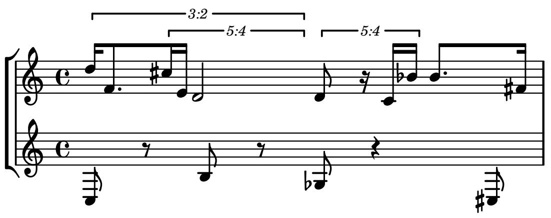
4. THE ABJAD OBJECT MODEL

Abjad models musical notation with *components*, *spanners* and *indicators*. Every notational element in Abjad belongs to one of these three families. Abjad models notes, rests and chords as classes that can be added into the container-like elements of music notation, such as tuplets, measures, voices, staves and complete scores. Spanners model notational constructs that cross different levels of hierarchy in the score tree, such as beams, slurs and glissandi. Indicators model objects like articulations, dynamics and time signatures that attach to a single component. Composers arrange components hierarchically into a score tree with spanners and indicators attached to components in the tree. [6]

5. BOTTOM-UP CONSTRUCTION

Abjad lets composers build scores from the bottom up. When working bottom-up, composers create individual notes, rests and chords to be grouped into tuplets, measures or voices that may then be included in even higher-level containers, such as staves and scores. Abjad affords this style of component aggregation via a container interface which derives from Python's mutable sequence protocol. Python's mutable sequence protocol specifies an interface to list-like objects. Abjad's container interface implements a collection of methods which append, extend or insert into Abjad containers:

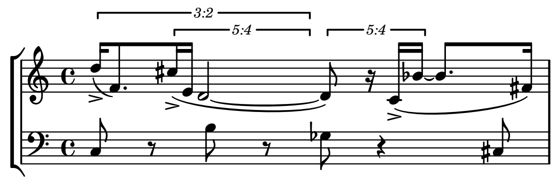
```
>>> outer_tuplet_one = Tuplet((2, 3), "d''16 f'8.")
>>> inner_tuplet = Tuplet((4, 5), "cs''16 e'16 d'2")
>>> outer_tuplet_one.append(inner_tuplet)
>>> outer_tuplet_two = Tuplet((4, 5))
>>> outer_tuplet_two.extend("d'8 r16 c'16 bf'16")
>>> tuplets = [outer_tuplet_one, outer_tuplet_two]
>>> upper_staff = Staff(tuplets, name='Upper Staff')
>>> note_one = Note(10, (3, 16))
>>> upper_staff.append(note_one)
>>> note_two = Note(NamedPitch("fs'"), Duration(1, 16))
>>> upper_staff.append(note_two)
>>> lower_staff = Staff(name='Lower Staff')
>>> lower_staff.extend("c8 r8 b8 r8 gf8 r4 cs8")
>>> staff_group = StaffGroup()
>>> staff_group.extend([upper_staff, lower_staff])
>>> score = Score([staff_group])
>>> show(score)
```



Notes and chords may be initialized with pitches named according to either American or European conventions. Notes and chords may also be initialized with the pitch numbers of American pitch-class theory or from combinations of Abjad pitch and duration objects. Unlike many notation packages, Abjad does not require composers to structure music into measures. All Abjad containers can hold notes, rests and chords directly.

Ties, slurs and other spanners attach to score components via Abjad's top-level `attach()` function. The same is true for articulations, clefs and other indicators. For example, after selecting the notes, rests and chords from each staff, individual components and slices of contiguous components may be selected by their indices within each selection. [7] Indicators and spanners may then be attached to those components:

```
>>> upper_leaves = upper_staff.select_leaves()
>>> lower_leaves = lower_staff.select_leaves()
>>> attach(Tie(), upper_leaves[4:6])
>>> attach(Tie(), upper_leaves[-3:-1])
>>> attach(Slur(), upper_leaves[:2])
>>> attach(Slur(), upper_leaves[2:6])
>>> attach(Slur(), upper_leaves[7:])
>>> attach(Articulation('accent'), upper_leaves[0])
>>> attach(Articulation('accent'), upper_leaves[2])
>>> attach(Articulation('accent'), upper_leaves[7])
>>> attach(Clef('bass'), lower_leaves[0])
>>> show(score)
```



When does it make sense for composers to work with Abjad in the bottom-up way outlined here? Instantiating components by hand in the way shown above resembles notating by hand and composers may choose to work bottom-up when doing the equivalent of sketching in computer code: when making the first versions of a figure or gesture, when trying out combinations of small bits of notation or when inserting one or two items at a time into a larger structure. For some composers this may be a regular or even predominant way of working. Other composers may notice patterns in their own compositional process when they work bottom-up and may find ways to formalize these patterns into classes or functions that generalize their work; the next section describes some ways composers do this.

6. TOP-DOWN CONSTRUCTION

What are the objects of music composition? For most composers, individual notes, rests and chords constitute only the necessary means to achieve some larger, musically interesting result. For this reason, a model of composition needs to describe groups of symbols on the page: notes taken in sequence to constitute a figure, gesture or melody;

[6] Abjad chords aggregate note-heads instead of notes. This corrects a modeling problem sometimes present in other music software systems: if chords aggregate multiple notes and every note has a stem then how is it that chords avoid multiple stems? Abjad chords implement the container interface described below to add and remove note-heads to and from chords.

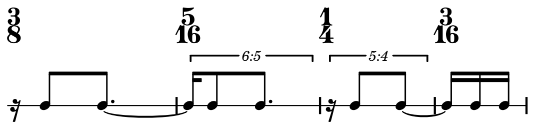
[7] Python allows indexing into sequences by both positive and negative indices. Positive indices count from the beginning of the sequence, starting at 0, while negative indices count from the end of the sequence, with -1 being the last item in the sequence and -2 the second-to-last. Subsegments of a sequence may be retrieved by slicing with an optional start and optional stop index. The slice indicated by `[1:-1]` would retrieve all of the items in a sequence starting from the second and going up until, but not including, the last. The slice indicated by `[:3]`, which omits a start index, retrieves all items from the sequence up until, but not including, the fourth.

chords taken in sequence as a progression; attack points arranged in time as the scaffolding of some larger texture. These entities, and the others like them, might result from a flash of compositional intuition that then requires detailed attention and elaboration.

Abjad invites composers to implement factories as a way of generalizing and encapsulating parts of one's own compositional process. In this way, composers can extend the system as they work to implement their own models of composition. Abjad also provides a variety of factory functions and factory classes that exemplify this way of working. These range from simple note-generating functions, like make_notes(), which combine sequences of pitches and rhythms to generate patterned selections of notes and rests, to more complexly-configured maker classes for creating nuanced rhythmic patterns or entire scores.

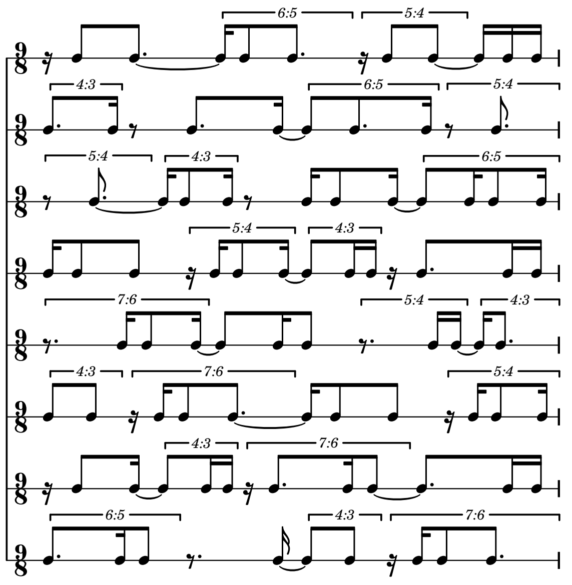
As an example, consider the rhythmmakertools package included with Abjad. The classes provided in this package are factory classes which, once configured, can be called like functions to inscribe rhythms into a series of beats or other time divisions. The example below integrates configurable patterns of durations, tupletting and silences:

```
>>> burnish_specifier = rhythmmakertools.BurnishSpecifier(
...     left_classes=(Rest, Note),
...     left_counts=(1,),
...     )
>>> talea = rhythmmakertools.Talea(
...     counts=(1, 2, 3),
...     denominator=16,
...     )
>>> tie_specifier = rhythmmakertools.TieSpecifier(
...     tie_across_divisions=True,
...     )
>>> rhythm_maker = rhythmmakertools.TaleaRhythmMaker(
...     burnish_specifier=burnish_specifier,
...     extra_counts_per_division=(0, 1, 1),
...     talea=talea,
...     tie_specifier=tie_specifier,
...     )
>>> divisions = [(3, 8), (5, 16), (1, 4), (3, 16)]
>>> show(rhythm_maker, divisions=divisions)
```



Once instantiated, factory classes like this can be used over and over again with different input:

```
>>> rhythmic_score = Score()
>>> for i in range(8):
...     selections = rhythm_maker(divisions, seeds=i)
...     measure = Measure((9, 8), selections)
...     staff = Staff(context_name='RhythmicStaff')
...     staff.append(measure)
...     rhythmic_score.append(staff)
...     divisions = sequencetools.rotate_sequence(
...         divisions, 1)
...
>>> show(rhythmic_score)
```

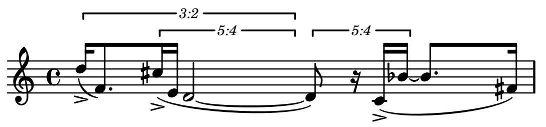


7. SELECTING OBJECTS IN THE SCORE

Abjad allows composers to select and operate on collections of objects in a score. Composers can select objects in several ways: by name, numeric indices or iteration. A single operation, such as transposing pitches or attaching articulations, can then be mapped onto the entirety of a selection.

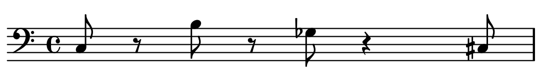
Consider the two-staff score created earlier. Because both staves were given explicit names, the upper staff can be selected by name:

```
>>> upper_staff = score['Upper Staff']
>>> show(upper_staff)
```



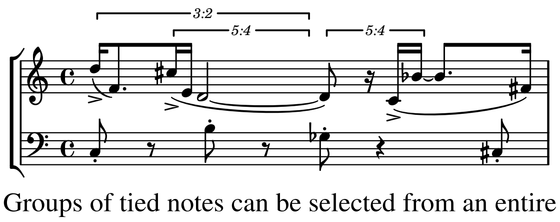
Using numeric indices, the lower staff can be selected by indexing the second child of the first child of the score:

```
>>> lower_staff = score[0][1]
>>> show(lower_staff)
```



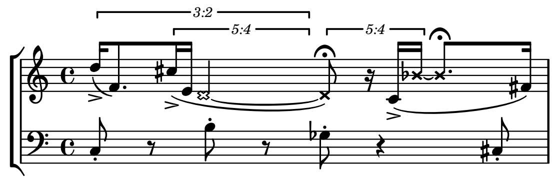
The top-level iterate() function exposes Abjad's score iteration interface. This interface provides a collection of methods for iterating the components in a score in different ways. For example, all notes can be selected from a single staff:

```
>>> for note in iterate(lower_staff).by_class(Note):
...     attach(Articulation('staccato'), note)
...
>>> show(score)
```



Groups of tied notes can be selected from an entire score. Abjad uses the term logical tie to refer to the collection of notes or chords joined together by consecutive ties. The 'logical' qualifier points to the fact that Abjad considers untied notes and untied chords as logical ties of length 1, which makes it possible to select untied notes and chords together with tied notes and chords in a single method call:

```
>>> for logical_tie in iterate(score).by_logical_tie():
...     if 1 < len(logical_tie):
...         attach(Fermata(), logical_tie.tail)
...         for note in logical_tie:
...             override(note).note_head.style = 'cross
...
>>> show(score)
```



8. PROJECT TESTING AND MAINTENANCE

Abjad has benefited enormously from programming best practices developed by the open-source community. As described previously, the extension of an existing language informs the project as a first principle. The following other development practices from the open-source community have also positively impacted the project and might be helpful in the development of other music software systems.

The literature investigated in preparing this report remains overwhelmingly silent on questions of software testing. None of the sources cited in this article reference software test methodologies. The same appears to be true for the larger list of sources included in [51]. [8] Why should this be the case? One possibility is that authors of music software systems have, in fact, availed themselves of important improvements in software test methods developed over the previous decades but have, for whatever reasons, remained quiet on the matter in the publication record. Perhaps the culture of software best practices now widely followed in the open-source community simply has not yet arrived in the field of music software systems development (and especially in the development of systems designed for non-commercial applications).

The use of automated regression testing in Abjad's development makes apparent the way in which tests encourage efficient development and robust project continuance. Abjad comprises an automated battery of 9,119 unit tests and 8,528 documentation tests. Unit tests are executed by `pytest`. [9] Documentation tests are executed by the `doctest` module included in Python's standard library. Parameterized tests ensure that different classes implement similar behaviors in a consistent way. Developers run the entire battery of tests at the start of every development session. No new features are accepted as part of the Abjad codebase without tests authored to document changes to the system. Continuous integration testing is handled by Travis CI [10] to ensure that all tests pass after every commit from every core developer and newcomer to the project alike.

The presence of automated regression tests acts as an incentive to new contributors to the system (who can test whether proposed changes to the system work correctly with existing features) and greatly increases the rate at which experienced developers can refactor the system during new feature development. Abjad currently comprises about 178,000 lines of code. The Abjad repository, hosted on GitHub, [11] lists more than 8.7 million lines of code committed since the start of the project. This refactor ratio of about 50:1 means that each line of code in the Abjad codebase has been rewritten dozens of times. The freedom to refactor at this rate is possible only because of the approach to automated regression testing Abjad has borrowed from the larger open-source community.

Testing benefits project continuance when the original developers of a music software system can no longer develop the system. Automated regression tests help make possible a changing of the guard from one set of developers to another. Automated tests serve as a type of functional specification of how a software system should behave after revision. While automated tests alone will not ensure the continued development of any software system, adherence to the testing practices of the open-source community constitutes the most effective hedge available to music software systems developers to fend against project abandonment in the medium and long term.

9. DISCUSSION & FUTURE WORK

The design and development priorities for Abjad outlined here derive from the fact that the developers of Abjad are all composers who use the system to make their own scores. Abjad is not implemented for the type of music information storage and retrieval functions that constitute an important part of musicology-oriented music software systems. Nor is Abjad designed for use in real-time contexts of performance or synthesis. Abjad is designed as a composers' toolkit for the formalized control of music notation and for modeling the musical ideas that composers use notation to explore and represent. For example, figure 1 shows a one-page excerpt from a score constructed entirely with tools extending Abjad and typeset with LilyPond. Although Abjad embeds well in other music software systems, future work planned for Abjad itself does not prioritize file format conversion, audio synthesis, real-time applications or graphic user interface integration. Future work will instead extend Abjad for object-oriented control over parts of the document preparation process required of complex scores with many parts. Future work will also extend and reinforce the inventory of factory classes and factory functions introduced in this report. We hope this will encourage composers working with Abjad to transition from working with lower-level symbols of music notation to modeling higher-level ideas native to one's own language of composition.

[8] AthenaCL [46] and Music21 [52] are important exceptions. Both projects are implemented in Python and both projects feature approaches to testing in line with those outlined here.

[9] `http://pytest.org`

[10] `https://travis-ci.org`

[11] `https://github.com/Abjad/abjad`

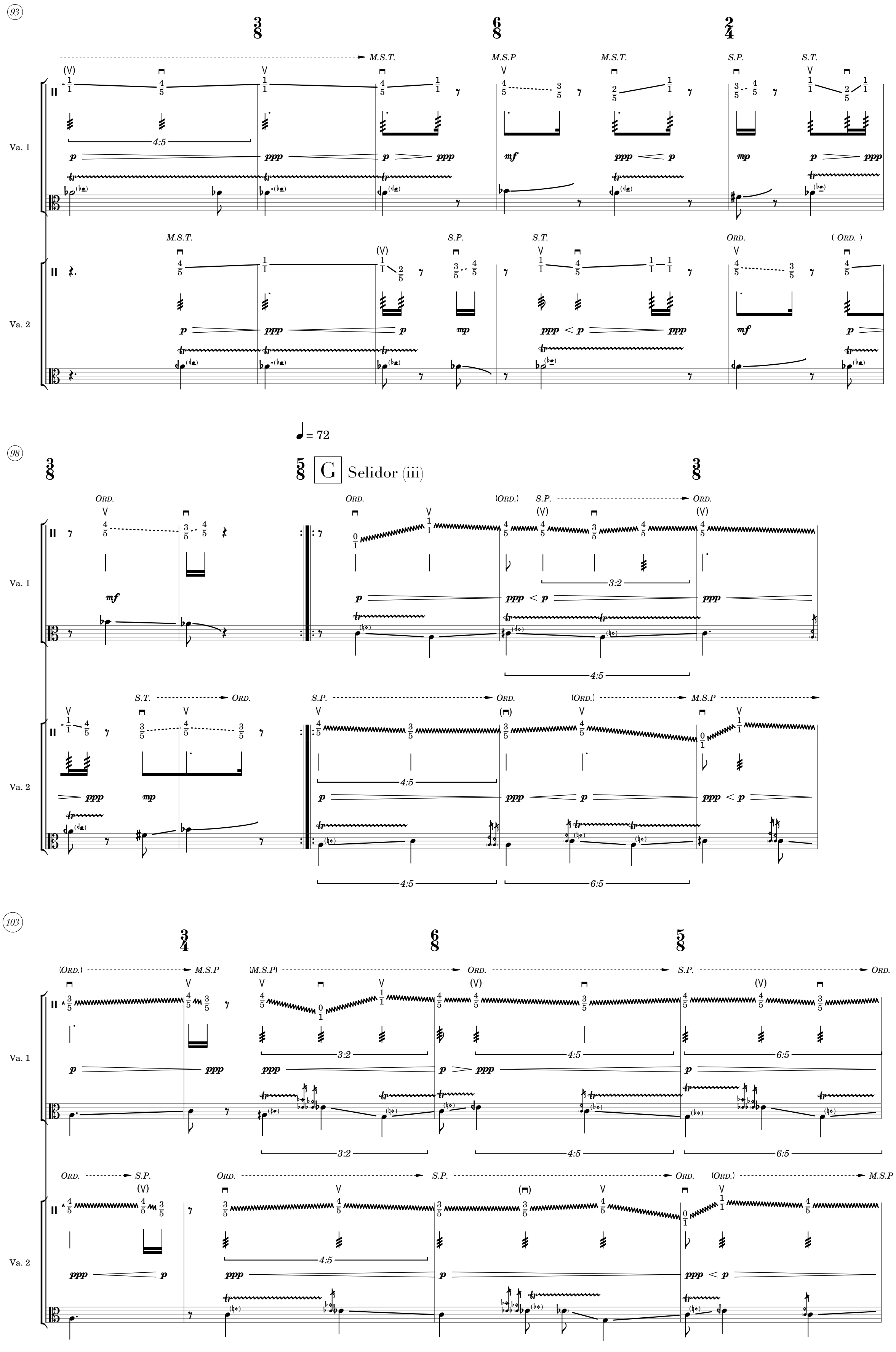


Figure 1. Page 8 from Josiah Wolf Oberholtzer's *Invisible Cities (ii): Armilla* for two violas (2015), created with tools extending Abjad. Source for this score is available at `https://github.com/josiah-wolf-oberholtzer/armilla`.

Acknowledgements

Our sincere thanks go out to all of Abjad's users and developers for their comments and contributions to the code. We would also like to thank everyone behind the LilyPond and Python projects, as well as the wider open-source community, for fostering the tools that make Abjad possible.

10. REFERENCES

[1] L. Polansky, M. McKinney, and B. E.-A. M. Studio, "Morphological Mutation Functions," in *Proceedings of the International Computer Music Conference*, 1991, pp. 234–41.

[2] Y. Uno and R. Huebscher, "Temporal-Gestalt Segmentation-Extensions for Compound Monophonic and Simple Polyphonic Musical Contexts: Application to Works by Cage, Boulez, Babbitt, Xenakis and Ligeti," in *Proceedings of the International Computer Music Conference*, 1994, p. 7.

[3] C. Dobrian, "Algorithmic Generation of Temporal Forms: Hierarchical Organization of Stasis and Transition," in *Proceedings of the International Computer Music Conference*, 1995.

[4] S. Abrams, D. V. Oppenheim, D. Pazel, J. Wright *et al.*, "Higher-level Composition Control in Music Sketcher: Modifiers and Smart Harmony," in *Proceedings of the International Computer Music Conference*, 1999.

[5] M.-J. Yoo and I.-K. Lee, "Musical Tension Curves and its Applications," *Proceedings of International Computer Music Conference*, 2006.

[6] S. Horenstein, "Understanding Supersaturation : A Musical Phenomenon Affecting Perceived Time," *Proceedings of International Computer Music Conference*, 2004.

[7] G. Boenn, M. Brain, M. De Vos, and et. al., "Anton: Composing Logic and Logic Composing," in *Logic Programming and Nonmonotonic Reasoning*. Springer, 2009, pp. 542–547.

[8] M. E. Bell, "A MAX Counterpoint Generator for Simulating Stylistic Traits of Stravinsky, Bartok, and Other Composers," *Proceedings of International Computer Music Conference*, 1995.

[9] M. Farbood and B. Schoner, "Analysis and Synthesis of Palestrina-style Counterpoint using Markov Chains," in *Proceedings of the International Computer Music Conference*, 2001, pp. 471–474.

[10] D. Cope, "Computer Analysis and Computation Using Atonal Voice-Leading Techniques," *Perspectives of New Music*, vol. 40, no. 1, pp. 121–146, 2002. [Online]. Available: `http://www.jstor.org/stable/833550`

[11] M. Laurson and M. Kuuskankare, "Extensible Constraint Syntax Through Score Accessors," in *Journées d'Informatique Musicale*, 2005, pp. 27–32.

[12] L. Polansky, A. Barnett, and M. Winter, "A Few More Words About James Tenney: Dissonant Counterpoint and Statistical Feedback," *Journal of Mathematics and Music*, vol. 5, no. 2, pp. 63–82, 2011.

[13] K. Ebcioglu, "Computer Counterpoint," *Proceedings of International Computer Music Conference*, 1980.

[14] A. F. Melo and G. Wiggins, "A Connectionist Approach to Driving Chord Progressions Using Tension," in *Proceedings of the AISB*, vol. 3, no. 1988, 2003. [Online]. Available: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.9086&amp;rep=rep1&amp;type=pdf`

[15] G. Wiggins, "Automated Generation of Musical Harmony: What's Missing?" in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999. [Online]. Available: `http://www.doc.gold.ac.uk/~mas02gw/papers/IJCAI99b.pdf`

[16] C. D. Foster, "A Consonance Dissonance Algorithm for Intervals," *Proceedings of International Computer Music Conference*, 1995.

[17] D. Hornel, "SYSTHEMA - Analysis and Automatic Synthesis of Classical Themes," *Proceedings of International Computer Music Conference*, 1993.

[18] M. Smith and S. Holland, "An AI Tool for Analysis and Generation of Melodies," *Proceedings of International Computer Music Conference*, 1992.

[19] M. Hamanaka, K. Hirata, and S. Tojo, "Automatic Generation of Metrical Structure Based on GTTM," *Proceedings of International Computer Music Conference*, 2005.

[20] P. Nauert, "Division- and Addition-based Models of Rhythm in a Computer-Assisted Composition System," *Computer Music Journal*, vol. 31, no. 4, pp. 59–70, Dec. 2007. [Online]. Available: `http://www.mitpressjournals.org/doi/abs/10.1162/comj.2007.31.4.59`

[21] B. Degazio, "A Computer-based Editor for Lerdahl and Jackendoff's Rhythmic Structures," *Proceedings of International Computer Music Conference*, 1996.

[22] N. Collins, "A Microtonal Tempo Canon Generator After Nancarrow and Jaffe," in *Proceedings of the International Computer Music Conference*, 2003.

[23] I. Xenakis, "More Thorough Stochastic Music," *Proceedings of International Computer Music Conference*, 1991.

[24] D. P. Creasey, D. M. Howard, and A. M. Tyrrell, "The Timbral Object - An Alternative Route to the Control of Timbre Space," *Proceedings of International Computer Music Conference*, 1996.

[25] N. Osaka, "Toward Construction of a Timbre Theory for Music Composition Composition," *Proceedings of International Computer Music Conference*, 2004.

[26] J. C. Seymour, “Computer-assisted Composition in Equal Tunings: Tonal Congnition and the *Thirteen Tone March*,” *Proceedings of International Computer Music Conference*, 2007.

[27] A. Gräf, “On Musical Scale Rationalization,” *Proceedings of International Computer Music Conference*, 2006.

[28] C. Ariza, “Ornament as Data Structure : An Algorithmic Model Based on Micro-Rhythms of Csángó Laments and Funeral Music Music of the Csángó,” *Proceedings of International Computer Music Conference*, 2003.

[29] W. Chico-Töpfer, “AVA: An Experimental, Grammar/Case-based Composition System to Variate Music Automatically Through the Generation of Scheme Series,” *Proceedings of International Computer Music Conference*, 1998.

[30] N. Collins, “Musical Form and Algorithmic Composition,” *Contemporary Music Review*, vol. 28, no. 1, pp. 103–114, Feb. 2009.

[31] J.-C. Derniame, B. A. Kaba, and D. Wastell, *Software Process: Principles, Methodology, and Technology*. Springer, 1999.

[32] I. Xenakis, *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press, 1992.

[33] L. Smith, “SCORE- A Musician’s Approach to Computer Music,” *Journal of the Audio Engineering Society*, vol. 20, no. 1, pp. 7–14, 1972.

[34] H.-W. Nienhuys and J. Nieuwenhuizen, “LilyPond, A System for Automated Music Engraving,” in *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*. Citeseer, 2003, pp. 167–172.

[35] H. H. Hoos, K. Hamel, K. Renz, and J. Kilian, “The GUIDO Notation Format- A Novel Approach for Adequatly Representing Score-level Music,” *Proceedings of International Computer Music Conference*, 1998.

[36] K. Hamel, “NoteAbility Reference Manual,” 1997.

[37] D. Psenicka, “FOMUS , a Music Notation Software Package for Computer Music Composers,” *Proceedings of the International Computer Music Conference*, pp. 75–78, 2006.

[38] ——, “Automated Score Generation with FOMUS,” *Proceedings of the International Computer Music Conference*, pp. 69–72, 2009.

[39] AVID. Plugins for Sibelius. [Online]. Available: `http://www.sibelius.com/download/plugins/index.html?help=write`

[40] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, “Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic,” *Computer Music Journal*, vol. 23, no. 3, pp. pp. 59–72, 1999. [Online]. Available: `http://www.jstor.org/stable/3681240`

[41] M. Laurson, M. Kuuskankare, and V. Norilo, “An Overview of PWGL, a Visual Programming Environment for Music,” *Computer Music Journal*, vol. 33, no. 1, pp. 19–31, 2009.

[42] A. Agostini and D. Ghisi, “Real-time Computer-aided Composition with BACH,” *Contemporary Music Review*, vol. 32, no. 1, pp. 41–48, 2013.

[43] L. Polansky, “HMSL (Hierarchical Music Specification Language): A Theoretical Overview,” *Perspectives of New Music*, vol. 28, no. 2, 1990.

[44] N. Didkovsky and P. Burk, “Java Music Specification Language, an introduction and overview,” in *Proceedings of the International Computer Music Conference*, 2001, pp. 123–126.

[45] H. Taube, “Common Music: A Music Composition Language in Common Lisp and CLOS,” *Computer Music Journal*, pp. 21–32, 1991.

[46] C. Ariza, “An Open Design for Computer-aided Algorithmic Composition: athenaCL,” Ph.D. dissertation, New York University, 2005. [Online]. Available: `http://books.google.com/books?hl=en&amp;lr=&amp;id=XukW-mq76mcC&amp;oi=fnd&amp;pg=PR3&amp;dq=An+Open+Design+for+Computer-Aided+Algorithmic+Composition:+athenacl&amp;ots=bHedXym8ZP&amp;sig=9i2RQINqIVr2Y7sjxeD9e74myxA`

[47] P. Berg, “PILE - A Language for Sound Synthesis,” *Computer Music Journal*, vol. 3, no. 1, pp. 30–41, 1979. [Online]. Available: `http://www.jstor.org/stable/3679754`

[48] E. Selfridge-Field, *Beyond MIDI: The Handbook of Musical Codes*. The MIT Press, 1997.

[49] NIFF Consortium, et al., “NIFF 6a: Notation Interchange File Format,” NIFF Consortium, Tech. Rep., 1995.

[50] M. Good, “MusicXML for Notation and Analysis,” in *The Virtual Score: Representation, Retrieval, Restoration*, ser. Computing in Musicology, W. B. Hewlett and E. Selfridge-Field, Eds. MIT Press, 2001, no. 12, pp. 113–124.

[51] J. R. Trevino, “Compositional and Analytic Applications of Automated Music Notation via Object-oriented Programming,” Ph.D. dissertation, University of California, San Diego, 2013.

[52] C. Ariza and M. Cuthbert, “Modeling Beats, Accents, Beams, and Time Signatures Hierarchically with Music21 Meter Objects,” in *Proceedings of the International Computer Music Conference*, 2010. [Online]. Available: `http://web.mit.edu/music21/papers/2010MeterObjects.pdf`

THE NOTATION OF DYNAMIC LEVELS IN THE PERFORMANCE OF ELECTRONIC MUSIC

Carlo Laurenzi
Ircam, Paris
Carlo.Laurenzi@ircam.fr

Marco Stroppa
Hochschule für Musik und
Darstellende Kunst, Stuttgart
stroppa@mh-stuttgart.de

ABSTRACT

The "sound diffusion" (or "sound projection"), that is, "the projection and the spreading of sound in an acoustic space for a group of listeners" [1], of works for solo electronics or for acoustic instruments and electronics (so called, "mixed pieces"), has always raised the issue of notating the levels to be reproduced during a concert or the correct balance between the electronics and the instruments.

If, in the last decades, some attempts were made by few composers or computer-music designers, mostly in the form of scores, none of these managed to establish a common practice. In addition, little theoretical work has been done so far to address the performative aspects of a piece, that is, to provide just the useful information to the person in charge of the sound diffusion.

Through the discussion of three historical examples and the analysis of two experiences we developed, we will try to identify some possibly general solutions that could be adopted independently on the aesthetic or technological choices of a given piece.

1. PRELIMINARY CONSIDERATIONS

The notation of electronic music has generated only few, often partial, essays. Most of the literature is either quite theoretical [2], or it delves into the automated translation of electronic sounds into a sort of graphical score, such as in [3]. These experiments were mainly aimed at providing ways to analyse purely electronic pieces more deeply than when simply listening to them, to account for the compositional process, or as an attempt to digitally preserve and archive cultural assets [4].

To our knowledge, little theoretical work has been done to tackle the more general issue of how to notate dynamic levels on a score that is to be read by the computer music performer (CMP) who will perform the electronics during a concert. The CMP does not need to be the composer or the first performer of the piece.

Although this task could be programmed on a computer and automated during the concert, a much better result can be achieved when doing it by ear. The listening and musical skills of a human being are, in fact, still much superior to what a machine can realize. The sound diffusion can be adapted to the acoustics of the hall, the properties of the loudspeakers, the whole audio system, the relationship between these and the acoustic image of the instruments on stage, whether they are amplified or not, and, finally, to the emotional reaction of the audience.

As a consequence, most of the time, the dynamic levels are controlled by ear (and by hand) by the CMP or the composer. Often they are only roughly sketched on the score. If a faithful recording will certainly help as a reference, the information is usually insufficient, especially in the case of particular spatial configurations that cannot be reproduced by a stereo recording.

Therefore, the most effective solution is to notate all the information about the sound diffusion directly on the score that will be used during the performance.

To delimit our scope, we will concentrate on the notation of dynamic levels and will not tackle the issue of notating other parameters used for real-time sound processing, such as, for instance, the transposition factor of a harmonizer.

1.1 Levels vs. loudness vs. musical dynamics

Objectively, levels are normally expressed in decibels, a logarithmic unit that is related to the ratio between the value of a given and of a reference sound pressure (usually, either the threshold of audibility, or the maximum available value in a given system)[1].

However, there are other ways to do it: from the point of view of the perception, the dynamic levels are called "loudness" and use phons (a unit that takes into account

[1] See http://en.wikipedia.org/wiki/Decibel (accessed 3/10/2015)

the psycho-acoustic effect of the equal-loudness curves ISO 226:2003)[2]; from a musical point of view, levels are called “dynamics” and use symbols such as *ff, mf, pp*.

Three important factors need to be taken into account: first, the same musical dynamics played by different instruments, or in different ranges of the same instrument, might yield different objective or perceptual levels; second, choices of interpretation play an important role and produce different absolute levels for the same musical dynamics, as pointed out in [5][3]; third, the perception of an acoustic instrument’s crescendo is always associated to the production of a richer and broader spectrum, that is, to a shift of the spectral “centre of gravity” toward a higher value. These spectral aspects differ specifically from each instrument and can be easily demonstrated by recording three sound files at three different dynamics (say, *pp, mf, ff*), clean them from background noises and finally normalize them. Even though they have the same maximum amplitude, their dynamics can be easily and correctly identified.

Hence, simply raising a fader will not be sufficient to convey a real feeling of crescendo, but rather of a sound getting closer. When notating levels into a performance score, which unit should be used: dBs, loudness or musical dynamics?

1.2 Level changes

The notation of levels changes (usually, albeit incorrectly, called crescendo or diminuendo) can use several strategies, like, for instance, crescendo or diminuendo symbols to illustrate the change between adjacent values (Figure 1a), simple straight lines, either with (Figure 1c) or without (Figure 1b) a reference scale of amplitude ranges for each level in the score, or, finally, simple small upward or downward arrows, eventually with some absolute values (Figure 1d-e).

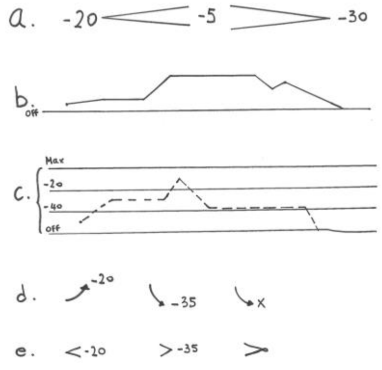


Figure 1. Different ways of notating changes of levels.

[2] See `http://en.wikipedia.org/wiki/Phon` (accessed 3/10/2015)

[3] “The absolute meanings of dynamic markings change, depend on the intended (score defined) and projected (actual) dynamic levels of the surrounding context”, [5] abstract.

These strategies clearly suggest that a compromise between space or information economy and score readability need to be found. Their usage also depends on the nature of the required movements: simply raising a fader to a given static level does not require the same precision as a jagged change over a longer period of time.

2. THREE HISTORICAL EXAMPLES

2.1 K. Stockhausen: Kontakte

Kontakte [6] was originally a 4-channel electronic piece composed in 1958-60 by Karlheinz Stockhausen. Soon after, the composer wrote a version for piano, percussion and the same 4-channel electronic material. The original score shows one of the first, composer-written, attempts to graphically notate the electronic material using unconventional, graphical signs. The second edition, published in 2008, adds some hints at the balance between the amplified instruments and the electronics. In Figure 2, a + above the piano means that the level of the amplification of that instrument should be raised until N (normal) is found.

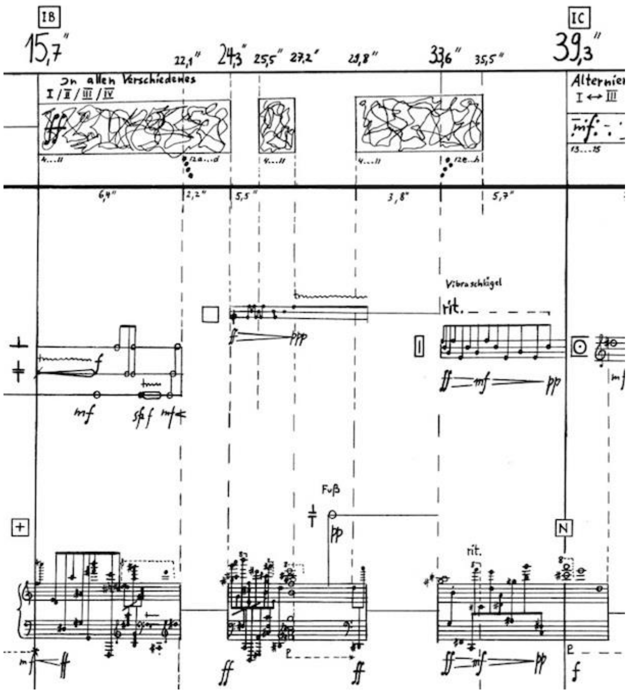


Figure 2. *Kontakte* (p. 1 excerpt, © Stockhausen Stiftung für Musik, Kürten, by kind permission).

Some gestures can be notated, but are hard to realize by hand, as they are very short, as the sudden reinforcement of the, respectively, electronics and marimba (+) in Figure 3.

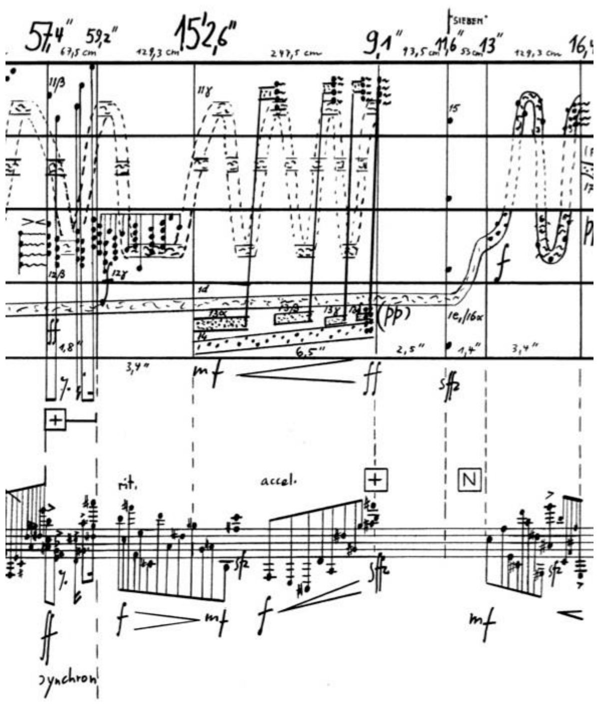


Figure 3. *Kontakte* (p. 17, excerpt, © Stockhausen Stiftung für Musik, Kürten, by kind permission).

On one occasion (Figure 4, page 32), after lowering the electronics (-), the composer explicitly asks for the channels II and IV to be reduced by ca. 5dB, because of a problem of balance in the original mixing, while the channels I and III remain at the N level.

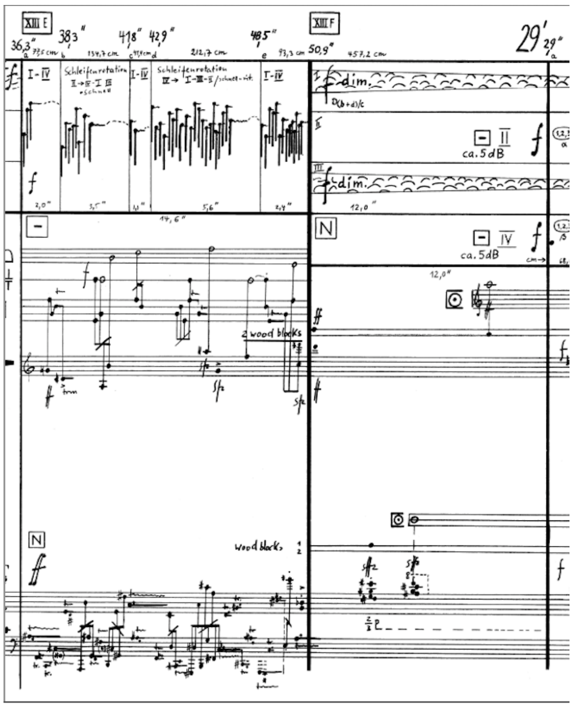


Figure 4. *Kontakte* (p. 32, excerpt, © Stockhausen Stiftung für Musik, Kürten, by kind permission).

To summarize: if the positions and panning of the microphones are very clearly specified in the technical notes that come with the score, information about the sound diffusion, added only in the second edition, is limited to +, - and N (normal) signs. However, this is already sufficient to have an idea of the sound diffusion.

2.2 L. Nono: *A Pierre* / *Omaggio a György Kurtàg*

The late mixed pieces by Luigi Nono make an extensive usage of simple, but continuous live-electronic treatments. Since the original score totally lacked information about the electronics, André Richard and Alvise Vidolin, who assisted the composer during several performances, together with Ricordi's editor Marco Mazzolini, embarked on the ambitious task of notating both the electronic setup and the sound diffusion in such a detailed way, that other people might play the piece without requiring other information than what is marked in the score.

In *A Pierre* [7], for bass flute, double bass clarinet and electronics (4 loudspeakers), the dynamics are marked using a mixture of the strategy shown in Figure 1b and musical dynamics, in spite of the fact, that the latter require both a level and a spectral change to be correctly perceived (Figure 5).

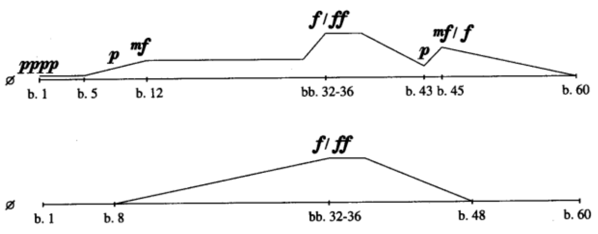


Figure 5. Level changes in L. Nono's *A Pierre*.

In another work, *Omaggio a György Kurtàg* [8], for contralto, flute, clarinet, tuba and electronics (6 loudspeakers), a further distinction is made between microphone faders (M1, M2, etc.), mainly used for sending the sound to the treatments, and output faders (L1-6). In addition, the portion of sound that needs to be recorded by a treatment is greyed in the score (Figure 6).

The notation is adequate to the needs of the composer, and many aspects of it can also be generalized.

2.3 P. Boulez: Anthèmes 2

The Universal Edition performance score of Pierre Boulez's *Anthèmes 2* [9], for violin and electronics, was realized by the composer's musical assistant at Ircam, Andrew Gerzso. Up to now, it is one of the rare examples that features a complete and detailed notation of the electronics (using dedicated staves for each electronic part or treatment). Together with the extensive technical manual, the score allows for the re-constitution of the electronics even without the original patch (Figure 7).

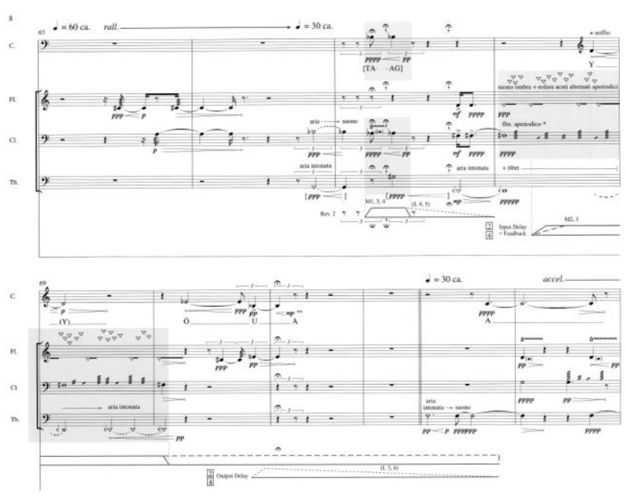


Figure 6.L. Nono's *Omaggio a György Kurtàg* (p. 8, © Casa Ricordi, by kind permission).

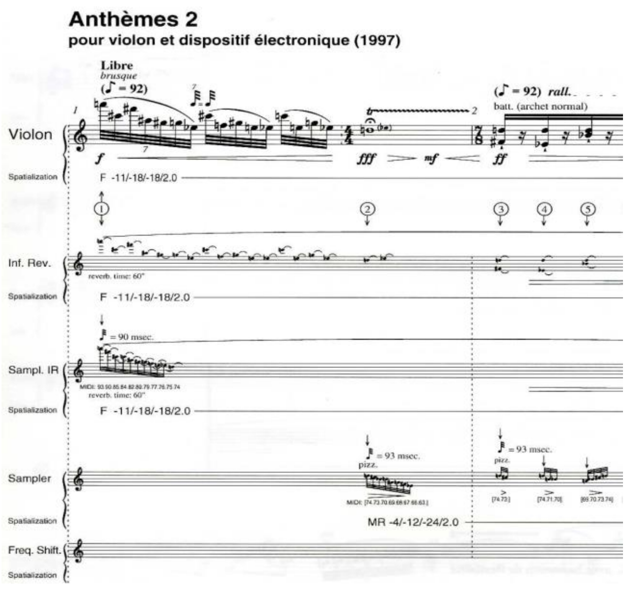


Figure 7. Beginning of P. Boulez's *Anthèmes 2* (© Universal Edition, Wien, by kind permission).

Surprisingly, there are almost no indications about dynamic levels: all the information is, in fact, contained in the Max patch for the piece. The balance between the violin and the electronics is explained in the technical manual and set in the patch. Levels are automated and changed globally, by recalling a different preset for each movement. The presets should be revised during the rehearsals, but, during the concert, only minor adjustments might be required from time to time.

This approach is related to those mixed pieces in which it is mainly the acoustic musician who is responsible for the amplitude of the real-time treatments; the interaction with the CMP, though still important, is therefore less crucial, the work being rather structured around pitches and timbral articulations.

It is therefore clear, that in *Anthèmes 2* the dynamic levels of the electronics play a different role as, for instance, in Nono's works, and, hence, do not need to be notated in the same detailed way.

3. HYPOTHESES

3.1 The case of *Spirali (1987-88)*

3.1.1 Setup

In Marco Stroppa's *Spirali* (Spirals) [10], for string quartet projected into the space, the electronics is constituted by a unique setup, exclusively made of six simultaneous, always active, types of reverb. Placed on stage as far as possible from the audience, the acoustic quartet, closely miked, is amplified and only heard through 4 or 6 loudspeakers around the audience, depending on the size of the hall (Figure 8).

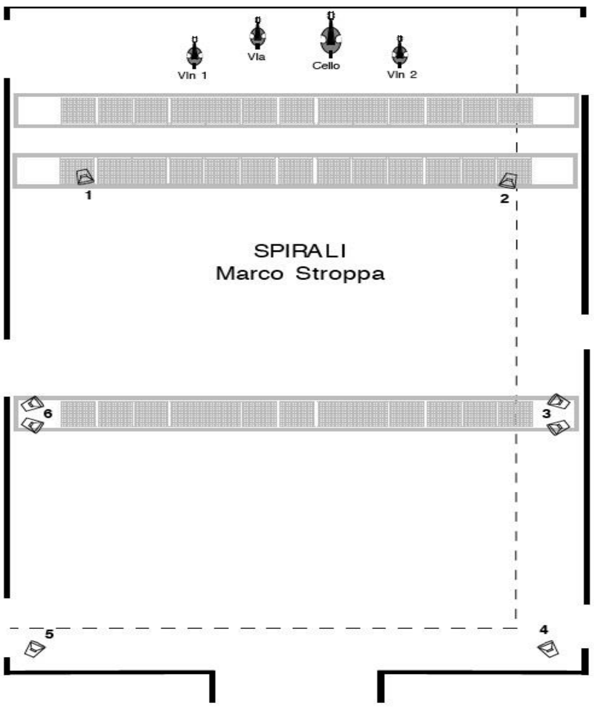


Figure 8. *Spirali*: setup with 6 loudspeakers.

Originally performed with analog equipment, *Spirali* was ported at Ircam by Serge Lemouton in 2005 as a Max patch with 18 control faders. The performance of the electronic part was a terribly virtuoso and risky undertaking and required an extensive study and clear skills! In 2013, Carlo Laurenzi integrated the Antescofo language[4] to the patch and automated some controls. This resulted in a more effective interface, with only 13 faders to move during the performance, although it is still quite challenging to perform.

[4]See `http://repmus.ircam.fr/antescofo/documents` for an abundant bibliography about *Antescofo* (accessed 1/28/2015).

3.1.2 Spatial taxonomy: space families

During the composition, Stroppa organised space into a personal taxonomy made of three space families: points (P), surfaces (S) and diffused space (D). He then related the six reverbs and the amplified instruments to it. Points correspond to the direct amplification of an instrument, to which correspond only one or two loudspeakers depending on the setup (Figure 9)

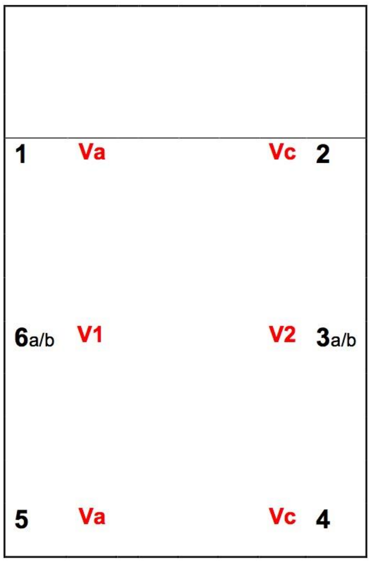


Figure 9. Points: double amplified quartet (6 loudspea-kers)

Surfaces use only the early reflections and cluster stages of reverberation. At each point two adjacent loudspeakers are added, providing a certain spread (called "width") to the sound image. The control of the width size is automated during the performance (Figure 10).

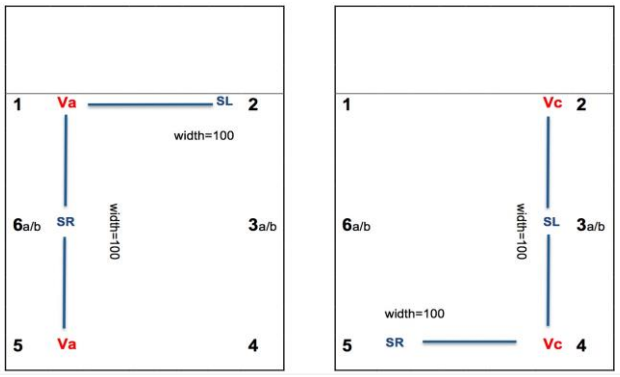


Figure 10. Surface: width spread for the viola and cello.

Finally, the diffused spaces only use the late reverberance, and produce a sound that seems to come from everywhere or… nowhere!

In the performance score, each instrument is considered as one of the voices of the electronics, and is "spatially orchestrated" by the CMP, that is, sent to one or another spatial family depending on what is being played. The final result is an augmented sound image that is not only much larger and deeper than usual, but it also dynamically varies during the performance. The spatial projection hence highlights the frequently used "spiral-like" materials, characterized by musical figures that present similar musical elements across the instruments at slightly different times.

3.1.3 Notational choices

Given these preliminary factors, and after 25 years of performance experience, a definitive musical score for the electronics was established and written immediately below the instrumental parts.

We decided to notate the composed spatial taxonomy directly, by associating a symbol (P, S or D) and a colour (blue, green or red) to each family. The other parameters (spatial width and reverb time) are automatized in Antescofo, but their change is mentioned above the instrumental score, near the event name (see Figure 11, e.254.1-2), since this proved to be a useful reminder for the CMP.

3.1.4 Reference Level

Our hypothesis for notating the dynamic levels is based on the crucial notion of "Reference Level" (RefLev). The RefLev is a perceptual, empirically established value. It depends not only on the audio setup and the characteristics of the hall, but also on the aesthetical preferences of the CMP. We define the RefLev as the level at which the points (the directly amplified instruments) sound "naturally amplified" in the hall and balanced between each other.

Once the RefLev for the points is specified, the RefLev for the other spaces is defined as the level at which they sound "naturally balanced" with the points.

When all the RefLev's are setup, the same physical position of the faders should sound equally loud, in spite of the differences (size of the instrument, position and type of microphones, nature of the spaces, and so on) for all the spaces[5]. This is, of course, a very personal estimation, as it is not easy to compare, for instance, the sound of an amplified violin, coming from one loudspeaker, with a reverberated sound of a cello coming from all the loudspeakers.

At the beginning of the rehearsals, the RefLev's must be empirically and precisely set up. In the score, they are notated with the letter "N" (normal). Notice that the same RefLev may produce a very loud sound, if the musicians are playing *fff*, or a very soft sound, if they are playing *ppp*.

[5] This position, as well as the dynamic curves, can be defined by the user in the patch, but, usually, it is located at about ¾ of a fader's length.

3.1.5 Level changes

Once the RefLev's are defined, all the other levels are notated as a dynamic difference with respect to them and marked with 1 to 3 "+" or "-" (that is, for instance, "+++" or "- -"). They are defined as three clearly different and perceptible dynamic layers: one +/– means slightly louder/softer than the RefLev, two +/– means clearly louder or softer, three +/– are extreme levels, from macro-amplified to barely amplified.

These levels are not absolute, but rather correspond to perceptual areas, and will, therefore, vary during the piece as a function of what kind of music is being performed. They indicate subjectively different "steps" in the amplification process: seven dynamic steps were considered as necessary and sufficient to accurately perform the sound diffusion of *Spirali*.

Since the changes between levels are not very complex, the traditional signs of cresc. and dim. were adopted, because they are expressive, use a space in the score that does not depend on the dynamic range and allow for the notation of a duration (see Figure 11).

3.1.6 Final score

Placed below the instrumental score, once the preliminary choices are clear, the notation of the electronics is quite straightforward (Figure 11).

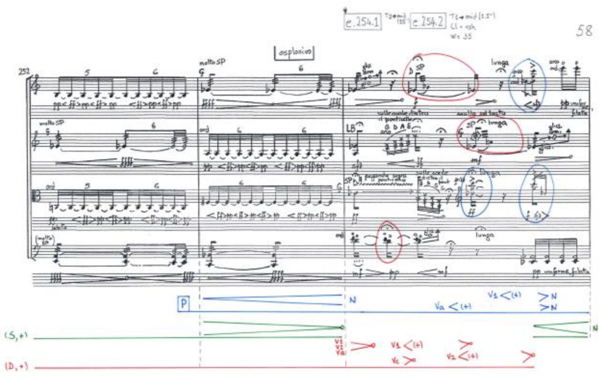


Figure 11. *Spirali*: manuscript score, p. 58 (© Casa Ricordi, by kind permission).

The usage of colours to identify the different spatial families turned to be a very important ergonomic feature, in order to improve the readability of the score. The relation between the notation and the physical gestures needed to operate the control faders becomes more straightforward and faster to learn.

In addition, the isolation of single elements in the instrumental score, using the same colour as the space they belong to, helps to focus on the correct timing and action to perform, especially if the passage is short and/or difficult to perform.

Finally, if printing a score in colours is still not very diffused, because of the production costs, generating a coloured PDF file and performing *Spirali* reading the score on a computer or a tablet already seems very reasonable.

Notice that the acoustic string quartet should not be aware of what is going on in the space, as the spatial changes risk to negatively influence the quality and accuracy of the interpretation. It just has to play!

3.2 Levels of sound synthesis: the case of *Traiettoria*

3.2.1 Setup

Traiettoria [11] is a 45' long cycle of three pieces for piano and computer-synthesized sounds written by M. Stroppa in the early 80s.

The electronics is solely made of eight stereo sound files (from ca. 3' to 7' long), which exclusively use additive synthesis and frequency modulation, with no reference to the piano's spectral structure. A strong connection with the instrument is established by "tuning" the electronic material to some harmonic structures played by the piano. The integration between the synthetic and the acoustic materials is very deeply structured, and can produce a compelling fusion, if the electronics is correctly performed!

The piano and the electronics are loosely synchronised by means of temporal pivots [12].

3.2.2 Spatial families

The sound diffusion of *Traiettoria* is composed of two main spaces:

a. a <u>reduced space</u>, made of the amplified piano (2 loudspeakers placed near the instrument) and of one loudspeaker facing the piano's sound board and placed under the instrument, from which a mono version of the electronics is diffused, so as to sympathetically interfere with the resonating strings.
b. an <u>enlarged space</u>, around the audience, uniquely reserved to the electronic sounds.

The constitution of the enlarged space was not specified in the original score, and could span from two loudspeakers behind the audience to a whole Acousmonium[6]. Ideally, the more loudspeakers are at avail, the more dimensions the enlarged space may have, and, therefore, the more subtle and expressive the spatial nuances can be. But the difficulty of the electronic performance is significantly increased!

After several decades of experience, and thanks to the work of Carlo Laurenzi at Ircam, the electronics was implemented in Max. As in *Spirali*, a spatial taxonomy was defined, but, this time, only as a result of the perfor-

[6] See `http://fr.wikipedia.org/wiki/Acousmonium` (accessed 1/28/2015).

mances with several different audio systems and configurations, and not when the piece was composed. Then, a suggested, standard taxonomy for the sound diffusion was defined: 7 families of spaces (totalling 11 main loudspeakers, see Figure 12). Each family is given a name and a symbol and is controlled by one fader: FC (Front Centre), Pf (Piano), U (Under the piano), F[R/L] (Front [Left/Right]), M[L/R] (Middle), R[L/R] (Rear), RC (Rear Centre). It is for this taxonomy that a new notation was established.

3.2.3 Notational choices

When *Traiettoria…deviata* was first published, it was provided with a unique, exhaustive notation of the synthetic sounds [13], a simple notation of the two main diffusion spaces (M=under the piano, D/S = left/right) and a double time staff (Tpo, Figure 13). The absolute times placed in the middle of the time staves are temporal pivots, the other markings belong to either the piano or the electronics[7].

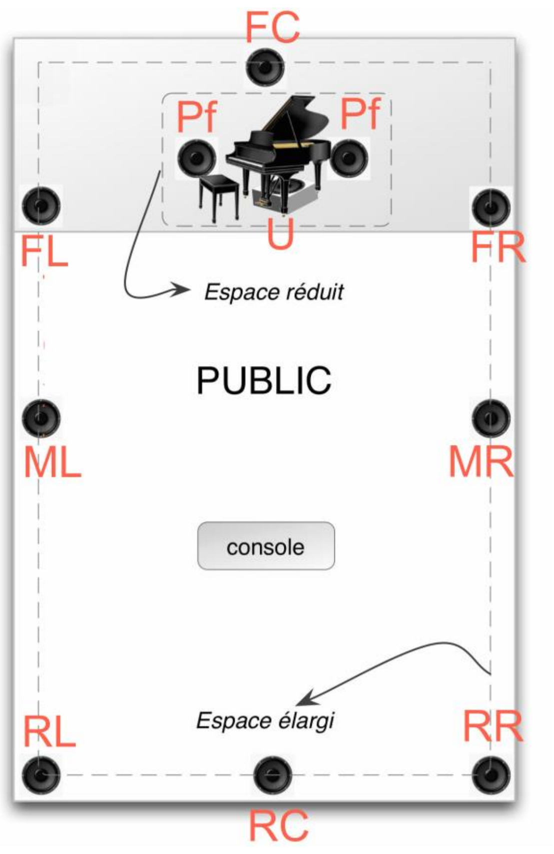


Figure 12. *Traiettoria*: standard audio setup.

Notice that the traditional cresc/dim signs are used, but that the composer explicitly asks for a shift of the spectral centre of gravity toward a higher region together with the movement of the faders. This was done with a HP-filter placed on the electronics' stereo input moved together with the fader.

As impressive as it may look, this notation proved not to be very practical for the sound diffusion. It contained too much information that was not required during a concert and too little information regarding the actual spreading of sound.

Finally, its "orchestral" appearance made it difficult for the pianist to grasp which sounds are easier to hear, and therefore to visually identify the essential cues corresponding to the temporal pivots to which the performance had to be synchronised. A more pragmatic and expressively efficient solution had to be found.

3.2.4 Reference Level

Based on our experience with *Spirali*, we defined a RefLev for *Traiettoria* as the subjective level at which the piano sounds "naturally amplified", and the electronics "naturally balanced" with it. However, here, it did not seem necessary to explicitly mark it in the score (with N). Three degrees of +/- indicate, as in *Spirali*, six perceptually different dynamics for the piano or the electronics.

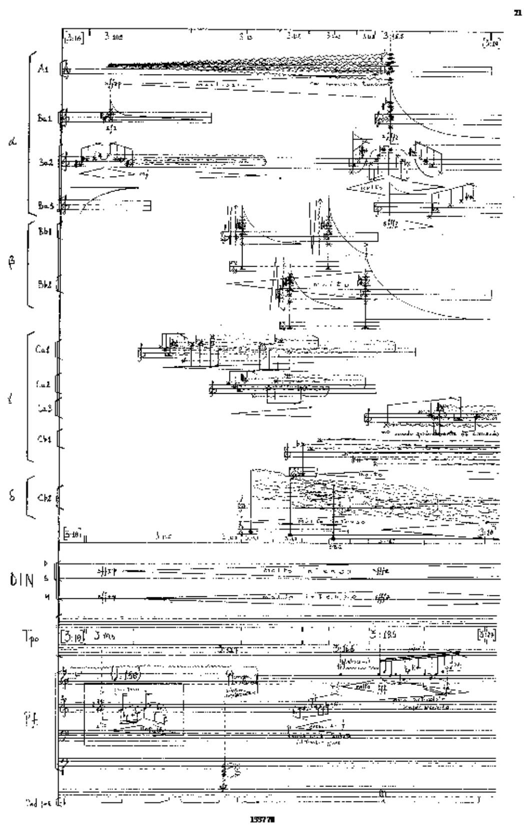


Figure 13. *Traiettoria…deviata*: original version, p. 21 (© Casa Ricordi, by kind permission).

[7] Since the electronics has to be tuned to the piano's A by slightly changing the reading speed, these times are not meant to be strictly followed, but to serve as an indication. Because of this, the usage of a stopwatch would simply not be precise enough. None of the pianists with whom we have worked ever used one during a concert.

During the performance of *Traiettoria,* the most difficult task is to find a musical balance between the sound in the hall and the piano (and some electronics) on stage. How to compare, for instance, an electronic sound coming from behind the audience with the piano? When the same level is indicated in the score, it is the task of the CMP to (subjectively) estimate the correct sound image and intensity.

3.2.5 Composition of the sound diffusion

Even though, in theory, there are as many ways to perform the sound diffusion of *Traiettoria* as there are concerts, the practical experience showed that some strategies were more musical and tended to be regularly repeated.

In the tradition of the acousmatic music, the sound diffusion is thought as a real orchestration of the electronic voices over a moving, imaginary space. Stroppa composed a precise hierarchy that organises not only the audio setup, but also the spatial form of *Traiettoria*.

For instance, *Traiettoria…deviata* starts with a barely amplified piano that gets increasingly louder, that is, more amplified. This yields a larger and larger sound image. When the electronics joins in, it fades into the piano's decaying resonance, and comes out only from U (see 3.2.2). Little by little, the constricted space of the electronics opens up to the Pf and the F groups, thus unfolding its image around the piano. It is only at 1'57 that the R group is activated. A detailed analysis of the spatial form of the sound diffusion of *Traiettoria* is beyond the score of this text, but it is important to remark that, since it is an important part of the composition of the piece, it needs to be precisely and correctly notated.

Each spatial group is represented by one fader on the control interface[8] and by one vertical position in the score. Since each group is identified by a letter, it needs to appear in the score only when it is active. In this way, the usage of the space within the page is more efficient.

3.2.6 Level changes

It did not seem necessary to find a more refined way to notate level changes than what was used in *Spirali*. In the few moments, where a random spread is needed, it is directly asked for by some text written in the score and each CMP can freely choose how to perform it.

3.2.7 Main/Secondary loudspeaker(s)

Together with the taxonomy explained in 3.2.2, the sound diffusion of *Traiettoria* extends the concept of loudspeaker. Each spatial family, identified by a letter, represents the "main loudspeaker", defined as the loudspeaker (or the couple of loudspeakers) that is heard as the main source of diffusion.

It is, however, always possible, depending on the characteristics of the hall or personal taste, to enlarge the focus of a single loudspeaker by diffusing the same electronic material into nearby loudspeakers (called "secondary loudspeakers"), at a softer level, so as to change the acoustic image of the main loudspeaker, without directly perceiving the other ones.

Being rather a performer's aesthetical choice, we decided not to notate this sound-diffusion technique, except when it had a compositional role.

3.2.8 Score

The final score is still under preparation, but concrete experiments and current sketches showed that simply notating the levels above the piano part was not sufficient to achieve a good performance and efficiently learning from the score.

After some tests, we found that adding a sonogram window of a mono mix of the synthetic sounds on top of the page was the best choice to correctly perform the electronics.

Even if a sonogram is very concise and cannot precisely represent pitched and rhythmic material, the most important temporal elements are still clearly identifiable and help both performers to follow the spectromorphological unfolding of the electronics. And if some special pitch or rhythmic structures need to be marked, it is always possible to locally add this information on the sonogram or between it and the dynamic levels.

Thanks to the very explicit images of the sonogram of synthetic sounds, learning the correct synchronization is no longer difficult (Figure 14).

When dealing with several sound files that are inherently unbalanced[9], the sound diffusion can become a tedious and cumbersome task, as each new sound would require a different position of the fader to compensate the inherent lack of balance.

To avoid this problem, a special solution, called "relative faders" (RelFad) was implemented in all the patches for Stroppa's electronic works. Before being multiplied by the value corresponding to the position on the control interface, each RelFad is first multiplied by a value written in the Antescofo score. In this case, if the written values are just right, it is enough to keep the fader at its neutral value (1.0). However, if unpredictable circumstances modify the perception of the diffused sounds, the RelFad can still be moved away from its neutral value.

[8] A MIDI mixer or an OSC-driven device, such as an iPad.

[9] For instance, because they are synthesized with radically different techniques and have extremely dissimilar spectral contents.

As a consequence, the movement of faders during the performance is greatly reduced, and the performance itself becomes more ergonomic and gesture-effective. The written values lay half way between the realm of the composition and of the interpretation and can always be very easily changed. One might also imagine to have presets of good values for different acoustical situations.

Since they were implemented, RelFad's have greatly improved the task of learning to perform the electronics of a mixed piece, and have helped to spread the sound diffusion technique to a larger community of CMP's.

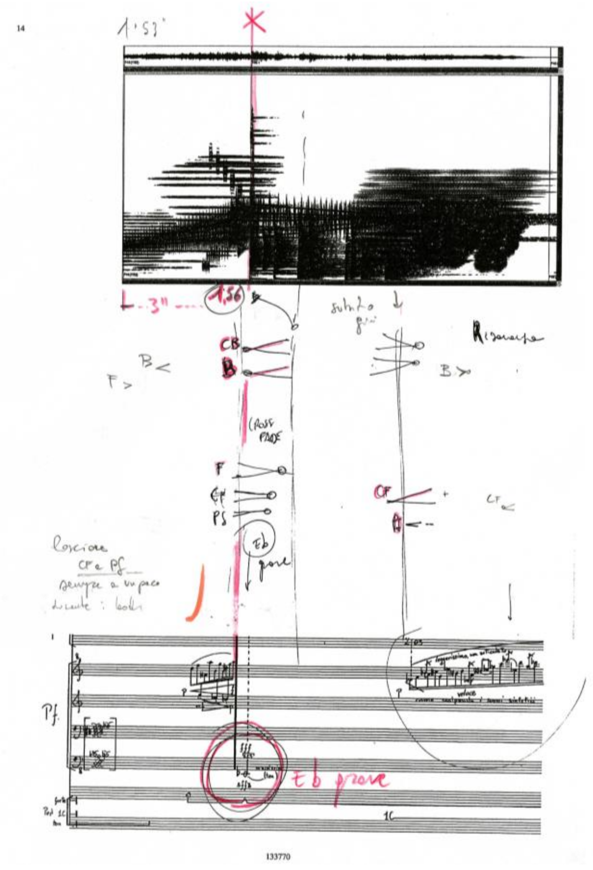


Figure 14. *Traiettoria* : sketch of the new electronic score. Relative faders

4. CURRENT STATE

The notation of dynamic levels in the performance scores of *Spirali* and *Traiettoria* was inspired by the late Nono's works, but the musical context is very different and has a totally diverse goal.

In Nono's works the notation was intended to approximately indicate the behaviour of the levels, in order to provide a schematic structure for the performance of pieces which allowed for a certain degree of improvisation from both the instrumental and electronic parts.

Stroppa, on the other hand, intends to confer a much higher responsibility to role of the CMP, who is required to possess a performance skill comparable to that of an instrumentalist. For this reason, the performance score must contain all the information needed to interpret the piece and accurately represent the time relationships between the acoustic instrument(s) and the electronics.

It is obvious that such a detailed performance score needs some time to be learnt and practiced.

Finally, this score may also have the crucial function, not only to effectively transmit precise information about the sound diffusion to other CMP's, but especially to make it possible to understand how to render a complex orchestration of synchronized spatial events between electronics and instruments.

Due to the complexity of the music and the amount of actions involved in the sound diffusion, learning the score by heart rapidly became a necessity. However, the performance score was still extremely useful during the learning phase and the rehearsals.

5. CONCLUSIONS

Our experience has shown that it is possible to find generalized and efficient symbols to notate the sound diffusion of electronic works, if it is not automated.

Our first step was to identify a spatial taxonomy adapted to a given piece, in order to find an intermediate layer of notation between the compositional concepts, the performance needs and the physical audio setup.

The next step was to define the meaning and the value of a RefLev for each situation and to notate all the other relative dynamic changes with respect to this subjective value. Introducing RelFad's also greatly improved the gestural aspects of a performance.

Our next step will be to extend this experience to the control of real-time treatments.

Acknowledgments

This research was partially made during Stroppa's sabbatical leave from the Hochschule für Musik und Darstellende Kunst in Stuttgart.

6. REFERENCES

[1] L. Austin, "Sound Diffusion in Composition and Performance: An Interview with Denis Smalley" in *Computer Music Journal*, vol. 24, no. 2, pp. 10–21, 2000.

[2] H. Eimert, F. Enkel, and K. Stockhausen, "Fragen der Notation Elektronischer Musik" in *Technische Hausmitteilungen des Nordwestdeutschen Rundfunks*, vol. 6, pp. *52-54*, 1954.

[3] G. Haus, "EMPS: A System for Graphic Transcription of Electronic Music Scores," in *Computer Music Journal*, vol. 7, no. 3, pp. 31–36, 1983.

[4] N. Bernardini, A. Vidolin, "Sustainable Live Electro-acoustic Music", in *Proceedings of the*

International Sound and Music Computing Conference, 2005. `http://server.smcnetwork.org/files/proceedings/2005/Bernardini-Vidolin-SMC05-0.8-FINAL.pdf` (accessed Jan. 28, 2015)

[5] K. Kosta, O. F. Bandtlow, E. Chew: "Practical Implications of Dynamic Markings in the Score: Is piano always piano?", in *Proceedings of Audio Engineering Society 53rd International Conference, London*, 2014.

[6] K. Stockhausen, *Kontakte*, Stockhausen Stiftung für Musik, Kürten, Germany, 2008 (`www.karlheinzstockhausen.org`).

[7] L. Nono, *A Pierre. Dell'azzurro silenzio, inquietum*, Casa Ricordi, Milano, 1985

[8] L. Nono, *Omaggio a György Kurtàg,* Casa Ricordi, Milano, 1983.

[9] P. Boulez, *Anthèmes 2,* Universal Edition, Wien, 1997.

[10] M. Stroppa, *Spirali,* Casa Ricordi, Milano, 1988.

[11] M. Stroppa, *Traiettoria…deviata, Dialoghi, Contrasti*, from *Traiettoria*, Casa Ricordi, Milano, 1982-88.

[12] J. Duthen, M. Stroppa, "Une Représentation de Structures Temporelles par Synchronisation de Pivots", in *Proceedings of the Symposium: Musique et Assistance Informatique*, Marseille, pp. 305-322, 1990.

[13] M. Stroppa, "Un Orchestre Synthétique: Remarques sur une Notation Personnelle", in *Le timbre: Métaphores pour la Composition,* J.B. Barrière Ed., Editions C. Bourgois, Paris, pp. 485-538, 1991.

AUTOMATED REPRESENTATIONS OF TEMPORAL ASPECTS OF ELECTROACOUSTIC MUSIC : RECENT EXPERIMENTS USING PERCEPTUAL MODELS

Dr David Hirst
School of Contemporary Music
Faculty of VCA and MCM, University of Melbourne
dhirst@unimelb.edu.au

ABSTRACT

Within this paper we firstly examine the determination of a number of temporal aspects of Electroacoustic Music, and their representations. Then various automated segmentation methods, for Harrison's *Unsound Objects*, are investigated. We find the multi-granular approach outlined by Lartillot *et al.*, combined with the use of MFCCs, is a very efficient and salient segmentation strategy for music structured predominantly according to timbre. Further, the 'Contrast' parameter is both versatile and effective in determining the granularity of segmentation.

INTRODUCTION

Traditional Electroacoustic Music is a studio-based artform involving the mixing of field recordings, processed field recordings, and synthesized sounds. Electroacoustic Music can also include performance of live electronic instruments in the form of laptops or other electronic devices and/or sensors.
This paper concentrates on studio-based Electroacoustic Music. Being largely an aural tradition, there is no widely accepted standard of notation or representation for this kind of music, either in the creation of the music, or the analysis of this kind of music. Our work seeks to explore ways in which signal analysis and/or perceptual models can assist in automating some aspects of the analysis of Electroacoustic Music in order to augment the aural analysis that is the predominant analytical method for this style of music.

Here we set out three recent attempts to automate analytical aspects of Electroacoustic Music associated with the temporal dimension of the music:

1. The representation of a measure of the activity within a section of an Electroacoustic musical piece, and the associated density of musical events.
2. The use of auditory models to derive a 'Rhythmogram Representation' of both short and long sections of music within a work.
3. Segmentation of Electroacoustic Music works, over a longer time-span, using the Music Information Retrieval Toolbox (MIRToolbox).

MEASURING SONIC ACTIVITY

The Problem Defined

While undertaking a recent analysis of Jonty Harrison's electroacoustic musical work, Unsound Objects [1] the initial phase involved analysing the acoustic surface to identify sound objects. The next phase required an examination of relationships between sound objects, giving rise to the following question: What propels the work along from moment to moment, section to section, scene to scene ? To help answer this question, I observed that an increase in sonic activity seems to elicit expectation in the listener that an important event is about to occur. There is a tension build up that seems to require a release the longer the build up goes on. But how can we measure something I have called "sonic activity" and, even better, how can we display sonic activity easily within a work ? Can some form of signal processing be used and be represented to assist in the interpretation of electroacoustic musical works ?

The Analytical Process

With Electroacoustic Music, the first part of an analysis can be described as analysing the acoustic surface. This involves "segmentation". Large scale segmentation into sections, and then small-scale segmentation of sound events from each other. In the analysis of Unsound Objects, the spectrogram and audio waveform displays were useful for the process. Sound events were annotated on the spectrogram and it was possible to get a time-stamped listing of the annotation layer, using the program Sonic Visualiser [2], which was then imported into a spreadsheet program (Microsoft Excel) and printed as a listing of all the annotations. The visual screens and printed time-stamped sound object listings became the data that facilitated detailed identification and specification of sound events within the aurally identified sections of the work.

The next phase of the analysis involved moving beyond the acoustic surface to examine structures, functions and motions between sound events. By "zooming out" to look at longer sections of the work, or carrying out "time-span reduction", we can observe changing sonic patterns over the course of the work. We can look at the different sections and ask questions like: What propels the work along from moment to moment, section to section, or scene to scene ? To help answer this question, we can observe that an increase in sonic activity seems to elicit expectation in the listener that an important event is about to occur. But how can we measure and, even better, display activity within a work ? Well the Sonic Visualiser program provides access to a suite of plugins of signal analysis. In the Unsound Objects article, I postulated that the type of analysis that seems to correlate best with sound object activity is a plot of "spectral irregularity" versus time.

There are several different methods for calculating the irregularity present within a spectrum, but essentially they both give a measure of the degree of variation of the successive peaks of the spectrum. Jensen, for example, calculates the sum of the square of the difference in amplitude between adjoining partials [3]. What I am postulating here is that where there is a large variation across the spectrum, partial to partial, then this can provide us with a depiction of a high degree of activity. Figure 1 depicts a spectral irregularity plot for the whole of *Unsound Objects*.

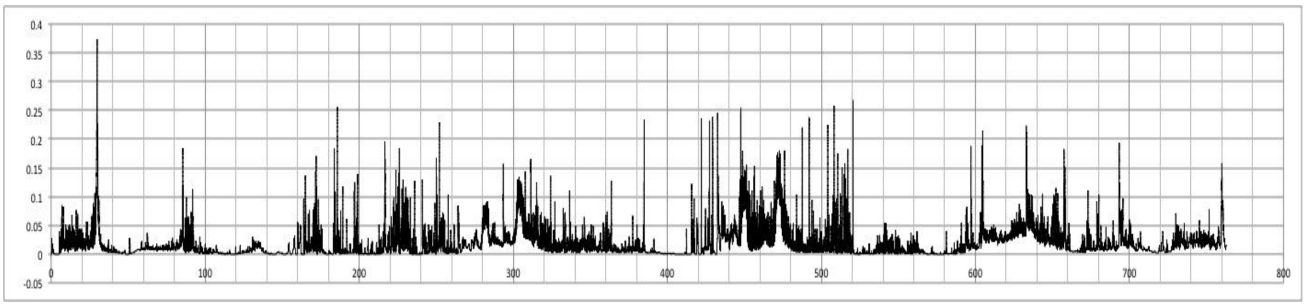


Figure 1 : A spectral irregularity plot for the whole of *Unsound Objects*.

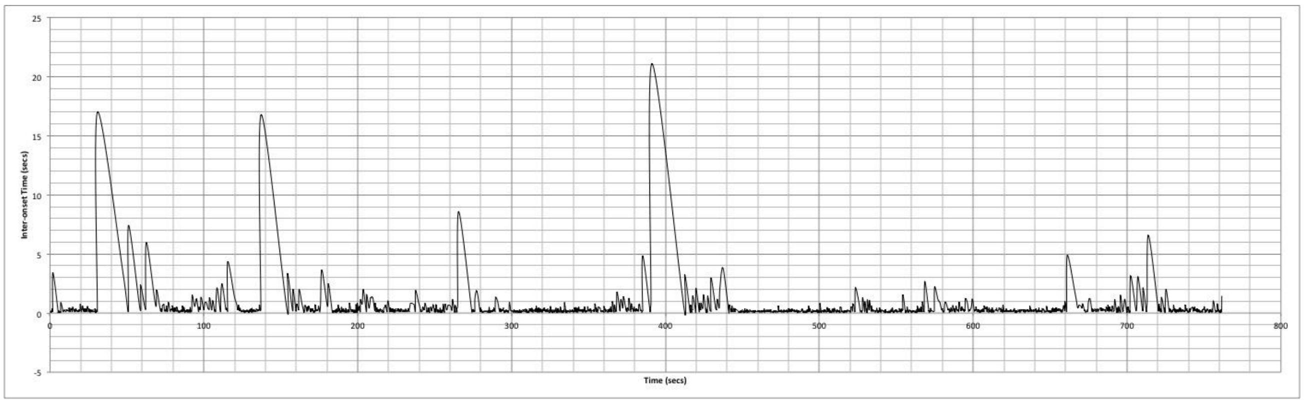


Figure 2 : Plot of Inter-onset Time *vs* Time (secs) for the whole of *Unsound Objects*.

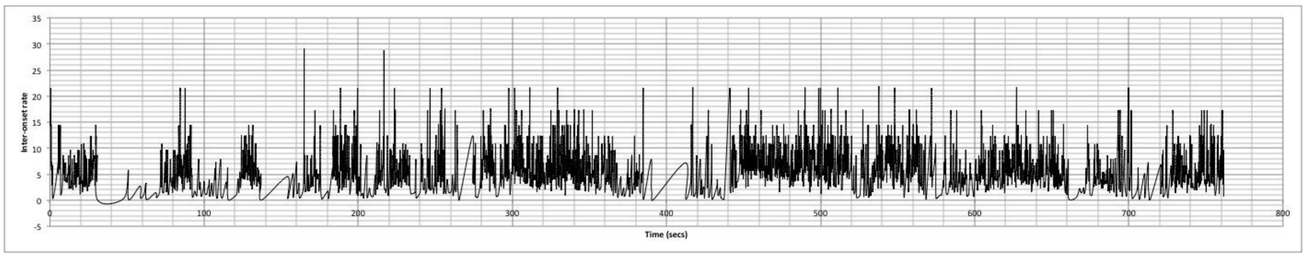


Figure 3. Plot of Inter-onset Rate vs Time (secs) for the whole of *Unsound Objects*.

The analysis of *Unsound Objects* then combined the use of spectral irregularity plots with aurally identified sections, within the work, to provide a detailed analysis of "activity" and to tabulate "sound types" for each section. This table showed "activity amount and type" and "selected sound object types". The work actually divides into two main halves and after the two halves were compared, a summary of sonic archetypes (in the form of mimetic archetypes and structural archetypes), sound transformations, functional relations, and sonic activity were discussed.

Determining Activity

The aim of the next study [4] was to seek an alternative method to the use of "spectral irregularity" for measuring activity in electroacoustic music.

In essence, activity could be defined as the number of sound events in a given time period. Therefore we are interested in the onset time of each sound event, and its duration. Let's start with onset time. What signal analysis tools exist for determining sound event onset time within a musical work ?

The program Sonic Visualiser, has a number of tools within it to perform such an analysis. Aubio onset detection (`aubio.org`) has eight different types which all produce a single list of time "instants" (vertical lines when plotted) of individual start times. This output can be exported to a spreadsheet. Their algorithm can be varied to suit the source material. The Queen Mary, University of London, in-built Sonic Visualiser onset detection algorithm lists three types of onset detector, but these are just the one detector with lots of variables: Program; Onset detection function type; Onset detection sensitivity; Adaptive whitening; Channel options for stereo files; Window size; Window increment; and Window shape. Output is an "onset detection function" which is a probability function of a "note" onset likelihood.

In developing a method for the detection of onsets in *Unsound Objects*, combining several forms of representation was found to provide a more reliable guide to data gathering rather than using any single plot. After some experimentation, the following combination was employed, using the Queen Mary algorithms:

1. RMS Amplitude.
2. Smoothed detection function: Time Values (displays probability function of onsets).
3. Note onsets: Time Instants. Program: Soft Onsets; Onset detection function: Complex Domain; Onset detection sensitivity: 60%; Adaptive whitening: Yes.

This resulted in the onsets (#3 above) aligning pretty well with the smoothed detection probability (#2 above), but with some low level noise swells failing to trigger the onset detector (#3 above).

The "time instants" data (#3 above) was exported, then imported into an Excel spreadsheet in order to be able to make further calculations such as "inter-onset times" (the time between onsets). Figure 2 shows a plot of Inter-onset Time versus Time for the whole of *Unsound Objects.* Its peaks show us where there are long breaks in the work, and give a pointer to how the work may be divided up in analysis.

Displaying time instants, however, only progresses us part of the way to obtaining a measure of event "activity". Inter-onset "rate" was then calculated and plotted, as shown in Figure 3. This provides us with a measure of the number of onsets per second, which, in turn, provides a guide to the amount of event initiation activity at a particular time within the work.

Implications of Activity Plots

Determining inter-onset time can give us a plot (Figure 2) that is useful in showing the main sections within a work. Calculating its reciprocal, inter-onset rate can generate a graph that provides some measure of the varying activity within an electroacoustic work (Figure 3). If we had graphed Figure 3 at the beginning of the analysis, we would have observed that the piece does divide into two, with little activity between about 390 and 410 seconds. The first half begins with three bursts of activity, followed by a longer, more active phase of increasing activity until the "mid-break". The second half is more continuously active until around 660 seconds, where the work has several less active periods, perhaps in preparation for the end of the piece.

In the previous analysis of *Unsound Objects,* sections were first determined aurally, then superimposed over the irregularity plot. Comparing the plot of inter-onset rate (Figure 3) with the irregularity plot (Figure 1) we can see that the piece appears to be much more active in Figure 3 than Figure 1, especially in the second half. The question remains as to which is a better measure of "activity" ? The inter-onset rate is probably a more accurate method, but it seems exaggerated. This is possibly because it doesn't take into account the loudness of the events. Perhaps if this plot (Figure 3) was modified by the RMS amplitude, then a more useful picture of "effective activity" may emerge. There are also inherent definition problems for "iterative" sound events, such as drum rolls or machine sounds. Is such a sound type one long event or many short events ? This phenomenon may skew the events per second data.

In terms of automating analysis, the inter-onset time plot (Figure 2) is very effective in identifying sections in a long musical piece, while the inter-onset rate (Figure 3) does provide a measure of active versus inactive depiction for various passages in a long piece.

The next step in this work was to examine activity and other temporal measures in other works, including more rhythmical pieces.

RHYTHMOGRAM REPRESENTATIONS

This section of our paper introduces work that is well documented in a paper from the ICMC in 2014 [5], but it will be very briefly summarized here to place our subsequent work on automated segmentation into a context of our ongoing work, and to demonstrate some contrasting and varied representations.

Having investigated activity plots, the aim of the next stage of our work was to continue our Segregation, Integration, Assimilation, and Meaning (SIAM) approach of employing a cognitive model [6], in combination with signal processing techniques, to analyse the "raw" audio signal, and more specifically, to depict time-related phenomena (beat, rhythm, accent, meter, phrase, section, motion, stasis, activity, tension, release, etc.). Such depictions should assist or enhance aural analysis of, what is essentially, an aural art-form.

After an extensive literature search, the use of the "rhythmogram" in the analysis of speech rhythm, and the analysis of some tonal music, seemed to fulfill the requirement of a cognition-based method that uses an audio recording as its input signal to produce a plot of the strength of events at certain time points.

The Rhythmogram

In my ICMC 2014 paper [5], I provided a thorough explanation of the rhythmogram, so I will only briefly summarise it here. The framework is documented in Todd [7], Todd & Brown [8] and Marr [9]. It makes use of quite a traditional auditory model where outer and middle ear responses are modelled by filtering, then gammatone filters model the basilar membrane. This is followed by the Meddis [10] inner hair cell model, which outputs the auditory nerve firing probability. It is then summed and processed by a multi-scale Gaussian low-pass filter system. Peaks are detected, summed and plotted on a time constant versus time graph, resulting in a plot known as a rhythmogram.[1]

Figure 4 shows an example rhythmogram for a repeating pattern of three short 50ms tones, followed by a 550ms period of silence, lasting 7 seconds.

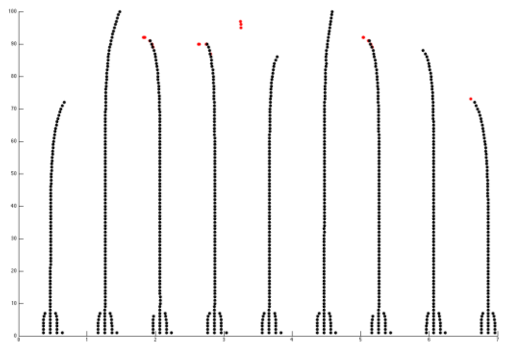


Figure 4. Rhythmogram for a repeating pattern of three short 50ms tones, followed by a 550ms period of silence.

Notable features of the rhythmogram model are:

- Consideration of sensory memory consisting of a short echoic store lasting up to about 200 to 300 ms and a long echoic store lasting for several seconds or more2.
- Each filter channel detects peaks in the response of the short-term memory units.
- The sum of the peaks is accumulated in a simplified model of the long echoic store.
- An "event" activation is associated with the number of memory units that have triggered the peak detector and the height of the memory unit responses.
- The hierarchical tree diagrams of Lerdahl and Jackendoff [12] have visual similarities to rhythmogram plots and so rhythmograms may help the researcher with gaining insights into the hierarchical structure of a musical work under investigation.
- Not only does the rhythmogram model detect the onsets of events, but it can represent other rhythmic grouping structures based on inter-onset times, changes in rhythm, and meter.
- Changing the analysis parameters allows the researcher to "zoom in" or "zoom out", to focus on short-term rhythmic details, or provide a representation of an entire section, or even a complete work.

In the case of the final point above, both of these levels of focus have been explored, and a summarised illustration of both short-term and long-term structures will be recapitulated briefly here.

Analysis of Normandeau's Electroacoustic works

This study utilised the MATLAB code written by Guy Brown, and adapted by Vincent Aubanel for the LISTA

[1] A version of Silcock's schematic [11] for Todd and Brown's model is shown in the Hirst (2014) ICMC paper [5].

[2] Todd (1994), pp. 34-35.

project [13]. The code makes use of the fact that it is possible to increase the efficiency of the computation and still obtain a useful, meaningful rhythmogram plot by using a rectified version of the input signal directly, i.e. bypassing the Gammatone filterbank and inner hair cell stages[3].

The electroacoustic works which were chosen for analysis in this study are collectively known as Robert Normandeau's *Onomatopoeias* Cycle, a cycle of four electroacoustic works dedicated to the voice. The *Onomatopoeias* Cycle consists of four works composed between 1991 and 2009, which share a similar structure of five sections and are of a similar duration of around 15 minutes. The works have been documented by Alexa Woloshyn [14], and by Normandeau himself, in an interview with David Ogborn [15].

Two types of analysis were performed. The first is a detailed rhythmic analysis of a short segment of one of the works. The second analysis zooms out to examine the formal structure of three pieces in the cycle and make comparisons.

Detailed analysis of a short segment of *Spleen*

The work chosen for detailed rhythmic analysis was the second work in the cycle called *Spleen* [16]. This work[4] was chosen as it has a very distinctive beat in various sections and it is slightly unusual for an electroacoustic work in that respect. Figure 5 shows a rhythmogram for the 13.5 second segment of *musique et rythme* from Normandeau's *Spleen*. The X-axis is time (in secs) and the Y-axis is filter number (from 1 to 100). For the full test parameters see [5]. For now we note that the minimum time constant was 10 msec, and the maximum time constant was 500 msec for this test.

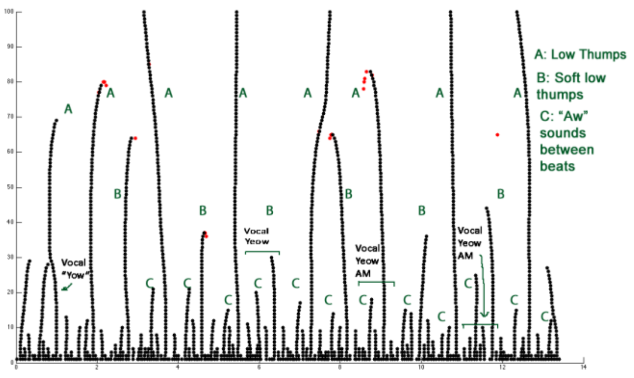


Figure 5. Rhythmogram for 13.5" of *musique et rythme* from *Spleen*.

Labelled as 'A' in Figure 5, the tallest spikes correspond with a "low thump", somewhat like a bass drum. Using these spikes we could even infer a tempo from their regularity. Labelled as 'B' and "soft low thumps" in figure 5, these softer peaks (B) are interspersed between the louder peaks (A) and are equidistant.

To summarise our observations further we can note that there is a rhythmic background of regular beats, consisting of low thumps, arranged in a hierarchy with softer low thumps interspersed. The "tempo" is around 66 bpm. An implied duple meter results from the loud-soft thump beats alternating.

Against this regular background is a foreground of vocal "yow" shouts. Less regular in their placement, the shouts become elongated to "yeow", and then amplitude modulated to add colour and variety. Although less regular in their placement, the "shouts" always terminate on a "thump" beat and thereby reinforce the regular pulse.

There are finer embellishments too, labelled 'C' in figure 5. This third level of spikes in the rhythmogram depicts events that are placed between thump beats and have a timbre that is somewhere between a saw and a squeaky gate. I'll describe these events as "aw" sounds, and they function as an upbeat to the main thump beat. This "one <u>and</u> two <u>and</u> three <u>and</u> four" pattern has a motoric effect on the passage. The presence of further, shorter, and regular spikes is an indication of more sound events which function to embellish the basic pattern.

Looking at the rhythmogram as a whole, for this passage, we can observe that it tells us there are regular time points in the sound, there is a hierarchy of emphasis in the time points (implying some meter), and a further hierarchy in the sense that there is a background of a regular part (the thumps) and a foreground of less regular vocal shouts. Both the background and the foreground have their own embellishments - anticipation of the events in the case of the former, and an increase in length and use of amplitude modulation, in the case of the latter.

Comparison of whole works from the cycle

The second part of this study involved the use of the rhythmogram in the representation and analysis of whole works. It turns out that the works of Robert Normandeau are ideally suited to this application as well. The *Onomatopoeias Cycle* comprises four works, which consist of the same basic form. Normandeau used the same timeline, but different samples, to create a cycle of works. In 1991 he composed the piece *Éclats de Voix* using samples of children's voices [15]. In 1993 came *Spleen* using the voices of four teenage boys, and in 1995 *Le renard et la rose* used the same timeline with adult

[3] See Todd (1994) "Appendix A.3.3 Input" p. 65.

[4] The first two mins of *musique et rythme* can be heard via the link on the electrocd site:
`http://www.electrocd.com/en/cat/imed_9920/`

voices. The final piece in the cycle is *Palimpseste*, from 2005, and it is dedicated to old age. The first three works were analysed, and rhythmograms were created for them.

As these works are each about 15 minutes long, a different set of analysis parameters was required from the analysis of just a 13.5 second excerpt. After a lot of experimentation, a suitable set of parameters was found. The reader can see [5] for further details, but significantly, the minimum time constant was 0.6 seconds, and the maximum time constant was 30 seconds. These parameters represent a "zoomed out" temporal view of the three pieces.

Figure 6 depicts the rhythmogram (Time *vs* Filter No.) for *Éclats de Voix* for its full duration of around 15 minutes. The alternating grey and white areas mark out the five sections that each piece is divided into - as tabulated by Woloshyn in her paper [14].

There is not the space within the confines of this paper to show the Rhythmograms for all three Normandeau works in the cycle. Neither is there the space to go into our detailed findings, however we can make some indicative comparisons in summary here.

Comparing *Spleen* with *Le renard* we observed similarities between the rhythmic profiles of sections 1, 3, 4 and 5. Comparing the rhythmograms from *Éclats de voix* and *Spleen*, there are some similarities of shape, especially in sections 3, 4 and 5. *Éclats* is more busy than *Spleen*, which is busier than *Le renard et la rose*. Finally, the contrasts become more exaggerated with each piece.

Remarks About Rhythmograms

This initial use of the rhythmogram in the analysis of electroacoustic music has demonstrated that the algorithm is capable of displaying the temporal organization of a short segment with details that may enhance analysis through listening. The algorithm is also flexible, given the careful selection of analysis parameters, in the sense that it can also be used on entire pieces to help elicit information regarding more formal temporal organisational aspects, and to make comparisons with other works.

Some of its short-comings are that it can't solve the separation problems of polyphonic music, rhythmograms can be awkward to interpret, and they also rely on aural analysis. Careful selection of analysis parameters is crucial in obtaining meaningful plots.

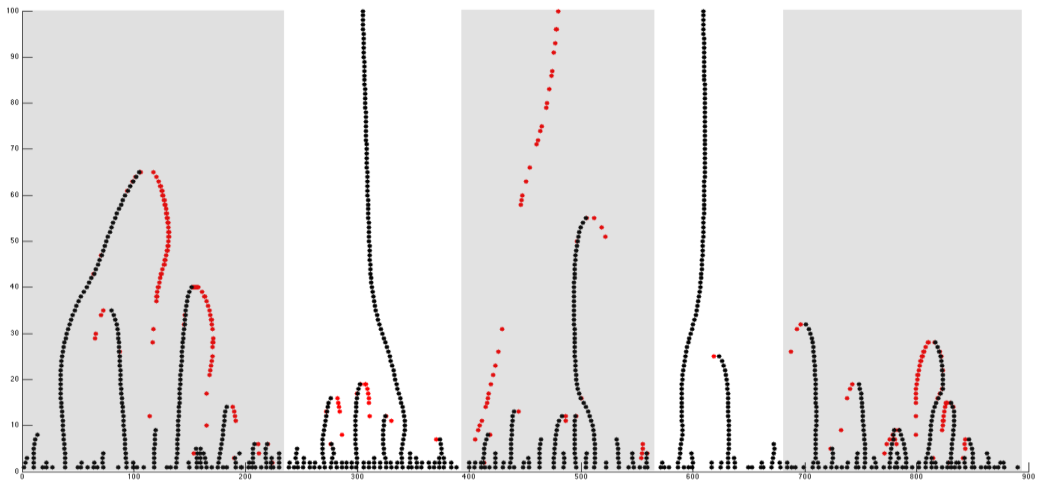


Figure 6. Rhythmogram of the whole of *Éclats de voix* from Normandeau's *Onomatopoeias* cycle.

AUTOMATED SEGMENTATION OF ELECTROACOUSTIC MUSIC

Following on from the investigation of the rhythmogram, the work on the entire Normandeau pieces brought up the research question of whether the segmentation of entire pieces into their sectional constructs could be automated somehow.

Recalling from section 2.2 above, the analysis of *Unsound Objects* began with **analysing the acoustic surface**. This process involves large-scale segmentation into sections, and then small-scale segmentation of sound events from each other.

To explore such segmentation, signal analysis routines from the MIRToolbox [17] were investigated as they represent a collection of auditory perceptual models on the one hand, and a modular approach in their selection and combination, on the other hand.

Automated segmentation Model

For large-scale segmentation, a method for media segmentation, proposed by Foote and Cooper [18], was used as a model. Their method focuses on the notion of self-similarity. Essentially, the spectrum of every time-segment of an audio work is compared with every other time-segment spectrum, and a "similarity matrix" is created for the whole work. Foote and Cooper [18] describe how the work can be divided into sections from the similarity matrix through the construction of a "novelty curve": 'To detect segment boundaries in the audio, a Gaussian-tapered "checkerboard" kernel is correlated along the main diagonal of the similarity matrix. Peaks in the correlation indicate locally novel audio, thus we refer to the correlation as a novelty score'.

Large peaks detected in the resulting time-indexed correlation are then labeled as segment boundaries.

Foote and Cooper go on to describe how they calculate similarity-based clustering to derive the signature of a musical piece, but our work has only proceeded as far as testing the segmentation technique within the electroacoustic musical realm.

Automated Segmentation in Practice Method I

Figures 7 and 8 demonstrate an example of a "novelty curve" and its accompanying segmented audio for the first 3 minutes of Harrison's *Unsound Objects* [19]. Figure 9 shows the sections derived by a human listener superimposed over the spectral irregularity plot for the same extract of *Unsound Objects*. Figure 9 is included for the sake of comparison between automated methods and a human analyst.

Using this segmentation method, the "kernel size" was manipulated to produce section lengths approximating the manual analysis. With a kernel size of 1250 samples, 7 segments were created in the first 3 minutes.

Comparing figures 8 and 9 we can observe that automated segments 1 and 2 (Figure 8) match Section 1 of the manual analysis pretty well (Figure 9). Similarly automated segments 3 and 4 seem to match Section 4, automated 5 and 6 line up with Section 3, and automated segment 7 matches the manual Section 4. At first glance then, this seems quite a useful method of segmentation. However, in deriving this representation, a convolution computation time of nearly 16 minutes is required for a "kernel size" of 1250 samples in the similarity matrix (quite a large kernel size). Clearly a more efficient method was needed.

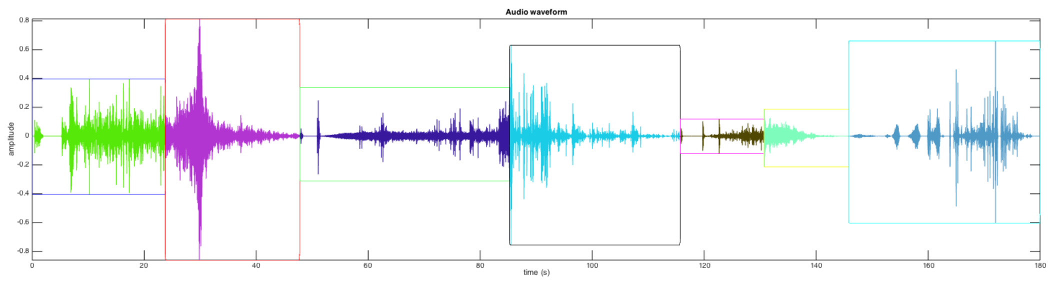


Figure 7. Novelty curve for the first 3 minutes of *Unsound Objects* – Method I.

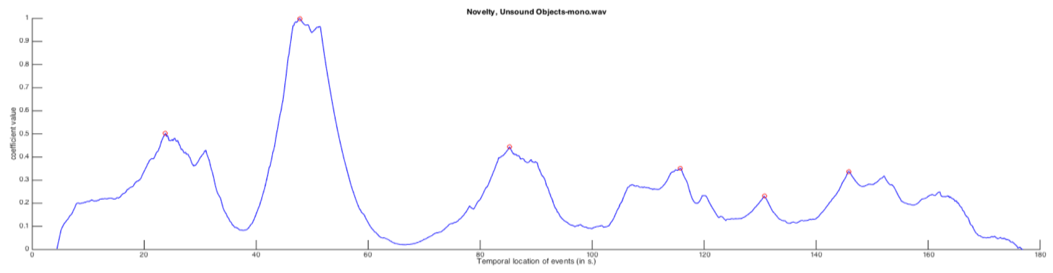


Figure 8. Audio waveform segmented using the novelty curve for the first 3 minutes of *Unsound Objects* – Method I.

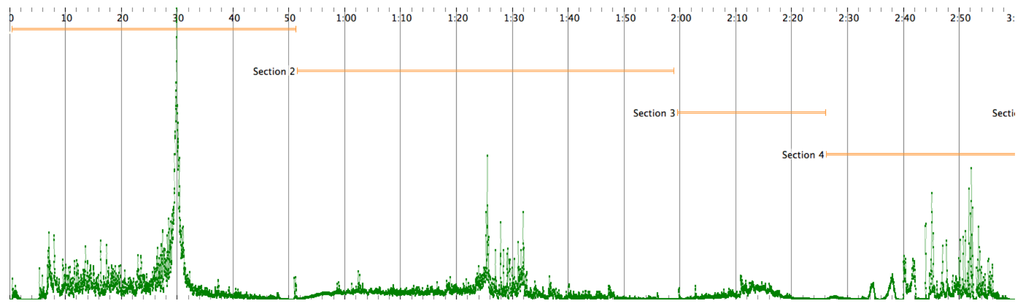


Figure 9 : Irregularity plot with section specification notated by a human listener for the first 3 minutes of *Unsound Objects*.

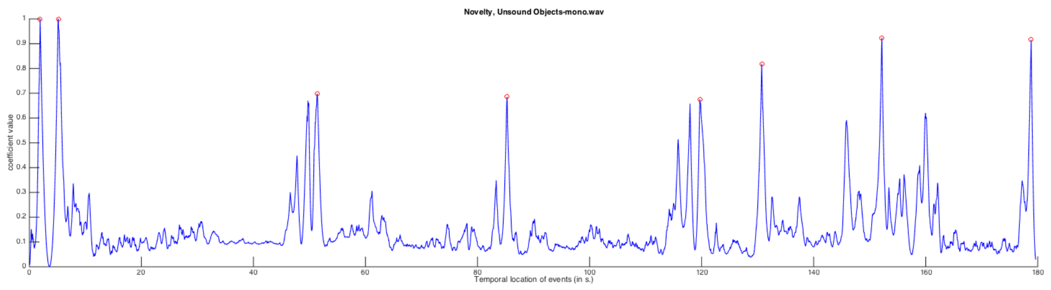


Figure 10 : Novelty curve for the first 3 minutes of *Unsound Objects* – Method II.

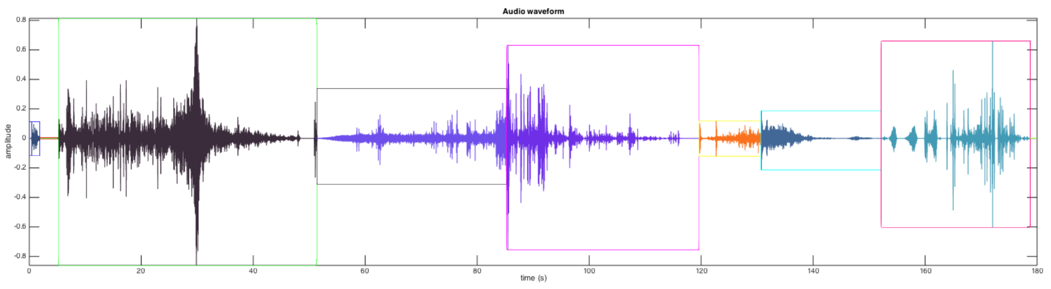


Figure 11 : Audio waveform segmented using the novelty curve for the first 3 minutes of *Unsound Objects* – Method II.

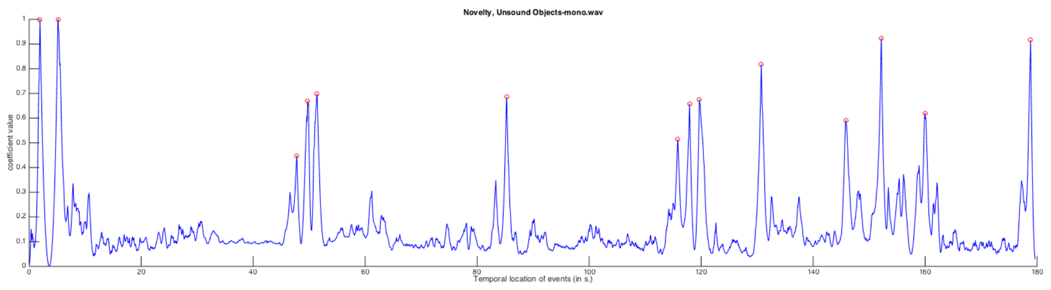


Figure 12 : Novelty curve for the first 3 minutes of *Unsound Objects* – Method II, lower Contrast value.

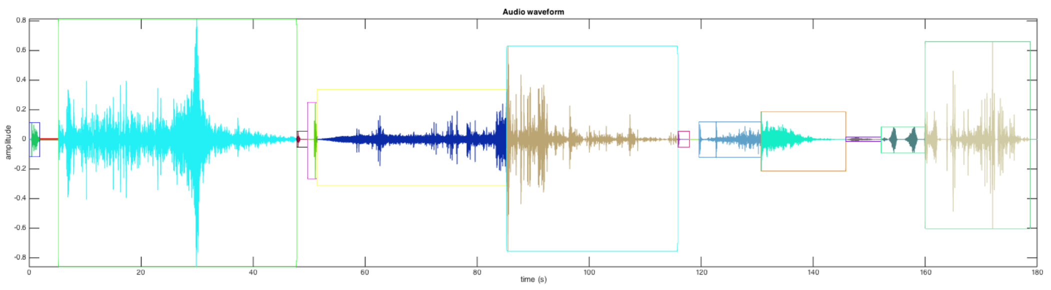


Figure 13. Audio waveform segmented using the Figure 12 novelty curve – Method II, lower Contrast value.

Automated Segmentation in Practice Method II

In Method I, segments are determined from peaks in the *novelty curve*. The *novelty curve* represents the probability along time of the presence of transitions between successive states, indicated by peaks, as well as their relative importance, indicated by the peak heights. For electroacoustic music, we use the spectrum as input to the similarity matrix specification routine. The Kernel based approach is described by Foote and Cooper [18] as follows: ‘Novelty is traditionally computed by comparing – through cross-correlation – local configurations along the diagonal of the similarity matrix with an ideal Gaussian checkerboard kernel.’ That is, every segment of the piece is compared with every other segment to look for similarities and differences. The sequence of operations is: audio in - spectrum - similarity matrix - novelty - convolution - peaks - segmented audio display - novelty score display.

Method II makes use of the simpler, multi-granular approach outlined by Lartillot, Cereghetti, Eliard & Grandjean [20]: ‘For each instant in the piece, novelty is assessed by first determining the temporal scale of the preceding homogeneous part as well as the degree of contrast between that previous part and what just comes next. The idea is to estimate the temporal scale of the previous ending segment as well as the contrastive change before and after the ending of the segment. The novelty value is then represented as a combination of the temporal scale and the amount of contrast’.

Using this multi-granular approach, the following MIRToolbox command yields the novelty curve shown in figure 10 and the segmented audio given in figure 11:

```
mirsegment(a,'Novelty','MFCC','Rank',1:10,'Contrast', 0.6)
```

Note that this method also uses the first ten Mel-Frequency Cepstral Coefficients (MFCCs) in order to decrease computation time, and the ‘Contrast’ level is set at 0.6. With this ‘Contrast’ value there are 8 segments identified in figure 11. These segments correlate quite well with the 4 sections shown in Figure 9 in the following way : Section 1 (segments 1-3); Section 2 (segments 4-5); Section 3 (segments 6-7); and Section 4 (segment 8).

It is also possible to vary the ‘Contrast’ parameter to segment on a shorter-term or longer-term event basis – using the same novelty curve. ‘Contrast’ is defined as: ‘A given local maximum will be considered as a peak if the difference of amplitude with respect to both the previous and successive local minima (when they exist) is higher than the threshold value specified’.

For example, by halving the ‘Contrast’ value to 0.3 (Fig. 12), six additional peaks in the novelty curve are included, and the audio is segmented into 14 segments (Fig. 13). This provides an effective means to vary segmentation from large sections to individual events, depending on the ‘Contrast’ value. In our examples, segmentation is on the basis of timbre, however pitch, rhythm and meter could also be used.

In contrast to the 16 minutes required to calculate segmentation using Method I, Method II is at least four times faster and more efficient.

CONCLUSIONS

Within this paper we have examined the determination of a number of temporal-related analytical aspects of Electroacoustic Music, and their representations. We calculated onset times, inter-onset times, and inter-onset rate for Harrison’s *Unsound Objects.* We explored the use of the “rhythmogram” as a means of hierarchical representation in the works of Normandeau’s *Onomatopoeias* cycle.

Finally we investigated various automated segmentation methods for *Unsound Objects.* We found the multi-granular approach outlined by Lartillot et al, using MFCCs, was a very efficient and salient segmentation strategy for music structured predominantly according to **timbre** (as opposed to pitch or rhythm). Further, the ‘Contrast’ parameter is effective in determining the granularity of segmentation – short events to long sections.

Acknowledgments

Many thanks to Vincent Aubanel, who generously shared his rhythmogram code, and to Olivier Lartillot and Petri Toiviainen for making their MIRToolbox publically available.

REFERENCES

[1] D. Hirst, “Connecting the Objects in Jonty Harrison’s Unsound Objects.*” eOREMA Journal, Vol 1, April, 2013.*

`http://www.orema.dmu.ac.uk/?q=eorema_journal`

[2] C. Cannam, C. Landone, and M. Sandler, “Sonic Visualiser: An Open Source Application for Viewing, Analysing, and Annotating Music Audio Files.” In *Proceedings of the ACM Multimedia 2010 International Conference,* pp. 1467-1468.

[3] K. Jensen, *Timbre Models of Musical Sounds,* Ph.D. dissertation, University of Copenhagen, Rapport Nr. 99/7, 1999.

[4] D. Hirst, “Determining Sonic Activity In Electroacoustic Music.” In *Harmony Proceedings of the Australasian Computer Music Conference 2014.*

Hosted by The Faculty of the Victorian College of the Arts (VCA) and the Melbourne Conservatorium of Music (MCM). 9 – 13 July 2014. pp 57-60.

[5] D. Hirst, "The Use of Rhythmograms in the Analysis of Electro-acoustic Music, with Application to Normandeau's Onomatopoeias Cycle." *Proceedings of the International Computer Music Conference 2014*. Athens, Greece, 14-20 Sept, 2014. pp 248-253.

[6] D. Hirst, *A Cognitive Framework for the Analysis of Acousmatic Music: Analysing Wind Chimes by Denis Smalley* VDM Verlag Dr. Muller Aktiengesellschaft & Co. KG. Saarbrücken, 2008.

[7] N. Todd, The auditory "Primal Sketch": A multiscale model of rhythmic grouping, *Journal of New Music Research*, 23: 1, 25-70. 1994.

[8] N. Todd, and G. Brown, Visualization of Rhythm, Time and Metre. *Artificial Intelligence Review* 10: 253-273. 1996.

[9] D. Marr, *Vision.* Freeman. New York. 1982.

[10] R. Meddis, Simulation of Auditory-Neural Transduction: Further Studies. *J. Acoust. Soc. Am*. 83(3): 1056-1063. 1988.

[11] A. Silcock, *Real-Time 'Rhythmogram' Display*. Report submitted in partial fulfillment of the requirement for the degree of Master of Computing with Honours in Computer Science, Dept. of Computer Science, University of Sheffield. 2012.

[12] F. Lerdahl, and R. Jackendoff, *A Generative Theory of Tonal Music*. Cambridge, Mass. MIT Press. 1983.

[13] G. Brown, and V. Aubanel, Rhythmogram MATLAB code written and adapted for the Listening Talker, LISTA, Project (See `http://listening-talker.org/`)

[14] A. Woloshyn, Wallace Berry's Structural Processes and Electroacoustic Music: A Case study analysis of Robert Normandeau's "*Onomatopoeias*" cycle. *eContact!* 13(3) 2010.

`http://cec.sonus.ca/econtact/13_3/woloshyn_onomatopoeias.html`

[15] D. Ogborn, Interview with Robert Normandeau. *eContact!* 11(2) 2009.

`http://cec.sonus.ca/econtact/11_2/normandeauro_ogborn.html`

[16] R. Normandeau, *Spleen*. On music CD *Tangram*. Empreintes DIGITALes, Montréal (Québec), 1994, IMED-9419/20-CD.

[17] O. Lartillot and P. Toiviainen, "A MATLAB Toolbox For Musical Feature Extraction From Audio" in *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07)*, Bordeaux, France, September 10-15, 2007. Pp DAFX 1-8.

[18] J. Foote and M. Cooper, "Media Segmentation using Self-Similarity Decomposition," in *Proc. SPIE Storage and Retrieval for Multimedia Databases*, Vol. 5021, pp. 167–75, January 2003, San Jose, California.

[19] J. Harrison, *Unsound Objects*. On *Articles indéfinis.* IMED 9627. emprintes DIGITALes, Audio CD, 1996.

[20] O. Lartillot, D. Cereghetti, K. Eliard, and D. Grandjean, "A simple, high-yield method for assess- ing structural novelty", *International Conference on Music and Emotion*, Jyväskylä, 2013.

THE COGNITIVE DIMENSIONS OF MUSIC NOTATIONS

Chris Nash
Department of Computer Science and Creative Technology,
University of the West of England, Bristol, United Kingdom
chris.nash@uwe.ac.uk

ABSTRACT

This paper presents and adapts the *Cognitive Dimensions of Notations* framework (Green and Petre, 1996) for use in designing and analysing notations (and user interfaces) in both digital and traditional music practice and study. Originally developed to research the psychology of programming languages, the framework has since found wider use in both general HCI and music. The paper provides an overview of the framework, its application, and a detailed account of the core cognitive dimensions, each discussed in the context of three music scenarios: the score, Max/MSP, and sequencer/DAW software. Qualitative and quantitative methodologies for applying the framework are presented in closing, highlighting directions for further development of the framework.

1. INTRODUCTION

Music and programming are two creative mediums mediated through notation. In both scenarios, notation is used to describe the behaviour of a system for subsequent execution – be that by computer or human performer. As a representation of a creative domain, notation shapes how the practitioner perceives and interacts with their art. In formal systems, such as computers and common practice music, the design of notation defines what actions and expressions are possible. However, the design of notation and techniques for manipulating can also dispose (or discourage) users to certain actions and formulations – what actions and expressions are easy.

This paper draws on research and findings in the psychology of programming and music HCI to describe a flexible approach to analysing and evaluating notations and user interfaces in a variety of digital and traditional musical practices. It begins with a discussion of the parallels between musical creativity (e.g. composition) and programming, before introducing the *Cognitive Dimensions of Notation* framework. [1] To demonstrate the application and adaptability of the framework, Section 4 explores sixteen core dimensions of notation use through three scenarios of notation-mediated music interaction: sketching and transcription using traditional score; audio/music programming using Max/MSP; and composition and production using a sequencer or digital audio workstation (DAW). Finally, Section 5 offers a survey of methodologies for applying the framework.

2. FROM PROGRAMMING TO MUSIC

There are many parallels between programming and creative musical scenarios such as composition, both in digital interaction and more traditional music practice.

Fundamentally, both practices can be mediated through notation. In Western music, formal training and practice is oriented around the musical score. Composers exploit the flexible affordances of pencil and paper to sketch and experiment with musical ideas, before transcribing their work more formally for communication to the performer, who interprets the notation to realise the written form as music (i.e. sound). The listener, as the consumer, does not see the notation. In programming, developers describe processes and interactive systems in source code, using symbol-based formal languages (such as C/C++, BASIC, or LISP). The code is compiled or interpreted by the computer to create a program that encapsulates some kind of functionality and processing of input and/or output. As in music, the end-user does not see the source code.

In both instances, the formal rules of the notation define what actions and entities can be represented with respect to the creative domain – music or program behaviour. The musical score developed over centuries to efficiently capture the formal rules of Western tonal music, during the common practice period (1600-1900). [2] While this covers a wide gamut of musical practices and styles, and continues to be relevant in modern styles, the format and conventions of the score implicitly shape the creativity of anyone working through it. [3, 4, 5]

Unlike music, no single standard programming notation exists; users have an element of choice over formalisms. Most coding languages are *Turing complete*, meaning they are practically capable of encapsulating any desired computer functionality. Thus, the issue with such notations is not what is possible, but what functionality is easy or quick to code, given the formal rules of the notation. [1] Different languages (and dialects) offer distinctions in syntax and semantics to facilitate different users and uses. For example: *BASIC* is designed using simple English keywords to be easily comprehended by beginners (at the expense of structure); *Assembler* more directly exposes low-level workings of hardware (at the expense of human-readability); and *object-oriented languages*, like *Java* and *C++*, are designed around creating modular systems and abstract data models that map onto user ontologies to enable notation of both low- and high-level concepts. As music notation similarly seeks to support beginners, instrument affordances, and flexible levels of abstract representation, it is instructive to analyse usability factors in notations for programming.

Beyond the format of notation, editing tools also impact the usability of a notation, and although text-based notations can be separated from code editors, other programming paradigms are more integrated with the user experience of the development environment. For example, *visual programming languages (VPLs)*, such as Max/MSP, are manipulated through a graphical user interface, the usability of which impacts how users perceive the language and its capabilities. Other coders develop using an *integrated development environment (IDE)*, offering unified platform for writing, building, running and debugging code. The integration of such tools allows code edits to be quickly tested and evaluated, accelerating the feedback cycle and thus enabling *rapid application development*, in turn facilitating experimentation and ideation. [6] Thus, any approach for analysing notation should likewise address factors in the UI.

In music, similar considerations can be made of the design of interactive modes supported by tools to manipulate notations – be that pencil and paper, ink and printer, or mouse and computer screen. Score notation supports composers in creating music, performers in interpreting it, scholars in analysing it, and learners in understanding it. In each case, practitioners use different techniques and tools to interact with the encapsulated music. Moreover, while music plays a functional role in many aspects of culture, it is also about personal, creative expression, and thus it is important to look at how the development of musical ideas is shaped by the design of notations. To consider this, the following section uses the analogue of programming to adapt an established analysis framework that might be used to reveal limitations, influences and opportunities in music notations and interfaces.

3. A USABILITY FRAMEWORK

The *Cognitive Dimensions of Notations* [1] is a usability framework originally developed by Thomas R. G. Green and Marian Petre, to explore the psychology of interaction with notation in the field of programming, breaking different factors of the software designer's user experience into cognitive dimensions that separately focus on affordances of the notation, but which collectively help to paint a broad picture of the user experience involved with editing code and crafting interactive software systems.

The definitions of each dimension (see Section 4) are borne from research in cognitive science, but shaped to operationalise the framework as a practical analysis tool for use by interaction designers, researchers, and language architects. [7] It is intended that each dimension describe a separate factor in the usability of a notation, offering properties of *granularity* (continuous scale; high/low), *orthogonality* (independent from other dimensions), *polarity* (not good or bad, only more or less desirable in a given context), and *applicability* (broader relevance to any notations).

In practice, these properties cannot always be met. [1, 7] Interactions between dimensions are evident, with either concomitant or inverse relationships. For example, low *viscosity* (~ ease of changing data) contributes to *provisionality* (~ ease of experimentation); whereas, higher *visibility* (~ ease of viewing) may reduce *hidden dependencies* (~ invisible relationships). Moreover, some dimensions are value-laden; intuitively it may be difficult to see how *error proneness*, *hard mental operations*, and *hidden dependencies* are desirable. However, knowledge of these relationships can be useful in solving usability issues, where a solution to one dimension can be addressed through a design manœuvre targeted at another.

The exact set of cognitive dimensions is not fixed, and various proposals for new dimensions, designed to capture aspects of a notation or user experience beyond the original framework, have been forwarded – many arising from its expanded use in other fields in and around HCI (non-programming interaction, tangibles, computer music). New dimensions should be measured against the aforementioned requirements, but their value is most effectively gauged by how much they reveal about the interaction context in question, and arguably the greatest contribution of the framework is that it provides a vocabulary and structure for discussing and analysing notation from multiple perspectives.

As an HCI tool (and in contrast to other usability methodologies), it allows both broad and detailed analysis of human factors in a notation or user interface, adaptable to different use cases and audiences. By considering each cognitive dimension in the context of a specific system, designers and evaluators can assess how the notation fits their user or activity type, whether that's making end-user systems easier to use [5] or making musical interaction more rewarding by increasing challenge. [8, 9]

For a detailed discussion of the background and definition of dimensions in the original framework, see [1]. For further publications on the subject, see the framework's resource site and associated bibliography.[1]

4. DIMENSIONS OF MUSIC NOTATION

In this section, sixteen core dimensions of the framework, adapted for a musical context, are detailed and discussed in the context of three common musical interaction scenarios. To evaluate both formal and informal music notation, each dimension is respectively reviewed in the context of the musical score and sketch (SCORE). The intersection of musical expression and programming is then similarly explored in the context of the Max audio synthesis environment (MAX/MSP). Lastly, the framework is used to review the user interfaces and experiences offered by mainstream end-user systems, through an analysis of digital audio workstation (DAW) and sequencer software (DAW). In addition to a description of the dimension, each is introduced with a simple question designed to encapsulate the definition in a form that can be used to capture feedback from end-users (e.g. a user survey [3, 8, 10, 11]).

4.1 Visibility

"How easy is it to view and find elements or parts of the music during editing?"

This dimension assesses how much of the musical work is visualised in the notation or UI, as well as how easy it is to search and locate specific elements. While hiding data will make it difficult to find, showing too much data can also slow the search. Pages and screens limit the amount of space available for displaying data, requiring a careful balance of visual detail and coverage.

[Related dimensions: *juxtaposability*, *abstraction management*, *hidden dependencies*, *conciseness/diffuseness*, *closeness of mapping*, *role expressiveness*.]

SCORE: In sheet music, all notated elements are visible on the page; there is no feature to dynamically hide notated elements, beyond using separate sheets. However, music is hidden on other pages, where page turns also present
challenges for typesetter or performer, if phrases continue over a join. This can be accounted for in layout, with forethought, but this increases the *premature commitment*. Things are easier for the composer, as a draft musical sketch need not cater for the performer, and pages can be laid side-by-side (see *juxtaposability*). Some aspects of the final musical form (e.g. expression and prosody of performance) may not be visually explicit in the musical score (see *closeness of mapping*).

MAX/MSP: As a visual programming language (VPL), *visibility* is a key dimension of Max, which explicitly represents the flow of audio and musical data. As in many programming languages, the visibility of process (code/data-flow) is prioritised over musical events (data). In Max, many elements of a system are not visualised, such as the internal state of most objects (e.g. default or current values). There is also no inherent linear / serial representation of musical time, making it difficult to sequence past or future events or behaviour. As such, Max best suits generative and reactive (live) applications.

DAW: Like most end-user software, DAWs offer a graphical user interface (GUI) that is inherently visual. However, different sub-devices (views) reveal or hide different properties of the music; no screen provides a comprehensive or primary notation. Notably, the arrange window is the only window designed to provide an overview of the whole piece, but filters low-level detail (e.g. notes), which must be edited through other interfaces (score, piano roll, data list). As a result musical data is dispersed through the UI and can be difficult to find, often involving navigating and scrolling through windows and views with the mouse. Arguably the primary and most expressive interaction medium for the sequencer is inherently non-visible: performance capture (MIDI/audio recording).

4.2 Juxtaposability

"How easy is it to compare elements within the music?"

Related to *visibility*, this dimension assesses how notated music can be compared against other data. Pages and moveable windows allow side-by-side comparison, albeit at some cost to *visibility*. How clearly elements and their purpose are represented will also affect how easy it is to compare notated passages (see *role expressiveness*). Music systems may also provide tools for non-visual comparisons – e.g. sound (see *progressive evaluation*).

[Related dimensions: *visibility*, *consciseness/diffuseness*, *role expressiveness*, *progressive evaluation*]

SCORE: Pages allow side-by-side comparison of elements, and the formal rules for encapsulating music make visual inspection an effective tool for assessing similarity (rhythmic patterns, melodic contour, etc.). However, some musical properties are distinguished more subtly in the visual domain (e.g. harmony, key, transposed parts – see *hidden dependencies*), requiring musicianship as well as notational literacy to enable effective comparison.

[1] http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/

MAX/MSP: Max's windowed system allows side-by-side comparison, so long as *abstraction* (sub-patching) is applied effectively. Groups of objects can be dragged next to each other, but this becomes cumbersome as the patch grows and objects are intricately woven and linked to surrounding objects (see *viscosity* and *premature commitment*). Broad visually similarity and functional similarity may not always align (see *role expressiveness*).

DAW: As in Max, windowed systems allow side-by-side comparisons, though sizing, scrubbing, and scrolling can be cumbersome in the face of many windows, a common issue in traditional linear sequencers [4, 12]. Most visualisations of musical elements are easy to compare to similar properties of other tracks, bars, etc., and generalised
representations (track automation envelopes) also offer a basis for comparison across different musical properties.

4.3 Hidden Dependencies

"How explicit are the relationships between related elements in the notation?"

This definition assesses to what extent the relationships and dependencies (causal or ontological) between elements in the music are clear in the notation. Showing dependencies can improve *visibility*, but there is often a trade-off with editing *viscosity*. For example, in programming, textual source code (e.g. C/C++) can be easily edited, but the relationships between sections of code, functions, and variables are not explicitly shown. However, in *visual programming languages (VPLs)*, objects and variables are linked using arcs, making their functional connection visually explicit, but making it harder to edit, once woven into the rest of the code. [1, 13]

[Related dimensions: *visibility*, *closeness of mapping*, *role expressiveness*, *viscosity*, *conciseness/diffuseness*]

SCORE: The *visibility* of the score ensures no actual data is hidden, except on separate pages, though the musical relationship between notated elements is not always explicit. Some elements are visually linked (e.g. slurs and phrasing) and there are other visual cues that events are related, as in the use of beams or stems to respectively bridge rhythmic or harmonic relationships. However, musical events are sensitive to context, as with dynamic marks, previous performance directions, and key changes – though a visual link between each individual note and the markings that affect its performance would be inefficient to notate explicitly (increasing the *diffuseness*).

MAX/MSP: A key attribute of all VPLs; the graphical connection of elements using patch cables explicitly identifies dependencies between Max objects, and help to show signal flow and the wider architecture of a patch. However, patch execution in Max is also affected by the relative placement and spatial relationship of objects (e.g. right-to-left processing of outlets), which is not visualised explicitly and can lead to unexpected patch behaviour that confuses users. While relations between objects are shown, its specific functional purpose is not explicit and the object's current state or value is hidden. For example, default values specified as arguments can be replaced by messages, but there is no visual indication the value of the object has changed from its displayed default value.

Use of sub-patching can also hide functionality, though this is a common trade-off with the additional expressive power offered by *abstraction* mechanisms. Moreover, as a data-flow environment (and in contrast to imperative programming, as in C++), musical time and the sequence of events are not visually explicit, hiding causal and timing relationships between musical elements.

DAW: The variety of different views and UIs designed for different purposes and perspectives can lead to a large number of hidden dependencies within DAWs. [3] For example, across the different screens and settings there are dozens of variables that impact the final volume of an individual note, and often no explicit visual link between them. Similarly, the routing of audio signals through a DAW is usually not visually illustrated, but dependent on the values of (potentially hidden) drop menus. Some DAWs have attempted to address this: *Tracktion* enforces a left-to-right signal flow where a track's inputs, inserts, send effects, outputs, and other processes are aligned in sequence (in a row) within the tracks of its arrange screen; whereas *Reason* takes a skeuromorphic approach using visual metaphor to the studio, enabling users to inspect and manipulate the wired connections on the back of virtual hardware devices.

4.4 Hard Mental Operations

"When writing music, are there difficult things to work out in your head?"

This dimension assesses the cognitive load placed on users. While this is one of the few dimensions with a prescribed polarity (to be avoided in a user experience), musical immersion, motivation, and enjoyment is predicated on providing a rewarding challenge commensurate with ability, such that music may be one of the few fields where this dimension is to some degree desirable.

[Related dimensions: *consistency, hidden dependencies*]

SCORE: Formal music notation carries a high literacy threshold, making the score inaccessible to untrained or novice users. Moreover, aspects of the score also require experienced musicians to solve problems in their head, such as applying key signature, deducing fingering, etc. (see *hidden dependencies)*. By not notating these elements, scores can be more concise, as well as less prescriptive for interpretation by performers. Interaction with music notation also draws heavily on rote learning and deliberate practice to develop unconscious skills and reflexive playing techniques that would be less efficiently or fluidly performed if mediated through notation.

MAX/MSP: While arithmetic and computation tasks can be offloaded to the Max, some aspects of patch behaviour must be carefully approached. Execution order and causal relationships are not visually explicit in a Max patch; users must comprehend the flow of processing to understand the behaviour of their program. Similarly, the lack of a timeline makes time a more abstract concept, making less process-oriented styles of music harder to create and conceive, unless mentally simulated by the user.

DAW: Pro audio software is created with design principles favouring usability and ease-of-use: to be accessible to musicians and non-computer audiences. The various sub-devices in a DAW allow users to edit data through a UI style suiting their background (score, mixer, piano roll, MIDI instrument, etc.). However, because complexity is hidden from the user, there is some risk of such systems becoming less flexible and more opaque; made of black boxes supporting established artistic workflows (see *closeness of mapping*, *premature commitment*). The apparent disjunction between usability and the virtuosity musicians embrace in other aspects of their practice (performance, score literacy, composition) may suggest that such users would accept the cost of developing skill, when more flexible approaches to musical creativity is the reward, and thus design heuristics based on virtuosity rather than usability may be more apt. [9, 14]

4.5 Progressive Evaluation (Audibility / Liveness)

"How easy is it to stop and check your progress during editing?"

This dimension details how easy it is to gain domain feedback on the notated work during editing. How complete must the work be before it can be executed?

In music, this is defined by what facilities are available to audition the sound or performance of the music. 'Liveness', another concept adapted from programming, defines the immediacy and richness of domain feedback available in the manipulation of notation [15, 16], and is a key factor in the user's feeling of immersion in the creative process and domains such as music. [11,17]

[Related dimensions: *provisionality*, *premature commitment*, *hard mental operations*]

SCORE: Musical feedback is available through manually playing, sight-reading, and auditioning the notated music using an instrument. Material can be evaluated through performance (possibly requiring transposition) on various instruments – commonly, a piano. Crucially, the piece needn't be complete (or 'correct') to audition individual phrases or parts. Moreover, lo-fidelity musical scores (sketches) allow unfinished, informal notation of ideas that can still be interpreted by the composer. There may, however, be a disparity between notated forms and a musical performance, where performers may add their own interpretations to the notes on the page (individual prosody, articulation, *rubato*, etc.). Simulation of material on a different instrument also relies on the composer's knowledge of the target instrument and related technique – e.g. a piano may be more or less musically flexible, and offer a different timbre to the target instrument.

MAX/MSP: The environment allows patches to be run at any time, though they must be coherent and syntactically correct to evaluate the sound design or music. Good programming practice encourages a modular approach that allows sub-components and simpler configurations to be tested individually, early in development, though its function and output might be abstracted from the final sonic intent of the code.

DAW: The timeline, mixer, playback and track controls (e.g. mute, solo) enable the user flexible control of listening back to musical data and auditioning individual edits. A piece can be auditioned as it is built up, track-by-track, bar-by-bar, or note-by-note, and there is no requirement that the 'solution' or notated form be musically correct or coherent to be heard. The rigid UI prevents the entry of non-sensical data, and informal or ambiguous directions (see *secondary notation*) cannot be auditioned. For digitally-produced music, the sound output offers an exact representation of the music notated in the UI.

Sequencers designed to accelerate the edit-audition cycle enable a higher level of liveness in the user experience of notation-mediated digital music systems, as evidenced by loop- and pattern-based sequencer software such as *Ableton Live* and most soundtrackers [11], which focus editing on shorter excerpts of music, shortening the feedback cycle. This contrasts the unbroken linear timelines of traditional sequencers, where (beyond the literally live experience of recording), interaction styles for editing and arranging parts offer lower liveness.

4.6 Conciseness / Diffuseness

"How concise is the notation? What is the balance between detail and overview?"

This dimension assesses the use of space in a notation. Both pages and screens have limited space, and both the *visibility* and *viscosity* of a notation suffer when data escapes from focus. Legibility may also suffer if the notation is simply shrunk or packed tightly, such that conciseness must normally be balanced with careful use of *abstraction* mechanisms. In music, composers need to be able to access every detail of a piece, but also able to get a sense of the 'big picture'. [3, 12]

[Related dimensions: *visibility*, *juxtaposability*, *hidden dependencies*, *abstraction management*, *consistency*]

SCORE: The score has evolved to provide a concise representation of music. Unlike digital notations, no abstractions or sub-views are available to hide detail; all elements are always visible, requiring economical use of space. Time is represented using a pseudo-linear scale, where notes are positioned within the bar to reflect relative position in a piece, but bar sizes are compressed such that sparse phrases consume less space. Musical time and page position are further decoupled through the symbolic representation of note duration, such that slow passages (e.g. of semi-breves) do not consume excessive space, but fast passages (e.g. of demi-semi quavers) are expanded to show the detail more clearly. This symbolic encoding of time, however, does lower the *closeness of mapping*, increasing the onus on literacy (*virtuosity*).

MAX/MSP: The layout and density of a Max patch is flexible, though readability suffers when objects are densely packed together or connecting patchcords obscure each other. When complex patches grow outside the confines of a window, *visibility* suffers and mouse-based interaction can be cumbersome. *Abstraction* mechanisms such as sub-patching are critical in managing complex systems and avoiding sprawling patches, but trade *diffuseness* over screen space for *diffuseness* over separate, possibly hidden windows.

DAW: The variety of notations and views in DAWs offer a varied level of *conciseness*. The arrange view sacrifices *visibility* of data to accommodate a broader overview of a piece in the UI. Part editors, like the score and piano roll interfaces, offer more detail (in a manner similar to the traditional score), but only partial views of the entire work. More generally, the lack of a comprehensive principle notation or interface means that information is diffused over different views within the program. Many DAWs do little to optimise window management, navigation, or searching, compounding interaction issues.

4.7 Provisionality

"Is it possible to sketch things out and play with ideas without being too precise about the exact result?"

This dimension assesses how easy it is to experiment with new ideas through the notation or UI, and how fully formed those ideas must be. Accordingly, it is a critical factor in a musical system's support for sketching, ideation, and exploratory creativity. [3, 5, 11, 16] In digital systems, an 'undo' facility significantly contributes to *provisionality*, allowing inputs and edits ('what if' scenarios) to be trialled and reversed, reducing *premature commitment* to a particular approach [1] – reducing the risk of trying new ideas. The dimension is closely related to *viscosity* and *progressive evaluation*, where the ease and flexibility of editing and auditioning similarly facilitates exploring new ideas. *Secondary notation* also offers the opportunity to make incomplete or informal remarks, but in a non-executable form that can't be auditioned.

[Related dimensions: *premature commitment*, *viscosity*, *progressive evaluation*, *secondary notation*]

SCORE: In a musical sketch, the affordances of paper and pencil support a powerful and flexible medium for capturing part-formed ideas. [5, 18] Pencil can be easily and quickly erased, facilitating experimentation and ideation. By contrast, the formality of the typeset, printed ink manuscript is less flexible and more permanent, used only to finalise a composition for archiving or communication (e.g. to performers). These two instances of score notation compliment each other in an established ecosystem that facilitates both composition (creativity) and performance (production) (cf. [19]).

MAX/MSP: The visual drag-&-drop, interactive debugging environment of Max facilitates its use as a rapid prototyping tool, useful in the exploratory design of new audio processing tools and synthesis techniques [13] – though some more involved musical constructs or expressions can be harder to develop or articulate quickly, reducing *provisionality* and ideation. Conversely, as a prototyping tool, Max's focus on experimentation and early stage creativity comes at the expense of subsequent stages of the creative process ("productivity" [19]): finalisation, refinement, and continued development of designs (e.g. for consumption by end-users, non-programmers, and other musicians) is normally conducted using other development tools (e.g. C/C++).

DAW: Like other authoring tools, DAWs offer multiple ways of quickly adding, editing and deleting elements in the document (i.e. musical piece). Moreover, the presence of 'undo' functionality makes it easy to backtrack actions, reducing the risk of experimenting with new ideas, encouraging ideation [1]. The primary mode of input – digital audio or MIDI performance capture – in combination with practically unlimited storage (length, tracks, etc.) represents an improvement in provisionality over historic recording techniques (e.g. tape). Users can also address issues in live recordings using advanced overdub tools, without recourse to re-recording entire performances. Offline editing, through part editors like score or piano roll, allows experimentation with different ideas, though such interfaces are not always optimised for the rapid entry, editing and auditioning of new material to support creative exploration of musical ideas. [11]

4.8 Secondary Notation

"How easy is it to make informal notes to capture ideas outside the formal rules of the notation?"

This dimension evaluates a system's provision for recording information beyond the formal constraints of the notation. As informal notation, data is typically not executable by computer or performer, and may only be related to the encapsulated piece / performance indirectly. Decoupled from the formal rules of expression in the notation, secondary notations often allow users to make freeform notes to support their edit process, though flexibly designed facilities may be used for a variety of purposes – including evaluation (peer feedback), working out problems, highlighting relationships in the notation, sketching rough high-level structure, aesthetic decoration, to-do lists, incomplete ideas, etc. In programming, code commenting is used to annotate code with useful labels, instructions, explanations, ASCII art, etc., helping to make the code more readable, but also as a form of communication between coders. As such, *secondary notations* should be designed to be as flexible as possible, to allow users to appropriate them for their own needs.

[Related dimensions: *provisionality, hard mental operations, hidden dependencies, role expressiveness*]

SCORE: The expressive freedom of pencil and pen marks on paper allow musical scores to be annotated with any additional information, such as personal notes, decoration, as well as irregular performance instructions. Formal notation places more constraints on what is representable, though written language can be freely used in performance directions. The human interpretation of scores enables a further degree of flexibility in applying and developing new terminology, such that informal notes that break from standard semantics may still be executable. Performers can also add their own notes to manuscripts to guide their own interpretation of the piece.

MAX/MSP: Like other programming tools, code comments are an important part of developing and maintaining Max patches. Max's visual medium supports annotations using free text (comment boxes), shaded areas, and imported images (bitmaps), used to explain workings, usage, or as decoration. However, drawing facilities are very limited in comparison to pencil and paper, and even digital graphics, with no provision for freehand sketching or drawing lines, arrows, or shapes (other than rectangles). Given the proven benefits of such affordances in other music notations (e.g. the musical sketch and score [5]), their omission in such a visual medium is surprising.

DAW: Despite the proliferation of different notational styles in DAWs, each UI is rigidly structured to fulfil a defined purpose and offer specific tools for editing the underlying data. Limited provisions for annotations are provided by way of labelling and colour-coding parts and tracks, and free text is often supported for meta-data, but few mechanisms are provided for flexibly annotating the music in any of the sub-notations or views, beyond those forms formally recognised by the program.

4.9 Consistency

"Where aspects of the notation mean similar things, is the similarity clear in the way they appear?"

This dimension defines how coherent and consistent the methods of representing elements in a notation or UI are. Consistency facilitates the learning of a system (see *virtuosity*), as users used to a style of presentation can apply knowledge learnt in one area to understand others. However, consistency may also be sacrificed to improve *conciseness*, *visibility*, or *role expressiveness*.

[Related dimensions: *conciseness*, *visibility*, *virtuosity*, *role expressiveness*, *abstraction management*]

SCORE: In sheet music, notated passages that are similar musically share similar visual cues, e.g. melodic contour, repeated passages, etc. Formal rules applied consistently likewise ensure recognisable and learnable conventions. However, compromises are made for *conciseness*, and to optimise the presentation of common expressions, at the expense of readability in less canonical works. For example, the symbolic representation of note rhythm in a passage completely alters if offset within the bar (e.g. moved by a quaver). Similarly, the representation of pitch depends on key; an identical phrase requires accidentals following a change of key signature. Both scenarios present limited issues in common practice music, but the inconsistency makes the notation harder to learn and understand, and the difficulty of using it outside its intended purpose encourages conformity, discouraging experimentation and creativity. Moreover, in digital use (notably MIDI sequencers), such inflexibility markedly reduces the usability of score notation, where systems are unable to unpick the expressive prosody in a captured live performance to display a coherent visual score.

MAX/MSP: By design, programming languages offer diverse paths to produce similar code functionality. Textual languages are based on rigid, carefully designed formal grammars that ensure basic low-level consistency among programming primitives, also enabling many syntactic errors to be identified during compilation. Max's collection of objects is less formally designed and, as the accumulation of several developer's efforts (and coding styles), less consistent. Inconsistencies exist in many areas, including object-naming schemes, inlet and outlet conventions, processing behaviour, message handling, audio quality (and level), and configuration methods. These nuances produce unanticipated code behaviour that increases the learning curve for novices. Objects behave like self-contained programs or plugins; black boxes that have to be mastered individually.

DAW: The added flexibility in the visualisation of data, in the various views afforded by DAWs inevitably comes at the cost of consistency of representation throughout the program. For example, volume might variously be represented as a MIDI value (0-127), automation value (0-1), gain (dBFS, e.g. -96dB to 0dB for 16-bit audio), or using graphics (colour, bar size, rotary knob angle). The trend towards skeuromorphic visual metaphors to electronic studio equipment similarly encourages inconsistencies in representation, drawing on the conventions of previously separate, loosely connected hardware devices. Moreover, while the advent of third-party plugins brings great advantages and creative flexibility, inconsistencies in control, representation, terminology, and interaction create usability issues and a fragmented user experience that is difficult to integrate with the host application.

4.10 Viscosity

"Is it easy to go back and make changes to the music?"

This dimension defines how easy it is to edit or change a notation, once data has been entered. A common example is *knock-on viscosity*, where making a simple edit to the notation requires further edits to restore data integrity. *High viscosity* prevents or discourages alterations, forcing users to work in a prescribed, pre-planned order (see *premature commitment*); *low viscosity* simplifies and encourages making changes, reducing the investment associated with trialling new ideas (see *provisionality*). Being able to easily explore and revisit ideas (ideation) is a key factor in supporting creativity [6, 19], requiring creative systems engender *low viscosity*.

[Related dimensions: *provisionality*, *premature commitment*, *progressive evaluation*]

SCORE: The provisionality of pencil marks simplifies the alteration, erasure and overwriting of notes and passages in a musical sketch. If more drastic changes are required, the reduced emphasis on neatness and third-party

readability allows the composer to strike out larger sections. Inserting new material is harder, but composers can similarly sketch the inserted passage where there is space (or on a new sheet) and note the insertion. Final manuscripts are intentionally more rigid, but performers can still annotate their copy with alternative instructions.

MAX/MSP: Simple changes to values and local objects are straightforward in Max. However, as patches grow and the interconnectedness of objects increases, Max suffers from *knock-on viscosity* [1], where one change requires further edits to restore patch integrity. For example, deleting, editing, or replacing objects removes all cords to other objects. Increased viscosity is a common trade-off in tools designed to avoid hidden dependencies, often seen in data-flow and visual programming languages like Max. As a graphical notation, changes to a patch often require the layout of a patch to be reworked to make room for object insertions, and to maintain readability. In text-based coding environments, such housekeeping is simplified by the inherent serialisation of code, but in VPLs like Max, leads to increased viscosity.

DAW: As with *provisionality*, the level of viscosity in DAW interaction varies between the interfaces and interaction modes of the sequencer. By itself, a tape recorder metaphor of recording a live performance makes it easy to erase and re-record a take, but harder to edit recorded data. Audio data can be processed (e.g. EQ, mixing, FX, splicing, etc.), but musical content (e.g. individual notes or harmonies) is not easily addressed or manipulated. Recorded MIDI data is easier to edit, though visual representations (e.g. score – see *consistency*) and interaction styles can be cumbersome and unwieldy for anything but simple edits. [11, 12]

4.11 Role Expressiveness

"Is it easy to see what each part is for, in the overall format of the notation?"

This dimension evaluates how well the role or purpose of individual elements is represented in the overall scheme of the notation or UI. Different elements may not be visually indistinct, or their function may be unclear in the way they are presented. For example, English language keywords in a menu or programming language can be used to express their function, whereas cryptic symbols or icons may need to be learnt. Alternatively, the visual design of GUI may impose a consistent aesthetic or

layout that fails to capture the diverse functionality encapsulated, or the relationship to other elements of the UI (see *hidden dependencies* and *closeness of mapping*).

[Related dimensions: *visibility*, *hidden dependencies*, *closeness of mapping*]

SCORE: While some aspects of the score may be inferred by listening to the music (such as a general sense of pitch and rhythm), most involve learning syntax and rote practice. Similarly, while some signs offer more expressive visual cues to role (crescendo and diminuendo hairpins; tremolo marks), many do not – clefs, accidentals, key signatures, note shapes, ornaments, and foreign terms symbolise complex musical concepts that require tuition. Once learnt, however, the symbols facilitate the rapid comprehension of notated music – e.g. different note shapes and beaming conventions provide clear differen-tiation between different note lengths. However, recent approaches to contemporary scores tend to exploit more expressive geometric forms, rather than new symbol sets.

MAX/MSP: The role of some specialised objects, notably user controls, is clear from their representation in Max. However, beyond caption and inlet/outlet configuration, Max offers little visual distinction in the representation of most coding objects, which appear as text boxes. Patchcords help to define the context and role of connected objects, and visual distinction is made between audio and message types (though not between int, float, list, or bang subtypes) – but, despite the unidirectional flow of data, flow direction is not depicted (e.g. using arrows).

DAW: Many aspects of DAW UIs rely on a degree of familiarity with studio equipment and musical practice. However, the graphical user interfaces of most packages make prominent use of expressive icons and detailed graphics to indicate the function of controls. Visual metaphor and skeuromorphisms are commonly used to relate program controls to familiar concepts. Image schema and direct manipulation principles are similarly applied to highlight interaction affordances, in the context of both music and generic computer interaction styles.

4.12 Premature Commitment

“Do edits have to be performed in a prescribed order, requiring you to plan or think ahead?”

This dimension defines how flexible a notation is with respect to workflow, and the process of developing ideas. Notations or system features that must be prepared or configured before use entail *premature commitment*. Notations with *high viscosity*, where it is hard to backtrack, also entail forward planning and commitment. In programming, an illustrative example is the need to declare variables and allocate memory before coding actual functionality (cf. C/C++ vs. BASIC).

[Related dimensions: *provisionality*, *viscosity*]

SCORE: A degree of viscosity in altering page layout means that some forward thinking is required to commit musical ideas to the page, which generally proceeds left-to-right, bar-by-bar. However, separate pages allow sections and movements to be developed non-linearly, and the *provisionality* of the musical sketch allows some flexibility with development of musical phrases and bars. Multiple approaches to composition are possible: horizontal (part-by-part), vertical (all parts at once, start to finish), bottom up (bar-by-bar), top down (musical form). Historically, the literacy and musical experience of composers meant that musical material was often part-formed before being committed to the page – either mentally, or through experimentation with instruments.

MAX/MSP: As a prototyping tool, Max supports experimentation with partially-formed design concepts. Often, however, audio processes will be designed with a plan or general architecture in mind; in Max, forethought with respect to *abstraction* (sub-patching) or layout benefits development, though housekeeping may be needed retrospectively (see also *viscosity*). The open-ended canvas allows patches to be flexibly extended in any direction, and a modular approach to programming allows piecewise development of complex systems.

DAW: Musical parts and audio segments can be easily inserted, moved, and copied in the arrange window, though complex phrases with overlapping tracks and automation can be difficult to split and re-sequence. Furthermore, the unified linear timeline and tape recorder metaphor encourages a linear workflow. [12] In modelling studio workflows, DAWs can be seen as transcription tools, rather than environments for exploratory creativity, where artists only turn to the recording process once a work has already taken form. [3,20] By contrast, pattern- and loop-based sequencers (*Live*, *FL Studio*, tracker-style sequencers) offer a flexible non-linear approach to developing and sequencing musical forms, facilitating digitally-supported creativity and flow.

4.13 Error Proneness

“How easy is it to make annoying mistakes?”

This dimension identifies whether the design of a UI or notation makes the user more or less likely to make errors or mistakes. These can manifest as accidental interactions with a program, or incoherent, unexpected musical results arising from vagueness or ambiguity in the notation (see *role expressiveness*). In programming, for example, a notation is error prone if its function markedly alters upon the addition/omission/position of a single character. Errors are broadly undesirable, but can lead to creative, serendipitous formulations in artistic expression. [21]

[Related dimensions: *hidden dep.*, *role expressivness*]

SCORE: In scoring, the literacy threshold means mistakes are more likely during early stages of learning. Aspects of *consistency* and *hidden dependencies* contribute to a user's propensity to make errors. However, like language, fluency with the notation reduces mistakes. Sketching, as a private medium for the composer, is also tolerant of errors; they are free to misuse or invent notation, which remains meaningful to them personally. When scores are used for communication, mistakes have consequences; but the impact on early creative process is minimal.

MAX/MSP: As a formal language, it is easy to make mistakes in Max, through the creation of ill-formed code. However, aspects of the Max UI make it more prone to errors in certain situations. As a graphical UI, mouse interaction is cumbersome, and Max attempts to avoid diffuseness with compact objects, such that selecting and connecting inlets or outlets using patchcords is awkward. As with the score, *consistency* issues and *hidden dependencies* also invite mistakes relating to coding semantics.

DAW: Recording performances in real-time heightens the likelihood of input errors, though facilities exist to correct or overdub recorded data, and the occasional mistake is often acceptable for the improved flow (musical and creative) afforded by live interaction with an instrument. As in Max, DAWs invite mistakes through dependence on the mouse, where delicate pointer control is required – many edits require targeting the edge of elements (track segments, notes, etc.), and the extent of such hotspots may be small and visually indistinct. Proximate and overlapping objects can be similarly difficult to target.

4.14 Closeness of Mapping

"Does the notation match how you describe the music yourself?"

This dimension assesses how well a notation's representation of the domain aligns with the user's own mental model. In a UI, this also applies to how closely workflows and interaction styles fit a user's working methods. Music perception and aesthetics are quintessentially subjective, making it difficult to encode a universally or intuitively acceptable formalisation, so notations and systems are built around common cultural practices. This can constrain the creative expression or affordances of a notation. To mitigate this, *abstraction mechanisms* may enable users to appropriate, redefine, and extend systems.

[Related dimensions: *role expressiveness*, *abstraction management*, *virtuosity*]

SCORE: While the score is not an intuitive representation that untrained users might themselves conceive or comp-rehend, it remains a widespread and established technique for notation in Western music. At the same time, the canonical score systematises music in a way that makes assumptions about the musical practices and aesthetics of its users, such that modern composers identify the format as a constraint on their personal expression and creativity. However, the flexibility offered by individual sketching techniques allows composers to invent and appropriate notation techniques for their own personal use.

MAX/MSP: The data-flow model of Max maps closely to diagrammatic forms used widely in signal processing, with a shared legacy in electronics and circuit diagrams. The inherent role of electronics in the studio, and representation of audio as voltage, also make this an analogy that musicians and producers can relate to. The functional and visual resemblance to generic flow charts further helps to make the programming environment accessible to non-technical users. However, for musical applications (rather than audio processing) such as arrangement and composition, the abstract representation of time offers a poor closeness of mapping to familiar representations of music. Similarly, for traditional programmers used to imperative programming (ordered sequences of instructions), scripting program behaviour over time is difficult.

DAW: For its intended audience of musicians and sound engineers, traditional sequencers and DAWs provide a strong closeness of mapping, using visual metaphors and interaction paradigms based on studio processes and audio hardware, to allow skills transfer. Notably, MIDI and audio recording tools focus interaction on musical instruments. However, in recent years, more computer-oriented musicians, with greater technical literacy, have begun to embrace tools that rely less on analogies to the recording studio and focus on the affordances of digital and computer music technologies – as offered by *Ableton Live* and *FL Studio*. Ultimately, engagement with music, as a personal experience, should be based on articulations of the music domain crafted by the user themselves, which the rising level of computer literacy might enable, as end-users increasingly engage with programming.

4.15 Abstraction Management

"How can the notation be customised, adapted, or used beyond its intended use?"

This dimension defines what facilities a system offers for appropriating, repurposing, or extending a notation or UI. All notations present an abstract model of a domain (e.g. music, software), providing a set of fixed abstractions representing basic objects (e.g. notes, parts) and properties (e.g. pitch, time, etc.) that enable the articulation of solutions (e.g. a piece). The creative possibilities are defined by what encapsulations of objects are possible and how easy they are to extend. Notations defined for a specific purpose fix the possible abstractions and ways of working. However, the opportunity to define new abstractions (e.g. in terms of existing ones) offers the user a way to develop their own toolset and facilitates the building of more complex solutions (e.g. by abstracting low-level detail), and helps to personalise and raise the creative ceiling of a system. [6] In programming, examples include defining custom functions and abstract data types (objects). In end-user computing, systems may support automation, macros, or plugins to enable users to add new functionality. Simpler abstraction mechanisms such as grouping and naming elements are also possible.

[Related dimensions: *visibility*, *closeness of mapping*, *role expressiveness*, *conciseness/diffuseness*, *consistency*]

SCORE: In sketching the piece during the creative process, composers are able to appropriate or invent new terminology of notation technique to describe music more *concisely* (composer shorthand) or to encapsulate unconventional musical devices and practices – only when it is transcribed for communication to a performer (or computer) must it conform to established notational forms.

The canonical score format is more limited; designed around common practices and conventions in formal music, but offers some support for grouping mechanisms (e.g. brackets, phrasing) and abstraction (e.g. custom ornaments). However, a composer can use the preface to a score to introduce original notation techniques and syntax, to instruct the performer's interpretation.

MAX/MSP: As a programming language, abstraction is a key technique for building and maintaining complex solutions. Max offers several provisions for abstracting and extending code: *sub-patches* allow embedded and external patches to be nested inside other patches, represented as new objects (linked using inlets and outlets); *externals* use a plugin model to allow new objects to be coded in C/C++ with defined inputs and outputs; and *presentation mode* allows patches to be rearrange, simplified and selectively exposed in end user-oriented UIs.

DAW: Sequencers and DAWs are designed to support specific working styles in music / production scenarios. Part editors support low-level editing of notes and other musical events. In other screens, higher-level abstractions are used to structure music (tracks, parts, etc.), with some provision for grouping and organising objects (e.g. group channels, folders, track segments). Most packages also support audio plugin formats that extend FX processing and synthesis options. However, few sequencers support more flexible abstraction mechanisms to facilitate interaction with notation, such as macros, scripting, or automation. Exceptions to this include *Live*, which can be integrated with Max, CAL Script in *Cakewalk SONAR*, and *Sibelius* plugins. In the tracker domain, *Manhattan* [23] offers end-user programming for music using an extended implementation of spreadsheet-style formulae.

4.16 Virtuosity / Learnability

"How easy is it to master the notation? Where is the respective threshold for novices and ceiling for experts?"

This dimension assesses the learnability of the notation, and whether it engenders a scalable learning curve – that is, a "low threshold" for practical use by beginners, a "high ceiling" for flexible expression by experts, affording "many paths" by which users can express themselves. In addition to supporting multiple levels of expertise and creativity, virtuosity should be understood in terms of the balance of challenge and ability experienced by the user. A slight challenge, relative to their ability, intrinsically motivates users and helps create the conditions for flow. [3, 9, 11, 22] Too much challenge and users become anxious; too little and they become bored. The best model for systems are based around "simple primitives" (building blocks) that can be easily understood by beginners, but flexibly combined to form more complex *abstractions* and functionality. [6]

[Related dimensions: *consistency*, *prog. evaluation*, *role expressiveness*, *closeness of mapping*, *error proneness*]

SCORE: The score has a steep learning curve and beginners require formal tuition and practice to master it. Novices can be discouraged from learning music by the literacy entry threshold. [3] The complexity of the notation reflects its relatively high ceiling and capacity to flexibly encapsulate a wide variety of musical styles and pieces, though contemporary and electronic composers can find traditional, formal syntax limiting. [2, 3, 12]

MAX/MSP: While programming languages often present a high threshold for novices, Max is explicitly designed for musicians, and uses a visual programming model to appeal to non-coders. Tutorials present beginners with simple patches that produce useful results, enabling a working knowledge to develop quickly. Innovative interactive in-program documentation and a strong user community supports both learners and practitioners. There are aspects of the environment that also impede learning (see *consistency*, *error proneness* and *hidden dependencies*). The creative ceiling for developing audio and music systems in Max is high, further supported by *abstraction* mechanisms – though audio programmers and more music-oriented users may graduate to other tools (e.g. C/C++, *OpenMusic*, *SuperCollider*).

DAW: Music and audio production packages are designed to provide a low threshold for musicians and those familiar with studios. The use of visual metaphor and direct manipulation principles allows knowledge transfer from these other practices [4], though users without such backgrounds may struggle. Packages provide a wide array of tools and features for a variety of purposes, though few users will have need of all features. The ceiling for musical creativity is relatively high, within the confines of conventional practices, though UIs are often optimised for specific workflows and techniques, and users are largely dependent on software developers to provide new opportunities for expression. Unlike the traditional score, and programming languages (like Max), users efforts to master authoring packages can be frustrated by a lack of continuity between versions.

5. PRACTICAL METHODOLOGIES

This section briefly surveys existing applications of the *Cognitive Dimensions of* Notations in musical contexts, highlighting both qualitative and quantitative methods for analysing notations and interaction.

Blackwell (with others [7-11, 16, 20, 24]) has used cognitive dimensions to highlight aspects of musical interaction in several settings, including music typesetting software [10, 20], programming languages [16, 24], and digital tools for composition (e.g. sequencers, trackers) [8-11]. In such treatments, the framework provides a language for discussing the affordances of notation, but has also lead to the development of tools to elicit feedback from end-users, such as questionnaires that probe dimensions in user-friendly, accessible language. [10] McLean's work on music and art programming languages similarly applies and develops the framework for analysis of new music coding notations and interfaces. [21]

Nash [3, 9, 11] extended previous qualitative analysis techniques to develop a quantitative approach to evaluating notations. Using a *Likert* scale, each dimension is formulated as a statement that users can agree or disagree with to a greater or lesser extent. The mean response from a large sample of users can then be used to plot a dimensional profile of the notation under evaluation. Figure 1 shows profiles for a survey of various music sequencer tools (*n=245*), not only highlighting relative strengths and weakness with respect to properties of each UI, but also revealing a general profile for music systems, where the trend may indicate the desired polarity of each cognitive dimension in music interaction. Moreover, the approach was combined with psychometric-style surveys of the experience of creative flow [22], using a battery of questions to also measure users' subjective experience of nine components of flow. Using cross-correlation and multiple-regression analysis, the results for individual flow components and dimensions of the notation were used to identify the key properties of notations facilitating flow, findings of which can be used to guide the design of immersive or embodied interaction systems. The study [3,11] suggests that key dimensions in the support of flow were *visibility* (visual feedback), *progressive evaluation* (audio feedback) and *consistency* (support for learning and sense of control) – as well as *virtuosity* (balance of skill and ability), *abstraction management* (high creative ceiling), *viscosity* (ease of editing), *premature commitment* (freedom of action) and *role expressiveness* (support for learning). The findings were used to propose a set of design heuristics for music systems based around the concept of virtuosity, rather than usability (see [3, 9]).

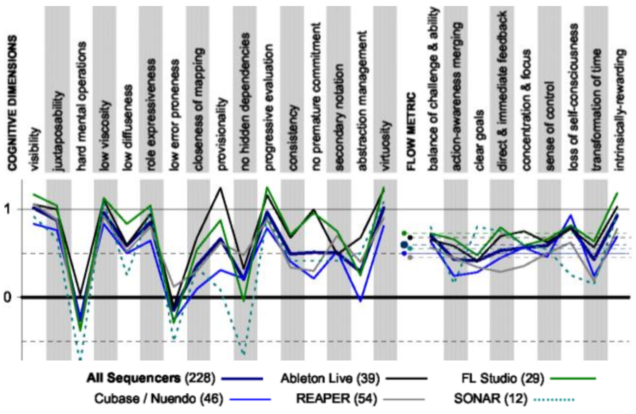


Figure 1 Cognitive dimension and flow profiles of music tools, based on quantitative user testing [3, 11].

6. CONCLUSIONS

This paper has presented a musical reworking of the *Cognitive Dimensions of Notations* usability framework, and suggested methods and tools for using it to analyse music notations, interfaces, and systems. Several applications have been identified that use the framework to provide insight into the human factors of notation-mediated musical systems, including creativity, virtuosity and flow.

Future work will focus on further use and development of the framework, including its application to other music interaction scenarios and systems, the evaluation of new dimensions, and research of other dimensional profiles in other music interactions. The growing intersection of music and programming practice is also likely to reveal other parallels between these creative domains that can further inform both theory and practice.

Acknowledgments

The author wishes to thank all those who supported and contributed to this research, especially Alan Blackwell, Ian Cross, Darren Edge, Sam Aaron, the Cambridge *Centre for Music & Science (CMS)*, and many other

researchers exploring and developing the CD framework in other domains. This research was funded by the Harold Hyam Wingate Foundation and UWE (Bristol).

7. REFERENCES

[1] T. R. G. Green, and M. Petre, "Usability analysis of visual programming environments: a 'cognitive dimensions' framework," *Journal of Visual Languages & Computing*, 7, 1996, pp. 131-74.

[2] G. Read, *Music notation: a manual of modern practice*, 1979, Taplinger Publishing Company.

[3] C. Nash, *Supporting Virtuosity and Flow in Computer Music*, PhD, University of Cambridge, 2011.

[4] M. Duignan, *Computer mediated music production: A study of abstraction and activity*, PhD, Victoria University of Wellington, 2008.

[5] M. Graf, *From Beethoven to Shostakovich*, New York: Philosophical Library, New York, 1947.

[6] M. Resnick et al., "Design principles for tools to support creative thinking," 2005, *NSF Workshop of Creative Support Tools*, pp. 25-36.

[7] A. Blackwell, "Dealing with new cognitive dimensions", *Workshop on Cognitive Dimensions*, Uni. of Hertfordshire, 2000.

[8] C. Nash, and A. Blackwell, "Tracking virtuosity and flow in computer music," 2011, *Proceedings of ICMC 2011*, pp. 572-582.

[9] C. Nash, and A. Blackwell, "Flow of creative interaction with digital music notations," in *The Oxford Handbook of Interactive Audio*, 2014, OUP.

[10] A. Blackwell, and T.R.G. Green, "A cognitive dimensions questionnaire optimized for users," 2000, *Proc. of PPIG 2000*, pp. 137-152.

[11] C. Nash, and A. Blackwell, "Liveness and flow in notation use," 2012, *Proc. of NIME 2012*, pp. 28-33.

[12] D. Collins, "A synthesis process model of creative thinking in music composition," *Psychology of Music*, 33, 2005, pp. 192-216.

[13] P. Desain et al., "Putting Max in perspective," *CMJ*, 17(2), 1993, pp. 3-11.

[14] J. A. Paradiso, and S. O'Modhrain, "Current Trends in Electronic Music Interfaces. Guest Editors' Introduction," *JNMR*, 32(4), 2003, pp. 345-349.

[15] S. L. Tanimoto, "VIVA: A visual language for image processing," *Journal of Visual Languages & Computing*, 1(2), 1990, Elsevier, pp. 127-139.

[16] L. Church, C. Nash, and A. F. Blackwell, "Liveness in notation use: From music to programming," *22*, 2010, *Proceedings of PPIG 2010*, pp. 2-11.

[17] M. Leman, *Embodied music cognition and mediation technology*, MIT Press, Camb., MA, 2008.

[18] A. J. Sellen, and R. H. R. Harper, *The myth of the paperless office*, MIT Press, 2002.

[19] T. Amabile, "The social psychology of creativity: A componential conceptualization," *Journal of personality and social psychology*, 45(2), 1983, pp. 357-76.

[20] A. F. Blackwell, T. R. G. Green, and D. J. E. Nunn, "Cognitive Dimensions and musical notation systems," 2000, *ICMC 2000: Workshop on Notation and Music Information Retrieval in the Comp. Age*.

[21] A. McLean, *Artist-programmers and programming languages for the arts*, PhD, Goldsmiths, 2011.

[22] M. Csikszentmihalyi, *Creativity: Flow and the Psychology of Discovery and Invention*, HarperCollins, New York, 1997.

[23] C. Nash, "Manhattan: End-User Programming for Music," 2014, *Proc. of NIME 2014*, pp. 28-33.

[24] A. Blackwell, and N. Collins, "The programming language as a musical instrument," 2005, *Proceedings of PPIG05*, pp. 120-130.

TUFTE DESIGN CONCEPTS IN MUSICAL SCORE CREATION

Benjamin Bacon
IDMIL, CIRMMT, McGill University
benjamin.bacon@mail.mcgill.ca

ABSTRACT

This paper introduces several examples of utilizing the information design concepts of Edward Tufte in musical notation and score-design. Tufte is generally considered a modern pioneer in the field of information design. Throughout several authoritative texts [1] [2] [3] [4], Tufte's work displays countless examples of successful and unsuccessful attempts of displaying information while also offering a few personal redesigns of especially troubled instances. Overall, Tufte reveals interesting concepts which could be useful when applied to designing musical notation systems. The author presents three personal notational examples which have been aided by Tufte's work. Information design is a vast multidisciplinary field which could provide composers and musicians with an abundance of technical approaches to complex notational challenges.

1. INTRODUCTION

The task of displaying information in a visual way is often a challenging one, riddled with difficult decisions, pitfalls, and corrupting influences [5]. All throughout history, experts and students of science and the arts alike have stumbled in their attempts to visually communicate ideas or concepts. The lessons of successful models of information design (see Figure 1) have often gone unnoticed, leading to poor work which could have been prevented. Edward Tufte's work in the field of information design brings examples, concepts, and lessons from across human history in an effort to show how the visual realm can be a powerful tool for communication.

Mr. Tufte's first book on data visualisation, *The Visual Display of Quantitative Information*, published in 1982 [1], stands as a landmark work in the field of data visualisation [6]. This book provides a clear and concise roadmap, richly packed with intellectual tools for the effective display of visual information. Elegant examples drawing from some of the most brilliant thinkers in history are beautifully displayed in order to showcase their qualities, paired in stark contrast with examples of poorly-executed displays.

Since the release of *Visual Display*, which is mostly focused on the logic of statistical evidence, Mr. Tufte has broadened his focus to include any design relying on visual reasoning strategies. This conceptual expansion has yielded equally acclaimed results in the realm of visual design [7], with his later three books [2] [3] [4] providing an abundance of compelling cases advocating for the ethical, aesthetically grounded, and truthful representation of visual evidence. His examples extend across many diverse applications including choreography, weather-maps, train-schedules, and the causal analysis of historically significant events.

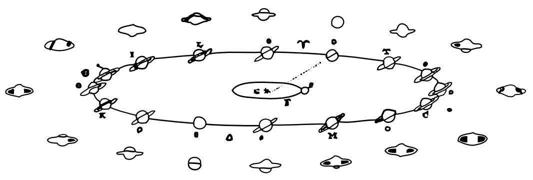


Figure 1. This orbital diagram by Christiaan Huygens in 1659 displays 32 images of Saturn across two different perspectives. The clarity of design in this diagram is frequently referenced by Tufte [1] as a superb example of *small multiple* design, as well as proof that good information design has existed for a long time.

1.1 General Principles of Design

While the examples Tufte draws upon to illustrate his ideas and notions in visual design span many cultures spread across hundreds of years, there are several main underlying themes which can be traced throughout his work. Among one of the most well known is the idea of *chartjunk*, which refers to the superfluous use of graphical elements (*i.e.* gradient shading or skeuomorphism) which serve no logical purpose in communicating the intended information. Chartjunk makes interpreting the intended information of a graphic difficult, since the eye is presented with extra non-essential stimuli. The creation of chartjunk can be avoided by making use of what Tufte coined the data-ink ratio, the *data-ink ratio*, which is the proportion of data-representative ink to total ink used in the graphic. The more ink that is used on non-representative graphics, the more the designer endangers the clarity of the graphic's intent.

In each of his publications, dozens of techniques and examples are shown describing how to layer, differentiate, and communicate information efficiently. The notions of chartjunk and data-ink ratios hold at their core the idea that graphical representations of information should maximise the space which they inhabit. This promotes the efficient transfer of ideas, un-obscured by the noise of poor design. Without a careful understanding of how graphical content interacts with itself, and indeed, other content, it is easy to obfuscate critical concepts while simultaneously detracting thinking and attention from the observer. While minimalism may come to mind when adopting Tufte's design strategies, he does not advocate for the avoidance of complexity. In fact, many examples are dense and complicated, showing at times thousands of data-points in a single image.

Above all, Tufte's principles ask the designer to employ techniques relevant to the cognitive task at hand. It is up to the designer to choose a methodology which communicates the primary message of the presented information. As Tufte clearly states [8]:

> If the thinking task is to understand causality, the task calls for a design principle: show causality. If a thinking task is to answer a question and compare it with alternatives, the design principle is show comparisons.

1.2 Information Design and the Score

The inclusion of diverse graphical examples in Tufte's books, spanning across many different disciplines and points in history, is an effort to showcase the universality of his theories. These graphic examples are meant to inspire and provoke thought on a variety of possible scenarios when designing, while also demonstrating how a technique can be used in a given discipline. Information design is interdisciplinary in nature, and can be applied to nearly anything where information is represented graphically.

Therefore, it would not require any stretch of the imagination to apply Tufte's principles when creating a musical score. Composers, especially of contemporary music, work with an ever-expanding palette of sonic parameters [9]. While most composers are not specifically attempting to display data or evidence for something that already exists, visuals are employed to provide reasoning for something that is about to happen (*i.e.* a performance).

Today, extended techniques, electronic processing, and even new instruments themselves place interesting demands on the composer. In many cases, complicated ideas must be communicated graphically to the performer, making careful distinctions between the representative elements of dynamic, technical, rhythmic, timbral, and pitched content. Further distinctions between micro- and macro-formal trends are often useful for performers, and add more demands to the score.

2. EXAMPLES IN SCORE-DESIGN

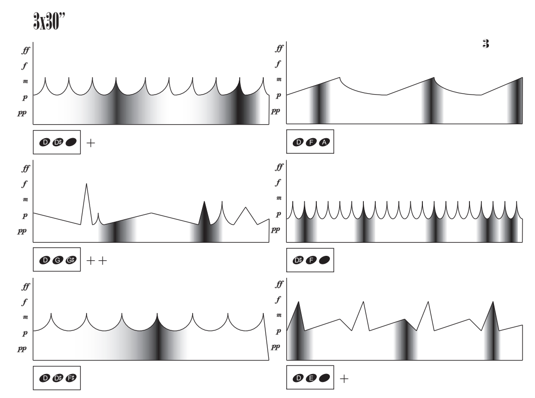
The following sections will discuss three examples of Tuftian design theories employed by the author in his own compositions, including one work from a scientific study. These examples will discuss specific musical ideas, and how information design theories can provide the composer with a useful toolbox for finding innovative solutions to graphical demands.

2.1 Graph-based Notation in de Chrome

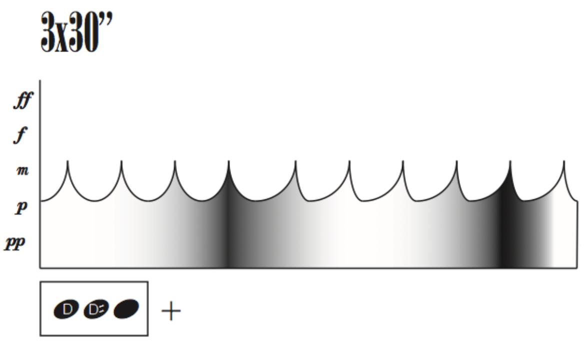
The first example inspired by Tufte's writings on information design is a piece entitled *de Chrome* written by the author in 2012, seen in Figure 2. This composition gives the performer a role in shaping the piece on a micro-level, while the larger form is dictated. Choices can be made by the performers on which content to perform and which pitches to sound, but are limited to a sub-phrasal level. Overall, the piece is to be performed by 3-5 players on any instrument.

This piece is comprised of graphical sub-phrases grouped together by page, which can be seen in Figure 2aa. Each *graph* depicts the dynamic contour of a sustained sound, with dynamic references located on the y-axis. The duration of each sub-phrase is indicated by the two numbers in the top left-hand corner page, best seen in Figure 2bb. In this case, the performers are instructed to choose any three sub-phrases (with no repeats), and to perform each one for thirty seconds; hence the notation of *3x30"*. After the completion of the phrases, the performers may move onto the next page. The gradient shading refers to the level of *timbral pressure* exerted in each phrase. Black indicates a heavy amount of force while white indicates little or no extra force. This can correspond to the pressure exerted on the bow, embouchure, etc.

The small box beneath the contour-graph contains the group of pitches the performer may choose from when sounding the sub-phrase. Note-heads with no indicated pitch are to be interpreted as *non-pitched* sustained sounds. This simply means the performed sound must, above all, not contain any tuned-pitch as detuned sounds are appropriate. Outside the box, percussive articulations are marked with the + symbol.



(a) *a*



(b) *b*

Figure 2. Subfigure *a* displays page 3 of *de Chrome*, which includes a collection of 6 sub-phrases. Subfigure *b* shows the first subphrase in detail.

2.2 Design Theories in de Chrome Score

The goal of *de Chrome* was to create a dynamic tonal landscape while guiding performers through their own decisions in interpreting the score. Since there are no temporal-metric markers higher than the sub-phrasal level, the concept of *small-multiples* was employed to give performers a better view each sub-phrase.

Small-multiples, popularized by Tufte in *Envisioning Information* [2], consists of a series of design elements showing multi-variate information. This design model directly encourages comparisons and promotes awareness between data-sets. The eye can easily move from one element to the next while maintaining a larger analytical perspective. In the case of *de Chrome*, the data is replaced by musical notation. To promote awareness and cohesion between performers in a graphically dominant score such as this, small-multiples bring each sub-phrase into view. The distinct differences between each sub-phrase can easily be identified, allowing each performer to blend, hear, and interact with one another.

In addition to the small-multiple design, the use of gradient shading to convey the timbral-pressure parameter was purposefully fitted within the negative space of the contour-line. Grey-scales are superior to colouring techniques in displaying hierarchical content. As mentioned by Tufte in *Visual Display*, grey-scales give an immediate multi-functional element to the inhabitant information, allowing for increased viewing resolution in a smaller space, and using less ink. An additional benefit to the increased information resolution of the employed visual techniques in *de Chrome* is the opening of white space for extra notation from the performer.

2.3 Scoring Bi-manual Action

The second score in this example is a percussion solo piece written by the author between 2013-2014. Entitled *Dextral Shift*, this work is focused on the bi-manual nature of percussion performance. With the exception of method books which are educational in purpose, most written percussion pieces are not too concerned with separating the left- and right-hands. *Dextral Shift* was conceptually inspired by the author's previous research in the field of laterality [10]. Laterality is an interdisciplinary field concerned with the behavioural differences between the left- and right-sides of the body. Everyone engages a preferred-side (*i.e.* left or right) when it comes to a specific task [11]. This score was designed with the two hands notated separately, with specific actions assigned based on the abilities dictated by handedness.

In Figure 3, two excerpted stems from the *Dextral Shift* score can be seen. This piece was written for four pitched temple-bowls in the left-hand, and one detuned tom-tom for the right-hand. On the left-side of the score along the y-axis are indicators denoting the notational domains of the hands. These domains are split by the center x-axis. The y-axis also doubles as an indicator of pitch/instrumental differentiation and dynamic intensity. In the left-hand, the closer a note is written to the center line, the lower the pitch. For the right-hand, notes written close to the center line indicate a striking area closer to the rim of the drum-head. Notes written further from the center line are to be played further from the rim of the drum-head. Dynamic markings also rely on the y-axis and are represented as a continuous line, shaded softly behind the note-heads. The markings for dynamics of each hand are again separated by the center line. The further from the center the grey-shading goes, the louder the dynamics for the notes placed over it.

2.4 Design Theories in Dextral Shift Score

Several important design strategies were employed in the creation of the *Dextral Shift*' s notational system. This piece is essentially two different scores presented as one. The left- and right-hands, except for a few short sections, are notated individually. Each hand is tasked with performing its own content on a different instrument using different techniques. This division was the primary goal of the composition, as it was intended to explore the physical difficulties of combining and separating the hands from each other.

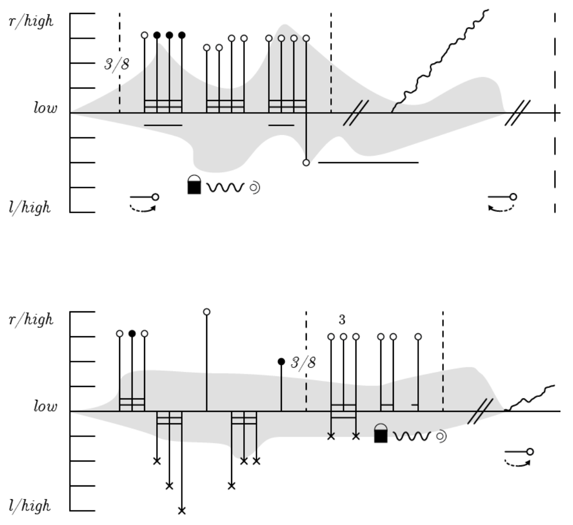


Figure 3. Two excerpted lines from *Dextral Shift*. These segments display the graphical shading techniques of the dynamics, as well as the split of the left- and right-hands over the center line (*i.e.* x-axis).

While keeping the hands apart was the primary goal, creating a comprehensive and unified systemic structure for the notation was also a top priority. This was achieved by assigning multiple variables to the two-axis notational platform. The score uses the same space and graphical language to display several different parameters all at once, in what Tufte describes as *layering and separation* [2]. Dynamic intensity and instrumentation share the same indicators (i.e. the high-low-high notation), which allowed for compact notational techniques. The grey-shaded dynamic contour freed space from the traditional dynamic notation techniques, which usually includes both graphical elements and text. The removal of such elements allowed for the mirrored note stemming from the center line. Further space was uncluttered by omitting traditional staff-lines as well. Instead, *staff-markers* are shown at the beginning of each musical system, of which there are three per page.

These multi-purposed graphic elements allowed for information to be concentrated along the center line, creating a reliable focal point for the performer. Comparisons, changes, and detailed compositional information are easily embedded within the score in an intuitive manner. Extra text which is usually required for dynamic or tempo markings, along with their own graphical content, are no longer necessary. The efficiency and compactness of this design strategy speaks directly to Tufte's design teachings in reducing administrative elements, while simultaneously increasing the data-ink ratio on the page. The system of notation in *Dextral Shift* places weighted emphasis on the most important elements of the score, while enabling graphical flexibility. The removal of staff lines clears-up the page and avoids the activation of negative space. Unused grid-space, especially those with pronounced lines, can often be confusing [12]. The unnecessary interaction of graphical elements can make the discernment of important information from non-important/existing information quite difficult [13].

2.5 Using Disinformation Design

The last example of using Tufte's design theories in creating a musical score comes from a scientific study performed by the author [14]. This example does not come from a specific piece intended for concert performance, but from a gestural analysis study on the effects of handedness. An experiment was devised in order to study how the hands are used when performing unrehearsed music, also known as sight reading. Insight from this study was used to better understand the role of the participant's internal-timing mechanisms in relation to their hand use.

In an effort to challenge the participants' sense of timing, the author composed a short rhythmic exercise. This exercise, roughly one minute in length, gradually increased in rhythmic complexity as it progressed. The beginning of the score contained simple 8th-note and quarternote rhythms and ended with a series of tricky multi-metered syncopations. Tuplets and other poly-metered rhythmic units require the performer to use delicate time-keeping techniques in order to perform a given passage accurately. The gradual change in difficulty allowed the performer to establish a comfortable counting routine before more difficult material appeared.

A major challenge in creating the experimental exercise score was developing a rhythmic environment which seamlessly moved from one segment to the next. Making the participants feel comfortable was important, as it would allow them to exhibit their most natural tendencies. Any extreme changes in notation or glaring multi-rhythmic tangles of obvious difficulty would undoubtedly put the participant at unease.

2.6 Design Theories in Sight-Reading Score

In designing how the score for the handedness experiment, tuplets and syncopated groupings were the primary mechanisms for the introduction of challenging material. Tuplets require complex counting strategies, even when performing rehearsed material [15], as they contain beats falling outside of the traditional beat-matrix. The design concept

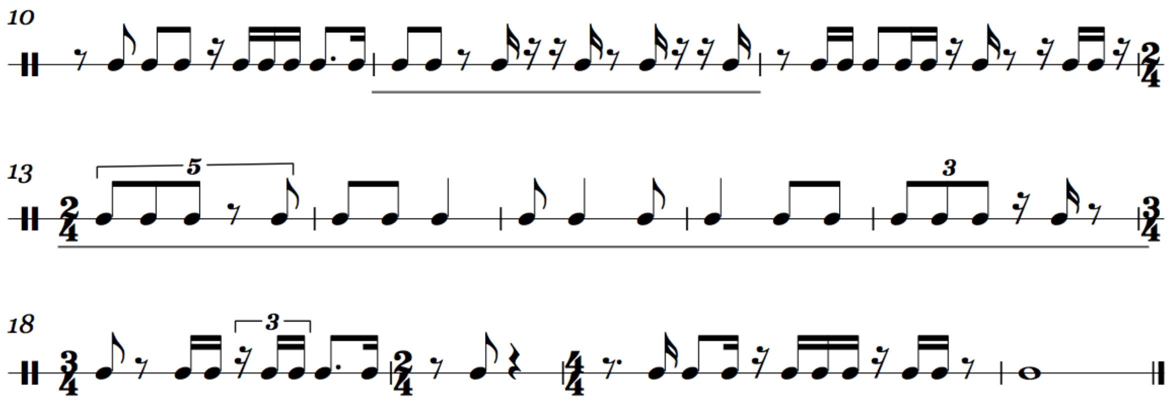


Figure 4. This is an excerpt from the handedness study music. The underlined segments highlight the areas where Tufte's concepts on disinformation design were employed.

of *disinformation design*, found in the book *Envisioning Information* [3] was used.

Disinformation design is in most ways the complete opposite of information design; the goal is obscure the truth or to produce an illusion. In Tufte's book *Visual Explanations*, an entire chapter is co-authored with magician Jamy Ian Swiss, as various examples of technical explanations and magic guidebook diagrams are discussed in detail. The notational methodology for the excerpt presented in the handedness study specifically disguised the introduction of new, and more challenging material.

In Figure 4, the underlined segments highlight areas where disinformation design tactics were employed. The first underline (measure 11) is a syncopated 3/4 pattern using 16th-notes. The measure begins with two 8th-notes, establishing a firm rhythmic foundation for the measure. The syncopated rhythm is written using 16th-notes only for the note-heads, and a collection of 16th-note and 8th-note rests for the spaces in between. The busy notation of this measure obscures a sense of regularity, and masks the metric identity of the measure. The graphical repetition of the 16th-note rest pairs serves as a kind of notational anomaly. When two rests are paired of equal value, they are usually grouped into one. Using two rests requires more subdividing by the performer which is reliant on internal counting strategies.

Measure 11 was repeatedly misplayed by the participants of the handedness study, as many had to suddenly dial-in on the resolution of their internal counting, which usually happened too late. The designed *error-zone* provided an opportunity to observe which hand would be used when intensified timing-based decisions needed to be performed. Consistent with previous findings on the matter [16], the preferred-hand performed most of the notes in this measure.

The second underlined segment seen in Figure 4 begins with a quintuplet figure with an 8th-note rest on the fourth beat. Following the quintuplet figure is another syncopated rhythm. The visual presentation of the syncopation is partially what makes it challenging. The isolated 8th-note on the quintuplet leads into the syncopated rhythms consisting of a 3/8 feel over several 2/4 measures. The graphical representation of the last quintuplet note in measure 13 leads into the next bar, much like the 8th-note in measure 15. While they look similar, one is bound to the beat-matrix while the other is not. This entire system (mm.13-17), was one of the most complicated segments to read. A clear majority of participants performed these measures with only their preferred-hand.

3. DISCUSSION

Musical content is represented graphically in a diverse and varied landscape, full of rich historical context and traditions. In musical notation, there is no right or wrong way to pursue or represent an idea. Composers often work with abstracted concepts in a visceral way, which is in turn reinterpreted by the performers and the audience. Conversely, information design is often grounded in verifiable data. Graphical elements are used in information design to give form to numbers and reveal trends. Information design is concerned with the visual presentation of evidence. For these differences, perhaps musical notation techniques may have remained separate from the quantitatively-driven world of information design. Music' s representation on paper is entirely arbitrary, and often self-containing. In contemporary music, the composer devises a new graphical language for each piece [17], largely shaped by what the composer wants to communicate.

Quantitatively speaking, the way in which traditional music is represented revolves around a grid-based system, where exact information can be presented. Timing and pitch can be precisely notated, but this system has been challenged to a great extent due to its limitations in displaying highly-specified technical information [18]. The limitations of grid-based notation gave way to graphical-based methods. Interestingly enough, the two systems have generally been segregated and have been thought to conflict with one another [19].

4. CONCLUSION

This paper sought to explore the possibilities of combining information design tactics, most notably those of Edward Tufte, and musical composition. Tufte's books trace common mistakes and important solutions in presenting information throughout history. These examples and lessons can provide musicians with a rich resource for solutions to displaying challenging musical material. As previously mentioned in Section 1.1, one of the primary questions designers of graphical content should ask is: *What is the thinking task*? The graphical representation of any given idea should help aid that task.

The examples presented herein were solely produced by the first author, as it was not the goal of this article to critique the works of others, or to highlight what makes a score good or bad. Composers are free to use any method or system necessary to express themselves, and in most cases the ends justify the means. Traditional western notation is highly customisable, and serves as the framework for a great deal of the contemporary music written today. It is, at its core, a highly successful and excellent example of information design. Furthermore, while it has the potential to be graphically difficult to navigate, western notation's visual grammar is widely recognized and familiar to its users. The difficulties of understanding its systemic structure are usually overcome in the early stages of a musicians career, allowing the experienced and professional to transcend any possible limitations of the notation in their music making.

Tufte's work is at its heart multidisciplinary, leaving an open framework for the interpretation of musicians. His ideas open the door to countless other persons and organizations who have discovered solutions to complex graphical questions. Musical composition is certainly complicated and multidisciplinary as well. Inspiration can be drawn from anything when writing music. The work of Edward Tufte and the world of information design has the potential to be a rich resource for imaginative compositions in the future.

5. REFERENCES

[1] E. R. Tufte, *The Visual Display of Quantitative Information*, 2nd ed. Cheshire, CT: Graphics Press, 1982.

[2] ——, *Envisioning Information*. Cheshire, CT: Graphics Press, 1990.

[3] ——, *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire, CT: Graphics Press, 1997.

[4] ——, *Beautiful Evidence*. Cheshire, CT: Graphics Press, 2006.

[5] ——, *The Cognitive Style of PowerPoint*. Graphics Press Cheshire, CT, 2003.

[6] Rudolf Schmid, "The Visual Display of Quantitative Information by E. R. Tufte," *Taxon*, vol. 38, no. 3, p. 451, 1989.

[7] A. Bose, "Visual Explanations by Edward Tufte," *The Indian Journal of Statistics*, vol. 59, no. 3, 1997.

[8] M. Zachary and C. Thralls, "An Interview with Edward R. Tufte," *Technical Communication Quarterly*, vol. 13, no. 4, pp. 447–462, 2004.

[9] T. Murail, "The Revolution of Complex Sounds," *Contemporary Music Review*, vol. 24, no. 2, pp. 121–135, January 2007.

[10] B. Bacon and M. M. Wanderley, "The Effects of Handedness in Percussion Performative Gesture," in *Proceedings of the 10th International Symposium on Computer Music Multidisciplinary Research*, 2013, pp. 554 – 560.

[11] M. Corballis, *Human Laterality*. Elsevier, 1983.

[12] M. Schrauf, B. Lingelbach, and E. R. Wist, "The Scintillating Grid Illusion," *Vision Research*, vol. 37, no. 8, pp. 1033–1038, 1997.

[13] J. Albers, "One Plus One Equals Three or More: Factual Facts and Actual Facts," *Search Versus Re-Search*, pp. 17–18, 1969.

[14] "Handedness in Percussion Sight-Reading," website, July 2014. [Online]. Available: `http://dl.acm.org/citation.cfm?id=2617995&coll=DL&dl=GUIDE`

[15] G. Cook, *Teaching Percussion*. Schirmer Books, 1988.

[16] M. Peters, "Hand Roles and Handedness in Music: Comments on Sidnell." 1986.

[17] C. Cardew, "Notation: Interpretation, etc." *Tempo, New Series*, no. 58, pp. 21–33, 1961.

[18] E. Brown, "The Notation and Performance of New Music," *The Musical Quarterly*, vol. 72, no. 2, pp. 180–201, 1986.

[19] P. Boulez, *Orientations: Collected Writings*. Harvard University Press, 1990.

NOTATION AS INSTRUMENT: FROM REPRESENTATION TO ENACTION

Eric Maestri
Université de Strasbourg, LabEx GREAM
Université de Saint-Etienne, CIEREC
eric.maestri@gmail.com

Pavlos Antoniadis
LabEx GREAM
Université de Strasbourg
IRCAM
katapataptwsi@yahoo.gr

ABSTRACT

The paper explores the hybridization of notation and instrument as a cognitive movement from representation to enaction. Features of such hybridization are latent in every notation, as a mix of descriptive and prescriptive functions. Current advances in the fields of computer music representation (interactive scores) and New Interfaces for Musical Expression, with precedents in graphic and action-oriented scores, are turning notation into a shared multimodal platform between composer and performer, liquidizing the limit between notation and instrument. We will present this dynamic rapport between scores and interfaces (haptic interactions, INScore, GesTCom, post-Klaus K. Hübler tablature notations of decoupled action-structures) in the light of theoretical models (enaction defined as navigation of affordances from the field of embodied and extended cognition, Leman's action-reaction cycle extended from instrument-making into notation, Veitl's conception of software as tablature, Atau Tanaka's definition of instruments as open-ended systems etc.). We are following an explicit line from new interfaces involving notation back to graphic and action-oriented scores, considering them in the theoretical framework of enaction.

INTRODUCTION: ONTOLOGY OF NOTATION TODAY

In an extension of its primordial role as recording of musical praxis and mnemotechnics, music notation today is still assuming the central position in the sophisticated communicative chain of conception, composition, performance and reception.

This role persists despite the 'performative turn' in musicology, which advocates the multiple nature of the musical work of art beyond an *Urtext* and into performances, recordings and improvisations [1, 2]; and despite the problematizations in view of music's medial extension, paradigmatically in early electronic music. [3]

The role of notation today could be described as one of attracting compositional activity and releasing performing activity. All compositional activity is aiming at the generation of notation, all performing activity is itself generated by notation, thus a linear model of musical communication.

Interestingly enough, the linear nature of this arrangement is perplexed by the omnipresence of performance inside composition and vice versa: From a composer's perspective, notation attempts to codify a future presence of performing bodies and instruments in virtue of their real absence in the act of composition; and from a performer's perspective, this set of virtual presences in the form of notation has to be deconstructed (through the understanding of the notation and of the composer's intention) and reconstructed in material presences.

Alternatively to this ambivalent communicative chain, notation can be viewed as equal constitutive part in a self–organized, feedback-loop dynamic system, in a formulation originating in the field of embodied/extended/enacted cognition [4]. At its current state of development notation can be thought of and further developed into *a shared multimodal platform for both composers and performers* in the form of a tablature and/or interface, that is: in the form of an instrument.

We will explore different manifestations of such hybridization, starting with Tomás and Kaltenbrunner *Tangible Scores* [4].

TANGIBLE SCORES AND GESTCOM AS COMPOSER AND PERFORMER PERSPECTIVES RESPECTIVELY

Tomás-Kaltenbrunner Tangible score: *inherent score* and multi-morphophoric sound elements

The question of an inherent-in-the-instrument score frames in new terms the problem of interaction design and affordance exploration of instruments and notations alike[1].

The problem of a differentiation between scores and interfaces is largely debated in the NIME community. A NIME designer develops a notation system that is inherent to the instrument. The designer thus cancels the difference between music composer and instrument maker: *the score is the instrument*. The definition is compatible with Atau Tanaka's definition of instruments as *open-ended systems*, whose architecture includes a structural-compositional layer, next to the input and output systems, mapping algorithms and sound synthesis systems.[5]

The example provided by *Tangible Score* highlights very well this particular evolution. He claims that different layers, namely the instrument and the score, accompany the interaction between the composer and the performer. However, the evolution of electronic instruments implies a radical change in this perspective: the construction of the instrument is not only an instrument-maker realization, but it becomes an act of composing.

Inherent scores are in this sense an expansion of what an instrument normally is: these instruments expand and reinforce their affordances, turning into objects acting in the sense of musical composition. The instrument implies gestures and sounds, exploding in a multiplicity of instrumental morphophoric elements. Duchez defines morphophoric: "The notion herein referred to as morphophoric - or form-bearing - element, has always and unfailingly guided musical action, that is to say strategies of production (inspiration, invention, representation, execution) and reception (listening, memorization). But this essential guidance is first of all only a more or less conscious, empirical practice based on immediate perception. Its efficiency, therefore, though direct and reliable, is limited, and it corresponds to what are generally called "primitive", orally-transmitted musics" [6].

[1] A demo of Tangible Score is available at : `http://vimeo.com/80558397`.

Graphic scores as proto-inherent scores

Tomás and Kaltenbrunner traces back the development of the notion of inherent scores in the 1960s and in particular in graphic scores.

The NIME designer programs the affordances. In this sense the instrument tends to be part of the composition, exactly as a graphic score was in the 50s or 60s. These scores are interfaces of interaction with the instruments: The sound result is open, but conducted by the graphic constructions prescribed by the score. *Inherent scores* are similar to *graphic scores*, despite the fact that the first are are sound producing and performable while the latter are only representational. As remembered by Tomás and Kaltenbrunner :

> […] performing became the creative exploration in freedom of the musical affordances, musical reactions or acoustic relations to the physical space performed, without the need of any kind of musical notation.

In this sense, inherent scores are evolutions of graphic scores, conceived as musical interfaces. Composers design the instrument, after Lachenman's motto: "*Konponieren heißt: ein Instrument bauen*".

The *tangible score* is the result of a compositional process that enacts gestures and strategies:

> We define a tangible score as the physical layer that is incorporated into the configuration of a digital instrument with the intention of conducting the tactile gestures and movements.

Thus, the tangible score influences and orients the process of enactment of the instrument: it affords tactile gestures and movements. In this sense this instrument embodies gestural scores.

However Tomás and Kaltenbrunner focuses mainly on the physical interaction, avoiding the problem of the acoustic one. For him *tangible score*,

> as a traditional score, it encodes a musical intention and delegates the decoding part to other agents.

That is partially true: a traditional score implies sounds that a gestural one does not. The score of a violin sonata is an encoding of the intention via the gestures, that leaves the decoding to another agent. However we must remark that we can't program differently the sound of a violin. In this sense the *tangible score* is not exactly traditional, but rather an exciting new extension of traditional possibilities. Each instrument has compositional constraints, but, until now, instruments are the result of historical and intersubjective evolution based on fundamental morphophoric elements – like pitches -; the *tangible score*, as mosts of NIMEs, is design on open morphophoric elements, that can be chosen by the composer or the performer, inventing in that manner different possible arrangements of *the score*.

GesTCom (Gesture Cutting through Textual Complexity)

A different example of a shared multimodal platform which amalgamates instrument, gesture and notation is the GesTCom. Its novelty lies in that it highlights the enactive potential of traditional musical scores from a performer- specific (rather than composer-specific) perspective.

It was developed in the course of a musical research residency 2013-2014 at the Ircam, as a prototype system based on the *a.* performative paradigm of embodied navigation of a complex score *b.* on the INScore platform and *c.* on the Gesture Follower [7]. The concept of corporeal (or embodied) navigation attempts to offer an embodied and medial performer-specific alternative to the classical UTI[2] paradigm. Instead of a strictly linear arrangement of its formants - understanding notation, then employing purposefully technique and then allowing, in the end, for expressive interpretation-, it proposes the conceptualization of learning and performance as embodied navigation in a non-linear notational space of affordances: The performer "moves" inside the score in several dimensions and manipulates in real-time the elements of notation as if they were physical objects, with the very same gestures that s/he actually performs. This manipulation forms indispensable part of the cognitive processes involved in learning and performing and transforms the notation. This transformation can be represented as a multilayered tablature, as in Figure 1.

b. INScore [8] is an open source platform for the design of interactive, augmented, live music scores.

INScore extends the traditional music score to arbitrary heterogeneous graphic objects: symbolic music scores but also images, texts, signals and videos. A simple formalism is used to describe relations between the graphic and time space and to represent the time relations of any score components in the graphic space on a *master/slave* basis.

It includes a performance representation system based on signals (audio or gestural signals).

It provides interaction features provided at score component level by the way of *watchable* events. These events are typical UI events (like mouse clicks, mouse move, mouse enter, etc.) extended in the time domain.

These interaction features open the door to original uses and designs, transforming a score as a user interface or allowing a score self-modification based on temporal events.

INScore is a message driven system that is based on the Open Sound Control [OSC] protocol. This message-oriented design is turned to remote control and to real-time interaction using any OSC capable application or device (typically Max/MSP, Pure Data, but also programming languages like Python, CSound, Super Collider, etc.)

A textual version of the OSC messages that describe a score constitutes the INScore storage format. This textual version has been extended as a scripting language with the inclusion of variables, extended OSC addresses to control external applications, and support for embedded JavaScript sections.

All these features make INScore particularly suitable to design music scores that need to go beyond traditional music notation and to be dynamically computed.

c. The Gesture Follower was developed by the ISMM Team at Ircam [9, 10]. Through the refinement of several prototypes in different contexts (music pedagogy, music and dance performances), a general approach for gesture analysis and gesture-to-sound mapping was developed.

The "gesture parameters" are assumed to be multi-dimensional and multimodal temporal profiles obtained from movement or sound capture systems. The analysis is based on machine learning techniques, comparing the incoming dataflow with stored templates. The creation of the templates occurs in a so-called learning phase, while the comparison of a varied gesture with the original template is characterized as *following*.

The *GesTCom*, equally rooted on embodied navigation, INScore and Gesture Follower, takes the form of a sensor-based environment for the production and interactive control of personalized multimodal tablatures out of an original score. As in the case of Embodied navigation (Figure 1), the tablature consists of embodied representations of the original (Figure 2). The novel part is, that those representations derive from recordings of an actual performance and can be interactively controlled by the player. The interaction schema takes the following feedback loop form (Figure 3).

More specifically, the input performative gesture produces four types of recorded datasets (gestural signals, audio, MIDI and video), which are subsequently used for the annotation, rewriting and multimodal augmentation of the original score. Those output notations are embodied and extended: They are produced through performative actions, they represent multimodal data, they can be interactively controlled through gesture and they can dynamically generate new varied performances.

[2] Acronym for Understanding-Technique-Interpretation

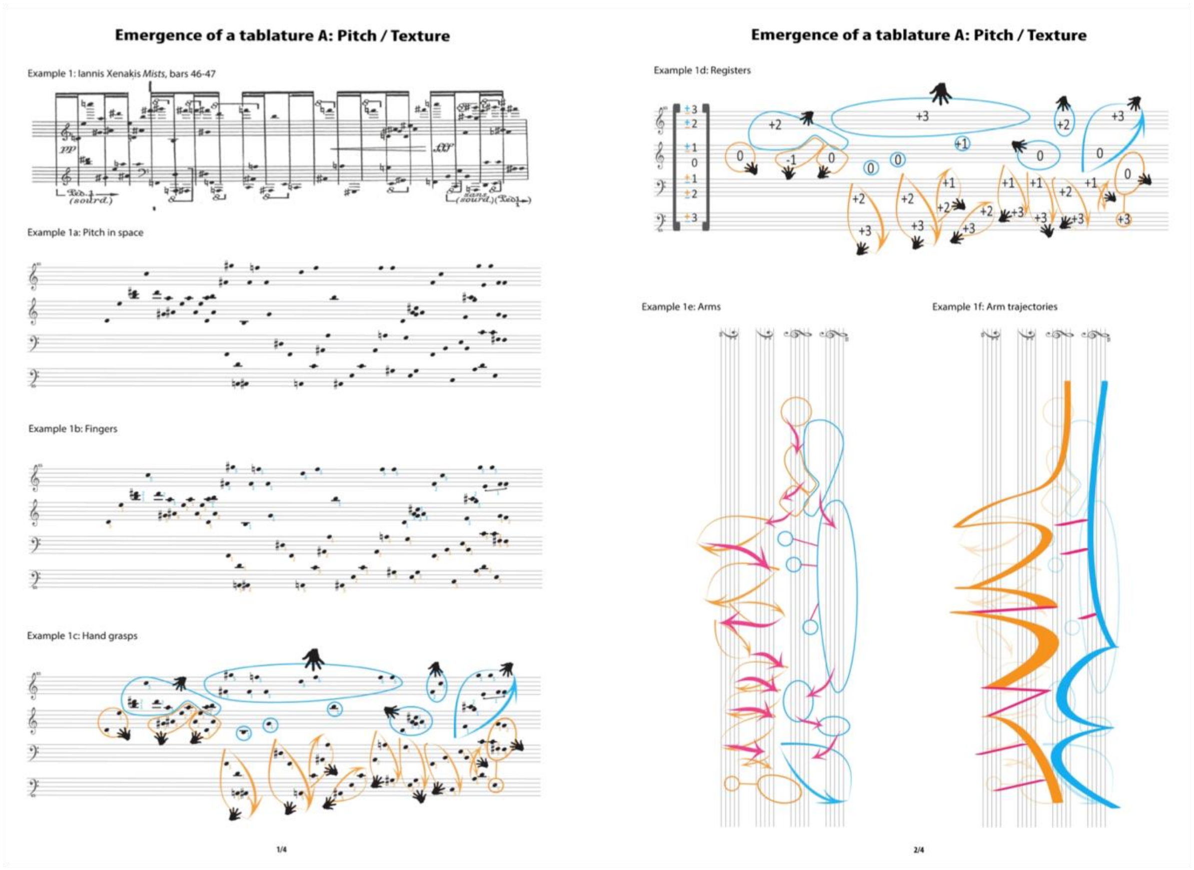

Emergence of a tablature A: Pitch / Texture
Example 1: Iannis Xenakis Mists, bars 46-47
Example 1a: Pitch in space
Example 1b: Fingers
Example 1c: Hand grasps
1/4
Emergence of a tablature A: Pitch / Texture
Example 1d: Registers
Example 1e: Arms
Example 1f: Arm trajectories
2/4

Figure 1. The embodiment of a Xenakian cloud / fingers-, hand-, and arm-layer in 1b, 1c, 1f respectively

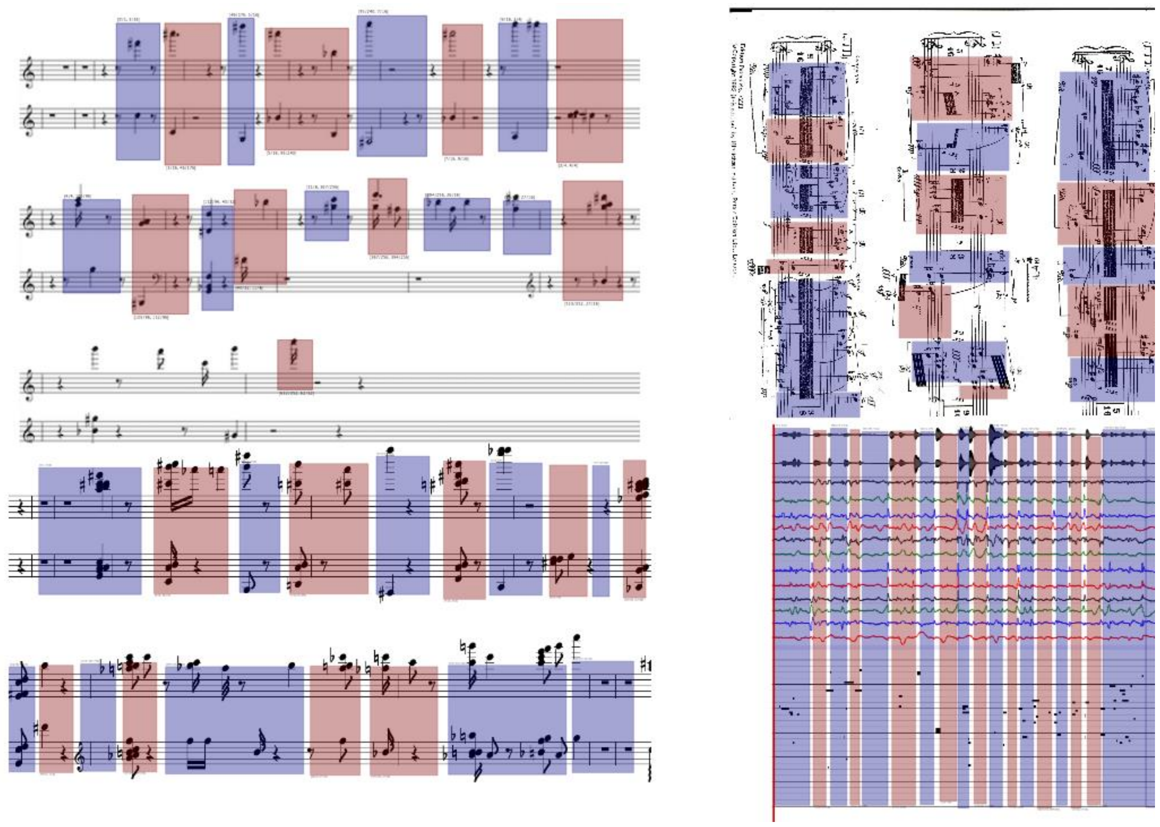


Figure 2. INScore tablature of combined representions. They can be synchronized with video and audio and interactively controlled. The player navigates between the several representations

They can be considered as the visualization and medial extension of the player's navigation in the score-space, creating an interactive feedback loop between learning and performance.[3]

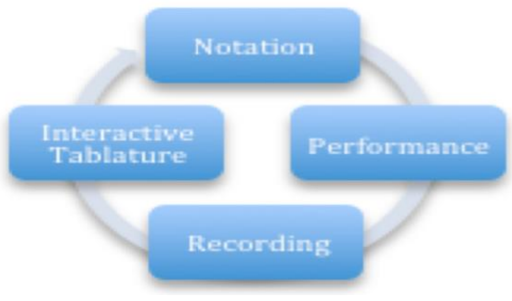


Figure 3. Interaction schema.

ACTION-BASED SCORES

Historical account

The relationship between notation and instrument, conceived as trigger of imagined and concrete gestures, emerges in various work since the 60s. The sonic invention of contemporary music restored the problem of notation, multiplying the number of possible morphophoric elements that inform the composition.

Our aim is to indicate some different perspective in an historical order. Even if incomplete, we remember three examples that seem to us to highlight the problem.

Lachenmann: Pression, for solo cello (1969)

One important composer that conceives composing as instrument making is Helmut Lachenmann. In a certain perspective the work of the German composer is inspired by phenomenology and primitivism, and more directly by the references to Schaeffer's *musique concrète*.

Lachenmann defined his music as *musique concrète instrumentale*. However, behind this intriguing definition, the composer is not making instruments, by exploring new possibilities that, still, are strongly idiomatic.

In *Pression*, a renowned piece for solo cello by Helmut Lachenmann, the composer invents new sounds and new writing for the cello. In this piece composed in 1969, the composer prescribes the cellist to play in unorthodox manners, precisely notating the gestures and the places that must be activated by the performer's gesture.The composer explores the instrument, exactly as the *tangible score* must be explored by the performer.

A milestone in subsequent developments towards the representation of independently organized, or decoupled, actions towards indeterminate sound results is offered by the work of Klaus K. Hübler, and in particular his article "Expanding String Technique" [11]. There, Hübler soughts to present a "completely new perspective on the instrument" through "an expansion of sound and technique that has its roots in the specific resources of the instrument and its manner of performance".

Aaron Cassidy: Second String Quartet (2010)

The activity of Aaron Cassidy is known for his original approach to the notation problem. Cassidy's scores invest deeply in the notion of musical gesture. Following the examples ok Hübler, Barrett and Frneyhough, Cassidy works on the instrumental decoupling: "a separation of the various activities of instrumental sound production" (Figure 4).

The work of Cassidy expands the approach that we highlighted in Lachenmann: Not only is there an exploration of instrumental affordances, but even an exploration of the performer's body affordances.

Therefore the instrument and the score are, means of representation and at the same time stimulation of the gestural content of the player's activity: Physicality is conceived in material terms.

In his *Second String Quartet* (Figure 4) Cassidy resumes the gestural parameters to a unique staff, making an evolution in relation with the former string quartet. The score has the role of being the interface for instrumental and physical enactement of the global musical body – the performer and the instrument. In this sense the score acts as a state space of gestural affordances. The relationship with sound, being open, reveals a coessential element with the *tangible score*: The graphics afford gestures on a known instrument, the string quartet, and the performer interacts with the sounds creating the acoustic output, that is not written in the score. In a similar manner the *tangible score* affords gestures creating open sounds possibilities. On the contrary Lachenmann indicates precisely the sound result.

On both our examples, the score is in the centre of the relationship between gesture and sound, being an abstract symbolic interface for physical movement, even if with different degrees of prescription.

3 A demo of GesTCom is available at:
`https://www.youtube.com/watch?v=KV9nQUhhyuI`

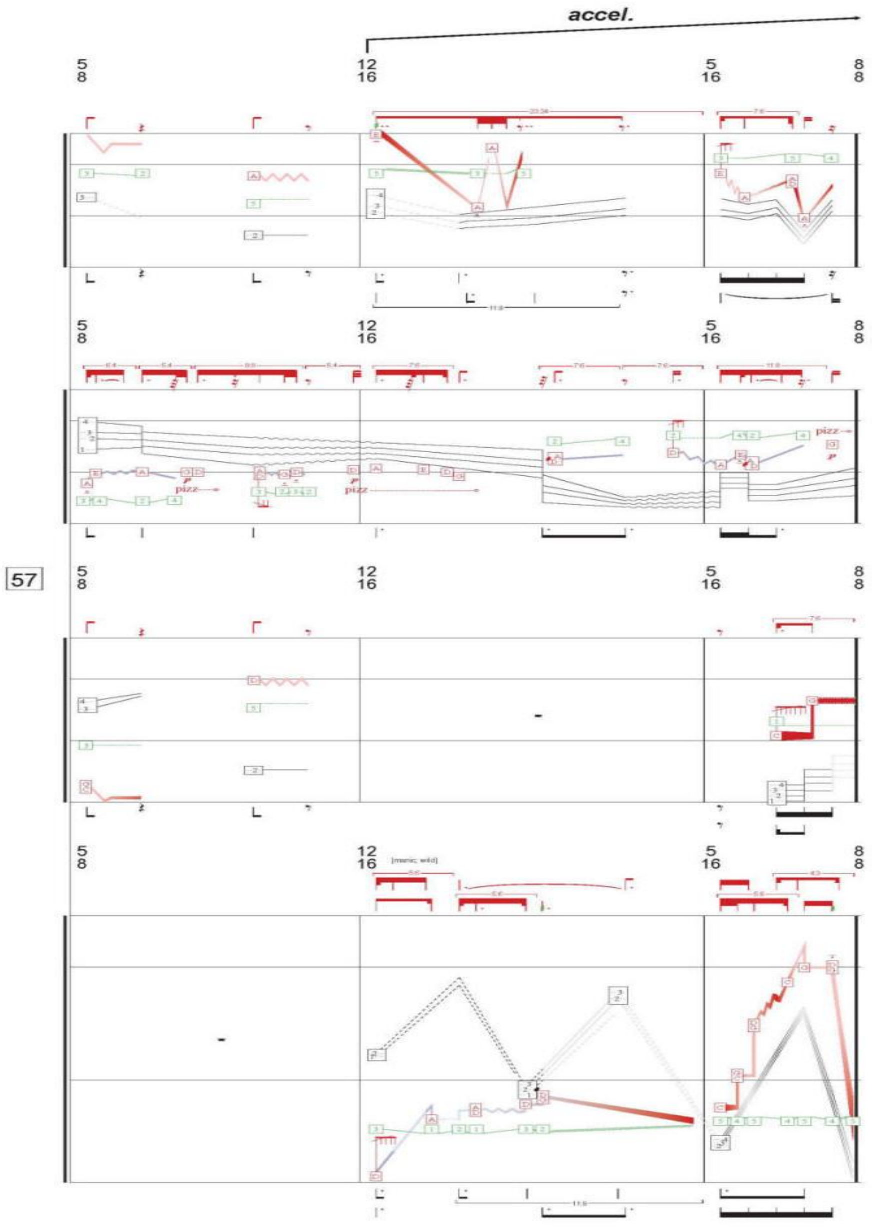


Figure 4. Aaron Cassidy's *Second String Quartet* page

THEORETICAL FRAMEWORK

Efforts of projection in time

We try to argue that scores, instruments and compositions seem to have a common essence. If scores are, or might be, abstract symbolic interfaces, and instrument concrete ones, we highlight how the recent evolution of new musical interfaces seems to make the limit fluid.

Scores and instruments not only collimate today in multimodal interfaces, but have, in our opinion, a common essence characterized by the typology of intentionality, based on the effort of projection of the maker: composition of scores or construction of instruments are forms of projection in time, based on enactive experience.

Making Musical instruments, making scores

Making of musical instruments involves action and perception; it also involves the understanding of the action-relevant value of sounds, the judgment of these sounds in view of musical ideals, and shaping physical environment that produces sound: projections of movements in virtue of the absence of physical presence.

The composer, the performer and the instrument-maker project the sound-object in time: they must project their subjective experience in an intersubjective dimension. Projection is expectation of reality based on past experience.

The action-reaction cycle proposed by Marc Leman as a paradigm for instrument making (and more widely, for music making and perception), frames theoretically, for us, this concept [12]. If the process of instrument making described by Leman as the synergic relationship between “Play, Listen, Judge and Change” is true, then the process of composition can be equally described. In fact

> While musical instrument is being built, a set of action-reaction cycles, which may occur on different time scales and perhaps in hierarchical order, transforms matter and energy into a cultural artefact for making music [Leman, 2007: 52]

There are forms of projections through writing that evolve in technology. Performers and composers are entailed in a similar form of projection, characterized by a different degree of distance from the gestural and sonic output. The projection is the conception of a process of accumulation of experience that comes to define the good shape of the instrument and of the score. Leman underlines the process as the Ratchet Effect:

> […] the actual process of instrument-making (ontogenesis) and the history of how an instrument evolves (phylogenesis) can be seen as the result of a repeated cycling of Play, Listen, Judge, Change. The action-reaction cycling is a dynamic model with the capacity to subsume a cumulative process similar to a ratchet effect [Leman, 2007: 54]

In our opinion, we can extend this model from instrument to notation, assuming that in both of them perception induces intentionality and anticipation: “the world is conceived from the viewpoint of action and prediction rather than merely based on the construction of gestalts”.

Scores are the result of a ratchet effect, in the sense that they simulate the economic growing of knowledge during the last centuries, similarly to the instruments. The abstraction of the musical practice in a few number of variables allows a global control of the instruments, that arrives to a certain control of the body of the performer. This kind of prescriptive approach is similar to the machines, that are totally, or almost, controlled. In this sense the composer uses the score as an instrument, as a temporal and physical interface of abstract interaction in time and space: *scores are extensions of the body of the composer in the body of the performer via the projection of the instrument represented by the score*. That creates a singular temporal dimension based on the absence and presence of the instrument: the composer constructs absences and the performer reconstructs the projected presences.

Notational system as performed system

We would like to suggest a framework of the definition of score as instrument, drowing a line between the programming of the sound result and the design of instrument and scores. We would like to argue that if scores are instruments, then this common essence is still developed in NIMEs.

As highlighted by Tomás and Kaltenbrunner, circuits are conceived as scores and instruments, because their combination implies specific sounds. This relationship is at the basis of the conception of synthetic instruments. Also for Max Mathews, computer is an instrument [13]; at the same time the computer is not a normal instrument, but it performs data that are memorized and activated.

In the case of NIMEs, the computer is still central. The computer controls the loudspeaker, but the musical interface controls the computer. It is a particular instrument that not only can be controlled by interfaces, like keyboards controls organs, but it can be programmed in infinite manners.

The interfaces have a role similar to that of scores: they generate information in real-time, but still record and encode data: interfaces are causal for scores.

Anne Veitl [14], following Cadoz's work [15], focuses on the notion of *causality*, that is the central element of the relation between scores and instrument. The comprehension and the definition of *causality* lies at the centre of the definition of the musical instrument. Veitl's model allows a kind of generalized instrumentality: highlighting the principle of *causality* fundamental, it becomes evident that instruments and score are part of the same causal process.

Criteria of a performed notational system

Considering the sound synthesis environments partitioned as score and instruments, Anne Veitl proposed to interpret softwares as notational systems.

Veitl proposed six criteria that seem to us to highlight some general properties of notational systems and instruments at the same time. These criteria stress the fact that softwares are notations, and, essentially, performable notations. A notational system is primarily :

a. *material*: it must be somewhere, memorized on a concrete and existing object, the paper or a hard disc ;
b. *visible*: that's why the machine language is not a notation, but softwares are visible ;
c. *readable*: it has to be read by a machine, a human being or both;
d. *performative*: it describes the action potential of a system. Softwares and computers are highly *performative* because

the material inscription is translated instantaneously in sound;

e. *systemic*: the signs, or the physical elements of the system can operate structurally ;

f. *causal*: notation must indicate and enable sounds. It must indicate the manner and the means necessary to produce the sound or the event.

In this sense, for Veitl, softwares are scores, thus NIMEs are expression of this essential character.

CONCLUSIONS

Technological advances have broadened our conception of notation and instrument as mutually shaping, action-oriented, open-ended systems, as much as they have contributed in their actual, material amalgamation.

Tomás' tangible score and Antoniadis' GesTCom offer instances of new interfaces-and-scores, which have historically followed up from graphic and action-oriented notations. In those instances, notation and instrument share common criteria (Veitl) and evolutionary cycles (Leman) beyond the prescriptive-descriptive classical dichotomy, materializing both representational and enactive cognitive features.

Eventually the very communicative chain and roles between instrument-makers, composers, performers and computer-music designers are to be genuinely rethought as cycles of synergy rather than linear models, with obvious implications for both pedagogy and creation in all respective fields.

REFERENCES

[1] N. Cook, *Beyond the Score: Music as Performance*. Oxford University Press, 2014

[2] A. Arbo and M. Ruta, *Ontologie Musicale*, Paris, Hermann, 2014.

[3] C. Seeger, “Prescriptive and Descriptive Music Writing”, *The musical Quarterly*, 44, 2, 1958, 184-195.

[4] E. Tomás, M. Kaltenbrunner, “Tangible Scores: Shaping the Inherent Instrument Score”, *Proceedings of the International Conference for New Musical Expression*, 2014.

[5] A. Tanaka, “Sensor-based Instruments and Interactive Music” pp. 233-255 in R. Dean, *The Oxford Handbook of Computer Music*. Oxford University Press, 2009

[6] M.E. Duchez, “An Historical and Epistemological approach to the Musical Notion of “form bearing” element”, *Contemporary Music Review*, 4:1, 199-212.

[7] P. Antoniadis, D. Fober, F. Bevilacqua, “Gesture cutting through textual complexity: Towards a tool for online gestural analysis and control of complex piano notation processing”, in *CIM 14 Proceedings* Berlin 2014

[8] D. Fober, Y. Orlarey, S. Letz: “INScore: An Environment for the Design of Live Music Scores”, *Proceedings of the Linux Audio Conference* - LAC 2012.

[9] F. Bevilacqua, N. Schnell, N. Rasamimanana, B. Zamborlin, F. Guedy: “Online Gesture Analysis and Control of Audio Processing”. In: J. Solis & K. Ng (eds.) *Musical Robots and Interactive Multimodal Systems*, pages 127-142. Springer-Verlag, Berlin Heidelberg, 2011.

[10] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, N. Rasamimanana: “Continuous realtime gesture following and recognition”. In *Embodied Communication and Human-Computer Interaction*, volume 5934 of Lecture Notes in Computer Science, pages 73–84. Springer Berlin and Heidelberg, 2010

[11] K. K. Hübler, “Expanding String Technique”, pp. 233-244 in CS Mahnkopf, F.Cox, W. Schurig Polyphony and Complexity, Wolke Verlag, 2002.

[12] M. Leman, *Embodied Music Cognition and Mediation Technology*, MIT Press, Cambridge, 2007.

[13] M. Mathews, “The Digital Computer as a Musical Instrument”, *Science*, 1963, pp. 553-557.

[14] A. Veitl, “Musique, causalité et écriture: Mathews, Risset, Cadoz et les recherches en synthèse numérique des sons”, *Musicologie, informatique et nouvelles technologies*, Observatoire Musical Français, Paris-Sorbonne, 2006, Paris.

[15] C. Cadoz, “Musique, geste, technologie”, in H. Genevois and R. de Vivo, *Les Nouveaux Gestes de la musique*, Editions Parenthèses, 1999, Marseille, pp. 47-92

TIMBRAL NOTATION FROM SPECTROGRAMS : NOTATING THE UN-NOTATABLE?

Andrew Blackburn
Fakulti Muzik dan Seni Persembahan
Universiti Pendidikan Sultan Idris
Tanjong Malim, Malaysia
andrew@fmsp.upsi.edu.my

Jean Penny
Fakulti Muzik dan Seni Persembahan
Universiti Pendidikan Sultan Idris
Tanjong Malim, Malaysia
jean@fmsp.upsi.edu.my

ABSTRACT

This paper outlines a research project currently underway in Malaysia that, through spectography, seeks to find models that might assist in the future development of a timbral notation. Located within the music creation and performance practices of the researchers, the project has elements of interculturality, which both enrich and inform the research. The authors consider the nature of a music score, the explicit and implicit information it carries, and how this impacts on the models being developed. The understandings elicited to date are not only located in music practice, but are underpinned and supported by the theoretical works of a number of theorists. The overall research project is broken down into smaller discrete sub-projects which are discussed, andcontextualized in the wider project. The paper includes a discussion of the score as artifact or 'thing' the relationships that are implicit within it, and the infinite potential it contains. Other outcomes are suggestive of a possible model of gestural notation which will be a further avenue of research. The paper concludes with suggestions of future research areas following the models of timbral notation being explored in this project.

1. INTRODUCTION

This paper is a brief exposé of a Fundamental Research Grant Scheme (FRGS) project, funded by the Malaysian Ministry of Education, being carried out at the Universiti Pendidikan Sultan Idris, Tanjong Malim, Perak in central Peninsular Malaysia – *Spectromorphological Notation - Notating the unNotatable? Modeling a new system of timbral and performance notation for ethnomusicological, musique-mixte and electroacoustic music compositions.* The focus of this fundamental research is broad, encompassing a range of intercultural, performance and sonic representation issues; and this report is necessarily of work in progress as the project is evolving clarity of direction and practical application. The project development and structure, research questions, reflections on the nature of the score and the creation of models for timbral representation are discussed.

This research is seeking answers to diverse timbre notation and music representation questions within three sub-projects that focus on, respectively, ethnomusicology, musique-mixte and electroacoustic music. Crucial to the first and second sub-projects is our interest in developing ways of representing timbre that can be understood from both Malaysian and Western perspectives of performance, and provide a live performance functionality. New compositions are being created as frameworks for these investigations that are experimenting with forms of notation that accommodate timbre as an addition to pitch and duration. A software independent means of notating electroacoustic music is a goal of the final sub-project. The project began in mid 2014, and the first and second sub-projects are currently underway. The third sub-project that focuses on electroacoustic notation will conclude in early 2016.

Denis Smalley (1994) begins to define timbre as "the attribution of spectromorphological identity" [1]. He points to the 'hazardous operation' of definition, of expanding the assumed notions of timbre based on acoustic sound and the trouble of refining and standardizing responses to such a complex element or identity. Within *Spectromorphological Notation: Notating the Un-notatable?* we are addressing both the acoustic and electroacoustic, aiming to create an investigative continuum that proceeds and informs from one to another. Elements of the study and documentation of the timbral characteristics of both traditional and modern instruments occurring in the initial stages of the research will lead to experimentation with notation and explorations of the relationships of score and performance. In the creation of new works, the transformation of the acoustic sound spectra through digital signal processing is extending this exploration into the electroacoustic context.

Further areas of exploration include an articulation of the relationship between the sound and context. This relationship is reflected in the scope of our definition of timbre based on Smalley's approach, and making recognition of Lasse Thoresen's assertion that we need to develop a terminology (and lexicon) to describe the "…phenomenology of music in experiential terms" [2]. This phenomenological approach to timbre was initially begun by Schaeffer and then carried forward by Smalley, and between the writings of all three, begins to accommodate the multiplicity of meanings of 'timbre': structural, contextual, analytical, tonal, and sound quality.

Timbral elements in musique-mixte works are central to interpretation and realization in performance, but often include somewhat vague or technology specific indications. The authors' experience as performers (flautist and organist) in the musique-mixte domain has prompted aspects of this study, and provides a practical basis for these explorations. In flute works, for example, timbre changes may be indicated by signs (often extended techniques) or words that can be highly evocative and poetic; the electronics may be indicated by effects or technical instructions such as fader control levels, or a particular form of synthesis. Where acoustic and electronic sounds merge, indications of timbre may become the 'property' of the software or mixing desk – the programmed effect. The authors suggest that a creative collaboration working within a performance environment to recreate the composer's intentions, rather than technical instructions, could be more effectively enabled with semiotically relevant timbral representation. In organ works, timbre is often suggested through assumed knowledge of historical performance practice[1], or specific stop indications combined with an understanding and knowledge of the instrument for which a piece was composed. In the works for organ and live electronics composed since 1998, the aural and spatial effect of the processing on the overall timbral environment is only 'discovered' in the space after all has been set up. A more specific representation of timbral effect in the score would allow the performers to adapt and optimally develop interpretation and technical set up according to the performance space.

Investigations of timbral descriptions of traditional instruments led us to Ngabut (2003) in *Kenyah Bakung Oral Literature: an Introduction* in which the author describes the odeng talang jaran (or jews harp) from the Borneo Kalimantan region. The description includes detailed descriptions of the instrument's construction (dimension, materials, and decoration), mode of playing, social function and many other cultural features, but makes only one reference to the actual sound of the instruments: "The sound produced resembles that of a frog" [3]. Assuming one knows the species of frog being referred to by the author, and what call it is giving, perhaps this is helpful. A motivating factor in this project is to try to find an objective, non-metaphorical process for notating the sound of the frog. Through spectrographic measurement we hope, as far as the visual can represent the aural, to find symbols and images that can communicate sound quality in all its complexity to a literate observer.

Referring to sound quality – its spectral content, sonic identity and recognition of source – Udo Will attests:

> "…It remains immensely difficult to 'talk about' them – oral cultures have no music theory. Things seem to be different in literate cultures, though. Through the very invention of writing systems, man has acquired means to cope with the elusiveness of sounds: the transformation from an aural-temporal form into a visual-spatial one. Sounds seem to be tamed and time seems more under control if treated spatially, however, this is only seemingly so because the accomplishments of such a transformation are limited and can at times be deceiving" [4].

Combined with the other informal explorations and considerations these comments became enabling texts to launch this exploration of timbral notation.

Central to the project is the music score itself – what is it, and what relationships the various participants each have with this thing or artifact? One common factor in all our understandings is of the score as an object of potential. The project is generating new questions and raising uncertainties about the nature or ontology of musical scores, as well as the syntactical conventions that exist in different cultures. Our references to Ingold and Foucault support this need for exploration. Kathleen Coessens calls the music score a "coded tool in the arts" and furthermore a score "…is a two-dimensional visual and coded artifact that allows for multiple performances or "resounding processes" by musicians…[and merging] the visual and the musical, the fixed and the dynamic, space and time" [5]. These are well-understood concepts, which confirm our (Western) cultural understandings of the ontology of a musical score. The project is also grounded in non-Western, oral-based paradigms: what does the score (as artifact or 'thing') mean within these cultures?

This project will explore the creation of models for the timbral and performance notation of music, incorporating both acoustic and electronic sound sources initially working with traditional instruments, then within contemporary Western Art Music research through the creation and performance of new musique-mixte and electroacoustic

[1] e.g. *Organo Pleno* for North German baroque instruments, or the *Tierce en Taille* of the Classical French organ tradition.

compositions using these possible models and systems."[2] The overall project consists of two conferences, bookending three sub-projects that, taken together, provide opportunities to envision the possibilities and value of timbral notation, aiming to create models from which to develop practical performance based scores, which are of value to participants in each area. The project's co-researchers are practitioners in ethnomusicology, acoustic, electroacoustic and musique-mixte as academics, creators and performers.

Already queries are arising regarding our ontological understandings of what comprises a score, and, how it functions and communicates, particularly over time. As Marc Battier, who presented at the project's opening conference in June 2014, observed

"… the preservation of a [musical] score is a big issue, and has implications for notation".

A score must be in a form which can be understood and read over long historical time frames, and in a form which allows long term archival storage and retrieval.

2. THE PROJECT

Research questions have evolved for each sub-project based on the following investigative parameters:

1. Can an intuitive notation system for electroacoustic music be developed from spectral analysis and spectromorphological representation?

2. What are the elements that composers, musicologists, performers require from a notation system and how can these be represented?

4. Can spectrographic analysis and software be used to provide a method for defining and identifying unique qualities of Malaysian indigenous instruments?

5. Can this information be used to 'describe' and notate the specific individuality of sounds, materials and performance methods in ways that expand the range and musical vocabulary of the ethnomusicologist?

6. What parameters of analysis can be defined to provide useful and universally 'understood' symbols using spectrographic softwares?

2.1 Issues Arising – a problem statement?

This research project is adopting a multi-faceted approach to exploring the possibilities of creating scores that describe and notate timbre and which might eventually come to some degree of functionality. The practice of the various co-researchers and the paradigm of their experience provide multiple sites and contexts for the research. These paradigms also encompass the realms of traditional and non-Western music performance, acoustic Western art music performance and music creation, and the environments of electroacoustic and musique-mixte. The range of modes of transmission of music and musical ideas is equally broad – being passed from one generation to the next, from creator to acoustic and electronic performance. Further, it encompasses oral and rote learning, common notation scores to software, and works dependent on the software that was used to create them as a way of preserving them. In these notation systems, with the exception of the electroacoustic performance software, there is no way of describing the quality of imagined sound – our 'frog call'.

What is notation and what is a score? Both are separate objects, but intertwined with cultural, ontological and semiotic inferences, all of which impact the artifact we call the score. In Western art music, a score is an artifact (often on paper, but perhaps in other media or in soft copy) used to communicate the musical ideas of the score's creator to the performer and, with an assumption of the performer's active creative input, to the listener. In traditional Malaysian music, we can describe the score as, more commonly, a series of memories and traditions, perhaps articulated mnemonically but not, until quite recently, written down. In this traditional music, pitch and rhythmic inaccuracies that arise from the use of common practice notation are considerable but, except that they are measured in spectrograms, beyond the scope of this presentation.

Our conception of the score as 'thing' connects the meaning of the score to Ingold's theory of 'correspondence' [6] drawing us to a significant difference between a score and a spectrogram – the spectrogram is an historical document – 'this sound **was** like this'. We can measure the sound that happened in this way, and read it as such. Contrarily, a music score (with its multiplicity of meanings) is a 'thing' of possibility [7]. It is a creator/composer's conception of some sounds that, if recreated in this or that way by the performer, has the possibility of generating non-verbal ideas and concepts in the minds of the performers and listeners. Manuella Blackburn suggests a new way of using the spectrogram to help generate compositional ideas in her exploration of the potential of spectromorphology and its associated language as a process for composition" [8]. She writes,

"… spectromorphology can be approached from an alternate angle that views the vocabulary as the informer upon sound material choice and creation. In this reversal, vocabulary no longer functions descriptively; instead the vocabulary precedes the composition, directing the path the composer takes within a piece. This new application is an attempt at systemization and an effort to (partly) remedy the seemingly endless choice of possibilities we are faced with when beginning a new work" [8].

[2] Blackburn (2014), `http://spectronotation.upsi.edu.my`

Blackburn's suggestion of the use of spectromorphology as a compositional tool suggests the possibility of changing the historic nature of the spectrogram into one of potential.

Other researchers have struggled with many of the issues that have arisen in our individual and collective deliberations. Rob Weale[3] in the EARS Glossary of terms, Spectromorphology notes there is both interdependence and dynamism in the word *spectromorphology*. Whist not reducing the historic quality of a spectrogram, this is helpful to this project for the conceptual development of a timbral score, as he describes spectromorphology as a tool for "describing and analyzing listening experience." He continues: "The two parts of the term refer to the interaction between sound spectra (spectro-) and the ways they change and are shaped through time (-morphology). The spectro- cannot exist without the -morphology and vice versa: something has to be shaped, and a shape must have sonic content" [9]. So there is the possibility of dynamism in a spectral score.

The score, if incorporating some form of spectrography, will probably contain graphics that also have semiotic qualities. Martin Herchenröder, in discussing the score of Ligeti's graphic score of the organ work *Volumina*, adds musical and performative gesture to the inherent quality of a score as he attests

> "…, it is a coherent system of signs [semiotics], whose details can all be translated into musical patterns. A look at the third page of the score of *Volumina* illustrates the cluster through visual analogy. The horizontal dimension corresponds to the flowing of time: The time sequence of musical events (according to the reading habits of the western world) is a left-right succession of notes. Thus, in principle, each event is fixed in time - the new cluster in the right hand as posits an approximately after 17 seconds, after another 10 seconds of complete, another 4 seconds later" [10].

It has been argued that this gestural quality is also semiotic and tied to the sonic gesture. The 'left-right' succession of symbols and their vertical location on the page indicating pitch (high/low) also has sonic inferences that offer potential for developing elements of performance notation [11]. Treating the score of *Volumina* as an *xy* graph for time and pitch, we can see that the evident gestures and sonic shapes are potentially useful in timbral notation. It is an area where the left-right and vertical associations could be helpful in 'notating' gestures, which, by their musical outcomes are also timbral. O'Callaghan and Eigenfeldt have demonstrated how spectral density can be implied within acoustic and musique-mixte compositions [12]. Combining colour, which can be ascribed various meanings, and graphic, gestural notation as outlined above is proving a rich potential model in creating gestural notation in the musique-mixte performance environment. This model is described in greater detail below.

2.2 The Sub-Projects

This research project is structured with three principal sub-projects, which, though operating in parallel, allow a sequential development of models and notational ideas. The applications used to create the spectrographs used in this project are Pierre Couprie's eAnalysis [13] and Sonic Visualiser [14].

2.2.1 Project 1 Ethnomusicology Project

The ethnomusicological sub-project, using spectrograms provides traditional music professionals with an objective understanding of the nature of the sound quality of specific instruments, and the musical or ritual context in which they prefer to use it. As a music tradition that is oral, transmission of music and pieces is achieved by rote, repetition, and aural memory. This research is not an attempt to standardize the sound of instruments. Instead, it adds to the knowledge of the Wayan Kulit artform, which is presently in a difficult phase. In parts of Malaysia, including one of its places of origin, Kelantan, it is banned. University programmes, such as those maintained by UPSI, are important in the continued artistic viability and vibrancy of Wayan Kulit (Director of Kelantan Arts and Culture Museum, personal communication in Penny/Blackburn FRGS The Imaginary Space, 2014). This spectrographic process is demonstrating the value of profiling instruments, allowing makers objective knowledge of the range of sounds preferred by the musicians who play and perform.

The first process within this sub-project has been to record the sound of, then create spectrograms of, traditional Malaysian *Wayang Kulit* shadow puppet music theatre. UPSI maintains a group of resident musicians specializing in this musical form. In performance, a group of four to six musicians and the master puppeteer are all located out of sight behind a large translucent screen, which is the stage for the shadow puppets. Our study includes an exploration or profiling of sounds preferred by professional traditional musicians in certain percussion instruments.

The orchestra of the *Wayang Kulit Siam* (as found in Kelantan, Malaysia) consists of percussion instruments including a pair of double-headed drums – gendang, a pair of single-headed goblet-shaped drums – gedumbak, a pair of vertically standing drums (gedug) hit with beaters, hand small cymbals – kesi, a pair of inverted gongs – canang, and, a pair of hanging knobbed gongs – tetawak. Melodic instruments include the serunai (a double-reed instrument, similar to the shawm) and a three-string spike

[3] www.ears.dmu.ac.uk/spip.php?rubrique28

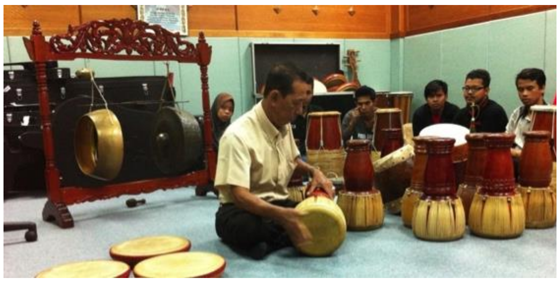


Figure 1. Testing the Gedumbak

bowed instrument – rebab. The instruments, while individually important, gain their true significance in an ensemble and dramatic context. When making recordings of various instruments, initially it seemed sensible to just record the instrument in a dry unadorned environment. However, in order for the *Wayang Kulit* leader (Pak Hussain) to make his assessments, the recordings that ended up being made were of the whole group playing

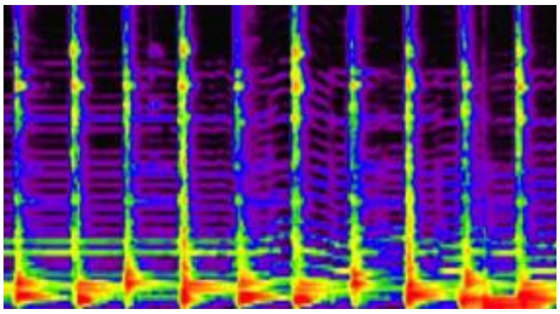


Figure 2. Spectrogram of Wayang Kulit ensemble –segment of recording focused on Gedumbak with strong onset feature.

while testing out the Gedumbak for different dramatic environments. Selecting instruments for their suitability in a given drama (normally, the stories are drawn from thirty of so traditional stories) means that the players are more interested in their collective role than the individual, so the recordings were made to reflect this. The gedumbak was close miked, and the rest of the ensemble sound was allowed to spill into these microphones. The longer red lines in the last section of this short segment show the moment when the serunai enters.

Why, for example, is one pair of Gedumbak preferred in one piece over another? Spectrograms can show a profile of the sound, which may then be attached to a musical (or in the case of *Wayang Kulit*) dramatic context. Spectrograms further show us that by using different modes of playing, different timbral qualities can be emphasized in the same instrument – brighter or more mellow and so on. Co-researcher, Mohd. Hassan Abdullah has pointed out that mnemonic forms of teaching and communicating musical content in Malaysian traditional music also imply different timbral and gestural modes of playing. So, we ask the question, can this content be given a visual (spectrographic) or written form, and applied in the other projects?

A second strand in this project investigates a 'Western' facet – the creation of a recorded catalogued of extended flute performance techniques, using a concert flute, which have been spectrographed and analysed for their characteristics. These characteristics are being extracted for the development of a form of spectral representation that can be adapted for use in common notation scores, particularly for acoustic instruments. This strand has been productive, opening ideas and knowledge that leads into the second sub-project, combining acoustic and electroacoustic musical contexts in new compositions.

2.2.2 Project 2 Musique Mixte project

The musical score as semiotic medium can be understood as an "infinite substance" [15**]** that activates the musician's ability to imagine and translate notation into a temporal unfolding of new knowledge and experience. As we look towards extending performance practices into new conceptual contexts and relationships, new paradigms that reflect and drive new expressions and activities evolve. Timbral notation as a context of change motivates explorations of shifting performative relationships, new ways of thinking and performing, and a reconceptualization of the score/performer relationship.

This part of the project will create models for spectrographic notation as performance scores. Analyses of notation, timbre and organology associated with chosen instruments and electronics (*musique mixte*) will be undertaken to develop a framework for investigating spectrographic analyses, evaluations and outcomes. New works will generate performance analyses through phenomenologically based studies, following the sound spectrum and performer responses to new musical works.

We question the role of the score as mediator between mind and sound [16**].** What information is conveyed through spectral timbre notation? What are the semiotic implications of sound codification? Is the information rigid, or a point of departure for the performer? A performer's notation needs clarity and embedded knowledge or information that directly communicates to them – that is clear, readable, interpretable, and informative of what the music is about. The multiple layers of a spectrograph emit different levels of information, multiple meanings, different streams of representation – all systems that require understanding and evaluations of the relations of the score. What can a performer expect – information of spectral density? Aesthetically, a spectrograph is a beautiful object – but just how effective and informative is it as timbral notation for the performer? Is it instructional, or suggestive, gestural, strictly coded or freely interpretable? Can a spectrograph be as revealing or evocative as a beautifully notated score? Can it evoke spatialities, memories, or sonic energies? What is the need for this as notation?

Investigating the recordings and single frame spectrographs of the Western flute extended techniques will allow us to experiment with the flautist to see how effective this is in the re-creation of timbres. The form of timbral representation on which we will focus does not consider fundamental pitches or duration, rather an emphasis of specific overtones. Pitch and duration are indicated using common musical notation. As a catalogue of sounds and acoustic performance techniques, the spectrographic series (see Figure 3) as a research process model provides some ways forward to link timbral representations to scores in a musique-mixte environment.

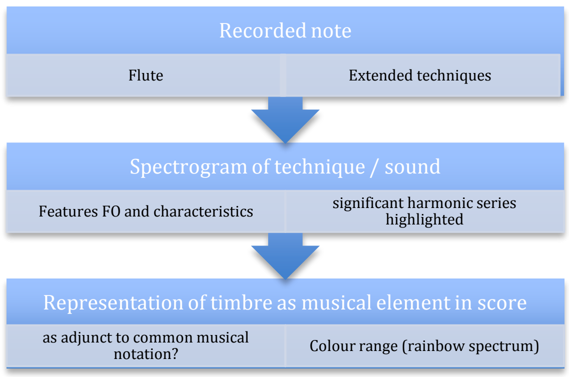

Recorded note
Flute
Extended techniques
Spectrogram of technique / sound
Features FO and characteristics
significant harmonic series highlighted
Representation of timbre as musical element in score
as adjunct to common musical notation?
Colour range (rainbow spectrum)

Figure 3. Process of model development from Flute Catalogue of extended techniques

According to Bhalke et al. [17], a single frame of a spectrograph contains information including:

(i). Fundamental Frequency;
(ii). Harmonic components;
(iii). An indication of the relative amplitude of the harmonic components;
(iv). Spectral Centroid.

Can the composer say to an instrumentalist "play your instrument to reproduce this quality of sound," indicating their musical ideas through spectrographic information? It is our sense that such compositional detail potentially denies certain instrumental ontologies. In art music, the instrumentalist brings many personal and musical contributions to the performance outcome – what we might loosely define as 'interpretation'.

Yolande Harris argues that sound "binds people together in space in a contextual manner" [18]. This concept of the score as relationship – between performer and notation, between composer and performer, between memories, communications, live sound, recorded sound, gesture, or cultural practices – interrogates and challenges our experience of performative modes and conventions. These are relations and ecologies that can be examined through concepts of heterotopia (Foucault), contexts of understanding (Gadamer) and correspondences (Ingold). Can a circle can be drawn around the score as space, and the spectrograph act as facilitator and activator of that space? In a recent study of intercultural music performance in Malaysia[4], heterotopia was articulated through the performative lens, the performance as a context for understanding artistic realisation of intercultural knowledge and experience. This space was posited as an ecology: a set of relationships, the music, the performance, a symbiosis of elements of the cultures, collaborations and connections that occur [19].

Digital media tends to handle music as encoded physical energy, while the human way of dealing with music is based on beliefs, intentions, interpretations, experience, evaluations, and significations [20], but the exploration of timbral notational elements and relations might activate questioning and re-assessment of values; the search of microstructures might lead to a search for sonic essences and deeper self understandings; new dimensions evolve, new ways of thinking and living (performing) result. These questions engage us with discovering the meaning of the music as new dimensions of musical practice open up.

2.2.3 Two models arising from Sub-Project 2?

Limiting the new content of notation to timbral qualities (and for now limiting its measurement to the 'single frame' timbre information outlined above Bhalke et al), the research teams are sensing that the information contained in a spectrogram is useful in determining the timbral quality of a sound at a given moment and dynamically over time. However, the uniform colour response of spectrographic software means the strongest elements of tone are always brightest and use fixed colour ranges. The spectrogram responds to relative prominence of tone with the same colour spectrum. To ask a performer to play a green or red sound on this basis is, therefore, meaningless. However we if (for example) ask a player to make a sound with the first and second harmonics (octave and fifth above the FO) emphasized (giving the tone a somewhat nasal quality), it could be indicated above common music notation in the form of, perhaps, a rainbow colour grid (i.e. red, orange, yellow, green, blue, indigo, and violet) related to the first seven frequencies of the harmonic series. Retaining common music notation, means that the target note (FO) would be black. An instrumentalist would need to acquire knowledge of the possible harmonic series for their instrument, and the instrumental technique required to produce such combinations of sound. Timbre indications could then be read as coloured dashes above musical phrases or individual

[4] *The Imaginary Space: Developing models for an emergent Malaysian/Western electroacoustic music* (2012-14). Fundamental Research Grant Scheme, Ministry of Education, Malaysia.

notes. This approach allows the retention of score relationships and its potential quality while providing the composer with a means of specifying timbral quality within their score.

Adapting this approach using graphic notation could include the dynamic quality of the spectrogram, which can include indications of duration, pitch, relative amplitude and the ASDR envelope. These could be incorporated into a form of notation that may resemble a colourised version of, for example, Ligeti's score of *Volumina.* The representation of music in this form might also be readable as a type of gestural notation, of pertinence to software instrumental performance, though this is a process currently being examined in our pieces. This approach must be considered only a starting point – a model for investigation.

2.2.4 Electroacoustic Music Project

The third sub-project is an exploration of the use of spectrograms to create a form of timbral notation, which could be used in electroacoustic compositions as a way of preserving the music independently of the software/hardware used to create them. As noted earlier, finding a mode of preserving a score is a major concern. One possible approach, and which culturally locates this research in South East Asia, is an exploration of the potential of adapting 'Uthmani' notation used in Qurannic recitation as a form of timbral or gestural notation. This exploration is not based around content, but is focussed on the context of how 'Uthmani' is used, written and recited 'through sound'. Hasnizam Wahid from UniMAS – Sarawak, and one of Malaysia's leading electroacoustic composers, is particularly focusing on this area. This project is yet to begin as the first two projects are creating many of the fundamental bases that must first be established. It is anticipated that this detailed research will begin in July 2015, continuing until the end of the year.

3. FUTURE PATHWAYS

Having identified some of the possible pathways for finding models of spectrographic or timbral representation in a score, this section suggests directions that this research might follow. They are not presented in order of preference or significance, but remain possibilities that address the outcomes of the research so far, outcomes yet to be realised and issues and meeting challenges so far identified.

If one were to wish for a software, and we will look at supporting software development in later research phases, it would be along the lines of a reverse-action of spectrographic software – i.e. a program such as eAnalysis currently takes an audio file and from it creates an image: is it possible to take that spectrogram and create an audio file to 'recreate' the sounds of the original file.

A simple outcome (though conceptually complex) would be to take some of the various software packages and have them sonify a spectrogram. Some simple experimentation with existing software packages, using Audio Paint [21] have been undertaken. The results using these are not promising. The concept might be helpful in realizing electroacoustic scores without access to the software used to create it. There are many issues and concerns at this juncture, which make this process one for a separate and continuing research project, developing and evolving models that might be forthcoming from this project. Some of the problems lie in impact of the space in which a sound is being projected and its influence on timbre. For multichannel electroacoustic works there is the question of how one will 'record' the original sound – as separate channels with individual spectrograms, which might then be reconstructed? Combined with the possibilities of the models outlined above, and acknowledging the many complexities, is a worthy goal to gain the ability to recreate fixed works long after the original software or hardware that created it is lost.

4. CONCLUSIONS

Our research to date seems to allow an optimistic attitude that spectrograms can be used as the basis of a timbral notation. The cultural significance of the score as an artifact and the relationships it implies – from composer/creator to performer to listener – must be accounted for in any new notation practices that develop to allow for specific timbral elements demanded by the composer. Our suggestion within instrumental contexts of a rainbow spectrum adds a new layer of complexity to the score, but we assert this enriches the various relationships established within the score's environs. The model of gestural notation appears to have the potential to provide a technically workable yet semiotically rich notational ontology, which will provide the basis for investigation in the electroacoustic/acousmatic context. In this sub-project, it is predicted that what Smalley describes as the discrimination of "…the incidental from the functional" [22] will be major areas of consideration. In many ways, findings relating to this project are the posing of more questions. Nevertheless, some elements of what will develop into models of timbral notation are suggesting themselves to the research group.

Acknowledgments

The authors would like to acknowledge the support of the following people and centres:

Co-researchers: Associate Professor Dr Mohd. Hassan Abdullah, Associate Professor Dr Hasnizam Wahid, Associate Professor Dr Valerie Ross.

Professor Marc Battier – Université de Paris (Sorbonne)

The Research Management and Innovation Centre at Universiti Pendidikan Sultan Idris, and its Director Associate Professor Dr Tajul Said.

Malaysian Ministry of Higher Education and the Fundamental Research Grant Scheme.

Research Assistant, Hafifi Mokhtar.

5. REFERENCES

[1] D. Smalley (1994) ”Defining Timbre - Refining Timbre” in *Contemporary Music Review* 10 (2) pp.35 – 48.

[2] L. Thoresen (2001/4) *Spectromorphological Analysis of Sound Objects. An Adaptation of Pierre Schaeffer’s Typomorphology.* The Norwegian Academy of Music p. 2.

[3] C.Y. Ngabut (2003) *Kenyah Bakung an oral literature:An introduction*. Last accessed 20/01/2015 http://www.cifor.org/publications/pdf_files/Books/social_science/SocialScience-chapter12-end.pdf

[4] U. Will *The magic wand of Ethnomusicology. Rethinking notation and its application in music analyses.* Accessed 20/01/2015 http://music9.net/download.php?id=6039

[5] K. Coessens (2014) “The Score beyond Music” in P. de Assis, W. Brooks, K. Coessens *Sound and Score: Essays on Sound, Score and Notation*. Leuven University Press: Ghent. p. 178.

[6] T. Ingold (2008) *Bringing Things to Life: Creative Entanglements in a World of Materials* accessed 25/01/2015 http://www.reallifemethods.ac.uk/events/vitalsigns/programme/documents/vital-signs-ingold-bringing-things-to-life.pdf.

[7] M. Blackburn (2009) *Composing from spectromorphological vocabulary: Proposed application, pedagogy and metadata* Accessed 30/01/2015 http://www.ceiarteuntref.edu.ar/blackburn, para 3.

[8] Ibid, para 1.

[9] R. Weale (N.D.) “Spectromorphology” in *ElectroAcoustic Resource Site.* Accessed 29/01/2015 on http://ears.pierrecouprie.fr/spip.php?rubrique28

[10] M. Herchenröder (1999) *Struktur und assoziation. György Ligetis Orgelwerke*. Schönau an der Triesting. Wien: Edition Lade, pp. 62-63.

[11] A. Blackburn (2013) “Sourcing gesture and meaning from a music score: Communicating techniques in music teaching” in *Journal of Research, Policy & Practice of Teachers and Teacher Education,* Vol 3, No 1, pp. 58 – 68, p.60.

[12] J. O’Callaghan & A. Eigenfeldt (2010) “Gesture transformation through electronics in the music of Kaija Saariaho” in *Proceedings of the Seventh Electroacoustic Music Studies Network Conference* Shanghai, 21-24 June 2010 www.ems-network.org. 2010.

[13] P. Couprie, eAnalysis http://logiciels.pierrecouprie.fr/?page_id=402

[14] http://www.sonicvisualiser.org

[15] D. Barenboim, (2009) *Everything is Connected*. London: Phoenix.

[16] M, Leman (2008) *Embodied Music Cognition and Mediation Technology*. Cambridge, MA: The MIT Press.

[17] D.G. Bhalke, C.B. Ramo Rao, D.S. Bormane, M. Vibhute (2011) “Spectrogram based Musical Instrument Identification Using Hidden Markov Model (HMM) for Monographic and Polyphonic Music Signals” in *ACTA Technica Napocensis Electronics and Telecommunications* Vol 52, No 12.

[18] Y. Harris (2014) Score as Relationships: From Scores to Score Spaces to Scorescapes, in *Sound and Score: Essays on Sound, Score and Notation*. Ghent: Leuven University Press.

[19] J. Penny (2015 upcoming) “The Mediated space: Voices of interculturalism in music for flute” in *Routledge International Handbook of Intercultural Research.* Eds P. Burnard, K. Powell, E. Mackinlay. Routledge: Abingdon

[20] M. Leman (ibid.)

[21] http://www.nicolasfournel.com/audiopaint.htm

[22] D. Smalley (ibid.) p.41.

COMPOSING WITH GRAPHICS : REVEALING THE COMPOSITIONAL PROCESS THROUGH PERFORMANCE

Pedro Rebelo
Sonic Arts Research Centre
Queen's University Belfast
p.rebelo@qub.ac.uk

ABSTRACT

The research presented here is product of a practice-based process that primarily generates knowledge through collaboration and exchange in performance situations. This collaboration and exchange with various musicians over a period of five years that constitutes a body of practice that is here reflected upon. The paper focuses on non-instructional graphic scores and presents some insights based on performances of works by the author. We address how composition processes are revealed in graphic scores by looking at the conditions of decision making at the point of preparing a performance. We argue that three key elements are at play in the interpretation of these types of graphic scores: performance practice, mapping and musical form. By reflecting particularly on the work *Cipher Series* (Rebelo, 2010) we offer insights into the strategies for approaching the performance of graphic scores that go beyond symbolic codification.

1. INTRODUCTION

Composition and performance practices involving the development of notation that operates differently from common music notation go back to the 1950's. Composers such as Mauricio Kagel, Karlheinz Stockhausen, Krzysztof Penderecki, John Cage, Earl Brown and Morton Feldman are commonly named as pioneers in this type of practice. These composers have typically engaged in graphic scoring during specific periods of their careers and have left bodies of work, which include innovative custom-designed notation alongside works using conventional notation. One needs only to reflect on the musical languages associated with these composers to realize the diversity of the aesthetic field laid out here. Graphic score practices in themselves cover a wide range of notational strategies, from simple extensions of common music notation to completely new models for the use of graphics as a device for communicating musical structures. This paper addresses works that are characterized by an approach to graphic notation that bypasses the symbolic and focuses on communicating musical structures in graphical form. This approach minimizes, or at times, completely abolishes instruction in favour of a freer approach to sharing and interpreting musical ideas. A deliberate decision to develop notational elements that are not conveying specific or determined performative actions has significant impact on the compositional process. Does it make sense to speak of a score that does not provide information to be read as commands for producing specific sound events? The relationship between the choice of notation and a composer's wider aesthetic project is discussed by Wadle :

> "the prescriptive notational innovations of Helmut Lachenmann, would reveal much about the composer's conceptualization of the performance techniques he calls for." [1]

The dynamics of determinacy and indeterminacy and their relation to notation are well known in the work of John Cage. [2] Cage arguably spent much of his career developing notational strategies that embody his philosophy of music. Mark Applebaum's extra-musical pictographic design informs gesture and form in his Metaphysics of Notation (2008) while handing over much of the musical decision making to the performer.

We argue that there are qualities in music communication which go beyond the symbolic and operate at a level of engagement which not instruction based. Both *Cardew's Treatise* (1962) and the iconic *December 1952* by Earl Brown, are notable examples of scores which raise more questions than answers and hence place the performer in a particular decision making situation. In this context, the contract between composer and performer is subverted to allow for a level of autonomy for the performer while preserving a sense of trust. One can argue that decisions about how a score is going to be approached are at play in all types of musical documents, including those based on common music notation. The types of decisions involved and the implication of specific choices to the sound result arguably come to the foreground in non-instructional graphic scores. In this paper we are particularly concerned with the qualities and characteristics of this decision making process and how they relate to the act of composing with graphics. In order to articulate this relationship we will begin not with the compositional process or intention but rather with a reflection on the dynamics of trust and engagement at the point when a performer decides to work with a non-instructional graphic score. Two distinct situations can occur which have a significant impact on subsequent performance preparation. This has to do with whether performer and composer are in communication with each other or not. In the first case, it is not uncommon for performers to need assurance that there is indeed no interpretative code behind the score. The assumption, even for performers who are accustomed with graphic scores, seems to be that the score is a mediator for a musical structure that pre-exists in the composer's mind. A situation in which performer and composer are not in communication is perhaps more illustrative of the process of performance preparation of these kind of works, seen as the performer arguably gains full autonomy. We will address three aspects, which determine how a score is transformed from a static document into an enabler for music performance in a creative ecology evolving musicians, instruments, venues, audiences etc... These three aspects focus on 1. cultural context and performance practice traditions, 2. relative connections/mappings between graphical and musical languages from the perspective of texture and gesture, and 3. the emergence of form as a derivation of the score's ability to frame musical time.

2. PERFORMANCE PRACTICE

It is important to bear in mind the relationship between composers and performers when it comes to the development of graphic scores. It doesn't take an exhaustive historical survey to recognise that the majority of composers interested in graphic scoring are also performers (John Cage, Barry Guy, John Zorn, Anthony Braxton, Mark Applebaum to name but a few). As such, traditional relationships of power and responsibility between these two roles begin to break down. As a composer engages in graphic scoring for his own performance practice, a culture of interpretation begins to emerge. In performance practice, the graphic score, or any type of score for that matter, becomes part of a broader musical experience.

The score is part of music making just as social relationships are. This musicking [3] determines a performative context in which the score is just one of many elements and doesn't necessarily gain the status of unquestioned authority it has in other musical traditions. The very function of a score as a symbol for 'the work' is in many instances also problematized with graphic scores. In her discussion of Cardew's Treatise, Virginia Anderson discusses the function of a score and what it represents for Cardew in contrast to Stockhausen (to whom Cardew was an assistant).

> "For Stockhausen, the performance is made in his service; the piece remains his and the performers should divine his intention even when it is not written down. For Cardew, the score is the responsibility of the performers once it is composed." [4]

This performer responsibility is exactly what we want to address through reflecting on the unspoken rules that emerge from any kind of music making. In the case of Cardew, his Scratch Orchestra (1962-72), set up to perform his other iconic work – The Great Learning – stands as a group of collaborators who commit to a rather specific ideology of music making and therefore share an approach to music which no doubt determines how the work with graphic scores unfolds. Cardew notably lays out his vision of social and musical dynamics in *A Scratch Orchestra : draft constitution* :

> "A Scratch Orchestra is a large number of enthusiasts pooling their resources (not primarily material resources) and assembling for action (musicmaking, performance, edification)." [5]

As with any music tradition, non-instructional graphic scores carry with them conventions and agency, which relate to how a specific performance lineage develops. As such, an understanding of this lineage becomes an important element in approaching graphic scores. Performance practice itself influences how a particular score is used.

3. MAPPING

Given the absence of the code that determines how a symbol on a page signifies a particular sound event, non-instructional graphic scores suggest an alternative way of relating graphics to sound. Returning to Cardew, the precision of the graphics and the importance of conscious decision making when preparing a score, is articulated in his *Treatise* handbook :

> "The score must govern the music. It must have authority, and not merely be an arbitrary jumping-off point for improvisation." [6]

The role of improvisation in the context of graphic scores is beyond the scope of this paper but it is nevertheless worth reflecting on how, for Cardew, the practice of improvisation stands opposed to the type of music making required when working with a score. One can however observe that most performers working with graphics would consider themselves improvisers, even though when performing a score, free improvisation is not the primary mode of engagement.

Without a code but still with the notion that the score governs the music, the graphic elements inevitably suggest a process of mapping, a set of relationships between the language of the graphics and a musical language (which is invariably situated in a particular performance practice as discussed above). This mapping can take the form of literal association (dense graphics – dense musical texture, graphical weight – musical dynamics, qualities of lines and shapes – musical gestures) or more formalised and codified strategies. In any case, the performer is faced with deciding on how this mapping will occur; either for a particular performance or a deliberate codification for a score to be repeated over multiple performances. In contrast to the work conducted in the area of parameter mapping in computer systems [7], the type of mapping discussed here is relatively unexplored. The mapping processes at question here implicate both multimodal perception, as explored in fields such as visual music [8], and musical practices and conventions, which range from cartoon gestural symbiosis in the music of Carl Stalling to mathematical translation of curves and textures in the work of Iannis Xenakis.

4. EXTRACTING STRUCTURE AND MUSICAL FORM

An element that is pervasive in the act of engaging with scores of any sort is the realisation of musical structure and form. This is partly to do with the relationship between music, as an ephemeral time-based phenomena and the physical score as an outside time artifact representing a sequence of events that can be seen at a glance. From the layout of the page to the palette of graphic elements employed in a score, a sense of structure is inevitably conveyed through framing (page layout, margins, relationship between pages) and placement of discrete elements (shape, colour, scale, repetition). It is in this domain that the compositional process is revealed. This happens as a process that shifts an understanding of a graphic score as a visual object to a musical one. An object which is made to speak the same language as all other elements of music making: the relativist language of ‘louder than’, ‘same as before’, ‘more dense’, ‘higher’, ‘lower’, ‘slower’, ‘faster’ etc… This relativism is particularly pronounced as performers face a score, which clearly contains musical information but no code to produce instructions. All decisions are then made from the score and in relation to the score.

5. REVEALING COMPOSITION

The three aspects at play when preparing a graphic score for performance as discussed above gradually reveal the compositional process and the making of the score itself. This process is driven by musical thinking of varying degrees of determinacy (i.e. more or less precise musical structures). It is also guided by a relationship with notation as material, its affordances and conditions. The ways in which different types of notation strategies enable composers to operate directly on musical elements to the extent that to compose and to notate can be seen as the same action, has been discussed elsewhere [9]. In order to better articulate this revealing of the compositional process we will refer to the work *Cipher Series* as an example.

> “*Cipher Series* is a collection of graphic scores that are displayed to audience and performers in accordance to a fixed temporal structure generated for each performance. The performance plays on the role of notation as a mediator of listening, setting up a performative condition based on interpretative strategies based on engagement by both the performer and the audience. The change from one graphic score to the next has immediate formal implications for the music and acts as a way of articulating shifts in musical material or interpretation strategy.” From Cipher Series’ performance notes (Rebelo, 2010)

As can be seen in the images below, *Cipher Series* employs line drawing (created by hand on a graphics tablet and vector graphics software) in a black and white paginated format. The score is a collection of pages, to be played independently or in sequence. The most common performance format is a pre-determined timed sequence for seven pages. Each page has a pre-determined duration between 40 and 90 seconds and the transition between pages is cued by a 10 second countdown. In this version of the work, the sequence is run twice. In the first iteration, the beginning 30 seconds from each page are recorded and then played back during the second. The sound projection of this playback is intended to be placed as close as possible to the instrument (e.g. loudspeaker inside the piano body) in order to expose the ambiguity of what is live and what is pre-recorded. By exposing a specific graphics-sound relationship twice we explore the very nature of mapping and interpretation. The moment a recording is triggered projecting the sound events made when that same graphic score first appeared, the performer is faced with the decision of whether to imitate her previous interpretation, complement it or indeed do something entirely different. The score of *Cipher Series* was conceived for audience display, which further exposes the decision-making process. By displaying the score the performer is following (without the cued countdown that triggers a change of page) the audience is also invited to derive their own mappings and musical structures.

The layout of *Cipher Series* on the page follows a number of conventions, which are apparent without the need for rules on interpretation. These include the landscape layout with orientation determined by legend at the bottom right corner. This mode of presentation suggests left to right reading although this is not specified. Each page presents a self contained musical sequence of events which can be played once or more times given a specific duration. A number of pages have relatively complex and detailed graphics, at times resembling eastern calligraphy. The density of events makes it practically impossible to engage in a "one-to-one" gestural mapping (i.e. one visual stroke determining one musical gesture) much as in Applebaum's *Metaphysics of Notation*. This is a deliberate attempt to invite the performer to engage with the score in ways other than scanning though events at a regular pace. In fact, in my own performances of the score I often focus on sub-sections of the page for repetition.

The most apparent compositional strategy employed here is perhaps the modular approach to the page as a frame for musical activity. In this context the transitions from page to page articulate the most striking musical changes. Even without a process of codification a performer preparing such a score will respond to the change of scale and texture evident in the difference between page 1 and page 2 below.

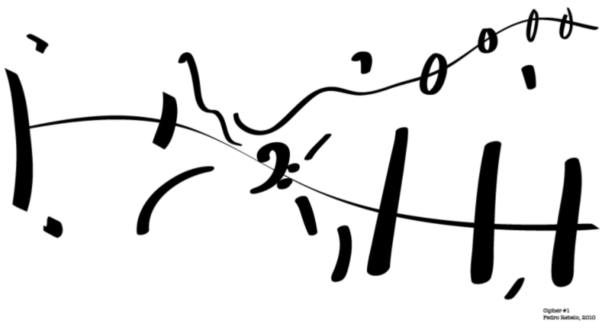


Figure 1. *Cipher Series*, p. 1 (Rebelo, 2010)

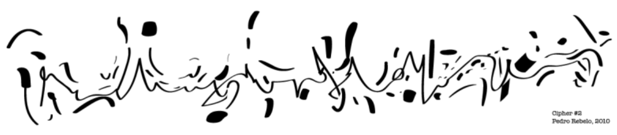


Figure 2. *Cipher Series*, p. 2 (Rebelo, 2010)

Cipher Series was the first in a sequence of works that share this type of graphical language (*Quando eu nasci*, and *Trio* both from 2011). These later works are designed for ensembles and develop the language to reflect a sense of musical parts, which inhabit the same. In *Trio* a simple colour scheme assigns each performer to a part while all other elements of the score remain non-instructional. Compositional strategies here reveal themselves also in the way the three parts relate to each other. Relationships of accompaniment, continuation, counterpoint, synchronisation can be derived from the score to inform musical performance.

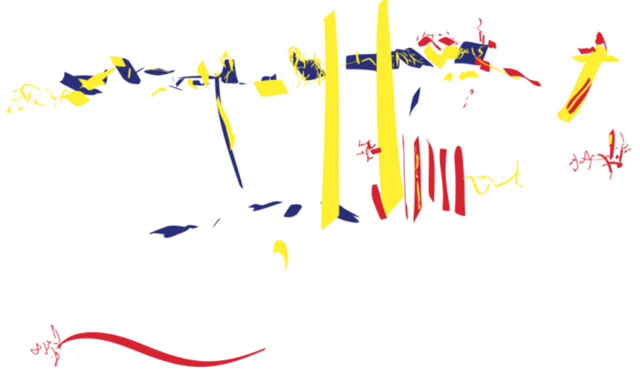


Figure 3. *Trio*, p. 1 (Rebelo, 2010)

6. CONCLUSIONS

By focusing on a type of graphic score practice that is deliberately un-codified and not based on the delivery of instructions for performance, this paper articulates the dynamics at play during the process of performance preparation. We argue that the autonomy transferred to the performer, or to be more precise, to the performance condition, is an act that reveals the compositional thinking behind a work. By bringing meaning into a score, a performer is following a roadmap created by a composer but deciding on how the journey is to unfold. The score as a roadmap gains the function of a document establishing musical circumstances, which within a performance practice become one of many elements determining the making of music. Composing with graphics ultimately reflects a desire to see the score not as the embodiment of "the work" but rather as a working document which only comes to live in the social workings of music making.

Acknowledgments

The insights discussed here derive from the experience of performing graphic scores with musicians such as Evan Parker, Elizabeth Harnick, John Eckhardt, Steve Davis, Franziska Schroeder, Basak Dilara Ozdemir, Ensemble Adapter, Ricardo Jacinto. Thank you all for you musical generosity.

7. REFERENCES

[1] Wadle, Douglas C. "Meaningful scribbles: an approach to textual analysis of unconventional musical notations." *Journal of Music and Meaning* 9 (2010). Pritchett, James. *The Music of John Cage*. Vol. 5. Cambridge University Press, 1996.

[2] Small, Christopher. *Musicking: The Meanings of Performing and Listening*. Wesleyan University Press, 1998.

[3] Anderson, Virginia. "'Well, It's a Vertebrate …': Performer Choice in Cardew's Treatise." *Journal of*

Musicological Research 25, no. 3–4 (December 1, 2006): 291–317.

[4] Cardew, Cornelius. “A Scratch Orchestra: Draft Constitution.” *The Musical Times* 110, no. 1516 (June 1, 1969): 617–19.

[5] Cardew, Cornelius (1971). Treatise handbook, including Bun no. 2 and Volo solo. London, Edition Peters.

[6] Hunt, Andy, Marcelo Wanderley, and Ross Kirk. "Towards a model for instrumental mapping in expert musical interaction." *Proceedings of the 2000 International Computer Music Conference*. 2000.

[7] Evans, Brian. "Foundations of a visual music." *Computer Music Journal* 29.4 (2005): 11-24.

[8] Rebelo, Pedro. “Notating the Unpredictable.” *Contemporary Music Review* 29, no. 1 (February 2010): 17–27.

ACCESS TO MUSICAL INFORMATION FOR BLIND PEOPLE

Nadine Baptiste-Jessel
IRIT-UT2
baptiste@irit.fr

ABSTRACT

In this paper we describe our approach to helping blind people access musical information. Guidelines of our approach are centered on information accessibility according to user disability. We present the process which allows musical information to be coded and converted so that it may be read, played and analysed by a blind musician. We focus our approach on the various levels of description of the score done by several codes and we exploit and describe existing results like BMML (Braille Music Markup Language) defined during Contrapunctus European project. We describe and comment on different scenarios using existing free conversion modules and software to obtain a score in BMML that may be read and manipulated by blind people using BMR (Braille Music Reader). We recommend the tutorials created during the Music4VIP European project.

1. INTRODUCTION

Some IT solutions exist to help blind people to access music, but analysis of these reveals both their utility and their limits. As Antonio Quatraro (blind musician) says, there are many factors which hinder the musical education of blind people - the lack of special needs training of teacher in mainstream schools and conservatoires, the difficulty of finding music scores in an accessible format and the persistent idea that music can be only learnt by ear.

Compared with existing methods of converting music into Braille like [1] and [2] our solution is based on the design of BMML (Braille Music Markup Language) [3]. To explain the process we first describe the principles of Braille music, in the next part the tools and code used to translate a score into an accessible format and in the final part we recommend the use of BME2 (Braille Music Editor) and BMR (Braille Music Reader) [4].

2. BRAILLE MUSIC PRINCIPLE

The rules used to create a Braille music score are presented in the New International Manual of Braille Musical Notation compiled by Betty Krolick [5]. It is important to note that, just as with conventional musical notation, this is an international code and so it is possible to exchange Braille scores between different countries. To explain the challenges involved in learning Braille music we divide the rules into three types: the simple rules, the presentation rules and the contraction rules. In this chapter we also describe BMML (Braille music markup language).

2.1 The simple rules

These are the rules used to transform music information into one or more Braille characters.

For example: the G clef is indicated by three Braille characters: >/l - although this information is not so important in Braille because the octave signs, rather than clefs on a staff, indicate the register of specific pitches in Braille music.

The octave sign is placed immediately before the note, for example the 4th octave mark is _

The name of a note is indicated by the four upper dots of a Braille character :

⠙	⠑	⠋	⠛	⠓	⠊	⠚
C	D	E	F	G	A	B

Table 1. *The Braille Name of note*.

The duration is indicated by the two lower dots, as shown below.

Whole notes and 16ths -

Half notes and 32nds '

Quarter notes and 64ths ,

Eighth notes and 128ths .

So a short simple score will be transcribed:

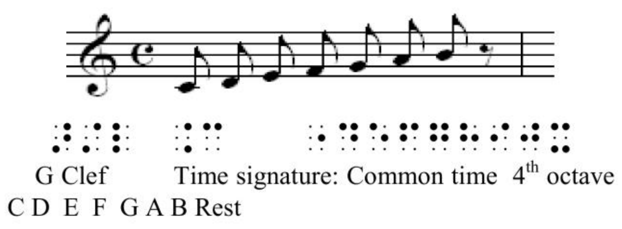


Figure 2. A simple score in Braille.

2.2 Presentation rules

Two presentations exist for keyboard instruments or other ensembles: bar over bar and section by section.

Bar over bar presentation presents a Braille line for each stave and the first note of each bar appears in parallel.

Section by section presentation presents a number of bars for one stave followed by a number of bars from the other.

These different presentation rules are available for all the score.

Other presentation rules exist to add in the Braille score the corresponding print page number to facilitate collaboration with sighted musicians.

2.3 Contraction rules

There are two types of contraction rules, dot reduction and character reduction. These different rules reduce the reading time and the number of pages in a Braille score. The rules are designed to help reader with a good knowledge of Braille music.

Example of dot reduction:

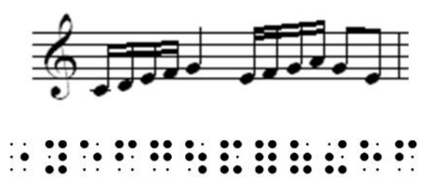


Figure 3. Example of Braille dot reduction.

The first note of the first double group is written with dots 3 and 6 but for the other note these duration dots are missing. The same reduction is not possible in the second part of the example because there would be an ambiguity with the last two notes.

Example of character reduction :

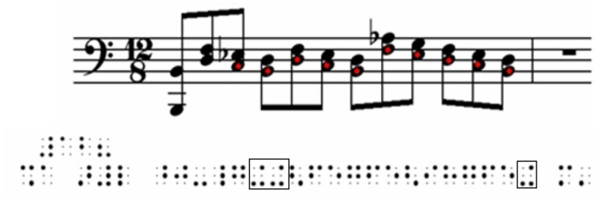


Figure 4. Example of Braille character reduction.

When the same interval appears several times the first interval sign is doubled and then one interval sign is placed at the end.

To store all the Braille information we created the BMML code during the Contrapunctus project.

2.4 BMML

BMML code was designed with following goals:

- to encode Braille structure and content as defined [3],
- to facilitate conversion from and to other music notation encoding such as MusicXML [4],
- flexibility to support different Braille music dialects.

The grammar of BMML is specified in [3] .Very briefly we can say that the BMML elements are of three types:

- a specific header in which is encoded all data relating to the document archiving and its structure,
- container elements which require a specific number of “children”. A child can be another container or a text element,
- text elements which represent the Braille text coded in Unicode.

BMML attributes are used to encode the meaning of each text element. A lot of them are required.

The following paragraph shows an very simple example of BMML but BMML can support more complex notation (tuplets, ornements, …) which permits its use by professional musicians.

3. A SCENARIO TO TRANSFORM A SCORE IN BMML

The objective of this scenario is to prove that it is possible to transform automatically a score in pdf format to a score in BMML. This example uses only free tools.

3.1 First step

The first step consists of finding a score in pdf format. It is possible to find this kind of score in an online library.

The score is Bach, Johann Sebastian, Minuet BWV Anh. 114 which is a public domain score found in the Petrucci Library site http://imslp.org/wiki/Notebooks_for_Anna_Magdalena_Bach_%28Bach,_Johann_Sebastian%29

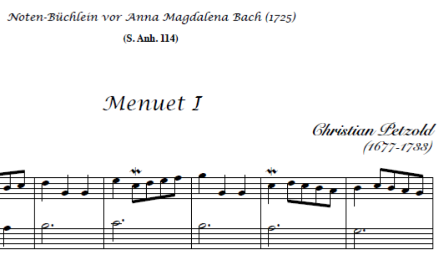


Figure 5. The *Minuet* in pdf.

3.2 Second step

We use the trial version of the Myriad-online.com product call PDFtoMusic Pro to convert the pdf score into MusicXML format.

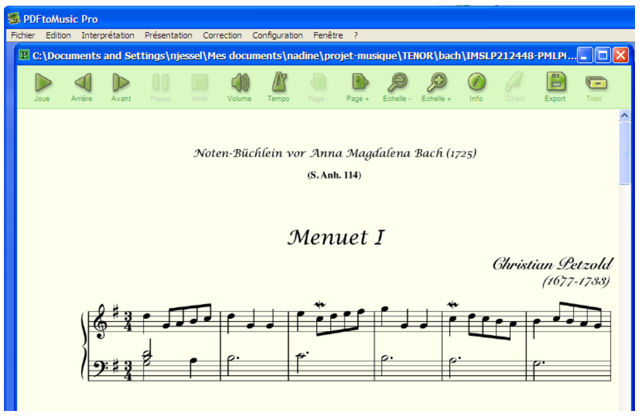
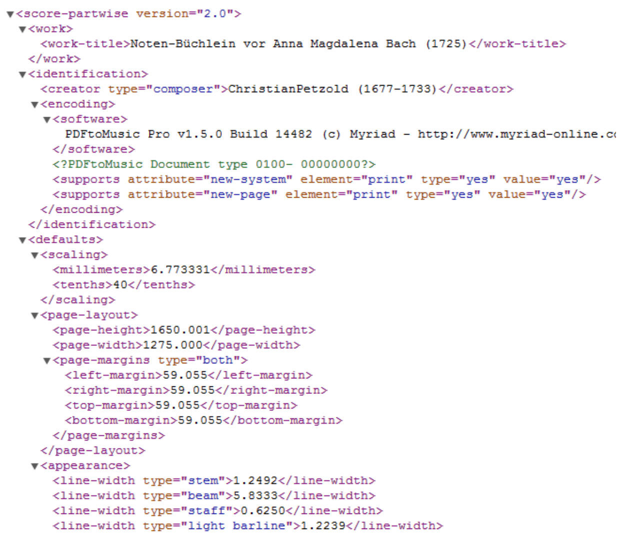

PDFtoMusic Pro
Fichier Edition Interprétation Présentation Correction Configuration Fenêtre ?
C:\Documents and Settings\njessel\Mes documents\nadine\projet-musique\TENOR\bach\IMSLP212448-PMLP...
Joue Arrière Avant Pause Arrêt Volume Tempo Page Page + Echelle - Echelle + Info Chant Export Tirer
Noten-Büchlein vor Anna Magdalena Bach (1725)
(S. Anh. 114)
Menuet I
Christian Petzold
(1677-1733)

Figure 6. The *Minuet* after music recognition.

The MusicXML document generated contains layout information and note information as follows:

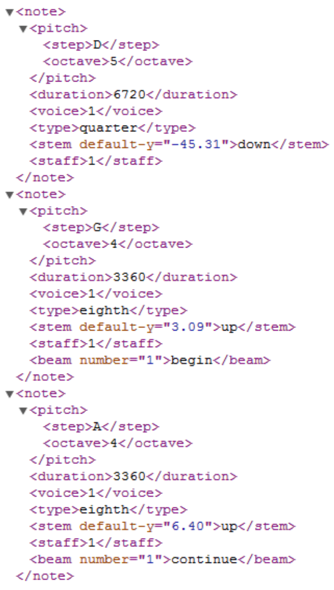


```
▼<score-partwise version="2.0">
  ▼<work>
     <work-title>Noten-Büchlein vor Anna Magdalena Bach (1725)</work-title>
   </work>
  ▼<identification>
     <creator type="composer">ChristianPetzold (1677-1733)</creator>
    ▼<encoding>
      ▼<software>
         PDFtoMusic Pro v1.5.0 Build 14482 (c) Myriad - http://www.myriad-online.c
       </software>
       <?PDFtoMusic Document type 0100- 00000000?>
       <supports attribute="new-system" element="print" type="yes" value="yes"/>
       <supports attribute="new-page" element="print" type="yes" value="yes"/>
     </encoding>
   </identification>
  ▼<defaults>
    ▼<scaling>
       <millimeters>6.773331</millimeters>
       <tenths>40</tenths>
     </scaling>
    ▼<page-layout>
       <page-height>1650.001</page-height>
       <page-width>1275.000</page-width>
      ▼<page-margins type="both">
         <left-margin>59.055</left-margin>
         <right-margin>59.055</right-margin>
         <top-margin>59.055</top-margin>
         <bottom-margin>59.055</bottom-margin>
       </page-margins>
     </page-layout>
    ▼<appearance>
       <line-width type="stem">1.2492</line-width>
       <line-width type="beam">5.8333</line-width>
       <line-width type="staff">0.6250</line-width>
       <line-width type="light barline">1.2239</line-width>
```

Figure 7. The layout information in MusicXML code.

In this first part of score we can see a lot of layout information which will not be found in the Braille score.

The note information with pitch, duration and octave signs is similar to the information in Braille.

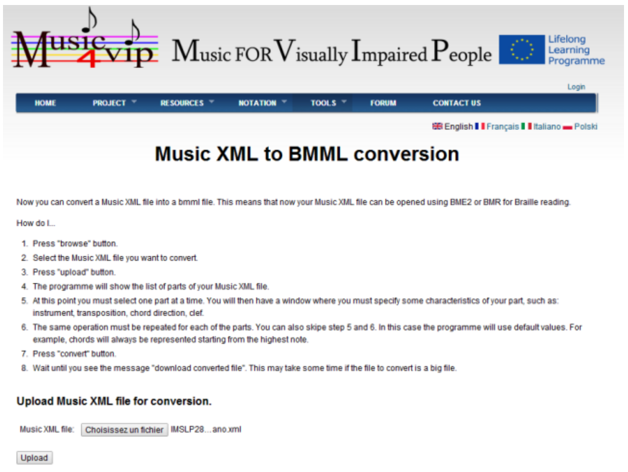


```
▼<note>
  ▼<pitch>
     <step>D</step>
     <octave>5</octave>
   </pitch>
   <duration>6720</duration>
   <voice>1</voice>
   <type>quarter</type>
   <stem default-y="-45.31">down</stem>
   <staff>1</staff>
 </note>
▼<note>
  ▼<pitch>
     <step>G</step>
     <octave>4</octave>
   </pitch>
   <duration>3360</duration>
   <voice>1</voice>
   <type>eighth</type>
   <stem default-y="3.09">up</stem>
   <staff>1</staff>
   <beam number="1">begin</beam>
 </note>
▼<note>
  ▼<pitch>
     <step>A</step>
     <octave>4</octave>
   </pitch>
   <duration>3360</duration>
   <voice>1</voice>
   <type>eighth</type>
   <stem default-y="6.40">up</stem>
   <staff>1</staff>
   <beam number="1">continue</beam>
 </note>
```

Figure 8. The note code in MusicXML.

3.3 Third step

We use the online tool provided on the Music4VIP website which converts the MusicXML score into BMML. It is a very simple on line tool which is accessible for blind people.


Music 4 vip
Music FOR Visually Impaired People
Lifelong Learning Programme
Login
HOME
PROJECT
RESOURCES
NOTATION
TOOLS
FORUM
CONTACT US
English Français Italiano Polski

Music XML to BMML conversion

Now you can convert a Music XML file into a bmml file. This means that now your Music XML file can be opened using BME2 or BMR for Braille reading.

How do I...

1. Press "browse" button.
2. Select the Music XML file you want to convert.
3. Press "upload" button.
4. The programme will show the list of parts of your Music XML file.
5. At this point you must select one part at a time. You will then have a window where you must specify some characteristics of your part, such as: instrument, transposition, chord direction, clef.
6. The same operation must be repeated for each of the parts. You can also skipe step 5 and 6. In this case the programme will use default values. For example, chords will always be represented starting from the highest note.
7. Press "convert" button.
8. Wait until you see the message "download converted file". This may take some time if the file to convert is a big file.

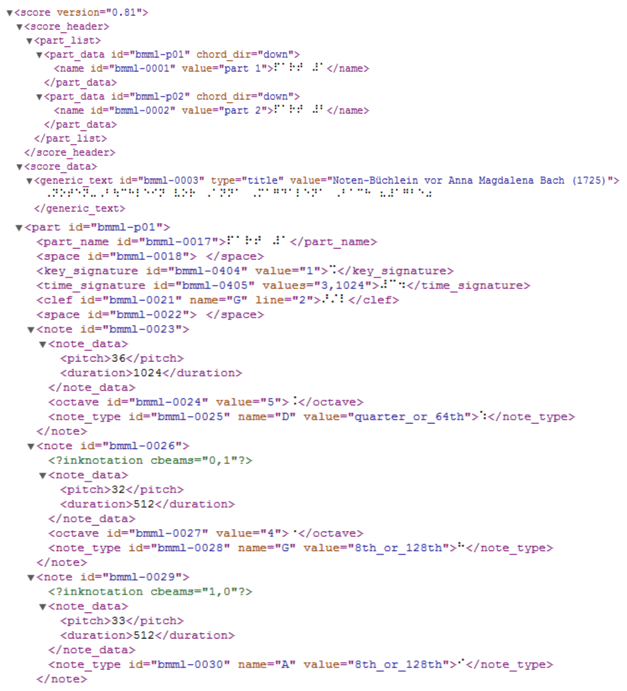
Upload Music XML file for conversion.

Music XML file: Choisissez un fichier IMSLP28....ano.xml

Upload

Figure 9 . MusicXML to BMML conversion tool.

The BMML file obtained is shown below.



```
▼<score version="0.81">
  ▼<score_header>
    ▼<part_list>
      ▼<part_data id="bmml-p01" chord_dir="down">
          <name id="bmml-0001" value="part 1">⠏⠁⠗⠞ ⠼⠁ </name>
        </part_data>
      ▼<part_data id="bmml-p02" chord_dir="down">
          <name id="bmml-0002" value="part 2">⠏⠁⠗⠞ ⠼⠁ </name>
        </part_data>
      </part_list>
    </score_header>
  ▼<score_data>
    ▼<generic_text id="bmml-0003" type="title" value="Noten-Büchlein vor Anna Magdalena Bach (1725)">
        [illegible]
      </generic_text>
  ▼<part id="bmml-p01">
     <part_name id="bmml-0017">⠏⠁⠗⠞ ⠼⠁ </part_name>
     <space id="bmml-0018"> </space>
     <key_signature id="bmml-0404" value="1">⠩</key_signature>
     <time_signature id="bmml-0405" values="3,1024">⠼⠉⠲</time_signature>
     <clef id="bmml-0021" name="G" line="2">⠜⠌⠇ </clef>
     <space id="bmml-0022"> </space>
    ▼<note id="bmml-0023">
      ▼<note_data>
          <pitch>36</pitch>
          <duration>1024</duration>
        </note_data>
        <octave id="bmml-0024" value="5">⠨</octave>
        <note_type id="bmml-0025" name="D" value="quarter_or_64th">⠱</note_type>
     </note>
    ▼<note id="bmml-0026">
        <?inknotation cbeams="0,1"?>
      ▼<note_data>
          <pitch>32</pitch>
          <duration>512</duration>
        </note_data>
        <octave id="bmml-0027" value="4">⠐</octave>
        <note_type id="bmml-0028" name="G" value="8th_or_128th">⠓</note_type>
     </note>
    ▼<note id="bmml-0029">
        <?inknotation cbeams="1,0"?>
      ▼<note_data>
          <pitch>33</pitch>
          <duration>512</duration>
        </note_data>
        <note_type id="bmml-0030" name="A" value="8th_or_128th">⠊</note_type>
     </note>
```

Figure 10. Example of BMML code.

The code indicates 2 parts in the head tag, a part for each hand for the keyboard. We can note container elements such as note_data and text elements such as note_type. We also see, for example, the arguments such as value=”4” for the octave and name=”D” for the name of the note.

Note the “inknotation” indication which refers to graphical aspects of the original musical notation.

3.4 Verification step and recommendation

To check the transcription, we can use either automatic tools or manual tools.

It is possible, for example, to create an automatic tool with the help of xquery to count the number of parts, bars or notes in order to establish easily whether any information has been lost.

Another automatic tool can compare the <step> tag in MusicXML with the attribute name in <note_type> in BMML to verify that they are the same.

To compare code we can also do the reverse transcription from BMML to MusicXML and compare the resulting graphic score with the original one. Some layout may be different – this is normal because the Braille code is not designed to store the graphical aspects of musical notation.

The following figures show the reverse transcription from BMML to MusicXML done with the online tool available on the Music4VIP site. Both in Melody Assistant and MuseScore there is a problem of text overlapping and we can also see that the stem direction of notes differs – all of which proves that the layout information is missing in the code.

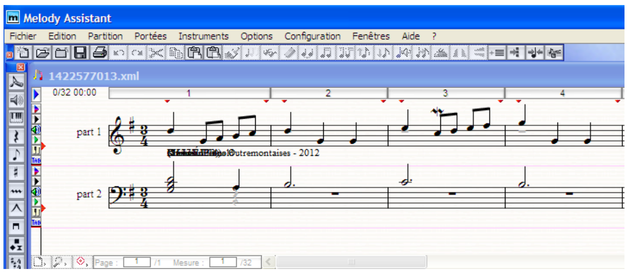


Figure 11. The MusicXML score in Melody assistant.

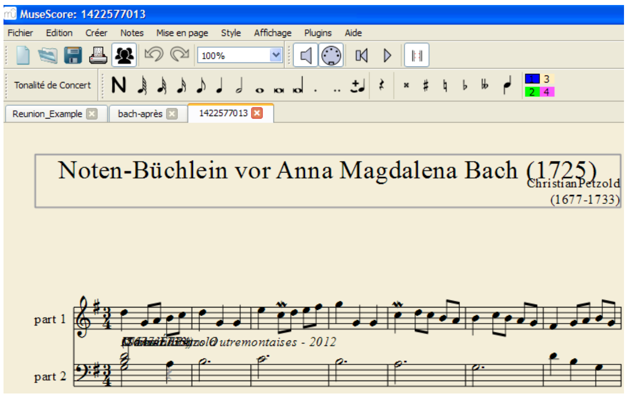


Figure 12. The MusicXML score in MuseScore.

If we download a MIDI File from http://www.free-scores.com/partitions_telecharger.php?partition=239

the graphic representation is not the same with either MuseScore or Melody Assistant.

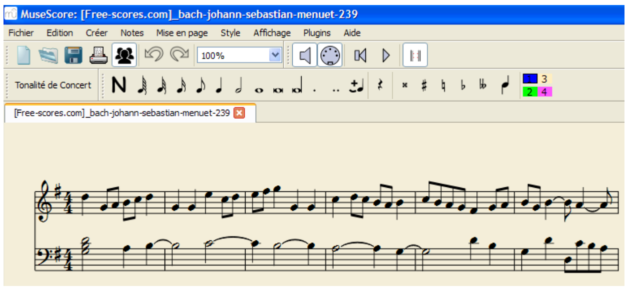

MuseScore: [Free-scores.com]_bach-johann-sebastian-menuet-239
Fichier Edition Créer Notes Mise en page Style Affichage Plugins Aide
100%
Tonalité de Concert
[Free-scores.com]_bach-johann-sebastian-menuet-239

Figure 13. The MIDI score in MuseScore.

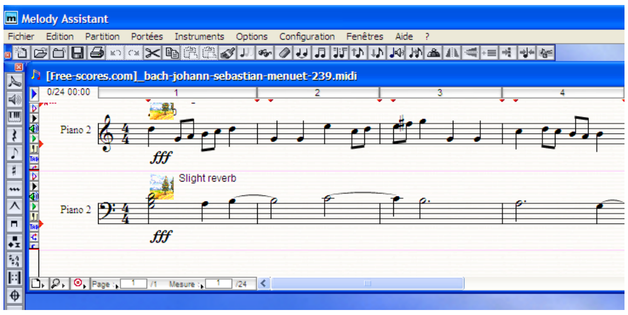

Melody Assistant
Fichier Edition Partition Portées Instruments Options Configuration Fenêtres Aide ?
[Free-scores.com]_bach-johann-sebastian-menuet-239.midi
0/24 00:00
Piano 2
fff
Slight reverb
Piano 2
fff
Page 1 /1 Mesure 1 /24

Figure 14.The MIDI score in Melody Assistant

The key signature is not the same in the two notation applications because the key signature is not explicit in MIDI file so the software has to interpret the information.

The time signature is the same in both applications but is not the same as the pdf file. This is an important problem because it implies a different meaning of the music. If we convert this file into BMML the musical information is very different from that obtained with the conversion of a pdf file. MIDI files have to be used with great care because they do not contain important information like fingering, slurs or ties.

In general, it will be of benefit to download digital scores from a reliable site like a library site. Having obtained a BMML file from whichever source a blind person can manage the score with a Braille reader or editor. We describe this process in the following section.

4. THE READER OR EDITOR USED BY BLIND PEOPLE

BMR is free software which permits blind users to read, learn and listen to music in a multimodal environment. Each piece of musical information can be accessed in Braille on a refreshable Braille display or by sound via MIDI or in a spoken form.

For a beginner, different kinds of Braille music elements may temporarily be hidden or a brief description of an unknown sign can be given.

With BMR the user can browse the score, add annotations, find parts and bars and skip through the score along hierarchical elements. He can, like a sighted person, have access to all the information contained in the score.

In the status bar of BMR we can read the musical information which corresponds to the Braille character which is after the cursor.

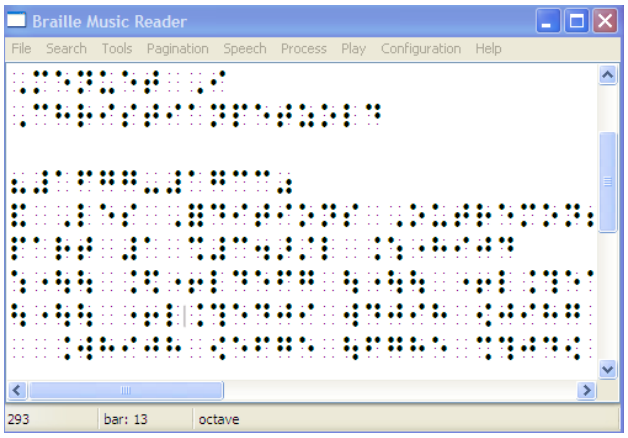

Braille Music Reader
File Search Tools Pagination Speech Process Play Configuration Help
293 bar: 13 octave

Figure 15.The Braille score in BMR

With BME2 [7] the same functionalities are available but, in addition, the user can write musical information in Braille. So users can create their own scores and produce BMML files. With the conversion module they can create MusicXML files and share them with sighted musicians. Of course, the result in graphic form will not be so well laid out as it would be if it had been originally produced in a conventional music editing application but the score will be immediately readable by a sighted musician and the minor formatting issues can be tidied up in a few minutes. This is enormously valuable for collaboration between sighted and blind musicians, whether they be teachers, students or members of a musical ensemble.

The way a blind person may access and make music without external aid is explained and demonstrated in the video found at : `http://www.music4vip.org/video_lesson_item/7`.

5. CONCLUSION

This paper describes how a blind user can access, convert, read and write musical scores. The conversion modules plus reading and editing tools are free, accessible and based on the BMML code. To obtain an available score in Braille it is necessary to convert a score into MusicXML format produced by an official editor or library. To facilitate the collaboration between sighted and blind musicians a reader with musical notation and Braille windows will be designed.

Acknowledgments

Part of this work has been undertaken in the European project Music4VIP. I wish to thank all the partners in the project.

6. REFERENCES

[1] D. Goto,T. Gotoh, R. Minamikawa-Tachino and N. Tamura, “A Transcription System from MusicXML Format to Braille Music Notation”, *EURASIP Journal on Advances in Signal Processing*, Volume 2007, Article ID 42498.

[2] `http://www.dancingdots.com/main/goodfeel.htm` (consulted 01/15/2015)

[3] B. Encelle, N. Jessel, J. Mothe, B. Ralalason and J. Asensio, “BMML : Braille Music Markup Language”, in *The Open Information Systems Journal*, 2009, pp. 123-35.

[4] `http://www.music4vip.org/braille_music_reader` (consulted 01/15/2015)

[5] Bettye Krolick, *New International Manual of Braille Musical Notation*, World Blind Union, 1996

[6] Michael Good, January 13, 2015, MusicXML Definition version 2.0, `http://www.musicxml.com/for-developers/musicxml-xslt/musicxml-2-0-to-1-1/`

[7] `http://www.veia.it/en/bme2_product2015`

NON-OVERLAPPING, TIME-COHERENT VISUALISATION OF ACTION COMMANDS IN THE ASCOGRAPH INTERACTIVE MUSIC USER INTERFACE

Grigore Burloiu
University Politehnica of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
gburloiu@gmail.com

Arshia Cont
MuTant Team-Project
IRCAM STMS UMR, CNRS, INRIA, UPMC
cont@ircam.fr

ABSTRACT

Integrated authoring and performing of mixed music scores, where musicians interact dynamically with computer-controlled electronics, is enabled by the *Antescofo* state-of-the-art software package. Composers are able to plan computerised actions through a dedicated programming language, and performances are then synchronised in real time. *AscoGraph* is the dedicated graphical interface that allows users to configure *Antescofo* behaviours and visualise their layout over a mixed music score. This paper presents developments in the direction of increased clarity and coherence of *AscoGraph*'s visualisation of computerised action scores. Algorithms for efficient automatic stacking of time-overlapping action blocks are presented, as well as a simplified model for displaying atomic actions. The paper presents the improvements in score readability achieved, as well as the challenges faced towards a complete representation of dynamic mixed scores in the *AscoGraph* visual environment.

1. INTRODUCTION

This paper describes a model of interactive visualisation for composition and performance of mixed music repertoire. Mixed music is commonly referred to as the live association of human musicians with interactive software/hardware during live music performance; as well as the authoring (composition) within this mixed medium. Among common practices of mixed music, *score following* has been an active line of research where the computer is equipped with a real-time machine listener that dynamically aligns a musician's performance to a pre-written music score and decodes performance parameters, which can be used to interpret and evaluate computerised actions. One can think of such a compositional paradigm as an extension of musical automatic accompaniment application, where accompaniment playback is replaced by programs acting on various aspects of sound and music computing.

Visualisation for mixed music is a challenging task for several reasons. The score is a joint combination of two main components: one that describes expected events from human musicians as extensions of classical musical notation; and one that describes computerised, or electronic actions. The two components are strongly-timed and most often aligned during authoring and synched dynamically during performance. Computerised actions in turn have heterogeneous time models: they can be discrete message passing, or continuous curves, or dynamic calculations. Their temporal ordering can be described as sequences of delays expressed in absolute or relative time; actions can be hooked sequentially (through delays inside a single sequence) or vertically (to an external event, condition or synchronisation pivot). Composers and performers are proficient at dealing with authoring and interpreting a variety of such parameters.

Enabling this level of expressivity in mixed music composition and performance is the goal of the *Antescofo* software, a state-of-the-art system for mixed music composition which integrates a score following engine [1] and a synchronous reactive programming language [2]. The user/composer is able to plan complex dynamic electronic actions which the system launches and controls during the performance, in sync with the live musicians' tempo. First described in [3], *AscoGraph* is the dedicated graphic development environment which enables visual feedback for authoring and performing *Antescofo* mixed scores.

The *AscoGraph* workspace consists of two main sections: the textual score editor and the graphical editor, which in turn is split into an instrumental section and an electronic

This project was partially funded by the French ANR INEDIT Project.

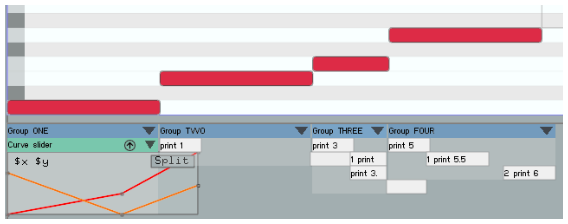


Figure 1. The "classic" action block display in *AscoGraph*. The musical timespans of Groups ONE, TWO and THREE are not represented clearly because of the overlapping of group blocks.

actions section. The instrumental view and electronic view are coupled in musical time along a common horizontal timeline. During a performance, *Antescofo*'s score follower determines the position of *AscoGraph*'s graphical cursor along the timeline.

This paper presents updates to *AscoGraph*'s electronic action view, developed with two directions in mind: (1) a clearer and more time-coherent visualization of *Antescofo* scores, and (2) a step towards a complete, self-contained visual notation format for mixed music scores. Section 2 presents the problem of overlapping action blocks, recast as a subset of the two-dimensional strip packing problem. The following section shows the three proposed algorithms for re-arranging action blocks. Section 4 tackles the issue of coherence between block width and musical time. We conclude the paper with an evaluation of the present model and future perspectives.

2. PROBLEM DEFINITION

We distinguish between physical time (measured in seconds) and musical time (measured in beats). The amount of physical time elapsed between actions depends on the tempo detected during performance, and on the active synchronisation strategies [2]. Meanwhile, *Antescofo* scores are specified in musical time. Since *AscoGraph* was primarily designed as a score visualisation tool, it employs a musical timeline. When a physical time unit is specified in a score (e.g. "after 2 seconds"), in order to display it *AscoGraph* must first translate it to an ideal musical time (e.g. "after 4 beats at 120bpm").

Fig. 1 shows an example of the original *AscoGraph* action block arrangement style. Here, each of the four notes (drawn in red in the instrumental view's piano roll) has one corresponding action group block. Durations can – and often do – differ between the length of a note and that of its associated electronic actions. While actions within a single group (e.g. Group FOUR) are stacked consecutively downwards, when two different action groups are partially concurrent, the second group is drawn over of the first. Consequently, the first group's duration is no longer clearly shown; things become even more confusing when overlapping automation curves (e.g. the one in Group ONE) are involved.

In order to rectify this loss of coherence and clarity, the need arises to stack action groups in downward non-overlapping order, similarly to how elements *within* groups are arranged. As the challenge becomes one of efficient management of 2D space, it is useful to describe it as a two-dimensional *strip packing* problem. A subset of *bin packing*, strip packing is used in areas ranging from optimizing cloth fabric usage to multiprocessor scheduling [4]. Algorithms seek to arrange a set of rectangles within a 2D space of fixed width and bottom base, and of infinite height. In our present case, the width of the strip corresponds to the total duration of the piece, and the rectangles to be placed are the action group blocks.

A particular constraint separates our problem from the rest of the bin packing literature. Unlike in existing bin packing problems, all *AscoGraph* action blocks must retain their *X* coordinate along the time axis. Since we are not allowed to "nudge" blocks horizontally, relying on existing packing algorithms becomes impractical.

3. PACKING ALGORITHMS

We introduce three new algorithms for stacked action group display in *AscoGraph*'s graphical editor. The user can switch between one of them and the original display style through the application's *View* menu. The appropriate option will depend on score complexity and the user's personal taste.

Please note: following bin packing convention, we shall consider the rectangles as being placed *on top of* the strip base. Naturally, in the *AscoGraph* environment the situation is mirrored and we build *downwards* starting from the upper border.

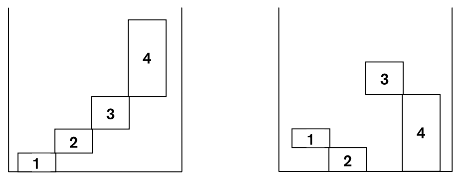
3.1 First Fit (FF)

The first option is the trivial solution of placing the blocks in the first space they will fit, starting from the base. The benefits of this option are speed and predictability: blocks are placed in the order in which they appear in the source code text, which is also their scheduled temporal order.

The downside can be intuited from Fig. 2a and b. We propose a worst-case scenario: a set of blocks with increasing heights and, for simplicity, all equal widths. While FF would stack them on top of each other (Fig. 2a), the optimal method would stack them two by two (Fig. 2b), so that the maximum height is given just by the final two elements.

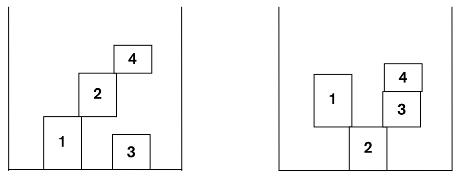
3.2 First Fit Decreasing (FFD)

Note that in the previous case, the optimal configuration can be reached by simply reordering the blocks by height.



(a) FF **(b)** OPT (FFD)

Figure 2. Horizontally constrained strip packing. Boxes are numbered in temporal (horizontal) order. This layout is arranged by FF in (a) and optimised by FFD in (b).



(a) FFD **(b)** OPT (FFDT)

Figure 3. Horizontally constrained strip packing. Boxes are numbered in temporal (horizontal) order. This layout is arranged by FFD in (a) and optimised by FFDT in (b).

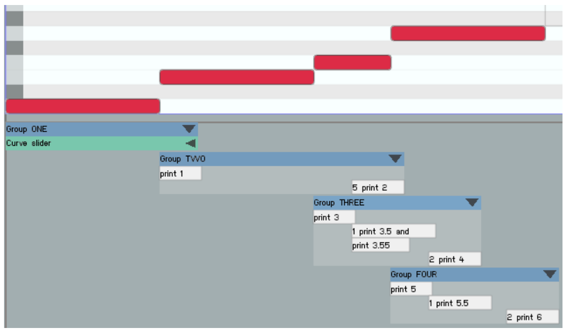
This insight lies at the root of the classic FFDH strip packing algorithm [5]. In our case, the FFD algorithm orders the blocks by non-increasing height, after which the First Fit process is applied. [1] Fig. 3a shows an FFD arrangement, along with the optimal solution at Fig. 3b.

3.3 First Fit Decreasing Towers (FFDT)

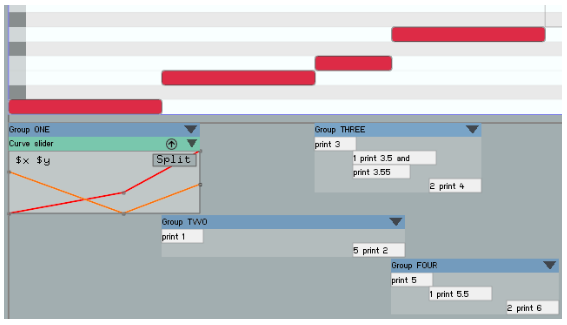
Again, the optimal configuration in the previous example points towards the next algorithm. We propose a greedy heuristic that builds upon FFD while tackling *AscoGraph*-specific situations like one action block sharing time with several blocks on both sides of it. The basic goal is to minimise gaps, such as the one between blocks 2 and 3 in Fig. 3a.

The FFDT algorithm first orders all blocks as in FFD. Then, action group *towers* are defined at the time-axis intersections between two or more group blocks. Their height is equal to the sum of the heights of their component blocks. For instance, in Fig. 3a and d the rectangle 2 is part of four towers: $\mathsf{T}_a\{r_1, r_2\}, \mathsf{T}_b\{r_2\}$ [2] $, \mathsf{T}_c\{r_2, r_3\}$ and $\mathsf{T}_d\{r_2, r_3, r_4\}$.

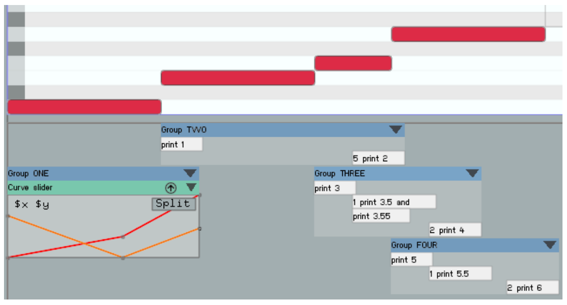
The entire width being now split along these virtual vertical strips (towers), we are able to refine the ordering of the blocks. The first criterion is the decreasing *maximum height* among the *towers* each block is a member of. If this maximum tower height is equal for two blocks, then the second criterion is decreasing *number of towers* each



(a) FF - action blocks are placed as close to the baseline as possible, in the order in which they appear in the code.



(b) FFD - blocks are ordered by height before being placed.



(c) FFDT - blocks are ordered according to a gap-minimisation heuristic before being placed.

Figure 4. The three *AscoGraph* packing options.

block is a member of. If this number is equal as well, we leave the FFD ordering (non-increasing block height) untouched.

By definition, the maximum tower height is a definite lower bound of any *AscoGraph* strip packing configuration. Therefore, in the FFDT heuristic the tallest tower will always be placed first, in an attempt not to overshoot this lower bound. Among its component blocks, the ones that are shared with many other towers are dropped closer to the base - the intention again being to maintain tower integrity as much as possible. Lastly, as with FFD, tall blocks are prioritised so as to fill gaps efficiently.

An *AscoGraph* use-case comparison of the three algorithms can be seen in Fig. 4. In it, we illustrate the orderliness of FF and the compactness of FFDT, with FFD a potentially useful compromise between the two.

[1] The difference to the classic FFDH algorithm is the absence of horizontal levels. New blocks are stacked at the minimum possible altitude rather than a common level.

[2] A minimal tower only contains one action block.

4. TIME COHERENCE OF ATOMIC ACTIONS

A basic element in *Antescofo*'s reactive language is the atomic action. Atomic actions can be part of larger dynamic constructs, but often they are simple messages to be triggered at a specific point in time. Since they are instantaneous, their visual representation taking up horizontal space on the action timeline is discordant. Moreover, as figures 1 and 4 show, they often clutter the workspace unnecessarily.

Our solution to more accurately represent action messages is to group all instances from a specific hierarchical level and display them on a single line as small circles, or conceptual *points*. When the mouse hovers over such a point, a list of the messages it contains is shown. Fig. 5 shows the expanded list for Group THREE; the messages are set at 3 different points in time, which is why 3 points are present in the message line.

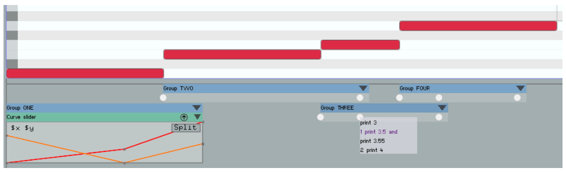
Our new model is fully time coherent and considerably clearer than before. The user experience improvement over the classic model becomes most obvious when dealing with complex scores with many messages - see Fig. 6. With the timeline fully zoomed out, the old model offers a less accurate overview of the activity in the electronic score. Action durations are impossible to estimate; the most egregious problem being at the final note of the score, where, with nothing to stop them, musically instantaneous message actions take up an inordinate amount of space in the timeline. Meanwhile, the new model neatly groups messages together and offers a clear view of action block distribution in time and individual durations.

5. CONCLUSIONS AND FUTURE WORK

We have shown an improved layout mechanism for electronic action groups over a musical timeline in *AscoGraph*. By stacking action group blocks we ensure information integrity and coherence, while expanding the vertical real estate used. The most basic stacking method, First Fit, is also the most easily readable option for scores of moderate depth. We also proposed two increasingly efficient stacking algorithms, FFD and FFDT, for scores containing larger concentrations of actions per time unit. While superior algorithms are technically conceivable (possibly a metaheuristic scheme built on top of FFDT), the present options were deemed appropriate for the practical use and the processing overhead of the *AscoGraph* software.

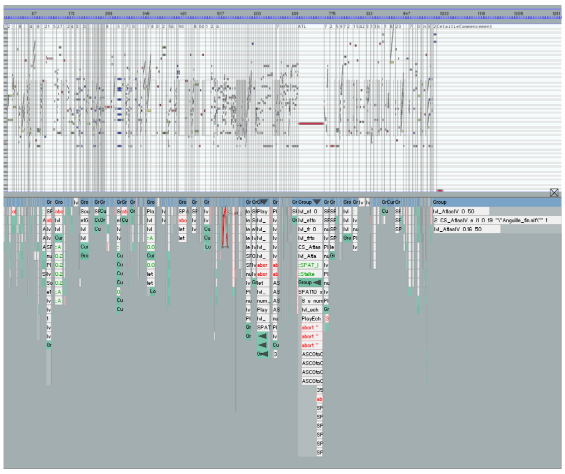
Finally, we have introduced a method of displaying related messages on a single line which preserves group hierarchy. The main advantages are time coherence and vertical compactness. Still, this model can be seen as a compromise in our quest for a completely specified, self-contained visual notation format which we proposed in the introduction. Dynamic constructs from the *Antescofo* language are in a similar situation. For instance, a Curve whose duration is a dynamic variable: in this case, *AscoGraph* cannot know its exact plot over time before execution.

Therefore, one direction of future research is a *performance simulation mode*, decoupled from the compositional

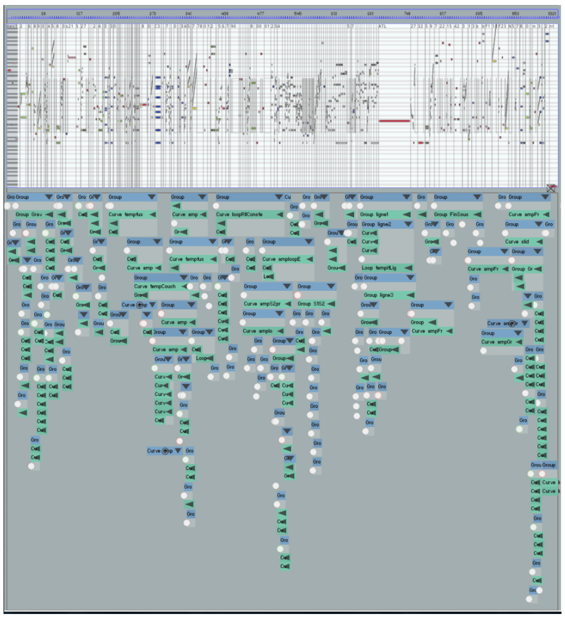


print 3
1 print 3.5 and
print 3.55
2 print 4

Figure 5. Time-coherent message circles display



(a) old model: blocks overlap, messages occupy horizontal space



(b) new model: blocks are stacked, messages are grouped in time-coherent points

Figure 6. Comparison of old and new *AscoGraph* models over a complex score.

display described thus far, in which all messages, Loops and other dynamic constructs are represented as they "happen" in an offline simulation. This function is currently in prototype form, having been first described in [3].

However, the need remains for a graphic compositional model that clearly describes dynamic behaviour and action results. With the growing crystallisation of *Antescofo*'s language into a mature, stable package, the path is now open for research in this direction.

6. REFERENCES

[1] A. Cont, "Antescofo: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." in *International Computer Music Conference (ICMC)*, Belfast, Ireland, Aug. 2008, pp. 33–40. [Online]. Available: http://hal.inria.fr/hal-00694803

[2] J. Echeveste, A. Cont, J.-L. Giavitto, and F. Jacquemard, "Operational semantics of a domain specific language for real time musician-computer interaction," *Discrete Event Dynamic Systems*, vol. 23, no. 4, pp. 343–383, Aug. 2013. [Online]. Available: http://hal.inria.fr/hal-00854719

[3] T. Coffy, J.-L. Giavitto, and A. Cont, "AscoGraph: A User Interface for Sequencing and Score Following for Interactive Music," in *ICMC 2014 - 40th International Computer Music Conference*, Athens, Greece, Sep. 2014. [Online]. Available: https://hal.inria.fr/hal-01024865

[4] R. Thöle, "Approximation algorithms for packing and scheduling problems," Ph.D. dissertation, Christian-Albrechts-Universität zu Kiel, 2008.

[5] J. Coffman, E. G., M. R. Garey, D. S. Johnson, and R. E. Tarjan, "Performance bounds for level-oriented two-dimensional packing algorithms," *SIAM J. Comput.*, no. 9, pp. 808–826, 1980.

[6] J.-L. Giavitto, A. Cont, and J. Echeveste. Antescofo a not-so-short introduction to version 0. x. [Online]. Available: http://support.ircam.fr/docs/Antescofo/AntescofoReference.pdf

DYNAMIC NOTATION – A SOLUTION TO THE CONUNDRUM OF NON-STANDARD MUSIC PRACTICE

Georg Hajdu
Center for Microtonal Music and Multimedia
Hamburg University of Music and Theater
georg.hajdu@hfmt-hamburg.de

ABSTRACT

This paper discusses *dynamic notation*—a method allowing, in a notation environment, instant switching between different staff views or *notation styles*, thus creating a common ground for practitioners of non-standard music, such as composers, performers, conductors and scholars. So far very few notation programs have explored this notion as much as it should have been. Therefore, we have implemented in the MaxScore Editor (a notation editor designed to run in Max or Ableton Live) a plugin structure for different notation styles based on a set of maps and queries executed during note entry and rendering—affecting music glyph choice and placement. We will give an in-depth analysis of the methods used for equidistant scales, non-octave tunings, music in just intonation as well as for instrument-specific layouts and will conclude with a description of a scenario in which dynamic notation was used for the transcription and performance of Alexander Scriabin's piano poem *Vers la Flamme* op. 72 by an ensemble of acoustic Bohlen-Pierce instruments.

1. INTRODUCTION

In his 2001 book *The Language of New Media* [1] media theorist Lev Manovich points out that *new media objects* need to fulfill certain criteria among which *variability* is related to *dynamic* delivery of content. New media objects can exist in multiple versions such as, in case of an audio recording, a high-definition 192kHz 64-bit file, a standard CD-quality file, a lossless ALAC or a lossy compressed mp3 file.

Variability is key to solving the conundrum practitioners of non-standard music are facing when performing such music. The prerequisite is that the music is created and/or delivered by a computer-based system capable of switching between different views or representations in real time[1].

Rudimentary dynamic notation is common amongst notation environments. Most typically, a program will let the users switch between regular and percussion notation, or piano roll view. Bach [3], PWGL [4], OpenMusic [5] and InScore [6] as well as the lesser known Siren [7] and CMN [8] represent music in various measured and non-measured ways but lack a simple plugin structure for adding new views dynamically, hence requiring a higher degree of meddling with the code to achieve results comparable to the MaxScore Editor.

2. NON-STANDARD MUSIC PRACTICE

The practice of music in non-standard tunings has been hampered by, besides the lack of appropriate instruments, a "confusion of tongues" in respect to how this kind of music is supposed to be notated. Notation should cater to the needs of the people involved: A player performs best if the notation is close to the layout of the instrument played. For that matter, guitarists have been using tablature for centuries and Harry Partch's notations for pitched percussion, for instance, are most often concerned with the topology of the instrument rather than actual pitch [9]. The notations for the Bohlen-Pierce clarinets which exist in two sizes (soprano and tenor) as well as the modified Bohlen-Pierce alto recorder also use fingering notation, taking advantage of the learned correspondence between finger position and sounding pitch on a traditional instrument. We can thus call this approach *instrumental notation*.

A composer or arranger works best when using a notation that represents the logic of a particular tuning. Sabat and von Schweinitz, for instance, have developed an elegant solution of representing frequency ratios in *just intonation* by designing a large set of accidentals [10]. Equidistant tunings in turn benefit from representing equal pitch distances as such. The diatonic Guidonian notation already poses difficulties when it comes to coherently representing whole-tone or atonal/12-tone music,

[1] We need to point out that our use of the term *view* differs from Dannenberg [2] who refers to "a data-structure that corresponds to a presentation."

but fails bitterly at non-octave tunings such as the Bohlen-Pierce scale. Equidistant notations, such as the Hauer-Steffens notation [11] have the advantage that transpositions and transformations of tone gestalts become evident, but have been rejected in history because of cultural and economic implications, and most likely also because of the cognitive mismatch between notation and the piano layout with its black and white keys. There is no reason, though, to shy away from introducing equidistant notation for tunings other than 12EDO (and its related, circle-of-fifths-based tunings). We dub this approach *logical notation*.

A conductor, finally, has different concerns as he/she needs to grasp the meaning of the different notation styles used in rehearsals and performances. A conductor needs to hear, identify and compare the sounding events to the score and to an internalized template—a feat facilitated by years of intensive training and practice. He/she may be best served by the representation of music in traditional Guidonian notation, enriched by an extended set of accidentals or indications of deviations written above the notes. This may also be the notation of choice for instruments such as standard string or wind instruments. We will name this approach *conventional* or *cognitive* notation as it depends on internalized templates.

As we are attempting to establish a taxonomy for notational approaches, we also need to concede that these distinctions are arbitrary to a certain extent. *Instrumental* and *conventional* notations have a common root originating from the *logic* of the music in practice when its notation was standardized.

2.1 Scenarios

One can conceive of the following scenarios in which dynamic notation may be welcome:

All musicians are reading from computer/tablet screens of either isolated devices or machines in a networked arrangement. Alternatively, only the person guiding the rehearsals/performance uses an electronic device while the other members of the ensemble read from paper-based print-outs. In the latter case, the responsibility lies in him/her to guide the communication on notational aspects. Finally, both scores and parts are paper-based, but they contain, on different staves, alternative representations of the music to be performed. Even in this case, a system capable of changing views in real-time can vastly simplify the process of creating scores and parts.

3. IMPLEMENTATION

A plugin structure for dynamic switching between notation styles has been implemented for the MaxScore Editor. MaxScore is a Max Java object designed and maintained by Nick Didkovsky since 2007 to bring music notation to the Max environment [12].

Since 2010 the author is developing an editor, which also interfaces with Ableton Live via Max for Live. As opposed to the Bach notation objects, which—being native Max externals—provide better integration in the Max environment, the MaxScore Editor is based on a hybrid approach consisting of the core mxj object and a periphery made of numerous C and Java externals, Max abstractions and JavaScript objects (forming the editor's GUI, among other functions; see **Figure 1**). The advantage of a hybrid system is its high degree of adaptability and versatility. As the communication between the core and the periphery is based on messages, they can easily be intercepted and reinterpreted according to current demand.

The editor handles notation styles like plugins and loads them as Max patches dynamically from a folder in MaxScore package hierarchy. It is thus very straightforward to add new styles to the existing repertoire.

Every notation style defines 5 attributes:

- A name of the notation style appearing in the style menu of the Staff Manager.
- The name of the Max patch containing pitch maps,
- The number of staff lines employed by the style.
- The micromap to be used for the rendering of accidentals.
- The name of the clef appearing on the staff.
- If a non-standard clef is being specified such as Bohlen-Pierce T-clef, an additional definition needs to be given in the form of clef name, glyph, x and y offsets as well as font name and size.

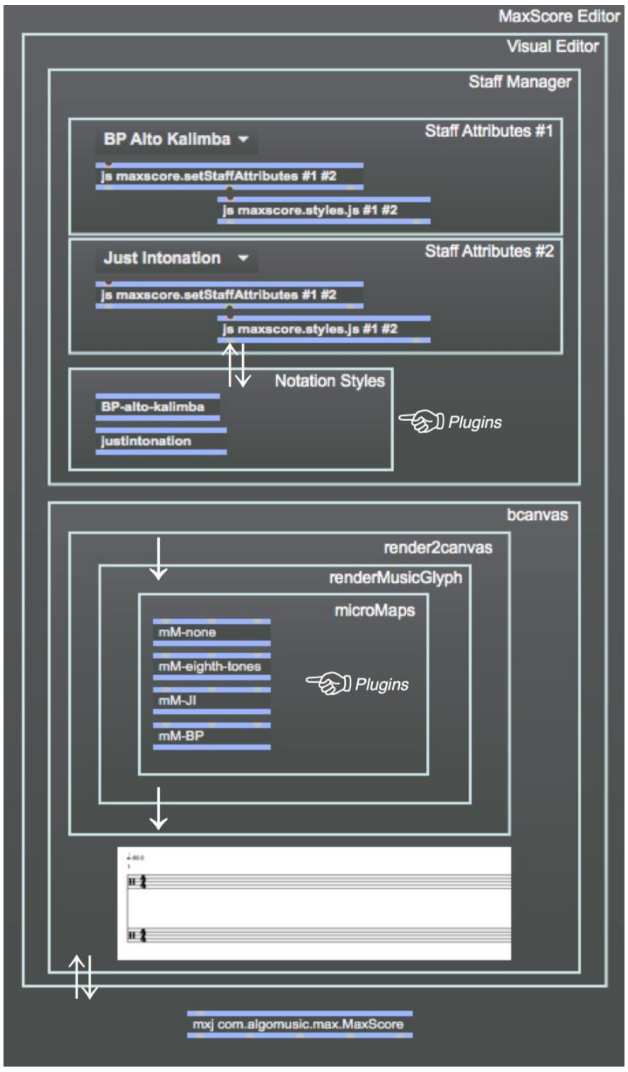


Figure 1. The nested structure of the MaxScore Editor plugin system.

3.1 Micromaps

Micromaps were introduced in 2011 to allow higher resolution representations of divisions of the semitone. While MaxScore's pitch attribute is stored, processed and played back in 32-bit floating-point precision, the drawing messages generated by the object are limited to quarter tones. Hence, MaxScore would fail to accurately represent music in eighth tones (the standard among spectral composers) or twelfth tones (used by composers such as Ivan Wyschnegradsky and Ezra Sims). Micromaps are Max abstractions that intercept drawing messages and query the pitch and accidental preferences attribute of the corresponding note. Based on this information and the notation style chosen by the user, a micromap sends new, more fine-grained drawing messages to the MaxScore canvas. Currently, the maximum precision is sixteenth tones in Sagittal notation, taking advantage of the enormous set of accidentals in the Bravura font, just recently released by Steinberg [13].

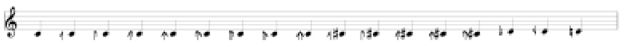


Figure 2. 16th-tone notation with the Sagittal font.

How is this mapping performed? This is best explained by an example: After entering a middle c a sixteenth note sharp (pitch = 60.125) the MaxScore object sends out a drawing message with 9 items (accidental, x, y, zoom, parent object, measure, staff, track and note indexes) such as:

“no_accidental 75.555557 81. 0.5 Note 0. 0. 0. 0.”[2]

The five last items are sliced off and a “getNoteInfo 0. 0. 0. 0.” query is sent to the core object. It returns a string in XML format which is being parsed and sent back to the micromap. Of the many note attributes, *pitch* and *accidental* information is being retained to calculate the pitch zone a particular accidental is applied to.

The zone index Z is given by this formula (*rnpc* = floating point remainder of natural pitch class, *n* = division of the semitone, sgn = -1 for ACCPREF = flat and sgn = 1 for ACCPREF = sharp):

$$Z = [\mathrm{sgn}\ \ rnpc\ \ n + \frac{1}{n+2}] \qquad (1)$$

In our case, the result would be

$$[1 \cdot 0.125 \cdot 8 + 0.0625] = 1$$

This value is fed into a zone-to-glyph-name lookup table (a Max coll object), which sends out accSagittal5CommaUp, the name of the Sagittal accidental in the Standard Music Font Layout specification on which the Bravura font is based [13].

This message is combined with the rest of the message into

“accSagittal5CommaUp 75.555557 81. 0.5 Note 0. 0. 0. 0.”

of which the first four items are further processed:

The zoom value scales the font size as well as x and y offsets. The accidental name is sent to another instance of a Max coll object which returns

“↿ -4 0 Bravura 24” (glyph, x offset, y offset, font name, font size).

[2] no_accidental messages will be ignored by the drawing engine, but are a prerequisite for mapping.

This information is then translated into three separate Max lcd messages:

1. “font Bravura 24.”
2. “moveto 71.555557 82.” (due to various reasons a y offset of 1 is applied to all glyphs)
3. “write ↿”

3.2 Notation Styles

For the representation of music in the Extended Helmholtz-Ellis JI Pitch Notation created by Sabat and von Schweinitz we had to go a step further. A problem arises when the MaxScore object is no longer capable of representing the correct *rnpc* in case of complex harmonic relationships such as 75/49 ($3^1 \cdot 5^2 \cdot 7^{-2}$). According to the logic of the Helmholtz-Ellis notation the interval size of 736.9 cents—when applied to a middle c—would have to be represented by an f with two accidentals, a double-sharp with two arrows down and a raise by two septimal commas (see **Figure 4**) The diatonic pitch class is calculated by moving along the circle of fifths, which in our case would be moving 13 ticks in clock-wise direction, thus amounting to an f double-sharp. This conflicts with the pitch class natively assigned by MaxScore which is g[3]. The solution was found by creating a specific Just Intonation *notation style*. Kuuskankare and Laurson [14] use a similar term (*notational style)* denoting changes in notation that include not only pitch but other aspects such as measured/non-measured notation.

A notation style is basically defined by two maps (map and inverse map) but also requires the definition of an additional attribute. This can be easily achieved in MaxScore where an unlimited number of dimensions can be added to notes and intervals first by applying the “setInstrumentDimension” message to a particular staff and then setting note dimensions values individually. Considering this, we have defined an additional *originalPitch* dimension which holds the pitch of a note regardless of how it is represented graphically in the score and is also used for playback.

After choosing a notation style from a menu in the Staff Manager, all notes for a given staff are passed to the inverse map of the abstraction of the *current* notation style (*default* for new scores) to restore the *originalPitch* attribute. Then all events are routed to the map of the abstraction of the *selected* notation style[4]. Here the *pitch* attribute is set to the position of the note it is supposed to occupy in the given notation style.

[3] MaxScore stops considering enharmonic spellings for ranges outside of double flats and sharps (maximum considered deviation = 225 cents; in case of 75/49 the deviation is 236.9 cents).
[4] NB.: Instead of current and selected, we could also use the terms old and new.

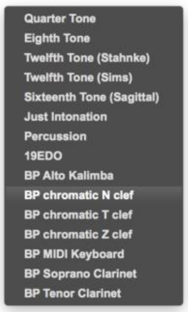


Figure 3. The style menu in the MaxScore Editor Staff Manager. The equal divisions of the semitone on top don’t require additional pitch mapping and are part of the *default* notation style.

Data flow in and out a plugin is controlled by a JavaScript object—one instance per staff. The object also receives messages when a note is being created, in which case, *originalPitch* is calculated by sending its *pitch* through the *inverse map*. For instance, when a note is created in T clef below the bottom staff line (pitch = 59), *originalPitch* will be immediately set to 49.98. Likewise, the *pitch* attribute will automatically be updated after a transposition, which, conveniently, can be done across different notation styles. While a *map*, generally, only receives *originalPitch* values from the JavaScript object, an *inverse map* receives a list of 9 current note, staff and measure attributes. The *inverse map* subsequently filters useful attributes, hence the dollar sign arguments in the notation style Max patches.

3.3 Examples

We will now more closely examine some of the notation styles implemented in the Max Score Editor.

3.3.1 Just Intonation

Definition: “"Just Intonation" justintonation 5 mM-JI default”

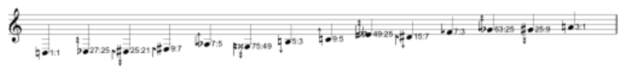


Figure 4. The Bohlen-Pierce scale in Extended Helmholtz-Ellis JI Pitch notation

As pointed out above, certain ratios lead to a situation where the actual pitch is more then 225 cents off its natural “anchor” tone. Having introduced *originalPitch* as an attribute for correct playback, the *pitch* attribute can now be “arbitrarily” moved to a tone above or below—thus displaying the correct pitch/accidental combination. The calculations involved are fairly complex and have been described in [12].

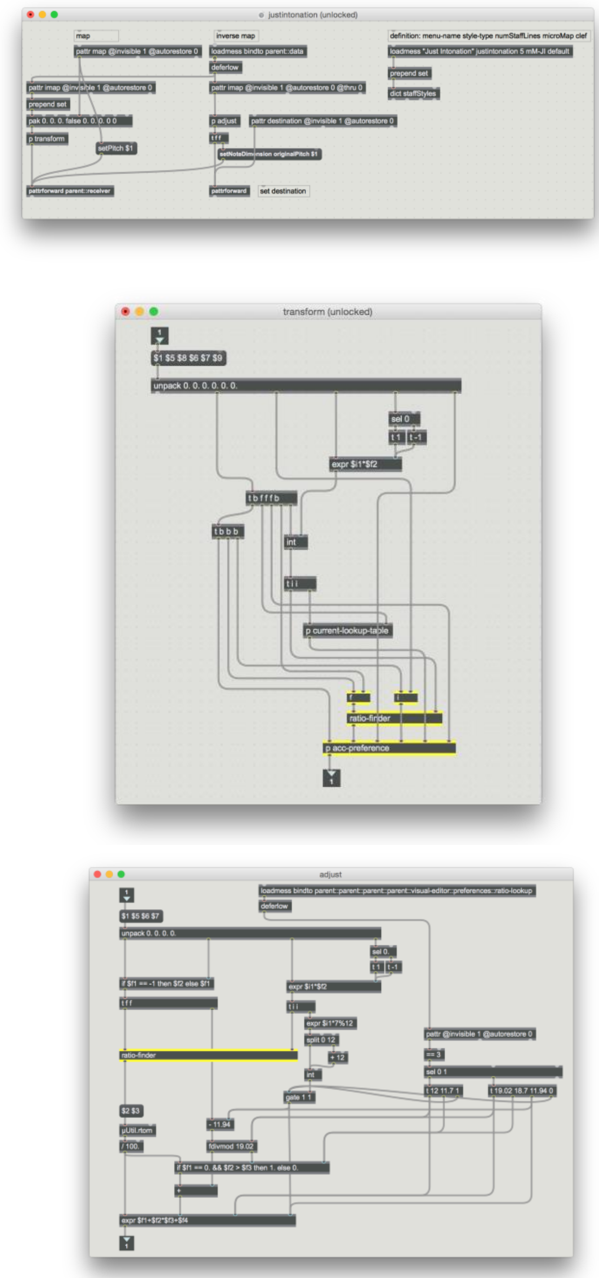


Figure 5. The just intonation notation style consists of nested Max patches illustrating the complexities of the calculations involved.

3.3.2 19EDO

Definition: "19EDO 19EDO 5 mM-none default"

Russian-American musicologist Joseph Yasser argued in his 1932 book *Theory of Evolving Tonality* that 19-tone music, in its just or equal tempered forms, constitutes the next logical step in the development of music [14]. While we can no longer subscribe to this claim, this tuning remains one of the popular ones, having been investigated by composers such as Easley Blackwood and Joel Mandelbaum. Its 19 tones form a closed circle of fifths and, thus, the scale possesses a diatonic subset and enharmonic alternatives for each black key, in addition to an e# and a b# between e and f and b and c, resp. The mapping is performed by:

1. Calculating the 19EDO scale step index with µUtil.PitchToStep abstraction, which is part of the author's µUtilities package bundled with MaxScore.
2. Extracting octave index and pitch class by dividing the index by 19 and passing the remainder through a lookup table yielding the 12EDO pitch class, accidental preference (sharp or flat) and enharmonic spelling for any of its 19 pitch classes.
3. Calculating pitch by multiplying octave index by 12 and adding the respective 12EDO pitch class and an offset.

E.g. for 7136 MIDI cents, the scale step index is 113. Divmod 19 yields 5 and 18. Feeding 18 into the coll returns "11 1 1", thus setting setAccPref to sharp and setAltEnharmonicSpelling to true. Pitch is 12 · 5 + 11 + 1 = 72, displayed as b#.

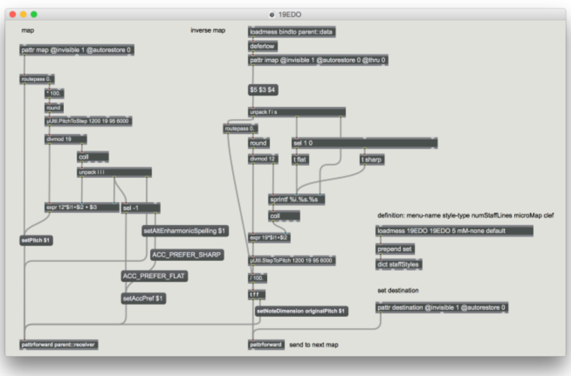


Figure 6. The 19EDO notation style takes *pitch*, *accidental* and *enharmonic spelling preference* into consideration.

3.3.3 Percussion

Definition: "Percussion percussion 5 mM-none percussion"

Most notation programs implement the percussion notation style in which MIDI notes (range 35 - 81) are mapped to the white keys between d4 and a5. Percussion notation uses certain positions redundantly, yet differentiates between classes of instruments by assigning various notehead shapes to the notes.

When switching **to** the percussion notation style the *originalPitch* attribute is sent to a Max coll (percussionMap) containing:

- The name of the instrument,
- Its (notated) pitch in percussion notation
- The corresponding notehead shape.

The map will now send three messages to the MaxScore core object: “setPitch value”, “noteheadTransform shape” and “setAltEnharmonicSpelling false”, the latter message to clear double sharps or flats should they have been set previously.

When switching **from** the percussion notation style the *pitch* and *notehead* attributes are evaluated and sent to another coll (inversePercussionMap) in order to clear notehead shapes and reconstruct the *originalPitch* attribute. The messages to MaxScore are “noteheadTransform NOTEHEAD_STANDARD” and “setNoteDimension originalPitch value”.

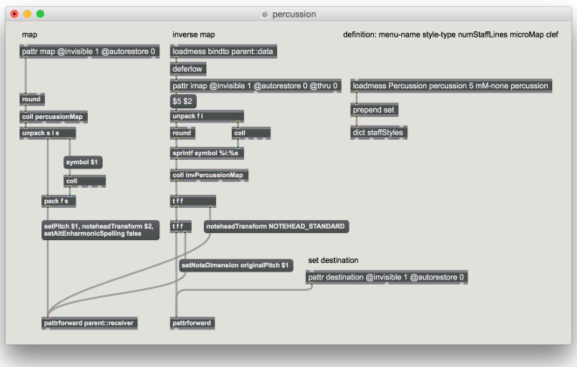

percussion
map
inverse map
definition: menu-name style-type numStaffLines microMap clef
pattr map @invisible 1 @autorestore 0
loadmess bindto parent::data
deferlow
pattr imap @invisible 1 @autorestore 0 @thru 0
loadmess Percussion percussion 5 mM-none percussion
$5 $2
round
prepend set
coll percussionMap
unpack f i
unpack s i s
round
coll
dict staffStyles
sprintf symbol %i:%s
symbol $1
coll invPercussionMap
coll
pack f s
t f f
setPitch $1, noteheadTransform $2, setAltEnharmonicSpelling false
t f f
noteheadTransform NOTEHEAD_STANDARD
setNoteDimension originalPitch $1
set destination
pattr destination @invisible 1 @autorestore 0
pattrforward parent::receiver
pattrforward

Figure 7. The percussion notation style consists of lookup table setting *pitch* and *notehead shape*.

3.3.4 Bohlen-Pierce T-Clef

Definition: “"BP chromatic T clef" BP-chromatic-T 6 mM-BP BP-T-clef”

Clef definition: “BP-T-clef T 2 -1 "Greifswaler Deutsche Schrift" 28”

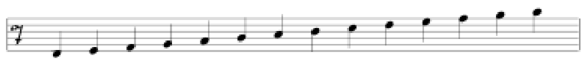


Figure 8. The Bohlen-Pierce scale in T clef maps the range of the tritave d3-a4 onto the six lines of Müller-Hajdu notation. Note that the lowest and highest notes look like d and a; this is a desirable, albeit coincidental trait.

We have described the design of a new notation system for the non-octave Bohlen-Pierce scale with 13 steps [16]. This scale, which was independently discovered by three people (Heinz Bohlen [17], John Pierce and Kees van Prooijen) in the 1970’s to 1980’s, is probably the most common and best-investigated scale of its kind. A number of acoustic instruments have been built since 2007 and a group devoted to the practice of this kind of music has been founded a year later in Northern Germany. It was shown by Pierce, Mathews et al. [18], Loui [18] and us [16] that this scale, substituting the octave (2:1) by a tritave (3:1), exhibits characteristics analogous to the 12-tone chromatic scale and its diatonic subsets and whose inherent relationships can be learned through repeated exposure. To allow for a new theory, we came up with a six-line staff, new note names, interval designations and clefs, which we call the Müller-Hajdu notation. There are three clefs, N, T and Z, for which we created corresponding chromatic notation styles (with notes either written without accidentals on a line or between two)[5].

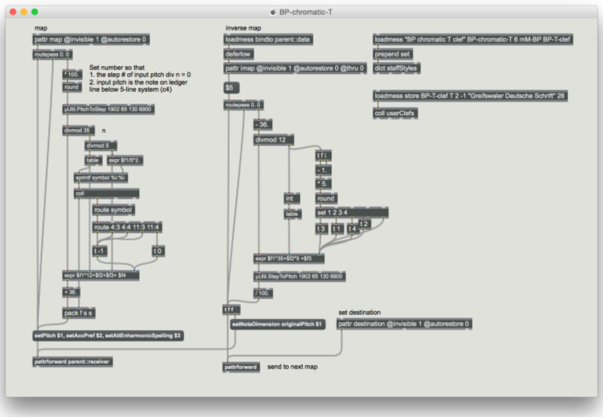

BP-chromatic-T
map
pattr map @invisible 1 @autorestore 0
routepass 0. 0
* 100.
round
Set number so that
1. the step # of input pitch div n = 0
2. input pitch is the note on ledger line below 5-line system (c4)
µUtil.PitchToStep 1902 65 130 6900
divmod 35
n
divmod 5
table
expr $f1/5*2.
sprintf symbol %i:%i
coll
route symbol
route 4:3 4:4 11:3 11:4
t -1
t 0
expr $f1*12+$f2+$f3+ $f4
+ 36.
pack f s s
setPitch $1, setAccPref $2, setAltEnharmonicSpelling $3
pattrforward parent::receiver
inverse map
loadmess bindto parent::data
deferlow
pattr imap @invisible 1 @autorestore 0 @thru 0
$5
routepass 0. 0
- 36.
divmod 12
t f i
- 1.
* 5.
int
round
table
sel 1 2 3 4
t 3
t 1
t 4
t 2
expr $f1*35+$f2*5 +$f3
µUtil.StepToPitch 1902 65 130 6900
/ 100.
t f f
setNoteDimension originalPitch $1
set destination
pattr destination @invisible 1 @autorestore 0
pattrforward
send to next map
loadmess "BP chromatic T clef" BP-chromatic-T 6 mM-BP BP-T-clef
prepend set
dict staffStyles
loadmess store BP-T-clef T 2 -1 "Greifswaler Deutsche Schrift" 28
coll userClefs

Figure 9. The Bohlen-Pierce T clef also allows for the microtonal subdivision of the BP base interval into 5 steps. It uses the mM-BP micromap for its single and double-shaft accidentals.

3.3.5 Bohlen-Pierce Clarinet Fingering Notation

Definition: “"BP Soprano Clarinet" BP-soprano-clarinet 5 mM-none default”

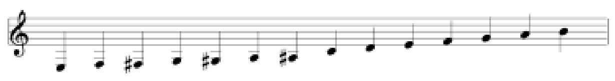


Figure 10. The Bohlen-Pierce scale in soprano clarinet fingering notation

Music written in N, T or Z clef can easily converted into Bohlen-Pierce clarinet fingering notation. There are two sizes of clarinets built by Canadian clarinet builder Stephen Fox [20], a soprano clarinet and a tenor clarinet, the former having the same size as a Bb clarinet and the latter having the size and shape of a basset horn. As the Bohlen-Pierce scale is based on the tritave, or just twelfth, the interval to which members of the clarinet instrument family will overblow, Fox was able to simplify the mechanics and thus proposed a fingering notation where certain notes are omitted in comparison to the Böhm and German systems.

[5] In case of the T clef we have even allowed for a microtonal division of the tempered Bohlen-Pierce scale step (146.3) into 5, virtually identical with the division of the octave into 41 steps (a well known tuning with a diatonic subset and a 24th scale degree being just the tiniest fraction higher than 3:2).

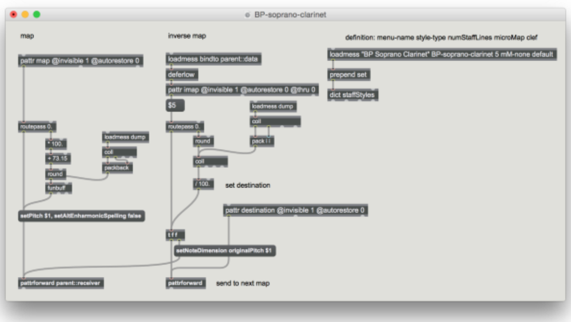

BP-soprano-clarinet
map
inverse map
definition: menu-name style-type numStaffLines microMap clef
pattr map @invisible 1 @autorestore 0
loadmess bindto parent::data
deferlow
pattr imap @invisible 1 @autorestore 0 @thru 0
loadmess "BP Soprano Clarinet" BP-soprano-clarinet 5 mM-none default
prepend set
dict staffStyles
$5
loadmess dump
coll
routepass 0.
* 100.
+ 73.15
round
funbuff
loadmess dump
coll
packback
routepass 0.
round
pack i i
coll
/ 100.
set destination
setPitch $1, setAltEnharmonicSpelling false
pattr destination @invisible 1 @autorestore 0
t f f
setNoteDimension originalPitch $1
pattrforward parent::receiver
pattrforward
send to next map

Figure 11. The soprano clarinet fingering notation style uses a lookup table to perform mapping.

3.3.6 Special Applications: Bohlen-Pierce Alto Kalimba

Definition “"BP Alto Kalimba" BP-alto-kalimba 5 mM-none percussion”

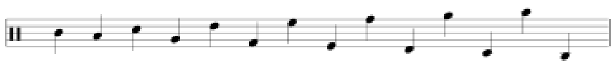


Figure 12. The BP Alto Kalimba notation style maps the ascending pitches of Bohlen-Pierce scale onto the centrifugal layout of the kalimba tines.

As mentioned before, new notation styles can easily be added such as in the case of the Hugh Tracey alto kalimba [21] whose 15 tines I tuned to the Bohlen-Pierce scale. For a percussionist, its pitches are best represented by notating the tines according to their alternating “centrifugal” layout with the longest tine in the middle being represented by a note in the middle of the staff.

This feat was accomplished by feeding *originalPitch* values through a µUtil.PitchToStep abstraction (to calculate the Bohlen-Pierce scale step index) and a lookup table, yielding the *pitch* to be displayed.

4. PRACTICAL APPLICATIONS AND FUTURE PLANS

One of my recent musical activities was the arrangement of the piano poem Vers la Flamme op. 72 by Alexander Scriabin [22] for 3 Bohlen-Pierce clarinets, Bohlen-Pierce guitar, double bass in Bohlen-Pierce scordatura, keyboard in Bohlen-Pierce layout, Bohlen-Pierce kalimba and tam-tam.

During the arrangement I:

1. imported a MIDI file found on www.kunstderfuge.com into the MaxScore Editor
2. mapped the tracks to the Bohlen-Pierce N, T and Z clefs
3. checked for motivic inconsistencies created by the automatic mapping and changed pitches where necessary
4. mapped the voices to various instrumental notations styles
5. extracted the parts for the musicians using the editor’s pdf generation capabilities.

Melle Weijters, the Amsterdam-based guitarist involved in the performance of the arrangement actually plays a 10-string guitar in 41EDO tuning. As the Bohlen-Pierce scale is actually a subset of this tuning, he only needed to find the correct positions on the fretboard. He therefore requested his part in T-clef with an additional empty 10-line tablature staff to manually notate finger positions. We are planning to automate this process and make it generally applicable to instruments with standard and non-standard numbers of frets and tunings. A number of papers have already dealt with the intricacies of automatic tablature transcription for guitar using genetic algorithms, neural networks and hill-climbing algorithms [23] [24].

Figure 13. The *Movinguitar* is a 10-string electric guitar in 41EDO tuning built by Dutch luthier *Lucid* for Melle Weijters.

Another interesting path to take would be exploration of graphical notation, such as the one employed by Karlheinz Stockhausen in his Elektronische Studie II [25]. The technological prerequisites have already been implemented in MaxScore, but it remains to be seen whether the effort of creating such a notation style or set of styles, for that matter, is justified in the light of the many individual solutions created by composers over the last few decades, or whether users would be best served by a separate specialized application. Sara Adhitya and Mika Kuuskankare have demonstrated a possible solution using macro-events in PWGL [26] for a piece by Logothetis.

Currently, our dynamic notations system is somewhat hampered by efficiency issues found in the Max JavaScript object. An effort will be spent to streamline the code and to replace it with Max C externals, if necessary.

5. CONCLUSIONS

We have developed for the MaxScore editor a plugin structure for dynamic notation that greatly facilitates the creation and practice of microtonal music in scenarios where composers, conductors and performers can no longer rely on a common notational reference internalized by years of training such as with the 12-tone system. Applying various styles in an arrangement for Bohlen-Pierce instruments proved to be a viable approach for editing, printing and rehearsing. More notation styles will be added as we further develop this version of the software, which currently is in a beta state and can be downloaded from http://www.computermusicnotation.com.

Acknowledgments

I would like to thank the *Behörde für Forschung und Wissenschaft Hamburg* for supporting our research in the framework of its *Landesforschungsförderung.* I would also like to thank Benedict Carey and Todd Harrop for proof reading.

6. REFERENCES

[1] L. Manovich, *The Language of New Media*. MIT Press, 2001.

[2] R. Dannenberg, "A structure for representing, displaying and editing music", *Proceedings of International Computer Music Conference*, 1986, pp. 153–160.

[3] A. Agostini and D. Ghisi, "Real-time computer-aided composition with bach", *Contemporary Music Review*, vol. 32, no. 1, 2013, pp. 41-48.

[4] M. Laurson, M. Kuuskankare, and V. Norilo, "An Overview of PWGL, a Visual Programming Environment for Music," *Computer Music Journal*, vol. 33, no. 1, 2009.

[5] G. Assayag and C. Agon, "OpenMusic Architecture," *Proceedings of the International Computer Music Conference*, 1996.

[6] http://inscore.sourceforge.net.

[7] http://fastlabinc.com/Siren/.

[8] https://ccrma.stanford.edu/software/cmn/cmn/cmn.html.

[9] H. Partch, *Genesis of a Music: Monophony*, University of Wisconsin Press, 1949.

[10] http://www.marcsabat.com/pdfs/notation.pdf.

[11] G. Read, *Music Notation: A Manual of Modern Practice.* Crescendo Book, 1979, p. 32.

[12] G. Hajdu and N. Didkovsky, "MaxScore – Current State of the Art," *Proceedings of the International Computer Music Conference*, 2012.

[13] http://www.smufl.org.

[14] M. Kuuskankare and M. Laurson, "Expressive Notation Package - an Overview," *Proceedings of the 5th International Conference on Music Information Retrieval,* 2004.

[15] J. Yasser, *Theory of Evolving Tonality*, American Library of Musicology, 1932.

[16] N.-L. Müller, K. Orlandatou and G. Hajdu, "Starting Over – Chances Afforded by a New Scale," in: *1001 Microtones*, M. Stahnke and S. Safari (Eds.). Von Bockel, 2014, pp. 127–172.

[17] H. Bohlen, "13 Tonstufen in der Duodezime," *Acustica,* vol. 39, no. 2, pp. 76–86, 1978.

[18] M.V. Mathews, J. R. Pierce, A. Reeves and L. A. Roberts, "Theoretical and experimental explorations of the Bohlen–Pierce scale," *Journal of the Acoustical Society of America*, vol. 84, p. 1214, 1988.

[19] P. Loui, Acquiring a New Musical System. PhD thesis, 2007. http://cnmat.berkeley.edu/library/acquiring_new_musical_system.

[20] http://www.sfoxclarinets.com.

[21] http://en.wikipedia.org/wiki/Mbira

[22] http://javanese.imslp.info/files/imglnks/usimg/5/52/IMSLP12740-Scriabin_-_Op.72.pdf

[23] S. Sayegh, "Fingering for String Instruments with the Optimum Path Paradigm", *Computer Music Journal,* vol. 13, no. 6, p. 7684, 1989.

[24] D. R. Tuohy and W. D. Potter, "Generating Guitar Tablature with LHF Notation Via DGA and ANN," *Proceedings of the 19th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE*, pp. 244-253, 2006.

[25] http://en.wikipedia.org/wiki/Studie_II

[26] Sara Adhitya, Mika Kuuskankare, "Connecting SUM with computer-assisted composition in PWGL: Recreating the graphic scores of Anestis Logothetis," *Proceedings of the Joint ICMC|SMC Conference,* Athens, 2014.

Webpages all accessed on January 28, 2015.