Improvements in bach 0.8.1, a User's Perspective

Jonathan Bell

Improvements in bach 0.8.1, a User's Perspective

Jonathan Bell
Aix Marseille Univ, CNRS, PRISM
Perception, Representations, Image, Sound, Music
Marseille, France
belljonathan50@gmail.com

ABSTRACT

This paper discusses recent uses of *bach* automated composer's helper), a Max library for algorithmic composition. In the context of the author's own works, audiovisual musical scores are (pre)-composed in *bach* (bach.score). In performance, the scores are sent to performers and synchronised to a shared common clock, in local networked music performances (npm), with the help of the SmartVox score distribution system. The 0.8.1 version of *bach* presents two major improvements which will articulate the structure of this article: on the lower level, *bach* now exposes an expr-like small language (*bell*: bach evaluation language for llls), which greatly simplifies algorithmic processes in Max. The case study of an algorithm for automatic cueing system for singers will be given as exemple. On the higher level, bach.roll now supports dynamics, thus revealing promising user-friendly playback possibilities, exemplified here with Ableton and the ConTimbre library.

1. INTRODUCTION

1.1 Technology for Notation and Representation

The present research situates itself within the realm of "Technologies for Notation and Representation (TENOR)", which emerged in France around 2014 under the initiative of Dominique Fober (GRAME-CNCM). The first conference of the same name took place at IRCAM in 2015, and enjoys ever since a growing community of researchers, developers, scholars, and composers. Amongst the latter (composers), the "slow but steady shift away from textualization in digital media" [1] inspires very diverse avenues of research, encompassing live generated scores [2] and their link to "comprovisation" [3] [4], realtime notation/algorithmic composition [5] [6], animated notation [7] [8], audio-scores [9] [10], augmented reality notation [11] [12] and distributed notation [13], to name a few.

1.2 Distributed Notation with SmartVox

Smartvox is web application dedicated to score distribution. Amongst the aforementioned categories, SmartVox best situates itself under the categories of audio-scores [14], distributed notation [15], and more recently augmented reality notation [16]. Obtaining best results with choirs (with or without instruments), it consists in distributing audiovisual mp4 scores (i.e. pre-composed audio and animated notation), and synchronising them to a shared common clock, through the browser of the singer's own Smartphones. Developed by Benjamin Matuszewski and the author at IRCAM in 2015, in the SoundWorks framework [17], SmartVox is based on a node.js server communicating with clients (smartphones) through the WebSocket protocol. The great majority of the scores sent and synchronised by SmartVox are composed by the author in the *bach* environment. Whilst *bach* is perhaps more often used for live-generated scores than for more traditional algorithmic composition, such live or ephemeral scores [18] have nevertheless endured criticism, which encourages the author to carry on writing fixed scores (i.e. the same score each time, although animated).

1.3 Algorithmic composition with *bach*

bach [19] is a library of external objects for real-time CAC (Computer-Aided Composition) and musical notation in Max. Providing Max with a large set of tools for the manipulation of musical scores, it has later been expanded with the higher-level Cage [20] and Dada [21] libraries. *bach* inherits from softwares such as Open Music or PWGL a lisp-like organisation of data, albeit with its own syntax called llls (Lisp-Like Linked Lists). The latest version of *bach* (version 0.8.1 released in 2019) exposes the *bell* language [22] [23], conceived specifically for llll (lists) manipulation.

2. BACH EVALUATION LANGUAGE FOR LLLL- AN OVERVIEW OF THE *BELL* LANGUAGE

2.1 General presentation

The *bell* language in bach arose from an observation that the Max patching environment can be cumbersome when formalising algorithmic compositional processes: "It has been clear since the beginning of *bach* that non trivial tasks - in algorithmic composition - require the implementation of potentially complex algorithms and processes, something that the graphical, data-flow programming paradigm of Max [...], is notoriously not well-suited to." [22].

Indeed, while the Max GUI can be extremely intuitive and efficient for many DSP processes, its data-flow paradigm

can make list (llll) formatting inefficient. As will be exemplified with the help of Fig.1, 2 and 3, bach.eval now allows for a single line of code to centralize list formatting, which would have formerly required dozens of objects, themselves most often bringing priority issues.

2.2 Writing notes and playing them

In some cases, composing with *bach* essentially comes down to formatting "addchord" messages (i.e. adding note on the score) containing all necessary information about where/how each note should be inscribed and displayed, and what Max should do when retrieving that information later during playback. Lisp inherited llls (lisp-like linked lists) can be represented as trees (see Fig. 1).

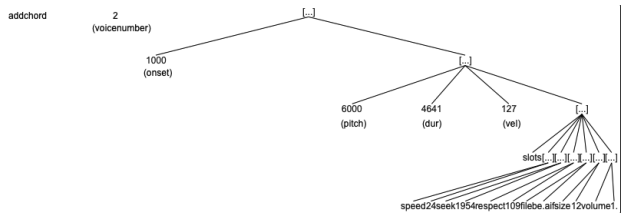

Figure 1. A tree representation of the following list: "addchord 2 [1000 [6000 4641 127 [slots [speed 24] [seek 1954] [respect 109] [file be.aif] [size 12] [volume 1.]]]]"

Before the existence of the *bell* language (mainly exposed in Max with the bach.eval object, the formatting of such list messages typically required the user to build that tree "from the ground up" (see Fig. 2, reading the patch from top to bottom, and Fig.1 from bottom to the top). The process implied for instance: 1/ appending together all slot content 2/ wrapping it together (adding a level of parenthesis) 3/ appending pitch, duration, and velocity 4/ wrap again 5/ prepend onset 6/wrap again 7/ prepend addchord and voicenumber....

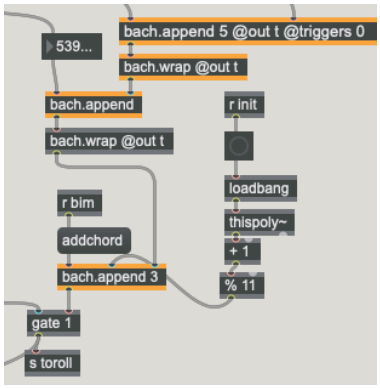

Figure 2. The object bach.wrap add a level of parenthesis around a list. Bach.append appends lists together. Formatting complex lists in this way was cumbersome and error-prone.

Such a process was often prompt to error, and now gives evidence of the fact that a single bach.eval expression makes the process much easier and versatile (see Fig. 3).

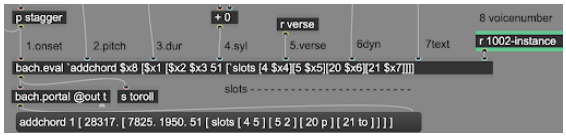

Figure 3. The *bell* language (mainly exposed through the bach.eval object) was conceived for lisp-inherited lisp-like structured messages. $x1, $x2... correspond to the different inlets of the object. *bach.eval* makes the construction of lisp-inherited parenthesis structures much easier than with the data-flow *bach.wrap* system.

The implementation of variables in the *bell* language constitutes another major improvement of *bach*. The ability to name variables in Max in this way (such as ONSET, or DUREE, as in the loop expressed in Fig. 4) helps again centralising information within simple expressions, which the message-driven send-receive Max functionality would have made cumbersome.

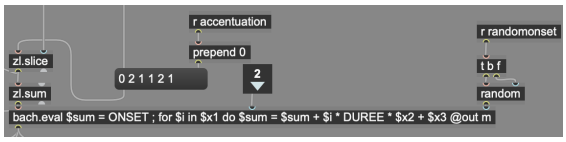

Figure 4. A loop calculates the onset of each syllable of a vocal line according to a starting onset (the variable "ONSET"), a given duration/tempo ("DUREE"), and prosodic accentuation (2 1 1 2 1 for long short short long short - 0 is used for initialisation).

2.3 Use cases in compositions for voices

According to the performer's feedback, and in comparison with the author's previous artistic output and experience with composing in *bach* (since 2015), the use of algorithmic processes (such as the ones exposed below) demonstrated much better results than "hand-made" former methods. Although in germ in *Common Ground* [1] (Fig. 5), a more systematic approach to algorithmic polyphony generation was used in a *Deliciae* [2] (Fig. 6), while discovering the new *bell* language.

Most polyphonic passages in *Deliciae* were generated inside a poly~ in Max, with each instance (i.e. each voice) receiving the same types of information (i.e. text, prosody, melodic contour and harmonic material) but only differing by vocal range (sopranos for instance cannot sing below middle C and so forth). Contrast in Renaissance polyphony often consist in alternation between homophonic passages and contrapuntal ones. Contrast therefore mainly consisted in opposing textures in which singers articulate and move from one pitch to the next at the same time (homophonic) or not (contrapuntal). This consideration highly influenced the parameters chosen at the foreground of the "composer's interface" [3]. ONSET is in milliseconds and correspond to

[1] Video of the performance: https://youtu.be/ZrLgbBw4xfU
[2] Video of Sevilla's performance: https://youtu.be/zxnznD0Gzo0
[3] See parameters tweaks on the right hand side for demonstration here: https://youtu.be/OKkiySEagm0

Figure 5. Common Ground. Photography: Simon Strong.

the position in the timeline where the generation is happening: as exemplified in the video 747618 ms correspond to 12'28". The term DUREE (French for duration) represents the duration of notes: the tempo speeds up when durations diminished. RANDUR sets the degree of randomness in durations, RANDOMONSET sets the degree of randomness in the onset of the first note of the phrase, DECAL sets the duration of delay between voices, positive value goes from low register to high, negative values from high (sopranos) to low (basses). STAGGER, finally, imitates a behaviour typical of the renaissance where two groups of voices are staggered or delayed by a given value. When the variables RANDUR, RANDOMONSET, DECAL, and STAGGER are set to 0, the algorithm generates a homophony. If only RANDUR increases, voices will start at the same time, but their duration will differ between each other. If only RANDOMONSET increases, they will all have the same duration but start at different times. If only DECAL increase, voice will enter at regular intervals from the bottom-up (and inversely if DECAL is negative).

Figure 6. Deliciae (Madrid version). Photograpy: Enrique Muñoz

2.3.1 Automatic cueing system

Since the beginning of SmartVox (see [24]), cueing the singers with what comes next appeared one of the main advantages of the system.

To identify appropriate moments for page turns and in order to cue the singers accordingly, the first step consisted in identifying the start and end of each phrase (see Fig. 7): with iterations on each note of the score two by two, we evaluate if the distance between two notes is superior to 600 ms or not. Fig. 8 and Fig. 9 illustrate how the decision is taken: in the first case it isn't (see Fig. 8, the two notes are close to one another) and nothing happens. On the following iteration however, the gap between two notes is wider than 600ms (see Fig. 9), so the messages "addmarker fin" and "addmarker debut" are sent to the end of the phrase and to the beginning of the next phrase respectively, resulting in adding green marker named *debut* and *fin* at the beginning and the end of the phrase respectively (see Fig. 8 and Fig. 9).

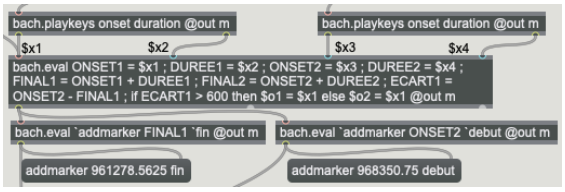

Figure 7. The script above adds markers only when two notes are separated by more than 600ms.

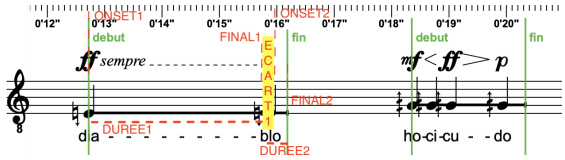

Figure 8. The first note (with lyrics "dia") has a duration that lasts until the beginning of the following note, (with lyrics "blo"). The distance between the two (ECART1, highlighted in yellow) is almost null.

When a performer has nothing to sing, this precious time is systematically used in the score to provide cues feeding the perfomer's headphone with what is coming next: using the markers previously generated to retrieve their onsets, if the pause is longer than the phrase to sing, (i.e. if the DURANTICIP is greater than DUR (see Fig. 10, and the "then" stance of the "if" statement in the code displayed in Fig. 12), then the cue will need to start at the onset corresponding to the difference between entrance of the singer (START) and the end of his phrase (END), with a 300ms break between the two. If on the other hand, the pause is shorter than the phrase to sing (see Fig. 11, and the "else" stance of the if statement in the code displayed in Fig. 12),

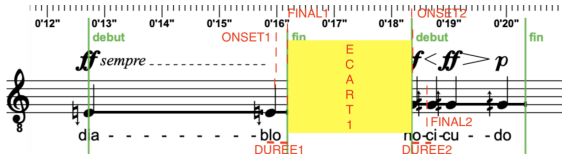

Figure 9. The two notes (with lyrics "blo" and "ho" respectively) are separated by a silence longer than 600 ms (ECART1 lasts a bit more than two seconds), therefore two markers are generated.

then the cue needs to start as soon as possible, i.e. as soon as the singers has finished the previous phrase (PREV):

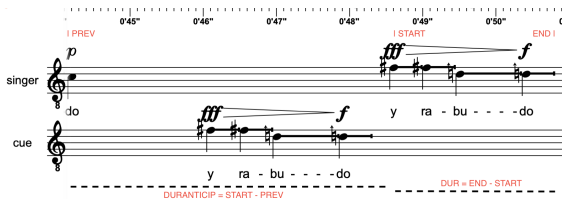

Figure 10. When the pause is long (or very long....) the cue needs to be provided as late as possible i.e. just before the singer's entrance. The corresponding onset value is 0'46" because START*2 - END = 48,5*2 - 51 = 46

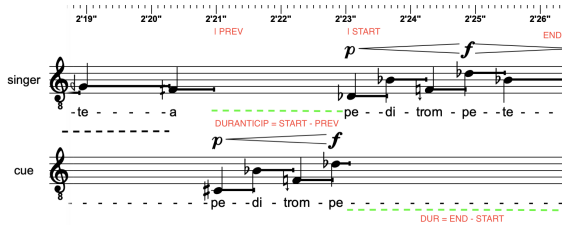

Figure 11. When the pause is short, the cue needs to be provided as soon as possible i.e. just after the previous singer's phrase (see the PREV variable).

```
START = $x2:($x1 1) ;
END = $x2:($x1+1 1) ;
PREV = $x2:($x1-1 1);
DUR = END - START ;
DURANTICIP = START - PREV ;
if DURANTICIP > DUR then ';
'tocue 'paste 2* START - (END + 300) 2
else '; 'tocue 'paste (PREV + 60) 2
```

Figure 12. Coding example in *bell*

Finally, the 'end' markers (the ones named 'fin', as in Fig. 8 at 0'16"200"') are used for score display: the domain to be displayed on the playing staff and on the preview staff of the *bach.roll*(i.e. the staff-line that is coming next, as for page turns). Fig. 13 shows a coding example for formatting messages aimed at displaying the right portion of the score at the right time on both staves.

Each time the cursor hits one of these markers, the domain display of both 'playing' and 'preview' staves are updated, provoking at the same time an alternation up and down between the position of those staves, so that the passive (or 'preview') roll looks like an anticipation of the active (or 'playing') one, resulting on a 2-staves display with constant preview. [4]

[4] See for instance the tenor part: https://youtu.be/NLpI_OpFcTs

```
'addmarker $x2:$x1
['play [$x2:$x1 ($x2:($x1+1)+ 200)]
'preview [$x2:($x1+1) ($x2:($x1+2)+ 200)]]
```

Figure 13. Message formatting for maker generation in *bell*.

3. DYNAMICS IN *BACH* 0.8.1, IN SEARCH FOR PLAUSIBLE INSTRUMENTAL WRITING MOCKUPS

It is often a danger for a composer to write only what he can hear on his computer, rather than from his imagination directly. First he will limit his palette to what his virtual instruments can do, but also these instruments will always draw his attention to how they sound best, rather than how the instrument they are emulating would sound. However, the simulations of a sample library like ConTimbre [5] slightly undermine this statement because of the infinity of timbre combinations it makes available to composers today. Taking advantage of *bach* 0.8.1 dynamics support, the following proposes an overview of some *mockup* strategies for instrumental writing, showing how *bach* may be particularly handy in the context of *musique mixte* (instrumental/vocal music with electronics), notably thanks the control of Ableton Live and/or ConTimbre.

3.1 *bach*-Ableton

The bach.roll object displays notation in proportional time (as opposed to bach.score in mensural notation), and outputs notifications of its playback status in real-time. These notifications can be interpreted in *Max For Live* in order to synchronize the notation for human players in *bach* with the electronic tape in Ableton Live. Figure 14 shows how Ableton's playback controls can be accessed through the live-set path of the live object model (LOM), which makes constant back and forth playbacks possible between composition of the vocal/instrumental lines in the score and electronic tape (fixed media) in Ableton, during the compositional process. Here *bach* ("master") sends playback notifications to *Ableton* ("slave") though the *live.object*.

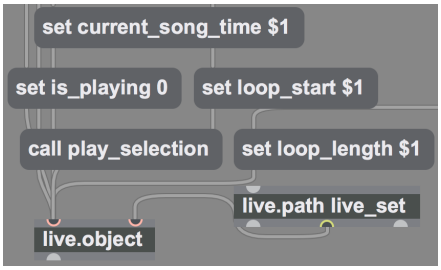

Figure 14. *Max For Live* device syncing *bach* and Ableton.

3.2 Controlling VSTs in Ableton via *bach*: the drawbacks of MIDI

We'll examine here the case where *bach* (the fullscore) is used within Max-for-live to control synthesisers or VSTs.

[5] https://www.ConTimbre.com/en/

One way to do this consists in hosting the score in one track (e.g. the master track), and adding one track per instrument e.g. violin on track 1, cello track 2...). Max-for-live devices can send messages to each other (but not signal), so the full-score track can send control data to various instruments (interpreted as MIDI by VSTs).

Numerous strategies can be used in *bach* to map symbolic data to dynamic change. The most common way will consist in using a function stored as slot [6] content, for instance dB function of time (see Fig. 15), to be retrieved during playback with the help of bach.slot2curve, thus mimicking the behaviour of automation lines in a DAW.

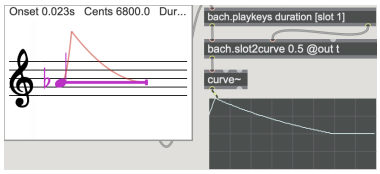

Figure 15. bach.slot2curve allows for automation curve retrieval.

Very similar results can be achieved by mapping the thickness or opacity of the duration line of the note to dynamic change, using this time bach.dl2curve. In this case the volume of the sound increases if the duration line gets thicker over time. Whilst the MIDI control data of the instrument (noteon pitch velocity noteoff...) is sent to the entry of the VST, such automation-like gain/volume control must be placed at its output. Some difficulties then emerge with this approach to MIDI-to-VST-instrument communication:

- MIDI velocity and volume/automation data tend to contradict each other (e.g. what is the velocity of a crescendo dal niente?)
- crescendos and diminuendo merely multiply the signal's value without changing its spectral content (Risset famously evidenced the correlation between loudness and brightness in his trumpet study [25])
- pitch-wise, MIDI communication needs to rely on pitch bend, which is relatively easy when dealing with finite values such as semi/quarter or eighth of tones. Pitch bend can however be more error prone than straightforward MIDI-cent communication (supported by contimbre through OSC communication) when dealing with non-approximated pitch value (e.g. making sure that 59.99 sounds "C" etc...).

3.3 *bach* Controlling ConTimbre via OSC

ConTimbre is a large database of orchestral sounds (87000 samples, 4000 playing techniques). Its use in conjunction with *bach* promisses being of interest for plausible mock-ups using instrumental extended technique, with clear possibilities in the realm of algorithmic composition. Considering one instrument (e.g. flute), each playing technique can be considered in *bach* as a list of slot information describing note-head, articulation, textual information to display on the score, and control data assigning that note to a specific playing technique of the ConTimbre library. The lisp-inherited data structure of *bach* allows to represent all this information as a tree. [7]

The OSC syntax of ConTimbre also allows for precise control over dynamic change. Dynamic changes will therefore simulate true crescendi or diminuendi by sample interpolation. In a trumpet crescendo for instance, the timbre or harmonic content of the sound will transform over time, from low to high regions of the spectrum. To do so directly from notation information retrieval, the dynamic of the note is treated differently whether it is stable (i.e. if the dynamics list contains only one symbol e.g. f as "forte"), or if it changes over time (e.g. if the dynamics list contains 3 symbols e.g. o <fff as "crescendo dal niente a fff"). The figure below routes dynamic information accordingly.

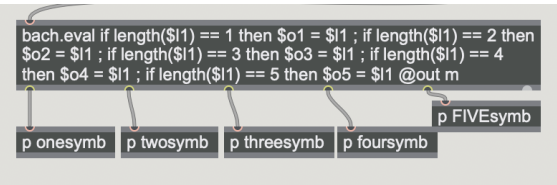

Figure 16. Dynamic information is routed according to the length of its list: "fff >o" contains 3 symbols.

If the dynamic information contains 3 symbols, it will send to ConTimbre a noteon message with its initial dynamic, followed by a "dynamic" message containing the duration of the note and the target value.

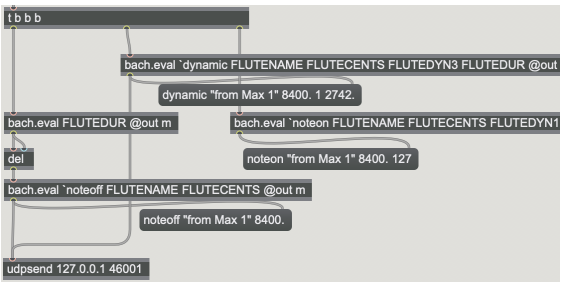

Figure 17. The message 'dynamic "from max 1" 8400. 1 2742' demands ConTimbre to reach velocity 1 in 2742ms"

3.4 *bach*, *dada* and attempts at using AI-inspired techniques

A playback engine such as the one described above (*bach-contimbre* could lead to interesting research in timbre combination and computer-aided composition, using techniques borrowed from AI in particular. Attempts are currently undertaken at classifying data with the help of the *dada* [21] library. [8]

4. CONCLUSIONS

Recent improvements of the *bach* library for Max open promising perspectives in the realm of algorithmic composition, with the implementation of the *bell* language in

[6] The concept of slot in *bach* consists in attaching various kinds of metadata to a single note.

[7] See https://youtu.be/cAtcMUKbQbs for demonstration.

[8] See for demonstration : https://youtu.be/UxaE$_u$KtXb8 and https://youtu.be/ByeIyRLnX-w.

particular. The seemingly anecdotal support of dynamics offers musical and intuitive playback control possibilities, and re-inforcing the link between historically-inherited musical notation and forward-looking algorithmic processes, which make *bach* an invaluable tool for composers today.

Acknowledgments

The author would like to thank Andrea Agostini, Daniele Ghisi, and Jean-Louis Giavitto from the latest improvements of *bach*. In memoriam Eric Daubresse who highly supported Cage, the first extention of the *bach* library.

5. REFERENCES

[1] G. Hajdu, "Embodiment and disembodiment in networked music performance," in *Body, Sound and Space in Music and Beyond: Multimodal Explorations*. Taylor & Francis, 2017.

[2] J. Freeman, "Extreme sight-reading, mediated expression, and audience participation: Real-time music notation in live performance," *Computer Music Journal*, vol. 32, no. 3, pp. 25–41, 2008.

[3] R. Dudas, "Comprovisation: The various facets of composed improvisation within interactive performance systems," *Leonardo Music Journal*, vol. 20, pp. 29–31, 12 2010.

[4] P. Louzeiro, "Improving sight-reading skills through dynamic notation – the case of comprovisador," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 55–61.

[5] A. Agostini and D. Ghisi, "Bach: an environment for computer-aided composition in max," in *Proceedings of the 38th International Computer Music Conference (ICMC)*, Ljubljana, Slovenia, 2012.

[6] G. Hajdu and N. Didkovsky, "Maxscore: Recent developments," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 138–146.

[7] C. Hope, "Electronic scores for music: The possibilities of animated notation," *Computer Music Journal*, vol. 41, pp. 21–35, 09 2017.

[8] D. Fober, Y. Orlarey, and S. Letz, "Towards dynamic and animated music notation using inscore," in *Proceedings of the Linux Audio Conference — LAC 2017*, V. Ciciliato, Y. Orlarey, and L. Pottier, Eds. Saint Etienne: CIEREC, 2017, pp. 43–51. [Online]. Available: inscore-lac2017-final.pdf

[9] S. Bhagwati, "Elaborate audio scores: Concepts, affordances and tools," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 24–32.

[10] C. Sdraulig and C. Lortie, "Recent audio scores: Affordances and limitations," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, C. Hope, L. Vickery, and N. Grant, Eds. Melbourne, Australia: Monash University, 2019, pp. 38–45.

[11] D. Kim-Boyle and B. Carey, "Immersive scores on the hololens," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, C. Hope, L. Vickery, and N. Grant, Eds. Melbourne, Australia: Monash University, 2019, pp. 1–6.

[12] Z. Liu, M. Adcock, and H. Gardner, "An evaluation of augmented reality music notation," 11 2019, pp. 1–2.

[13] R. Gottfried and G. Hajdu, "Drawsocket: A browser based system for networked score display"," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, C. Hope, L. Vickery, and N. Grant, Eds. Melbourne, Australia: Monash University, 2019, pp. 15–25.

[14] J. Bell, "Audio-scores, a resource for composition and computer-aided performance," Ph.D. dissertation, Guildhall School of Music and Drama, 2016. [Online]. Available: http://openaccess.city.ac.uk/17285/

[15] J. Bell and B. Matuszewski, "SmartVox. A web-based distributed media player as notation tool for choral practices," in *Proceedings of the 3rd International Conference on Technologies for Music Notation and Representation (TENOR)*. Coruña, Spain: Universidade da Coruña, 2017.

[16] J. Bell and B. Carey, "Animated notation, score distribution and ar-vr environments for spectral mimetic transfer in music composition," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'19*, C. Hope, L. Vickery, and N. Grant, Eds. Melbourne, Australia: Monash University, 2019, pp. 7–14.

[17] N. Schnell and S. Robaszkiewicz, "Soundworks – A playground for artists and developers to create collaborative mobile web performances," in *Proceedings of the first Web Audio Conference (WAC)*, Paris, France, 2015.

[18] S. Bhagwati, "Vexations of ephemerality." in *Proceedings of the 3rd International Conference on Technologies for Music Notation and Representation (TENOR)*. Coruña, Spain: Universidade da Coruña, 2017, pp. 161–166.

[19] A. Agostini and D. Ghisi, "A max library for musical notation and computer-aided composition," *Computer Music Journal*, vol. 39, no. 2, pp. 11–27, 2015.

[20] E. D. A. Agostini and D. Ghisi, “cage: a high-level library for real-time computer-aided composition,” in *Proceedings of the International Computer Music Conference (ICMC)*, Athens, 2014.

[21] D. Ghisi and C. Agon, “dada: Non-standard user interfaces for computer-aided composition in max,” in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR’18*, S. Bhagwati and J. Bresson, Eds. Montreal, Canada: Concordia University, 2018, pp. 147–156.

[22] J.-L. Giavitto and A. Agostini, “Bell, a textual language for the bach library,” in *ICMC 2019 - International Computer Music Conference*, New York, United States, Jun. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02348176

[23] A. Agostini, D. Ghisi, and J.-L. Giavitto, “Programming in style with bach,” in *14th International Symposium on Computer Music Multidisciplinary Research*. Marseille, France: Mitsuko Aramaki, Richard Kronland-Martinet , Sølvi Ystad, Oct. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02347961

[24] J. Bell and B. Matuszewski, “SMARTVOX - A Web-Based Distributed Media Player as Notation Tool For Choral Practices,” in *TENOR 2017*, Coruña, Spain, May 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01660184

[25] J.-C. Risset and D. Wessel, “Exploration of timbre by analysis and synthesis,” in *The Psychology of Music*, D. Deutsch, Ed. Academic Press, 1999, pp. 113–169. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00939432