# SIGN/E: SYMBOL-BASED INTERFACE FOR GRAPHIC NOTATION EDITION FOR ABLETON LIVE

**Pierre-Luc Lecours**
Université de Montréal
Laboratoire formes • ondes (LFO), CIRMMT
pierre-luc.lecours@umontreal.ca

**Nicolas Bernier**
Université de Montréal
Laboratoire formes • ondes (LFO), CIRMMT, Hexagram
nicolas.bernier@umontreal.ca

**Evan Montpellier**
Université de Montréal
Laboratoire formes • ondes (LFO)
evan.montpellier@gmail.com

**Philippe-Aubert Gauthier**
Université du Québec à Montréal
CIRMMT, Hexagram
gauthier.philippe-aubert@uqam.ca

## ABSTRACT

This article outlines the functionality of SIGN/e (Symbol-based Interface for Graphic Notation Edition), a set of Max for Live devices allowing to easily create, read and animate graphic scores within Ableton Live. It addresses the need for an efficient DAW-integrated notation system suited for electronic, instrumental and mixed-media practices. Taking advantage of the intuitive Ableton Live workflow, SIGN/e allows the creation and parametric control of individual symbols and textual elements as well as management of global visualisation properties. This architecture allows the construction of dynamic synchronized graphic scores while benefiting from Ableton Live's timeline and automation system. SIGN/e is built entirely with standard Max objects to ensure long-term stability and compatibility. The visual rendering pipeline relies on Jitter and OpenGL, with shaders written in Gen to handle specific needs such as grid generation, cropping, and compositing. SIGN/e provides a flexible environment for writing and performing animated scores. This article summarizes SIGN/e's conceptual foundations, technical structure, and potential for expanding contemporary notation practices.

## 1. INTRODUCTION

Technological and computational advances over the past decades have led to the development of numerous software tools dedicated to the creation, editing, and playback of musical scores, most of them revolving around Common Western Notation (CWN), while offering limited options for idiosyncratic graphic score creation.



SIGN/e (Symbol-based Interface for Graphic Notation Edition) is designed for the creation, animation, and playback of graphic scores within Ableton Live. SIGN/e consists of a set of AMXD devices that allow users to import, modify, move, and animate graphic symbols (raster image files like JPGs, TIFFs and PNGs) in a 2D space that scrolls horizontally in sync with Ableton Live's timeline. The device is the outcome of several iterations of research-creation projects focused on notation for Ensemble d'oscillateurs [1]. Since 2016, the ensemble's activities have contributed to electronic music group performance practices, the development of graphic notation systems, and the design of digital tools dedicated to the transmission of electronic music [2]. Notably, the ensemble previously led to the development of Graphic Score Reader (GSR), a device enabling the playback of scrolling graphic files in Ableton Live. The conception and development of SIGN/e build on this knowledge. This article will first outline a brief overview of contemporary software initiatives in graphic, animated, and interactive notation to contextualize SIGN/e's design history, technical specificities and functionality[1].

## 2. SCORE NOTATION EDITOR AND READER OVERVIEW

There are several notation software options available; the most well-known, such as Finale, Sibelius, MuseScore, and Dorico, are specializing in CWN, offering only limited options for graphic score creation.

Other initiatives, like OpenMusic [3] and LilyPond [4], offer more flexible and programmable approaches to composition and musical notation. OpenMusic employs object-oriented programming, while LilyPond relies on a textual syntax to produce scores that meet professional quality standards. While these applications offer options to output graphical elements, their focus remains rendering CWN scores.

[1] Please also see this video summarizing SIGN/e functionality: https://youtu.be/WrAxyXMXBoQ

A small number of plugins that allow graphical notation symbols have been integrated into music production environments, such as MaxScore2 [5], which operates within Ableton Live and Max. MaxScore2 supports graphical notation, interactive scores and CWN through the BACH library [6], another Max package. BACH provides advanced computer-assisted composition features, including graphic notation and real-time score generation.

Other score-reading applications have emerged to encompass the diversity of notation practices, including non-conventional notation. The simplest form includes tools such as forScore that remain specialized in CWN while allowing the display of graphic scores as PDF documents.

Applications like Decibel Score require the composer to create a score using external graphics editing software and import it into the program, which is then converted into a scrolling score with animation features that can be displayed on multiple synced iPad devices [7]. While these tools provide features such as scrolling score display and multi-device synchronisation that are beneficial for performance, their reliance on the use of external graphics editing software for composition may be technically challenging for some.

Other complex software environments such as IanniX [8] and Ossia Score [9] offer solutions for event sequencing and real-time score generation. While these platforms allow dynamic interaction with other software, facilitating synchronization and the triggering of sonic or visual events, their complex graphic and code-based approach can present steep learning curves for some artists or offer too many options for those wishing to focus only on creating a graphic score.

SIGN/e has been developed to fill the gap within this spectrum of graphic notation possibilities by serving as an accessible and easy-to-use platform that works in conjunction with the digital music production capabilities of a Digital Audio Workstation (DAW).

## 3. FROM GSR TO SIGN/E

Developed between 2018 and 2021 for "Ensemble d'oscillateurs" at the Université de Montréal, Graphic Score Reader (GSR) is a Max for Live plugin designed for real-time scrolling display of graphic scores. The project was directed by Nicolas Bernier and programmed by Evan Montpellier.

GSR supports a variety of image formats. Each Max for Live device can contain up to twelve score files, allowing sequential playback of multiple scores in a concert performance context.

Multiple devices can be loaded simultaneously to project different synchronized versions of the same graphic score (e.g. a clean version for the audience and an annotated version for performers). Score playback can be synchronized with Live's transport or controlled using automation curves. In 2023, Ensemble d'oscillateurs' instrumentarium evolved from sine wave oscillators to analog synthesizers and "no-input" mixers. This transformation introduced new considerations for musical writing and raised unprecedented challenges, paving the way for the creation of SIGN/e.

Like other similar applications, GSR lacked flexibility in score editing as it forced the use of external graphics editing software. Thus, adjusting scores was a cumbersome process: each modification required returning to the graphics software, making corrections, re-exporting, and re-importing the updated score into GSR. Furthermore, the use of external graphics editing programs can be prohibitive for individuals without prior knowledge of such tools. SIGN/e emerged from a growing need to simplify the process by merging writing and reading within the same environment. Furthermore, the integration of SIGN/e into a DAW allows for extensible modular control by using Ableton Live's parameter mapping functionality, as will be detailed in this article.

## 4. SIGN/E: SYMBOL-BASED INTERFACE FOR GRAPHIC NOTATION EDITION

### 4.1. System Overview

SIGN/e operates through four interconnected devices:

1) SIGNe-Screen.amxd serves as the central hub for visualization and global control;
2) SIGNe-Symbol.amxd manages individual graphic elements and their dynamic properties;
3) SIGNe-Text.amxd allows textual notation and animation;
4) and SIGNe-Background.amxd enables the addition of a fixed or scrolling image that can serve as a score template or other visual reference created using external graphics software.

This architecture allows composers and performers to integrate visual notation directly into a DAW (Figure 1).

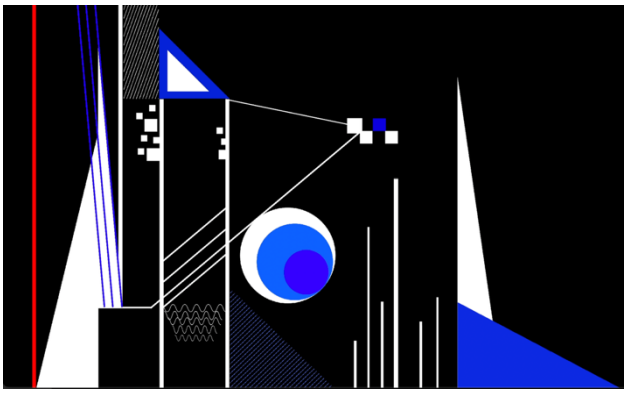

**Figure 1**. Still from an animated graphic score by Alexandre Sasset-Blouin, entirely created with SIGN/e for Ableton Live.

By combining visual design tools with the automation and sequencing capabilities of Ableton Live, SIGN/e provides virtually endless possibilities. For instance, one could easily map Ableton Live's random LFO to the height of a symbol so its position would randomly change over time, in synchronicity or asynchrony with Ableton Live's

timeline; or one could map an envelope follower to the opacity of a symbol so that input from an acoustic instrument would control the appearance and disappearance of a graphic symbol. SIGN/e fosters dynamic interaction between sound and image via its integration with Ableton Live's features, supporting both structured and improvisational approaches.

Its architecture facilitates idiosyncratic forms of notation and performance, particularly in contexts where conventional notation is inadequate for representing and communicating the musical ideas.

### 4.2. Architecture and Workflow

The SIGNe-Screen.amxd device acts as the host for all SIGNe-Symbol.amxd, SIGNe-Text.amxd and SIGNe-Background.amxd devices added to the session. SIGNe-Screen.amxd generates the video output window where graphic elements are displayed and can be manipulated directly on the screen with the mouse, while providing access to global score parameters via the device's settings window, e.g. background color, quantization, time ruler, etc. SIGNe-Symbol.amxd, SIGNe-Text.amxd and SIGNe-Background.amxd are placed on MIDI tracks. Each device corresponds to a single visual element. The parameters of each symbol are automatable, editable and mappable like any Ableton Live plugin. Depending on preference, the user may choose to use one track for each device or add multiple devices to a track and select the one to automate using Live's Device Chooser dropdown menu.

To use the SIGN/e devices, a SIGNe-Screen.amxd should always be added onto a single MIDI track to display the video window. Subsequent SIGNe-Symbol.amxd devices are added to other tracks, introducing placeholder elements that can be replaced through the Symbol and Background image selector, or the Text editor.

### 4.3. Global Controls

SIGNe-Screen.amxd (Figure 2) allows users to manage the global settings of the score (Figure 3) via the settings menu. The overlay menu provides access to display, navigation and project parameters. Additional options include among others, Freeze All, which locks all symbols against manual repositioning.

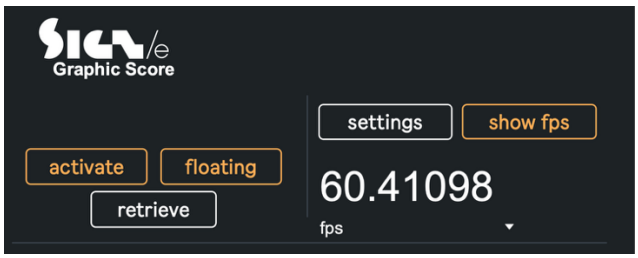


**Figure 2**. SIGNe-Screen.amxd device

The device setting menu also gives access to a session editable quantized grid, a time ruler showing measure numbers on the bottom of the score, and a scrolling playhead synchronized with Ableton Live's transport. Window management options include buttons to make the window float over other windows and to restore it to a default position, as well as settings to adjust the background colour and a global framerate indicator.


**Figure 3**. SIGNe-Screen.amxd settings menu

Playback can follow Ableton Live's transport or be driven by automation curves. Score duration is defined by a numeric value in measures. Driving the score via automation curves allows the score to scroll both backward and forward as the Live set plays. The scrolling position parameter can also be mapped to a controller to facilitate manual control over the movement of the score, making it possible to move through sections as quickly or slowly as one wants. More options are accessible through the device settings interface, and their functions are covered in the user manual.

### 4.4. Symbol-Level Controls

SIGNe-Symbol.amxd, SIGNe-Text.amxd, and SIGNe-Background.amxd manage the properties and behaviours of individual graphic elements shown in the score window. SIGNe-Symbol.amxd provides visualization of the selected symbol, along with its controls: rotation, position, scale, opacity, layer depth, and its repetition (Figure 4).

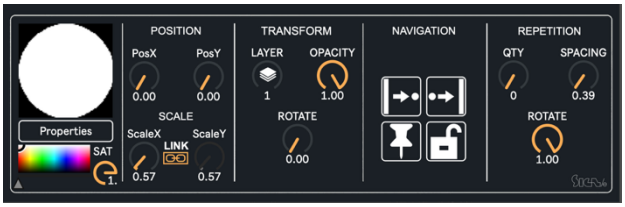


**Figure 4**. SIGNe-Symbol.amxd device

The device properties allow users to choose a symbol and a pattern while accessing numerous parameters (Figure 5). Colour settings support interpolation between two user-selected colours with automation capability, allowing alternation between the two selected colours via automation or any mappable Ableton Live device (LFO, envelope follower, etc.). The pattern parameters such as tiling and opacity can be adjusted dynamically: for instance, a circular symbol could have a square tiling pattern that would grow bigger over time to emphasize a timbre mutation within the duration of a musical gesture (the gesture symbolised by a circle in this example). The device also offers quantization tools that constrain symbols' movement along the X axis according to rhythmic divisions and the Y axis based on vertical divisions.


3

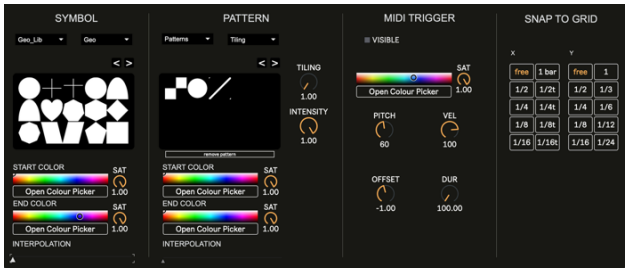


**Figure 5**. SIGNe-Symbol.amxd device properties

SIGNe-Symbol.amxd and SIGNe-Text.amxd contain MIDI trigger functionality, which extends the interactive potential of each symbol, linking visual elements to sonic events in real time. A MIDI trigger is a point that moves together with a symbol or text object in the graphical score. When a trigger passes the playhead, the corresponding device emits a MIDI note. By default, a MIDI trigger is invisible, but it may optionally be visualized as a vertical line. A trigger always moves in sync with its symbol, but may be displaced horizontally relative to the symbol, allowing manual calibration of the timing between the visual movement of the symbol and its MIDI output. A dedicated part of the device menu allows detailed editing of MIDI trigger parameters, including pitch, velocity, duration, and visual attributes.

Common parameters like rotation, scale, and opacity can also be modified using mouse and keyboard shortcuts for a more efficient workflow.

The SIGNe-Text.amxd device allows you to add text to the score (Figure 6). Text can be used for annotating the score, for passing indications to musicians, to create text scores, or to use text as a purely visual element. The device offers similar position and parameter controls to SIGNe-Symbol.amxd, but includes specific text editing options such as font selection, letter spacing, and the ability to manually adjust the bounding box used for mouse selection.

Finally, the SIGNe-Background.amxd (Figure 7) device offers the option to add a background image that can serve as a score template or other visual reference. The image can be set as fixed or configured to scroll in sync with the other visual elements via automation or by following the session transport.


**Figure 6**. SIGNe-Text.amxd device

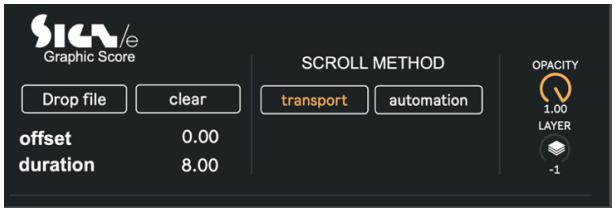


**Figure 7**. SIGNe-Background.amxd device

### 4.5. Integration of Custom Graphics

SIGN/e includes a default library containing several categories of useful notation symbols, including simple geometric shapes, arrows, and lines. Additionally, SIGN/e supports importing user-defined graphic elements, which can be added by placing PNG files in the USER directory within the application's LIBRARIES folder. Recommended specifications include a resolution of 300 dpi and dimensions of approximately 500 × 500 pixels to maintain visual quality when scaling graphical elements. This feature allows composers to personalize their scores and expand the visual vocabulary beyond the default libraries.

## 5. SOFTWARE FRAMEWORK

### 5.1. Development Environment and Design Principles

SIGN/e is implemented entirely using only standard objects included in the official Max distribution. While bringing some limitations, this choice was made to ensure long-term maintainability and cross-platform stability, avoiding dependencies on third-party externals. Core operations such as filename parsing and path construction rely on CPU-based Max objects combined with the Live Object Model, which provides access to Ableton Live's internal state.

### 5.2. Rendering Pipeline and Visual Architecture

SIGN/e utilizes Max's Jitter framework to establish an OpenGL rendering context. Although the score is visualized in two dimensions, the system operates within a 3D space using orthographic projection (a method of rendering a 3D scene in 2D that involves drawing the scene via rays that are perpendicular to the camera plane and parallel to each other, as opposed to having all the rays converge at a single point, as in a pinhole camera or eye) to eliminate perspective distortion. This architecture leverages Jitter's robust feature set, providing native support for geometric primitives, meshes, and procedural text, along with intrinsic parameters for position, rotation, scale, colour, and blend modes. By enclosing these objects in Max for Live devices, the system links Jitter's visual manipulation features directly to Live's automation, undo history, and save systems.

To simulate a strict 2D environment, a custom mouse interaction scheme restricts transformations to a camera-facing plane. A pure 2D fragment shader approach was explicitly discarded; unlike Jitter's object-oriented structure, a fragment shader-based system would require dynamically creating parameters and shader inputs for each new


4

graphical object via the Max API. This method was rejected to avoid instability and ensure reliable data retention.

All graphic elements are positioned in a plane perpendicular to the 3D camera. The camera moves horizontally in synchronization with Ableton Live's transport, ensuring that spatial coordinates correspond directly to musical time. While standard object properties are controlled via OpenGL, advanced features, such as grid generation, time rulers, and playheads, are implemented using shaders written in Max's Gen language, which compiles to GLSL for efficient GPU execution.

### 5.3. Parameter Management and Automation

Compositionally relevant parameters are exposed through Max for Live User interface (UI) objects, allowing integration with Ableton Live's automation system. This integration provides several benefits: parameter changes are recorded in real time, stored within Live sessions, and included in the undo stack. Automation data can be authored manually or generated algorithmically, then edited post-recording. For data types unsuitable for direct storage via UI objects, such as strings, the system employs Max's pattr preset mechanism, which interfaces with Live's parameter storage framework.

### 5.4. Modular Architecture and Device Structure

SIGN/e employs a modular architecture that distributes visualization and control across distinct devices. This strategy reflects established practices in 3D content creation and game development where 3D objects are represented as discrete nodes; when selected, the user can edit the parameters of the corresponding object. By allocating one device per symbol, SIGN/e maintains organized automation data and enables scalable resource management. Although this methodology may introduce complexity in extensive projects, it adheres to workflows commonly found in advanced graphics software.

### 5.5. Symbol Representation and Mesh Design

Symbols are rendered as textures mapped onto geometric meshes stored in FBX format, with UV coordinates configured for camera-facing display. Initially limited to quads, the mesh set was expanded to include circles, triangles, and diamonds to improve mouse interaction and sorting when multiple symbols overlap in space. Textures support alpha channels allowing non-rectangular shapes to be displayed without revealing the mesh shape, and associations between meshes and textures are currently hard-coded.

### 5.6. Graphical interface interaction

Jitter's native system for mouse and keyboard interaction, particularly through the jit.gl.handle object, is not optimized for pseudo-2D objects. The absence of axis-locking for rotation makes it nearly impossible to rotate an object exclusively around the axis perpendicular to the camera, often resulting in off-axis orientations. Furthermore, mouse selection in jit.gl.handle relies on checking the mouse's position in screen coordinates against an object's world-aligned bounding box, which is always a quad aligned to the render context's axes. When two non-square symbols with alpha channels are placed close together, overlapping bounding boxes can cause incorrect object selection, leading to a frustrating user experience.

To address these limitations, a custom interaction system was developed using Jitter's physics engine to generate collision volumes that approximate the actual shape of the meshes used for rendering symbols. This approach significantly improves selection accuracy and introduces support for axis-locked 2D rotation, providing a more intuitive and reliable interaction model for score editing.

### 5.7. Performance and optimization

Given that SIGN/e is intended for use in real time presentation scenarios (i.e. concerts), performance is a serious consideration. While the current design that relies on a one-to-one relationship between Max for Live devices and graphical objects is convenient from a UX and development perspective, this architecture has limitations in terms of performance. Each instance of a Max for Live device in a Live session requires a certain amount of available memory, with more complex devices requiring more memory. In their current state, each SIGN/e-Symbol device occupies about 50 MB of memory. Since this memory use scales linearly as more devices are added to the set, it imposes a hard limit on the number of symbols that can exist within a given score before performance degrades.

Given the memory limitations of a mid-range laptop, SIGN/e can approximately accommodate 300 SIGNe-Symbol.amxd devices. Hence, the users must work within these limits, e.g., use a limited number of symbols per session; use the fixed background image feature to reduce RAM usage; or separate a score into multiple sessions. These limitations vary according to the available RAM of the computer used. We are currently exploring several optimization strategies for the next development stages.

## 6. CONCLUSIONS

SIGN/e is designed for artists (composers, musicians, designers) who want to create graphical music scores within the Ableton Live architecture using a simple and accessible interface. The simplicity and accessibility of the proposed tools aim to make graphical notation available to creators who are not familiar with often complex and costly graphics editing applications such as Adobe Illustrator or Affinity Designer.

While we understand that SIGN/e's exclusive integration with Ableton Live could be seen as a drawback for non-Ableton Live users, we believe that integration with a widely-used DAW is key to contribute to the democratization of graphic scores. By embedding scoring tools into a program that a critical mass of musicians uses every day, SIGN/e should appeal to a community that may not typically write scores or possess a background in contemporary notation. This immediacy removes the friction of switching contexts; users who might never seek out

standalone scoring software may be compelled to experiment with graphical notation simply because it is available within their native production environment.

Furthermore, the modular, device-based approach chosen for SIGN/e provides the opportunity to add more devices in the coming years and allows users to create custom ones that can be integrated into the SIGN/e architecture, thus making the application customizable and virtually endlessly expandable to meet specific needs.

By providing an accessible and flexible environment for creating animated graphic scores, the system bridges the gap between conventional notation tools and the creative demands of graphics-oriented compositions. Its architecture not only simplifies the process of score creation but also allows dynamic interaction between sound and image, supporting both written and improvisational approaches.

Future developments will focus on extending the device library, enabling user-defined meshes, and improving interaction models to further enhance customization and usability. Through these iterations, SIGN/e aims to contribute to the evolution of performative notation and to promote new forms of collaborative and experimental music-making.

**Acknowledgments**

This article is part of the Performative Notation project directed by Nicolas Bernier and funded by Fonds de recherche du Québec – Société et culture (FRQSC), a research project held at Laboratoire formes • ondes (LFO) at Université de Montréal between 2024 and 2027. Pierre-Luc Lecours would like to thank the Social Sciences and Humanities Research Council of Canada (SSHRC) for funding his doctoral research between 2021 and 2024 and the Centre for Interdisciplinary Research in Music, Media, and Technology (CIRMMT).

## 7. REFERENCES

1. N. Bernier, “Observations on Performing Sine Waves with an Oscillator Ensemble,” *Leonardo*, vol. 55, no. 2, pp. 161–165, 2022.

2. P.-L. Lecours and N. Bernier, “Sound Synthesis Notation Applied to Performance: Two Case Studies,” in *Proc. of the Int. Conf. on Technologies for Music Notation and Representation – TENOR'24*, Zurich University of the Arts, 2024, pp. 142–149.

3. J. Bresson, C. Agon, and G. Assayag, “OpenMusic: Visual programming environment for music composition, analysis and research,” in *Proc. of the 19th ACM Int. Conf. on Multimedia*, Scottsdale, Arizona, 2011, pp. 743–746.

4. H.-W. Nienhuys and J. Nieuwenhuizen, “LilyPond, a system for automated music engraving,” in *Proc. of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, vol. 1, 2003, pp. 167–171.

5. G. Hajdu and N. Didkovsky, “MaxScore: Recent developments,” in *Proc. of the Int. Conf. on Technologies for Music Notation and Representation – TENOR*, vol. 18, 2018, pp. 138–146.

6. 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.

7. C. Hope, L. Vickery, A. Wyatt, and S. James, “The DECIBEL Scoreplayer—A Digital Tool for Reading Graphic Notation,” in *Proc. of the First Int. Conf. on Technologies for Music Notation and Representation – TENOR'15*, 2015, pp. 58–69.

8. T. Coduys and G. Ferry, “Iannix aesthetical/symbolic visualisations for hypermedia composition,” in *Journées d'Informatique Musicale*, 2004.

9. J.-M. Celerier, P. Baltazar, C. Bossut, N. Vuaille, J.-M. Couturier, and M. Desainte-Catherine, “OSSIA: towards a unified interface for scoring time and interaction,” in *Proc. of the First Int. Conf. on Technologies for Music Notation and Representation – TENOR'15*, 2015, pp. 81–90.