THE MUSIC PERFORMANCE MARKUP FORMAT AND ECOSYSTEM

Axel Berndt
Center of Music and Film Informatics
Ostwestfalen-Lippe University of Applied Sciences and Arts
Detmold University of Music
axel.berndt@th-owl.de

ABSTRACT

Music Performance Markup (MPM) is a new XML format that offers a model-based, systematic approach to describing and analysing musical performances. Its foundation is a set of mathematical models that capture the characteristics of performance features such as tempo, rubato, dynamics, articulations, and metrical accentuations. After a brief introduction to MPM, this paper will put the focus on the infrastructure of documentations, software tools and ongoing development activities around the format.

1. MOTIVATION

The performance of a musical piece transforms the musical raw material (typically a symbolic representation such as Common Western Music Notation) into a sounding output or equivalent audio signal. One and the same piece of music can be performed in many different ways. Playing the raw material exactly as notated is one special kind of performance, often referred to as "robotic" or "machine-like". Human musicians' performances typically involve more complex transformations that only partly derive from the notation. Those are subject to active research in musicology, in particular in the fields of performance research, Historically Informed Performance Practice, and Music Information Retrieval.

Typical questions are: How can the connection between audio document and musical score be drawn? How can the musical realization of a performer be described and put in relation to printed performance instructions and the performances of other musicians? Especially in the light of the musical culture of the past century, which has been strongly coined by audio media, this research field gains in importance and demands more and more urgently for suitable tools. Yet there is still a lack of a common, open data standard that would allow systematic and comprehensive access to the phenomena of music performance and facilitate the publication and re-use of research results in the variety of application and research contexts.

Based on several years of prior basic research and model development, the Music Performance Markup (MPM) format was developed to address this problem. MPM enables several novel research designs. E.g., the "simulation" or "reconstruction" of a performance allows the experimental testing of auditory impressions and hypotheses about how a performance was precisely made and by which expressive means a certain effect was achieved.

However, in order for MPM to become an accepted and useful tool in the communities, we have made further efforts beyond the mere definition of the format. An ecosystem of software tools, guidelines, sample encodings and tutorials was developed of which this paper will report, thus, providing an overview and good entry point for new users.

2. DESCRIBING MUSICAL PERFORMANCES

How can a musical performance be described? The description of acoustic phenomena on a colloquial level is problematic in scientific discourse. A subjective listening impression may be described as a "strong ritardando from measure x to measure y", but lacks the necessary precision in several respects, particularly in terms of initial and final tempo, initial and final timing position, and the specific course of the tempo transition. Another performer may play a ritardando at the same musical position that is just as strong but completely different in its execution and effect. On the other hand, a precise tempo curve may be derived from timing measurements in the audio data. This would usually render a very noisy tempo curve, because it is the sum of several timing features (tempo, rubato, asynchrony, agogic accents, and unsystematic elements often referred to as human imprecision) and measurement imprecision.

These two opposing perspectives can be found in all the formats used in music description and performance research. Accordingly, they can be roughly divided into (1) measurement series formats and (2) symbolic formats.

The most extreme form of a measurement series format is certainly the audio recording itself. Without a considerable effort of analysis, however, abstract performance concepts such as tempo, rubato, dynamics, etc. cannot be inferred from it. This is therefore a low-level representation, which by itself does not provide any information about higher-level structures in the series of measurements.

The first step in inferring volume and tempo characteristics is audio signal analysis. Note onset positions are determined algorithmically or manually, inter-onset intervals are measured, spectra are evaluated, amplitude curves are

generalized to envelopes, peak and sustain levels of individual tones are measured, etc. Examples of such analyses and the resulting complex series of measurements can be found in performance research publications [1]. A tool widely used in empirical performance research is the Sonic Visualiser [2]. The resulting measurement series are often stored in CSV (Comma-Separated Values) format [3]. An extensive data collection in this format is the Mazurka-aBL dataset with roughly 2000 sound recordings, annotated with loudness and tempo information [4]. The CSV format is often used for comparing analysis algorithms, such as in MIREX [5, 6].

For more complex data structures that can no longer be represented in CSV, the JSON format is a popular alternative [7]. It is easy to process, human readable and is frequently used to relate measurement series to each other, such as in audio-to-audio and audio-to-score alignment [8].

In contrast to the very detailed data of measurement series formats, which provide no immediate information on high-level structures, are the symbolic formats. They describe the abstract structures, but leave out any specific detail. Although they enable efficient communication about music, listening impressions and performance concepts, which for the most part suffices everyday use, they lack the aforementioned precision required in scholarly discourse. To take up the above example: When does a ritardando become a "strong" ritardando? The degree of abstraction becomes particularly striking when a computer generates music playback directly from the score and the result is described as "mechanical".

Symbolic formats are used mostly to encode music notation. Most notation software and Digital Audio Workstations (DAWs) use proprietary formats that are optimized for their particular needs. Open formats are MusicXML [9], ABC [10], and Lilypond [11]. In addition, specialized formats such as Humdrum [12] and MEI [13] have become established in musicology and music editing [14].

Compared to other symbolic formats, the Standard MIDI File format [15, 16] has a special position. It represents musical information as control messages. Each note is represented by a NoteOn-NoteOff pair. The volume of notes is specified by numerical values which are interpreted by synthesizers and converted into actual amplitude values. Temporal indications are made on a tempo-independent grid similar to the musical time measure and are converted into physical time values (milliseconds) only in connection with tempo messages. Also other parameters of musical expression (tonality, mixing, DSP effects etc.) can be implemented as a series of controller messages. However, meta-structures in those domains are not represented, so that MIDI clearly corresponds to a measurement series format in this respect and is also practically used for such purposes. An example for this is CrestMusePEDB, a database of several hundred piano performances [17]. Such recordings are typically made with MIDI-fied instruments. The relatively low numerical resolution of MIDI's 8 and 7-bit numerical values and the fact that volume and controller values are interpreted differently by each synthesizer [18] are two main criticisms of the format, which is nevertheless indispensable in today's practice.

MEI and Humdrum, too, can be extended into hybrids, as proposed by Devaney and Gauvin [19]. This is achieved by augmenting the symbolic data by measurement data, e.g. timestamps for notes. However, these measurements are not decoded and linked to corresponding larger structures leaving the gap between both perspectives open.

The essential achievement of MPM is to combine low-level and high-level perspective. Here, the high-level description serves to systematically break down the complex interplay of various performance concepts and features, which manifests itself in such detailed measurement series. This descriptive approach opens up new perspectives for a broader, scientific discourse on music performance.

3. A BRIEF INTRODUCTION TO MPM

MPM is designed as a tandem partner or complement for symbolic music formats such as MIDI, MusicXML and MEI. While these encode the score to be interpreted, MPM describes its musical performance. MPM chooses a model-based approach for this. Each description primitive is based on a mathematical model that emulates its corresponding performance feature. For a detailed introduction to these models see [20]. Currently, the corpus of models defined in MPM and supported by fundamental research comprises the following feature types:

Timing: tempo (incl. discrete and continuous tempo changes), rubato, asynchrony, random/unsystematic deviations from exact timing,

Dynamics: macro dynamics (incl. discrete and continuous dynamics changes), metrical accentuations, random/unsystematic deviations from exact dynamics,

Articulation: with absolute and relative effects on tone duration, loudness, tuning and timing (e.g. agogic accents) as well as random/unsystematic fluctuations of duration and tuning.

These models enable MPM to bridge the previously described gap between the two opposed approaches (measurement series versus symbolic representation) as they disambiguate the high-level concepts by explicit mappings. Conversely, they can also be used for resynthesis.

MPM allows to define several performances for one and the same piece of music. In the basic structure of such a performance a first peculiarity of MPM's conception becomes apparent. Performance instructions can be defined *globally* for all parts and *locally* for a single part. If local and global information of the same domain compete, the local dominate. This makes it possible to describe polyphonic performances in which, e.g., a solo instrument has its own freedom of expression while the accompanying orchestra follows the global instructions of a conductor.

Both, the global and part environment, subdivide into *header* and *dated* information. The header contains style definitions, i.e. lookup tables to map literals ("Allegro",

```
<?xml version="1.0" encoding="UTF-8"?>
<mpm xmlns="http://www.cemfi.de/mpm/ns/1.0">
  <performance name="a performance"
               pulsesPerQuarter="720">
    <global>
      <header>
        <tempoStyles>
          <styleDef name="famous conductor">
            <tempoDef name="Allegro"
                      value="133.0"/>
            <tempoDef name="Adagio"
                      value="67.4"/>
          </styleDef>
        </tempoStyles>
      </header>
      <dated>
        <tempoMap>
          <style date="0.0"
                 name.ref="famous conductor"/>
          <tempo date="0.0" bpm="Allegro"
                 transition.to="Adagio"
                 meanTempoAt="0.7"
                 beatLength="0.25"/>
          <tempo date="28800" bpm="87.45"
                 beatLength="0.25"/>
        </tempoMap>
        <dynamicsMap>
          <dynamics date="0.0" volume="80.0"/>
        </dynamicsMap>
      </dated>
    </global>
    <part name="Solo Violin" number="1"
          midi.channel="0" midi.port="0">
      <header/>
      <dated>
        <dynamicsMap>
          <dynamics date="0.0" volume="92.0"/>
        </dynamicsMap>
      </dated>
    </part>
  </performance>
</mpm>
```

Listing 1: A short MPM code example.

"mf", "staccato" etc.) to numeric values. The *dated* environment, on the other hand, is the place where performance instructions are specified and assigned to metrical positions (corr. MIDI ticks) and musical elements (e.g. articulations to notes via XML IDs). These instructions are organized in sequential lists, so-called *maps*, one for each feature type (e.g. tempoMap, dynamicsMap, rubatoMap, metricalAccentuatioMap). Listing 1 demonstrates this structure. The parameterization of the style definitions and performance instructions derives from their underlying mathematical models and translate to attributes in the XML encoding.

To give an impression of one of MPM's feature models, the code example involves a tempo slowdown from "Allegro" to "Adagio". The course of tempo curves is modelled with power functions in the interval $[0.0, 1.0]$. Attribute "meanTempoAt" specifies the relative position between start and end date of the transition where the curve passes the mean tempo, in this case after 70% of the time frame (0.7). Detailed documentation of all features, syntax and models is given on the official website. [1]

For music editions, MPM is a tool for the philological registration and critical examination. This also leads directly to use cases in library and archiving contexts. For the analysis of individual performances, the model-based approach of MPM provides feature classes that enable an abstracted and differentiated review. MPM does not primarily serve a purely positivistic quantification of music. Coupled with the possibilities of applying the modeled performances to symbolic music data to output them as expressive MIDI and audio, it offers, in the sense of *transformative digital intermedia studies* [21] a tool for a hermeneutic approach to performance analysis. As an experimental tool, it can serve to (re)construct performances that have not survived as audio documents but in textual form, e.g. in music-practical treatises and performance scores. Beyond a purely scientific use, this also shows potential for application in digital music production.

The model-based description approach of MPM also motivates new research questions that could not be systematically addressed so far. For instance, the sharp differentiation of timing, dynamics, and articulation into several subcategories and their description by means of mathematical models had led to questioning the common understanding of inégalité (the unequal playing of notes of equal value) as a pure timing feature [22, 23], which is of relevance for Historically Informed Performance Practice. Studies based on the models were eventually able to demonstrate a multifaceted interplay of micro-timing shifts, accentuation, and changes in tone duration [24]. Further research questions address, e.g., playing inaccuracies beyond classical timekeeping and synchronization studies (incl. variations in loudness, intonation, and articulation), the interaction of an ensemble depending on exterior conditions (such as acoustics and mutual visibility), and related questions of timbre research. In addition, analyses on larger corpora of performances can lead to new and more differentiated insights into music-historical change processes and formative characteristics of individual performers and schools.

4. DEVELOPING AN ESCOSYSTEM FOR MPM

The primary application areas of MPM are musicological edition and performance research as well as computer-aided music production. The development of tools for the creation, analysis, presentation and further use involve also computer science and especially MIR are further application areas. When designing the ecosystem around MPM, the goal was to reduce the barriers to adoption and productive use of MPM as much as possible.

4.1 Supporting Work in XML Editors

In the domain of digital music editing, working with XML code and dedicated editors such as Oxygen XML is common practice and requires a detailed format documentation and guidelines. This work is supported by convenience tools, namely code completion and live validation. Both require an appropriately comprehensive schema definition. The MPM schema including its documentation was speci-

[1] https://axelberndt.github.io/MPM/, last access: July 2021.

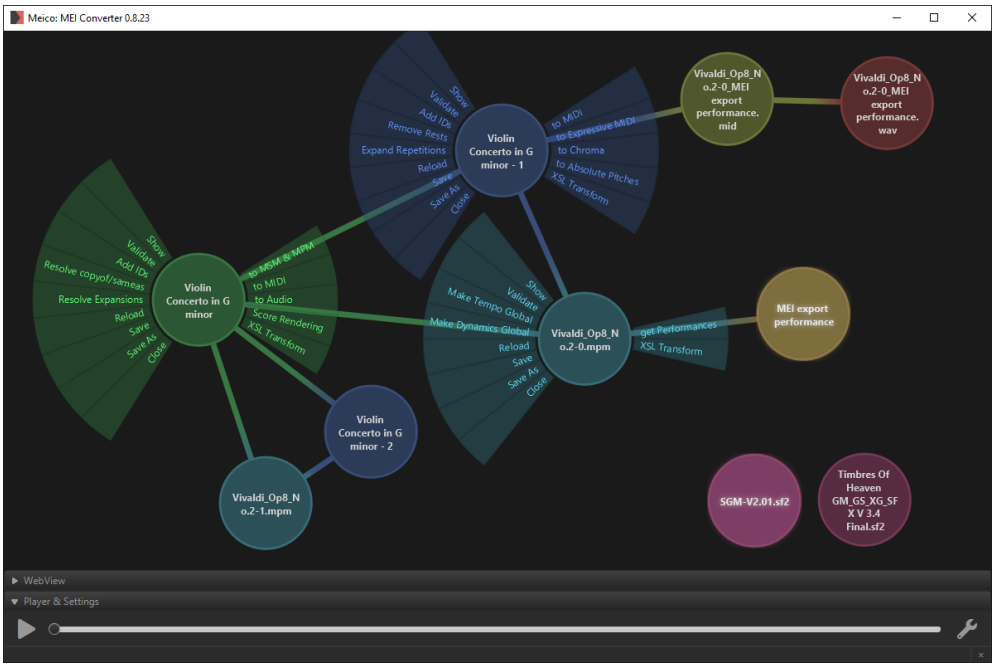

Figure 1. Screenshot of the meicoApp. From the MEI data (left, green) MSM (dark blue) and MPM (light blue) are exported. The MPM here contains one performance, "MEI export performance", from which the MSM can be rendered "to Expressive MIDI" (top, yellow). The MIDI data is rendered to audio (top right, red). At the bottom right, two external soundfonts are included, the left one is activated.

fied in the TEI ODD [2] meta-language, which can be compiled into several other schema languages, such as RNG, XSD and DTD. This provides a maximum compatibility with all commonly used XML parsers. In addition to the purely syntactic validation, several Schematron rules enable content-based validation. Typical errors, such as missing references, value range violations, and invalid value combinations, are detected during validation and communicated by meaningful warning and error messages.

The documentation is supplemented by several sample encodings, i.e. example projects with which users can experiment. A continuous integration and custom-developed transformer translate updates to the MPM schema immediately into the website (documentation and guidelines) and release assets (incl. an RNG and XSD compilation). The format is published under the open source licenses BSD-2 and CC-BY-4.0 and is open to future extensions from the communities.

4.2 Software Development Tools

The basis for efficient software development for a data format and its integration into existing software is the Application Programming Interface (API). The API enables convenient data access and processing (e.g. by means of predefined data structures and functionality for parsing, creating, processing and storing) without the application developers having to implement the underlying XML processing or mathematical models themselves. Thus, the API also represents a reference implementation. The MPM API was implemented in Java as part of the converter framework *meico* [3] [25]. Meico is an established conversion tool in the music encoding community and comes with some features that mesh well with MPM-related functionality.

In particular, meico offers the currently most comprehensive MEI-to-MIDI export. Therefore, it utilizes a proprietary intermediate format, Musical Sequence Markup (MSM). Its basic structure (global/part, header/dated) parallels that of MPM. In MSM all note information (without performance data) is represented. Meico's MEI-to-MSM export was extended to convert all performance instructions to MPM data. MIDI data can also be converted to MSM. Thus, via MSM, MPM can already be used in tandem with MEI, MIDI and MSM itself.

Furthermore, a full-featured *performance rendering engine* has been integrated into the MPM API. This allows the MPM performances to be applied to MSM-encoded scores and rendered into expressive MIDI sequences. These can then be played directly in meico, exported as MIDI files and converted to audio (WAV, MP3). For MIDI playback and MIDI-to-audio rendering, third-party soundfonts (SF2, DLS) can be used or MIDI playback can be passed on to an external MIDI port. Thus, further processing (e.g. by external synthesizers) and music production in a DAW are immediately feasible.

However, meico is not only a programming library, but also provides two application programs in the form of the *meicoApp*. The command line application is integrated by some users into their XML editor as an external call (e.g. for proof-listening of music encodings) and is also used

[2] `https://wiki.tei-c.org/index.php/ODD`, last access: July 2021.

[3] `https://github.com/cemfi/meico`, last access: July 2021.

in scripting environments (e.g. in Python programs). The graphical application is used as a stand-alone tool and offers more interaction possibilities as well as more versatile conversion paths, see figure 1. The meico library is also the core component of the following software tool.

4.3 Graphical Editor and Analysis Tool

For end users, work with MPM should not be restricted to XML editors, even if this is principally always possible. Rather, productive authoring and analysis work is to be facilitated by an efficient, graphical user interface. This motivated the development of the MPM Toolbox application. We followed a rather traditional user interface engineering process. First, paper mockups were used to try out different interface approaches and play through usage scenarios. This was complemented by experiences and feedback from a workshop with musicologists and editors during Edirom Summer School 2020. This resulted in the following observations and corresponding design decisions.

1. A schematic representation of the MPM data must be present for navigation purposes, and it must also be fully interactive. But the visual focus is on the graphical score.

2. The MSM data, i.e. the pure note information, is needed for matching time indications, voice assignments and references, so it must also be represented. However, since the focus is on work in the performance domain, the MSM does not need to be interactive, nor should it take up much display space.

3. The interaction is to be focused on the score display. This is where new performance instructions are created and performance instructions that already exist in the MPM are positioned. This is done in two different application contexts, (a) the free creation of performances (creative use case) and (b) the analysis of performance scores, i.e. the interpretation of signs in the autograph (analytical use case).

4. Not all attributes of performance instructions can and should be visualized in the score, as this quickly overloads the display space and impairs readability. Consequently, dedicated editor dialogs are needed for creating and editing them, which may employ their own visualizations to illustrate the values set.

5. The visual placement of performance instructions in the score is of limited value with respect to their assignment to musical parts and temporal placement. For instance, an instruction may clearly precede a note, but apply only from that note on.

In accordance with the first two points, the MSM and MPM data were each implemented in a tree visualization, placed fairly slim on the left and right border of the application window. These interface widgets can be both minimized and detached from the layout, allowing users to freely place them and arrange their workspace. The MPM tree is fully interactive. So it is possible to do all work directly in it. For some types of information, such as style definitions, there is no visual representation in the musical notation practice anyway. These can only be located in the MPM tree.

The third point presented a particular challenge. While a generated score image would suffice for the creative use case (3a), the analytical use case (3b) involves working with a preexisting score image. In addition, the initial plan was only to generate the score image from MEI data using the Verovio music engraving software [26], which practically excludes some other input sources which MPM Toolbox should also support. Therefore, it was decided that the score image will be imported as image data (currently supported are JPG, GIF, PNG, BMP, and PDF). Those can be generated from all other formats with widely available tools and converters. Thus, the MPM Toolbox satisfies both use cases (3a and b) equally. A built-in score rendering solution may, nonetheless, be added later on.

For interaction in the image space, it is necessary to know the positions of notes and performance instructions and to link them to the MSM and MPM data. This is an opportunity to incorporate Optical Music Recognition (OMR) techniques. Depending on the condition of the autograph (clean print versus handwritten manuscripts with many additional markings), the recognition performance will be of different quality. Therefore, it must always be possible to perform or correct this work manually. Consequently, this manual linking was implemented with an efficient input procedure. It automatically iterates over the notes (MSM) and plays them while the user marks their position in the score image; the same for MPM. An OMR solution remains open as a future extension or third party contribution.

The linking of the note and performance information in the image space creates a point cloud. If, in the further course, performance instructions are created or linked in the image space, these can be set in relation to the surrounding elements and added to the point cloud. In doing so, voice assignment and time position are read from the surrounding linked elements and set as default values. This eliminates the need for additional user input in most cases. According to point 5, however, this can nonetheless be changed in the editor dialog.

On large and complex score page, such a point cloud can become very extensive. During interactions in the score, the points located in the local environment of the cursor must regularly be determined. To ensure smooth interaction, the point cloud must be organized in an efficient data structure. For this purpose, different standard approaches (such as BSP Trees and Quadtrees) were analyzed. A peculiarity of the application context here is that interactions mostly take place in a local environment, are not scattered "chaotically" across the music sheet, but often even follow the musical sequence. The next interaction is very likely to occur near the position of the previous. An efficient data structure for this type of interaction is the Orthant Neighborhood Graph [27], which was eventually implemented in the MPM Toolbox.

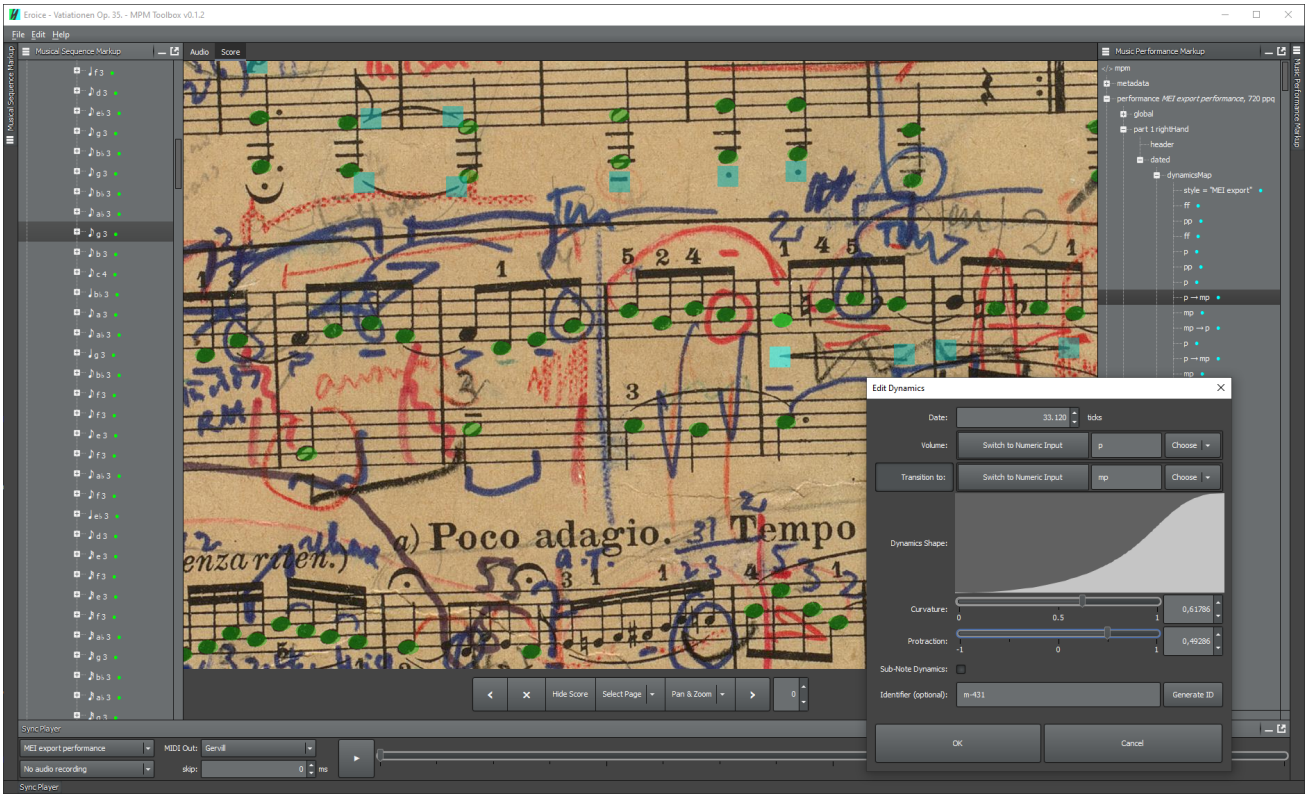

Figure 2. Screenshot of the MPM Toolbox. The green note overlays link notes to pixel positions. The blue squares link performance instructions. In the dynamics editor dialog the course of a crescendo is specified.

According to point 4, it is not practical to display all the information of the performance instructions directly in the score, since it is usually printed very compactly anyway and the remaining space is important for readability. For this reason, the annotations created with MPM Toolbox are implemented as semi-transparent and compact overlays. The detailed settings of the performance instructions are done in separate editor dialogs. Each of these dialogs not only provides the respective input options and visualizations, but also actively ensures that the input is valid, i.e. validates against the MPM schema. Numerous smaller convenience functions support the user in this process.

Of course, the created performances can also be listened to via the built-in player widget, sent to an external MIDI port (synthesizer, DAW) and exported as expressive MIDI and audio file. The player widget further allows the playback of audio recordings. Both, performance rendering and audio recording can even be played synchronously. This feature serves the purpose of listening analysis. A performance description can be iteratively developed and adjusted to approximate the listening impressions of the audio recording.

Figure 2 shows a screenshot of MPM Toolbox's graphical user interface. Source code and executable release assets are published under the GNU GPL 3.0 license. [4]

[4] `https://github.com/axelberndt/MPM-Toolbox`, last access: July 2021.

5. SUMMARY AND NEXT STEPS

MPM allows for the creation of highly detailed music performances. The format is accompanied by a comprehensive schema definition, documentation and sample encodings. An API, including converter and performance rendering engine, provides the basis for efficient software development around MPM. For example, the API is already in use as a generator for expressive music performances. For productive authoring and analysis work apart from XML editors, the graphical editor software MPM Toolbox was developed.

As a supplement to the MPM documentation, as well as a practical introduction to working with MPM Toolbox, a tutorial project is currently being conceived. In addition, MPM Toolbox will be supplemented by a comprehensive module for performance analyses in audio recordings. Options for interoparability with Sonic Visualizer are being investigated, e.g. import of onset detection data.

We offer introductory workshops for users and collect feedback on possible enhancements from the community regarding new performance features for MPM as well as its tools. The MPM project is open source and welcomes suggestions and contributions from the communities.

Acknowledgements: The author wishes to thank Tilo Hähnel, Benjamin W. Bohl, Simon Waloschek, and Peter Stadler for their support and contributions. This project is funded by the Fritz Thyssen Foundation.

6. REFERENCES

[1] H. von Loesch and S. Weinzierl, Eds., *Gemessene Interpretation: Computergestützte Aufführungsanalyse im Kreuzverhör der Disziplinen*, Staatliches Institut für Musikforschung Preußischer Kulturbesitz. Mainz, Germany: Schott, 2011.

[2] C. Cannam, C. Landone, M. B. Sandler, and J. P. Bello, "The Sonic Visualiser: A Visualisation Platform for Semantic Descriptors from Musical Signals," in *7th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2006.

[3] Y. Shafranovich, "Common Format and MIME Type for Comma-Separated Values (CSV) Files," *RFC 4180 (Informational)*, Oct. 2005.

[4] K. Kosta, O. F. Bandtlow, and E. Chew, "MazurkaBL: Score-aligned loudness, beat, expressive markings data for 2000 chopin mazurka recordings," in *Proc. of the 4th Int. Conf. on Technologies for Music Notation and Representation (TENOR)*, Montréal, Canada, 2018, pp. 85–94.

[5] J. S. Downie, K. West, A. F. Ehmann, and E. Vincent, "The 2005 Music Information retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview," in *Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2005, pp. 320–323.

[6] J. S. Downie, "The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research," *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247–255, 2008.

[7] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," Internet Engineering Task Force (IETF), Tech. Rep., Dec. 2017, rFC 8259.

[8] S. Waloschek and A. Hadjakos, "Driftin' down the scale: Dynamic time warping in the presence of pitch drift and transpositions," in *Proc. of the 19th Int. Society for Music Information Retrieval Conf. (ISMIR)*. Paris, France: Int. Society for Music Information Retrieval, Sept. 2018.

[9] M. Good, "MusicXML specification," https://github.com/w3c/musicxml [last access: May 2021], Dec. 2017.

[10] C. Walshaw, "The ABC Music Notation," http://abcnotation.com/, last access: May 2021, 2017.

[11] 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)*, Firenze, Italy, May 2003, pp. 167–171.

[12] D. Huron, "Music Information Processing using the Humdrum Toolkit: Concepts, Examples, and Lessons," *Computer Music Journal*, vol. 26, no. 2, pp. 11–26, 2002.

[13] A. Hankinson, R. P., and I. Fujinaga, "The Music Encoding Initiative as a Document-Encoding Framework," in *Int. Society for Music Information Retrieval Conf. (ISMIR)*. Miami, Florida, USA: Int. Society for Music Information Retrieval, Oct. 2011, pp. 293–298.

[14] A. Hadjakos, J. Iffland, R. Keil, A. Oberhoff, and J. Veit, "Challenges for Annotation Concepts in Music," *Int. Journal of Humanities and Arts Computing*, vol. 11, no. 2, pp. 255–275, 2017.

[15] R. A. Moog, "MIDI: Musical Instrument Digital Interface," *Journal of the Audio Engineering Society (JAES)*, vol. 34, no. 5, pp. 394–404, 1986.

[16] MIDI Manufacturers Association, "The Complete MIDI 1.0 Detailed Specification. v. 96.1," MIDI Manufacturers Association, La Habra, CA, Tech. Rep., 1996.

[17] M. Hashida, E. Nakamura, and H. Katayose, "Constructing PEDB 2nd Edition: A Music Performance Database with Phrase Information," in *14th Sound and Music Computing Conf. (SMC-17)*. Espoo, Finland: Aalto University, July 2017, pp. 359–364.

[18] R. B. Dannenberg, "The Interpretation of MIDI Velocity," in *Proc. of the Int. Computer Music Conf. (ICMC)*. Tulane University, New Orleans, USA: International Computer Music Association, Nov. 2006, pp. 193–196.

[19] J. Devaney and H. L. Gauvin, "Encoding music performance data in Humdrum and MEI," *Int. Journal on Digital Libraries*, pp. 1–11, Oct. 2017.

[20] A. Berndt, "Formalizing Expressive Music Performance Phenomena," in *Works in Audio and Music Technology*, A. Berndt, Ed. Dresden, Germany: TUDpress, Sept. 2015, ch. 4, pp. 97–128.

[21] O. Eide, *Media Boundaries and Conceptual Modelling: Between Texts and Maps*. New York, NY: Palgrave MacMillan, 2015.

[22] J. J. Quantz, *Versuch einer Anweisung, die Flöte traversière zu spielen*. Bärenreiter, 1752, Reprint (1997), H. Augsbach.

[23] S. E. Hefling, *Rhythmic Alteration in Seventeenth- and Eighteenth-Century Music. Notes Inégales and Overdotting*. New York, NY: Schirmer Books, 1993.

[24] A. Berndt and T. Hähnel, "Studying Music Performance and Perception via Interaction," in *Works in Audio and Music Technology*, A. Berndt, Ed. Dresden, Germany: TUDpress, Sept. 2015, ch. 5, pp. 129–153.

[25] A. Berndt, S. Waloschek, and A. Hadjakos, "Meico: A Converter Framework for Bridging the Gap between Digital Music Editions and its Applications," in *Audio Mostly 2018: 13th Conf. on Interaction with Sound—Sound in Immersion and Emotion*, Glyndŵr University. Wrexham, North Wales, UK: ACM, Sept. 2018.

[26] L. Pugin, R. Zitellini, and P. Roland, "Verovio: A Library For Engraving MEI Music Notation Into SVG," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)*. Taipei, Taiwan: Int. Society for Music Information Retrieval, Oct. 2014.

[27] T. Germer and T. Strothotte, "The Orthant Neighborhood Graph: A Decentralized Spatial Data Structure for Dynamic Point Sets," *Communications in Computer and Information Science*, vol. 21, pp. 41–55, 01 2009.