A TOOLBOX FOR STORING AND STREAMING MUSIC-RELATED DATA

Kristian Nymoen
fourMs - Music, Mind, Motion, Machines
Department of Informatics
University of Oslo
krisny@ifi.uio.no

Alexander Refsum Jensenius
fourMs - Music, Mind, Motion, Machines
Department of Musicology
University of Oslo
a.r.jensenius@imv.uio.no

ABSTRACT

Simultaneous handling and synchronisation of data related to music, such as score annotations, MIDI, video, motion descriptors, sensor data, etc. requires special tools due to the diversity of the data. We present a toolbox for recording and playback of complex music-related data. Using the Sound Description Interchange Format as a storage format and the Open Sound Control protocol as a streaming protocol simplifies exchange of data between composers and researchers.

1. INTRODUCTION

In this paper we introduce a set of tools that have been developed for working with music-related data. Our goal with this software is primarily to provide a set of tools for researchers working with music-related body motion, but we also see the potential for using the tools in other research areas. We started working on the tools in 2008, and the development has continued over the last years together with our research on music and movement [1, 2, 3]. The need for a common method of storing and sharing data related to musical movement was discussed at a panel session at the International Computer Music Conference 2007 [4], and further emphasised at a seminar in May 2010 at IRCAM, Paris, where several researchers from around the world working with music and motion, and sound spatialisation were present. A common denominator for this seminar was to come closer to a scheme for describing spatio-temporal aspects of music. The tools we are presenting were revised after this seminar with the intention of making them easy to use for the research community.

Section 2 introduces previous research and gives an overview of why these tools are needed, and what has already been done in the field. In section 3, the different types of data we are working with are discussed. Section 4 introduces the tools. Finally, in section 5, we conclude and point out the future directions of the development.

2. BACKGROUND AND MOTIVATION

In our research on music-related body motion, we are often faced with situations where we want to study data from several devices at the same time. We will start this section by looking at two use cases that summarise some of the challenges in the field and the tools needed.

2.1 Use cases

a. The music researcher
A researcher is interested in studying the movement of a pianist by using an optical infrared motion capture system and record MIDI events from the piano. By themselves, the MIDI and motion capture data is trivial to record. However, synchronising the two, and being able to play them back later, or even scrubbing through the recording, keeping the MIDI-data and the motion capture data aligned, is not as trivial. Motion capture data is typically recorded at a sampling rate of 100–500 Hz, while the MIDI data stream is event driven and only needs to be stored each time a MIDI event takes place. Thus, using a common sampling rate for MIDI data and motion capture data would mean recording a lot of redundant data. The setup becomes even more complex when the researcher wants to record data from other sensors and audio/video as well.

b. The composer
A composer wants to develop a system for modifying sound in real-time. Let us say that the composer has hired a violin player who is wearing a position sensor and using a bow equipped with an accelerometer. She wants to develop a system that modifies the violin sound in real-time, based on output from the position sensor and the bow accelerometer data. Having the musician available at all times to perform can be expensive, as the musician would typically have to spend quite a lot of time waiting for the composer to make adjustments in the mapping between motion and sound. The composer would benefit from being able to record both the sound and the sensor data, and to play them back as a single synchronised performance.

Both of these examples show us that there is a need for a flexible system that is able to record different types of data from an arbitrary number of devices simultaneously. Further complexity is added when multiple representations of the same data is required. For instance, the researcher could be interested in the coordinates of the hands of a piano player in relation to a global coordinate system, but

also in relation to a coordinate frame defined by the position of the piano, or the center of mass in the pianist's body. The natural complexity of music introduces needs for various simultaneous representations of the same data.

Existing formats for working with these types of data have advantages and disadvantages, and there is no agreement between researchers on how to share music-related motion data. For motion capture data, the most widespread format is C3D. [1] Unfortunately, C3D does not allow for storing or synchronising music-related data and media. The Gesture Motion Signal [2] format has been developed to handle low level data in a musical context, but does not handle higher level data. The latter is handled well with the Performance Markup Language, [3] but this format does not meet our requirements when it comes to audio and video synchronisation.

An approach similar to or own has been implemented in OpenMusic [5]. Bresson et al. have implemented a solution for storing and streaming sound spatialisation data in the Sound Description Interchange Format (SDIF). This seems to be a promising solution, and we hope to keep collaborating on SDIF descriptors for spatio-temporal data.

2.2 GDIF

The Gesture Description Interchange Format (GDIF) has been proposed for handling the diversity of data related to music and motion [6]. The name GDIF might be somewhat misleading, as this is neither a format per se, nor is it limited to only gesture-related data. Rather, it is a concept and an idea for how data, and particularly data related to musical movement, can be described and shared among different researchers.

This concept includes a hierarchical structure, where the raw data (i.e. the data that one receives directly from the sensor or interface) is stored at the bottom layer. Above this layer is a so-called *cooked layer*, where certain processing has taken place. This can be anything from simple filtering or transformations, to more advanced analysis. Other layers may include segmentations or chunks [7] and even higher-level descriptors such as expressivity, affect and mood.

So far, GDIF development has been concerned with conceptual issues, and it has been up to the user to define how to implement storage and streaming. Some guidelines have been suggested, one of them being the approach implemented in the system we are presenting in this paper. We are using the Sound Description Interchange Format for storing and the Open Sound Control protocol for streaming GDIF data [4]. These formats will be presented in sections 2.3 and 2.4.

2.3 SDIF

The Sound Description Interchange Format (SDIF) was proposed by researchers at IRCAM and CNMAT and has been suggested as a format for storing GDIF data [4, 8]. This file format describes a sequence of time-tagged *frames*. Each frame consists of an identifier indicating what type of frame it is, the frame size, the actual data and zero-padding to make the frame size a multiple of eight bytes [9]. The frames are further structured into *streams*. These streams are series of frames, and all streams share a common timeline. Inside each frame, the actual data is stored as strings, bytes, integers or floating point values in one or more 2D matrices.

[1] `http://www.c3d.org/`
[2] `http://acroe.imag.fr/gms/`
[3] `http://www.n-ism.org/Projects/pml.php`

2.4 Open Sound Control

Open Sound Control (OSC) is a protocol for real-time audio control messages [10]. Conceptually, OSC shares many similarities with the SDIF format, as it describes a way of streaming time-tagged bundles of data. Each bundle contains one or more *OSC messages*, each message containing an *OSC address* and the actual data in a list format. The OSC address contains a hierarchical structure of human readable words, separated by slashes, making it simple to work with and share data between researchers and musicians (e.g. `/mySynth/pitch 120`).

3. DATA TYPES

We are working with many different sorts of data. Part of GDIF development is to define data types that are as generic and at the same time as well defined as possible. In other words, data types in GDIF recordings must be defined in such a way that they are open enough for different use, and at the same time detailed enough to leave little or no doubt about what sort of data that is contained in a GDIF stream.

Frames and matrices in SDIF streams are identified by a four letter type tag. This introduces some challenges when it comes to describing data. By convention, the first letter should be X for non-standard SDIF streams, leaving us with three letters to define the frame type and matrix type we are working with. Although it makes sense to distinguish between the two, our current implementation makes no distinction between the frame type and the matrix type. This means that the current system only allows a single data matrix inside each frame, and the frame automatically adapts the type tag from the matrix it contains. This has been sufficient in our use so far, but it would make more sense to let the frame type identify the stream (e.g. according to input device) and the matrix types define the data within each matrix (e.g. position, orientation, etc.).

For our matrix type tags, we have chosen to let the second letter determine the main data category, e.g. "P" for position data. The third letter denotes the dimensionality of the data, e.g. "2" if we are only tracking horizontal position. The fourth letter lets us know if the stream contains delta values of the original data. This number denotes derivative level, for instance "1" if the stream is the first derivative of the original data. This means that an XP32 matrix would contain 3-dimensional data, of the second derivative from the original position stream (i.e. acceleration).

We are sometimes interested in the absolute value of a vector, i.e. the length of the vector independent of the direction. This type of matrix is denoted by replacing the

third letter in the type tag with an "A". To formalise, this gives us the general case:

$$\mathrm{XPjd}[n] = \mathrm{XPj(d-1)}[n] - \mathrm{XPj(d-1)}[n-1]$$

$$\mathrm{XPAd}[n] = \sqrt{\sum_{i=1}^{j} \mathrm{XPjd}[n][i]^2}$$

and as an example, the specific case:

$$\mathrm{XP31}[n] = \mathrm{XP30}[n] - \mathrm{XP30}[n-1]$$

$$\mathrm{XPA1}[n] = \sqrt{\sum_{i=1}^{3} \mathrm{XP31}[n][i]^2}$$

where d denotes the derivative level, *n* denotes the frame index in a sequence of frames, *i* denotes the dimension index at frame *n*, and *j* denotes dimensionality of the stream.

In addition to streams describing position, velocity, etc., GDIF data types include everything from raw data from sensors to higher level descriptors. Table 1 displays a selection of the GDIF data types we are currently working with. A more complete list of data types can be found at the wiki that has been set up for GDIF and SpatDIF development. [4] It should be noted that these are our suggestions, and we welcome a discussion on these data types.

Table 1. A selection of GDIF data types.

Tag	Description
XIDX	Referring to a certain event in a series of events, e.g. triggering a sound sample from a sample bank.
XP30	3-dimensional position stream.
XP31	3-dimensional position stream. 1st derivative. (i.e. velocity calculated from position data)
XPA1	x-dimensional position stream. Absolute value of 1st derivative.
XOQ0	Orientation stream, four quaternion values.
XA30	3D acceleration stream. Used when working with systems that provide acceleration data as raw data.
1MID	MIDI stream, already defined in the SDIF standard
XEMG	Electromyography sensor input.
XMQ0	Quantity of motion stream.
XMA1	Area of motion stream. First derivative.

The system accepts all measurement units. However, we recommend using the International System of Units (SI) whenever this is possible. This will make it easier for researchers to share GDIF recordings.

4. IMPLEMENTATION

The tools presented in this paper are based on the SDIF tools in the FTM library, [5] mainly `ftm.sdif.write` for recording and `ftm.track` for playback [11]. They are implemented in Max as modules in the Jamoma [6] framework. These frameworks provide solutions for OSC and SDIF. The two main modules in the toolbox are the recording module and the playback module.

[4] `http://xdif.wiki.ifi.uio.no/Data_types`
[5] `http://ftm.ircam.fr`
[6] `http://www.jamoma.org`

The recording module, based on `ftm.sdif.write`, is designed for writing matrix-formatted data into separate streams in an SDIF file (Figure 1).

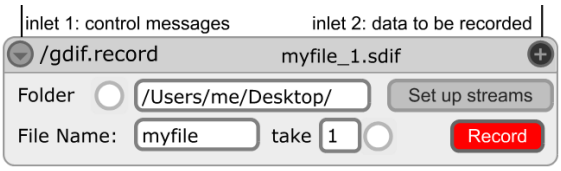

Figure 1. The record module

Different streams are separated by different OSC namespaces (e.g. `\stream\0`, `\stream\1`). The internal components of the recording module are created dynamically based on the user's selection of streams from a drop-down menu in the GUI. The user may customise the stream types that are available in the drop-down menu by editing a simple text file. Using a script language that is specific to the Max environment, stream definition commands and data descriptions are generated dynamically and sent to the `ftm.sdif.write` object whenever the user inputs a command or selects streams. The internally used OSC-routing objects as well as the `ftm.sdif.write` object are also created dynamically whenever the user chooses a different selection of data types. Figure 2 displays a simplified flowchart of how the record module works internally.

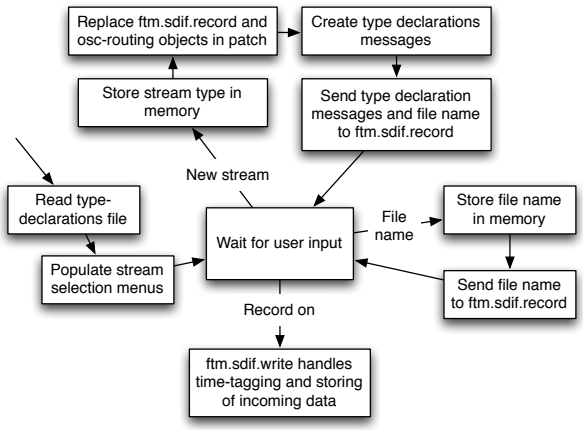

Figure 2. Simplified flowchart of the scripting system in the record module

The playback module displayed in Figure 3 is based on the `ftm.track` object. When an SDIF file is loaded into the playback module, an `ftm.track` object is created for each stream in the file. The data that is streamed from each track object is converted from the FTM float matrix format to Open Sound Control bundles using the OSC tools developed at CNMAT [10]. OSC does not support streaming matrices, hence each matrix row is separated as an instance number with its own OSC sub-address, e.g. first row gets the address `/XPOS/1`, second row `/XPOS/2`, etc. The user may set a custom buffer size for the OSC time tag to compensate for network latency and jitter. This buffer is set to a default value of 10 milliseconds.

The modules provide the user with a simple user inter-

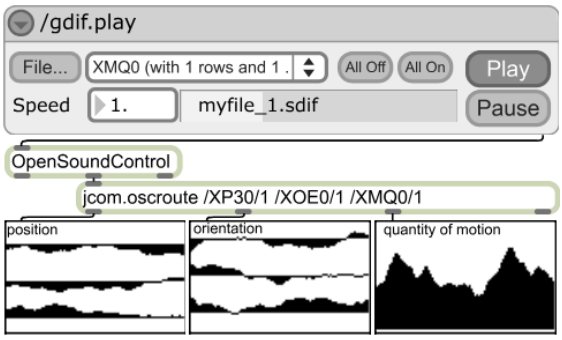

Figure 3. The playback module streaming a *3D position stream*, an *euler orientation* stream and a *quantity of motion* stream

face. Rather than having to record data into separate unsynchronised buffers, the user can now record data into a single container without worrying about synchronisation. The presented tools are open source, and can be downloaded by checking out the Jamoma repository from github, [7] or directly from the project website. [8] Since the data is recorded as SDIF files, users may benefit from tools like EasDIF [9] for analysis and post processing.

5. CONCLUSIONS AND FUTURE WORK

This paper has presented challenges we are facing when studying music-related body motion, and our solution to some of these problems in the form of a software toolbox. This toolbox includes a flexible module for making synchronized recordings of music-related data, and a module for playing back the data in real-time. The implementation makes the GDIF recording setup fast and easy, and makes this type of technology available to less experienced Max users.

Future development includes:

- Separating frame types as independent definitions. This will allow describing the stream type according to the device (e.g. a motion capture stream), and each frame can contain different data matrices (e.g. a position matrix and an orientation matrix).
- Human readable OSC namespace for data from the playback module (currently using the SDIF type tag).
- Integration of the Jamoma dataspaceLib for conversion between different data representations [12].
- Implementing simple data processing like automatic filtering and calculating absolute values.
- Develop a sequencer-like visual display, allowing zooming, editing, etc.
- Database for storing large collections of GDIF data.

6. ACKNOWLEDGEMENTS

Thanks to the developers of Max, Jamoma, FTM and OSC for providing a good frameworks for implementing these tools. Thanks also to the reviewers for valuable feedback.

[7] http://github.com/jamoma/Jamoma
[8] http://www.fourms.uio.no/software/jamomagdif/
[9] http://sourceforge.net/projects/sdif/

7. REFERENCES

[1] A. R. Jensenius, "GDIF development at McGill," McGill University, Montreal, Canada, COST ConGAS – STSM report, 2007.

[2] K. Nymoen, "A setup for synchronizing GDIF data using SDIF-files and FTM for Max," McGill University, Montreal, Canada, COST SID – STSM report, 2008.

[3] A. R. Jensenius, "Motion capture studies of action-sound couplings in sonic interaction," KTH, Stockholm, Sweden, COST SID – STSM report, 2009.

[4] A. R. Jensenius, A. Camurri, N. Castagne, E. Maestre, J. Malloch, D. McGilvray, D. Schwarz, and M. Wright, "Panel: the need of formats for streaming and storing music-related movement and gesture data," in *Proceedings of the 2007 International Computer Music Conference*, Copenhagen, 2007.

[5] J. Bresson, C. Agon, and M. Schumacher, "Représentation des données de contrôle pour la spatialisation dans openmusic," in *Actes de Journées d'Informatique Musicale (JIM'10)*, 2010.

[6] A. R. Jensenius, T. Kvifte, and R. I. Godøy, "Towards a gesture description interchange format," in *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression*. Paris, France: Paris: IRCAM – Centre Pompidou, 2006, pp. 176–179.

[7] R. I. Godøy, "Reflections on chunking in music," in *Systematic and Comparative Musicology: Concepts, Methods, Findings. Hamburger Jarbuch für Musikwissenschaft*. P. Lang, 2008, vol. 24, pp. 117–132.

[8] M. Wright, A. Chaudhary, A. Freed, S. Khoury, and D. L. Wessel, "Audio applications of the sound description interchange format standard," in *AES 107th Convention*, 1999.

[9] M. Wright, A. Chaudhary, A. Freed, D. Wessel, X. Rodet, D. Virolle, R. Woehrmann, and X. Serra, "New applications of the sound description interchange format," in *Proceedings of the 1998 International Computer Music Conference*, Ann Arbor, 1998, pp. 276–279.

[10] M. Wright, A. Freed, and A. Momeni, "OpenSound Control: state of the art 2003," in *Proceedings of the 2003 conference on New Interfaces for Musical Expression*, Montreal, Canada, 2003, pp. 153–160.

[11] N. Schnell, R. Borghesi, D. Schwarz, F. Bevilacqua, and R. Müller, "FTM – complex data structures for Max," in *Proceedings of the 2005 International Computer Music Conference*, Barcelona, 2005, pp. 9–12.

[12] T. Place, T. Lossius, A. R. Jensenius, N. Peters, and P. Baltazar, "Addressing classes by differentiating values and properties in OSC," in *Proceeding of the 8th International Conference on New Instruments for Musical Expression*, 2008.