M.U.S.I.C. (MUSICAL UTILITY SOFTWARE FOR INTERACTIVE CREATIONS): THE MULTIPURPOSE MOTION-SOUND MAPPING PLATFORM

Joseph Ryan, Steven Landry, Jaclyn Barnes, Myounghoon Jeon

Mind Music Machine Lab,
Michigan Technological University,
Houghton, MI 49931 USA
{jdryan; sglandry; jaclynb; mjeon}@mtu.edu

Chung Hyuk Park

Assistive Robotics & Tele-Medicine Lab,
The George Washington University,
Washington, DC 20052 USA
chpark@gwu.edu

ABSTRACT

With the advent of embodied interaction, research on interactive sonification has rapidly been increasing. There have been consistent efforts on the enhancement of motion-sound mappings, but still more improvements can be made. The goal of the Musical Utility Software for Interactive Creations (M.U.S.I.C.) platform is to provide a flexible, robust, and easy to use interactive GUI for taking motion data from other devices or software and mapping them to sound or music. In the M.U.S.I.C. platform, users can perform actions when the data meet user-generated conditions. Taking inspiration from the previous motion-sound mapping software, Interactive Sonification Markup Language (ISML), the M.U.S.I.C. structure allows for changes to be both dynamic and static, be it to load the program from launch or to make changes in real-time. Furthermore, any device's data can be used on-the-fly. After introducing the design and implementation of M.U.S.I.C., we discuss its comparisons with ISML and future plans.

1. INTRODUCTION

By supplementing or replacing traditional cognitivism, embodied cognition/interaction has emerged as a new theoretical paradigm in Cognitive Sciences [1] and Human-Computer Interaction (HCI) [2] and so does in interactive sonification. In sports, aerobics [3] and rowing in a boat [4] were translated into sound. In arts, movement of performing artists [5] and dancers [6] made aesthetic melody lines. Also, in Human-Robot Interaction gestures of children and robots were sonified to reflect their interactions [7]. In these various scenarios, auditory display researchers are specifically interested in motion-sound mappings. In the present paper, we introduce Musical Utility Software for Interactive Creations (M.U.S.I.C., https://github.com/trimlab/ilSoP-GUI-Interface) to offer a more flexible, robust, and easy to use graphical user interface for making mappings between motion data from external devices (e.g., sensors, cameras, robots, etc.) to sound or music. We expect that this work can help non-programmers easily configure and design the interactive sonification in real-time without hard coding. Also, other researchers and developers can update and contribute to making M.U.S.I.C. more powerful and accessible.

2. INTERACTIVE SONIFICATION MARKUP LANGUAGE (ISML)

Figure 1: ISML example.

To make efficient motion-sound mappings, we first developed the Interactive Sonification Markup Language (ISML) [8] (Figure 1). The primary goal of ISML was to enable researchers with non-programming backgrounds (e.g., cognitive scientists, artists, musicians, etc.) to configure mappings in an accessible way. ISML was designed for the immersive Interactive Sonification Platform (iISoP) where artists and technologists collaborate with each other [9]. The project focused on how researchers and artists could use sound in a 3D virtual environment to interactively communicate with users. The iISoP project utilized motion capture data via a Vicon tracking system and thus, ISML mainly communicated with the Vicon tracking system. This system allowed users to use their entire body as a controller. The motion capture data were taken in as input and mapped onto both visual and auditory displays in a number of ways.

ISML files can be considered as an activity flowchart or state machine defining application behavior in response to external inputs. ISML bears similarity to scripting languages since it is designed to automate the behavior of iISoP's sound generation application. Specifically, it is an example of an audio synthesis scripting language. Languages of this type already exist. To name a few, examples include the ChucK

audio programming language [10], the Csound [11], the commercial Reaktor software, and the MPEG-4 Structured Audio standard [12]. Some of these systems use graphical user interfaces for their scripting languages, while others are textual. Additionally, some of them support live coding, the ability to change the program's behavior while it is running. By comparison, ISML scripts can be created in a graphical or textual manner. To the best of our knowledge, ISML was the only scripting language specifically designed to translate physical movements into dynamically generated sound, which makes it very different from other audio scripting languages.

Using ISML to the mappings is much more efficient and accessible to non-technical researchers than attempting to reprogram the system for every new experiment with new specifications. Through our empirical experiences, we have found that using ISML could save 20-40% of time, compared to hard coding after 3-4 specification developmental cycles. For more details of ISML, see [8].

The ISML platform allowed researchers to create items that had both conditions and actions. Using this platform, researchers were able to do some basic object comparisons and interactions. ISML allowed the user to create simple 1:1 mappings between tracked objects and pre-defined MIDI notes and melody lines. However, unfortunately, developers and users were still required to hard code some features into the audio code and manually adjust ISML scripts. It was then realized that development was almost entirely being done in code or via text file manipulation, which was the opposite of the original intention of ISML. Moreover, the manipulations had to be made offline. The formation of the audio program was a static load up. In other words, users had to choose a file that they made at the compile time and could make no changes while the program ran. Features were program specific and could not be shared between users. This made it very hard for more than one person to use the platform effectively. The ISML script generator was also separated from the audio program. Therefore, users had to use multiple programs just to produce any sound. Another problem was the lack of data feedback that was received from the program. Data were calculated and used by the system, but the values were not displayed to users. This made it unnecessarily difficult to accurately assess system actions and make useful changes. The software lacked transparency and provided little feedback for development and testing of sonification scenarios. Finally, even though ISML's structure was very logical, it was somewhat confusing to new users, specifically for non-programmers. These issues led to many complications and difficulties when developing motion-sound mappings for the iISoP project.

To solve these issues, the Musical Utility Software for Interactive Creations (M.U.S.I.C.) platform was created. It serves as an interactive platform intended not only for the iISoP project, but also other interactive sonification projects in order to increase the robustness and usability of the entire system so that anybody can develop future applications in a more convenient way.

3. M.U.S.I.C. STRUCTURE

The structure of the M.U.S.I.C. platform is similar to the structure of the ISML platform. However, it has more

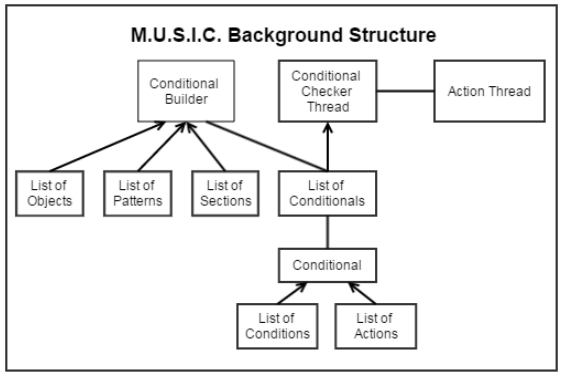

Figure 2: M.U.S.I.C. platform's basic structure.

```
Procedure conditionalChecker:
  variable doActions;
  begin
    repeat
        set doActions value to true;
        repeat
            if condition is not met:
              set doActions value to false;
                go outside of repeat
            end if
        until conditionList has no condition
              left

        if doActions is true:
            perform all actions

          end if

   until conditionList has no conditions left

  end
```

Figure 3: Pseudocode of the conditional checker.

features to bridge the gaps of execution and evaluation. The structure of M.U.S.I.C. is depicted in Figure 2. The platform is still built around the ability to create conditional statements for motion-to-sonification. A conditional is a combined set of conditions and actions. Conditions are user-defined criteria that are constantly checked against the stream of incoming data. Different actions, such as playing a specific note or melody, are triggered when specific conditions are met. When all conditions are met, all actions are performed in separate threads. This allows for multiple conditionals to be run concurrently without slowing the speed of execution for assigned actions. Conditionals are built in a block of code, called "conditional builder". The conditional builder has a reference to all of the current objects, patterns, and sections as well as a list of previously made conditionals. All of these references act independently of the conditional builder, which allows changes to occur at any point in the development process.

Each conditional has a flag variable, the value of which determines if the conditional is enabled or disabled. The conditional checker is a separate thread of program that has a reference to the conditional list and checks all conditionals if enabled or not. This process is shown in Figure 3, a pseudocode of the conditional checker. It checks for every enabled conditional, and then performs all actions within that condition. The flag variable doActions is set to true so that all actions can be performed even when no conditions are enabled. This in turn allows for chosen actions to be executed continuously, if desired.

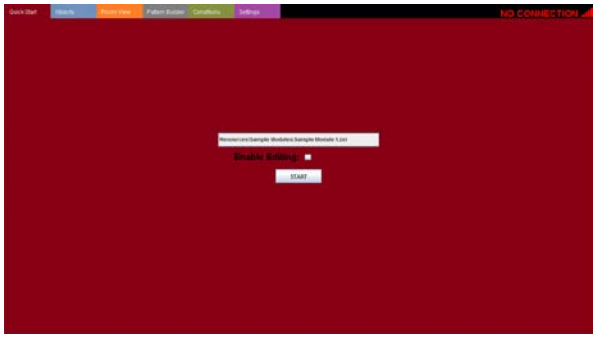

Figure 4: The Quick Start View.

Figure 5: The Objects View.

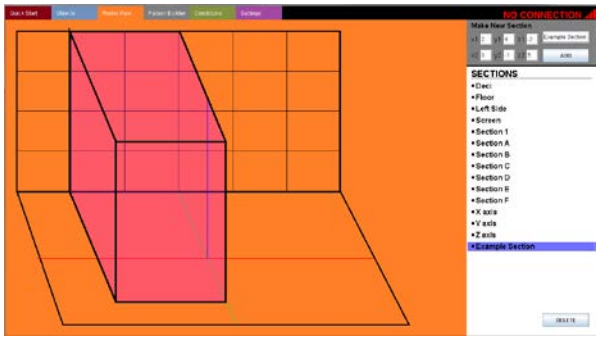

Figure 6: The Room View.

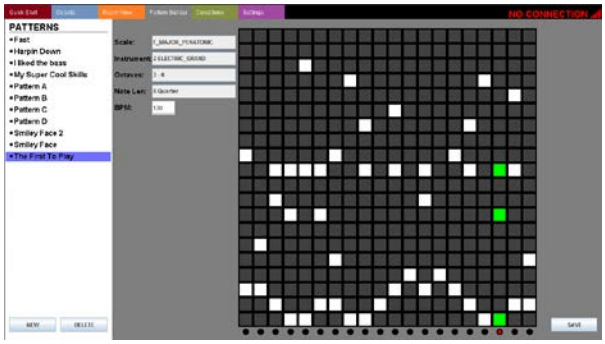

Figure 7: The Pattern Builder View.

Figure 8: The Conditional View.

4. M.U.S.I.C. GRAPHICAL USER INTERFACE

Following M.U.S.I.C.'s structure, the platform uses an interactive interface that allows non-programmers to "program" their modules without touching code. The M.U.S.I.C. GUI is split up into sub-sections called tabs, each of which has specific functions. Upon launching the program, the user will see the "Quick Start" tab (Figure 4). This tab is for starting up a previously created module. On this page, users can also learn whether they have a connection to the motion tracking devices (e.g., Vicon sender) and they are receiving the live data. The "Objects View" (Figure 5) allows users to view and access all of the different objects' data that the M.U.S.I.C. platform is tracking. Users can also toggle listening to certain objects, as well as adjust their color for the "Conditional View". In the "Room View" (Figure 6), users can divide the space into sections for later condition checking in the conditional builder, e.g., dividing the room into left and right halves so that conditional such as "play middle C when left hand is in the left half of the room" can be created. In the "Pattern Builder" tab (Figure 7), users can create their own MIDI patterns to use as actions of the conditionals. Users can simply click a box on the grid marking a note to play a pattern in a right timing. The simplicity of this view makes it easy for users to make their own patterns and sounds without additional music programs. It can also be easily connected/extended to external sound programs. The "Conditional View" (Figure 8) is our conditional builder. This is where users take all of their sections, patterns, and objects to make conditionals. As the current program stands, there are two types of conditions: 'in-section' and 'value'. The in-section condition checks to see if the object is within a created section. The value condition lets users choose from the list of data points and make a comparison between those numbers. The currently available actions are to play a single note or to play a previously defined pattern.

5. THE COMPARISONS OF ISML AND M.U.S.I.C.

Although the main conceptual structures between ISML and M.U.S.I.C. are similar to one another, there are major differences between the two platforms (Table 1).

Table 1: Comparison between ISML and M.U.S.I.C.

Functionality	ISML	M.U.S.I.C.
No programming knowledge required		✓
Interfaces with Vicon tracking System	✓	✓
Graphical user interface	✓	✓
Faster than text file editing		✓
Can view tracked data live		✓
Dynamic adjustments		✓
Can build conditionals	✓	✓
Easy to use		✓
Can be utilized internet free		✓

There were difficulties faced when using ISML for extended development. One was simply the speed of script development. In most cases, developers had to start with ISML and then go to the generated text file and edit it manually. M.U.S.I.C. solves this problem by having an enhanced graphical user interface (GUI), which not only gives users direct visual/audio feedback for the work that they have done, but also enables them to simply make complex features without writing a single text file. In our experience, it is faster for most users to create conditionals in the GUI than in text files. In addition, the M.U.S.I.C. platform provides the users with feedback on the objects being tracked (the Objects View in Figure 5), which is particularly helpful for debugging. By allowing users to dynamically adjust patterns, sections, and conditionals while sonifying data in real-time, M.U.S.I.C. solves the static manipulation problem that ISML has had. Most importantly, any user can use this platform without having a programming background.

6. THE FUTURE OF M.U.S.I.C.

M.U.S.I.C. is currently in its infancy. There are considerable potential applications and future directions that this platform can go. First and foremost, we will add more conditions and more actions that can be utilized. The structure makes it very convenient to add new conditions and actions. In addition, when the algorithms for gesture recognition are created (e.g., drawing a circle in 3D space), it is just a matter of adding a new condition option for users to utilize. Next, adding more objects to interface with the platform, such as a muscle sensor (e.g., Myo) or brain sensor (e.g., EPOC), can add a new dynamic of biological data tracking. Code optimization to further improve performance as well as running usability tests to improve user experience are also planned. The sub-section GUI, such as the Room View tab, could be greatly improved by a better graphics engine, such as the Unity Engine. Since this application is currently built in Java, developers are exploring the option of moving this application to Android as well. Conditionals are very useful concepts and could be generalized to other tasks and domains that would benefit from interactive sonification. For example, in the driving domain, additional auditory feedback could be provided based on driving performance. Similar applications are possible for athletic training, physical rehabilitation, and human-robot interaction.

7. REFERENCES

[1] Davis, J. I. and Markman, A. B. (2012). The future of embodied cognition. Topics in Cognitive Science, 4, 4, 685-793.

[2] Marshall, P., Antle, A., Van Den Hoven, E., and Rogers, Y. (2013). Special issue on the theory and practice of embodied interaction in HCI and interaction design. ACM Transactions on Computer-Human Interaction (TOCHI), 20(1).

[3] Hermann, T. and Zehe, S. (2011). Sonified aerobics: Interactive sonification of coordinated body movements. Proceedings of the 17th International Conference on Auditory Display, Budapest, Hungary.

[4] Schaffert, N., Mattes, K. and Effenberg, A. O. (2011). The sound of rowing stroke cycles as acoustic feedback. Proceedings of the 17th International Conference on Auditory Display, Budapest, Hungary.

[5] Jeon, M., Landry, S., Ryan, D. J., & Walker, J. W. (2015). Technologies expand aesthetic dimensions: Visualization and sonification of embodied penwald drawings. In A. L. Brooks, E. Ayiter, and O. Yazicigil (Eds.), Arts and Technology. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST) vol. 145. (pp. 69-76). Springer International Publishing.

[6] Landry, S. and Jeon, M. (2017). Participatory design research methodologies: A case study in dancer sonification. Proceedings of the International Conference on Auditory Displays (ICAD2017), PA, USA, June 20-23.

[7] Zhang, R., Barnes, J., Ryan, J., Jeon, M., Park, C. H., and Howard, A. (2016). Musical robots for children with ASD using a client-server architecture. Proceedings of the 22nd International Conference on Auditory Display (ICAD2016), Canberra, Australia, July 2-8.

[8] Walker, J. W., Smith, M. T., and Jeon, M. (2015). Interactive sonification markup language (ISML) for efficient motion-sound mappings. Proceedings of 17th International Conference on Human-Computer Interaction (HCII2015), Los Angeles, CA, August, 2-7, 2015.

[9] Jeon, M., Smith, M. T., Walker, J. W., and Kuhl, A. S. (2014). Constructing the immersive interactive sonification platform (iISoP). In N. Streitz and P. Markopoulos (Eds.), Distributed, Ambient, and Pervasive Interactions (DAPI), LNCS 8530, pp. 337-348. Switzerland: Springer International Publishing, 2014.

[10] Wang, G. (2008). The ChucK Audio Programming Language. Dissertation. Princeton University, New Jersey.

[11] Boulanger. R. (2000). The Csound Book. MIT Press.

[12] Scheirer. E.D. (1998). The MPEG-4 structured audio standard. Acoustics, Speech and Signal Pro-cessing, 1998. Proceedings of the 1998 IEEE International Conference on. Vol. 6. IEEE.