NAVIGATING ONTOLOGICAL STRUCTURES BASED ON FEATURE METADATA WITH THE SEMANTIC MUSIC PLAYER

Florian Thalmann, Alfonso Perez-Carrillo, György Fazekas, Geraint A. Wiggins, Mark Sandler
Centre for Digital Music, Queen Mary University of London
`f.thalmann@qmul.ac.uk`

ABSTRACT

The *Semantic Music Player* is a cross-platform mobile app that investigates new ways of playing back music on mobile devices, especially unpredictable, context-dependent, and interactive ways. It takes realtime decisions based on structural information and analytical data represented using Semantic Web technologies, and reacts to sensor and user interface inputs.

1. INTRODUCTION

Within less than a decade, mobile devices have developed into compact multi-sensory computers, bringing computing power and novel ways of interaction, such as multi-touch gestures, accelerometer control, or geolocation tracking, into everyone's hands and pockets. [3] Yet, the music listening experience is only slowly adjusting to the new environment and its capabilities. Although mobile music apps often add functionality adopted from social media, such as recommendation schemes and shareable playlists, the listening process itself is often no different from how it was with a Walkman in the early eighties.

In this demonstration, we introduce the *Semantic Music Player (SMP)* with which we investigate novel ways of experiencing music on mobile devices. The SMP is a cross-platform mobile app built with Ionic [1], ngCordova [2], JavaScript, the Web Audio API [3], as well as Semantic Web technologies, and is designed to play back music in *spontaneous*, *unpredictable*, *context-dependent*, and *interactive* ways. Figure 1 illustrates the structure of the application along with the involved technologies.

2. DYNAMIC MUSIC OBJECTS

The SMP builds on the notion of the *Dynamic Music Object (DYMO)*, a type of *Digital Music Object* [2] that is

[1] http://ionicframework.com
[2] http://ngcordova.com
[3] http://www.w3.org/TR/webaudio/

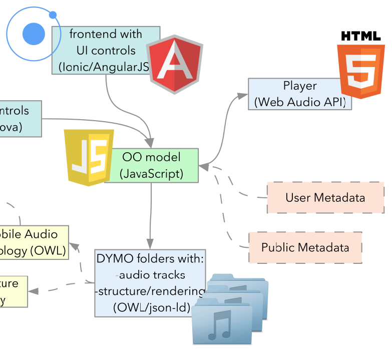
frontend with UI controls (Ionic/AngularJS)
HTML
Player (Web Audio API)
sensor controls (ngCordova)
OO model (JavaScript)
User Metadata
Public Metadata
Multitrack Ontology
Mobile Audio Ontology (OWL)
DYMO folders with: audio tracks structure/rendering (OWL/json-ld)
Audio Feature Ontology

Figure 1. Application structure of the Semantic Music Player.

flexible and modifiable and that typically sounds different each time it is played back. More specifically, we define DYMOs as

- a bundle of *music files*
- *analytical data* extracted from the files using MIR techniques
- a *structural definition* relating the audio and the analytical data and enabling a number of modifiable musical parameters
- a playback configuration called *rendering*, which maps controls to parameters

Both the structural definition and the rendering are represented using the Web Ontology Language (OWL) [4] and can be queried and searched on demand via SPARQL. [5] Figure 2 shows an example of a DYMO structure representing a simple multi-track mix that offers parameters on various hierarchical levels. [6] Whenever a higher-level parameter is changed, lower-level ones are adjusted accordingly, relative to the higher-level one. [7] With the example DYMO in the figure, amplitudes can for instance be controlled for the main `mix` object, which changes the overall amplitude, but also for the `riddim` object, which merely

[4] http://www.w3.org/TR/owl2-overview/
[5] http://www.w3.org/TR/sparql11-query/. The so-called *Mobile Audio Ontology* defines these structures which are based on CHARM, an abstract music representation system. [4]
[6] The definitions in the figure are expressed in (pseudo-)Turtle, a textual syntax for OWL ontologies (http://www.w3.org/TR/turtle/). The `a` keyword refers to the `rdf:type` property, which defines instances of OWL classes, written in capitalized words, e.g. `DYMO` or `Amplitude`. Properties begin with a lowercase letter, e.g. `hasParameter`.
[7] This is analogous to the relative transformation of satellites as for instance described in [6].

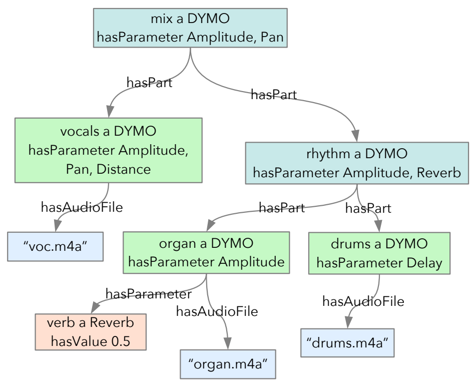

Figure 2. A sample Dynamic Music Object.

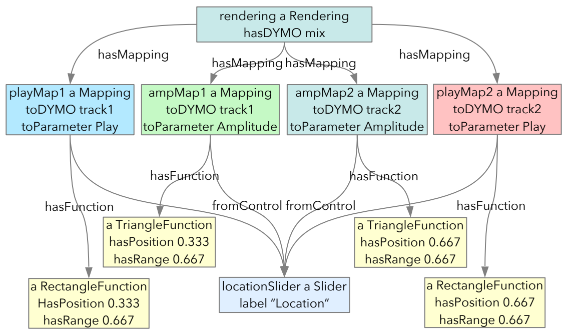

Figure 3. A sample rendering of a DYMO with two tracks.

affects organ and drums. The example DYMO also has other parameters, some of them informed by analytical features extracted and represented in OWL using Vamp plugins [1], for instance a bar segmentation parameter.

3. DYMO RENDERINGS AND MAPPINGS

Once we have a structural definition of a DYMO we can specify how it will be played back by defining one or more *renderings*, which consist of a set of mappings from features or controls to any of the available DYMO parameters. Currently supported controls include *sensor* controls (accelerometer, compass, geolocation, etc), *UI* controls (sliders, buttons, toggles, etc), and *autonomous* controls (statistical, graph navigation, AI). Figure 3 shows a rendering where one slider is mapped to various parameters of a DYMO representing two tracks. The two rectangle function mappings decide when each of the tracks is started or stopped while the triangle function mappings control the amplitudes leading to a cross-fade. A graph of the result is shown in Figure 4. The generality of the framework also allows for more adventurous structural definitions and renderings. [8] Multi-dimensional mappings, for instance, can be used to create intricate results, e.g. a two-dimensional generalization of the rendering defined above could create geolocation mappings similar to the ones defined in [5].

[8] In order to facilitate the definition of DYMOs and their renderings and make them intuitively accessible to musicians, producers, or distributors we are also working on the so-called *Dymo Designer*, a simple visual interface that allows users to define the mappings graphically, for a multitude of use cases.

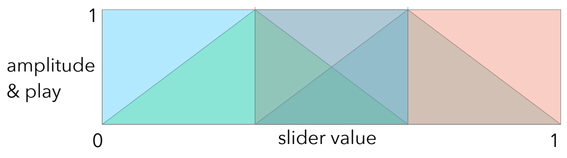

Figure 4. Graph of the mappings in Figure 3.

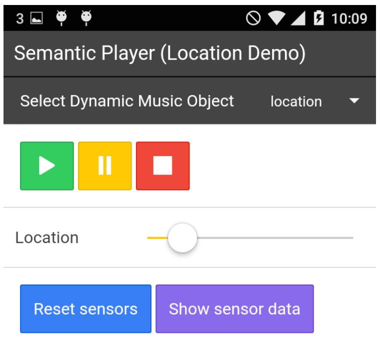

Figure 5. The GUI dynamically generated for the rendering in Figure 3, shown on an Android device.

For any rendering and its DYMO, the interface of the Semantic Music Player is generated dynamically. Figure 5 shows the interface generated for the rendering shown in Figures 3 and 4, consisting of just one slider labelled *Location* in addition to the standard interface buttons.

4. REFERENCES

[1] Chris Cannam, Mark Sandler, Michael O Jewell, Christophe Rhodes, and Mark d'Inverno. Linked data and you: Bringing music research software into the semantic web. *Journal of New Music Research*, 39(4):313–25, 2010.

[2] David De Roure. Executable music documents. In *Proceedings of the 1st International Workshop on Digital Libraries for Musicology*, pages 91–3, 2014.

[3] Georg Essl and Michael Rohs. Interactivity for mobile music-making. *Organised Sound*, 14(2):197–207, 2009.

[4] M. Harris, A. Smaill, and G. Wiggins. Representing music symbolically. In *Proceedings of the IX Colloquio di Informatica Musicale*, Venice, 1991.

[5] Adrian Hazzard, Steve Benford, and Gary Burnett. Sculpting a mobile musical soundtrack. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, Seoul, 2015.

[6] Florian Thalmann and Guerino Mazzola. Visualization and transformation in general musical and music-theoretical spaces. In *Proceedings of the Music Encoding Conference 2013*, Mainz, 2013. MEI.