Introducing the Music Access Platform

Gavin Wood and Aeron Buchananc
Lancaster Logic Response Ltd.
19 Kendall Crescent, Oxford OX2 8NE UK
i@gavwood.com, aeron@lancasterlogicresponse.com
http://lancasterlogicresponse.com

Music is an ever present component of most people's lives from the background soundtrack in cafés to the prominence of live stage performances. While composition and appreciation is well understood, its generation and analysis are very human intensive processes. Computer automation in this area is cutting edge research and many promising approaches have been proposed. However, being an academically taxing area, these techniques and their associated utility too often remain inaccessible to those outside the immediate research discipline.

The Music Access Platform, a project funded in part by the UK government’s Technology Strategy Board, aims to deliver the fruits of music analysis research to those from other, disparate disciplines including multimedia design, lighting and concert design, choreography and the media. By raising awareness we hope that the platform can effect the leveraging of existing talent and facilitate exciting new possibilities in cross-media performance.

We will discuss driving factors and the requirements needed to reach this goal given the disparate nature of the end-users. Aspects of music annotation, description and specification will be discussed both in terms of the creative disciplines and cross-referenced with the audio-analysis disciplines. An overview of the access platform's interface design along with usage examples will be described, including our proposals for work flow and system interaction and augmentation. Finally, a case-study will be discussed and used to illustrate how such a platform could be utilised to deliver a choreographed performance at once expressing both the intent of the musician and that of the lighting artist with minimal need of any technical knowledge.


1. INTRODUCTION

The enjoyment of music can be enhanced by reinforcing its structure through the stimulation of our other senses; e.g. seeing a musician create the music imparts resonances in the dynamics and progression of a piece. Likewise with a choreographed dance performance or a crafted visualisation. The range in effort to create this supplementary media is great; musicians performing on stage naturally conduct themselves with resonance to the music that they play, however the hours spent (a recent light show reported by Grenoble (2012) took up to 4 hours of programming per minute of music) choreographing a lighting display or an animated music visualisation betrayed effort far from trivial. In many cases human performers are not available, nor perhaps even appropriate, and thus there is a place in a cross-media context for computerised tools that can be used to create media to accompany musical pieces. They may be either fully automatic, requiring no extra intervention or as a basis for ambitious projects where an artist may utilise the system allowing them to concentrate effort on elevating the work to new aesthetic heights. In this paper we describe our work on exactly such a system of tools.

Music is rich in information: as listeners, the way in which our subconscious interprets the structure and texture in a piece of music determines how interesting and entertaining we find that music. It is the artistry of composers and performers that creates the structure in everything from the melody and rhythm of the piece as a whole to the texture and dynamics of individual notes. It is helpful to consider our enjoyment of music as coming from our brain’s extraction of information from a sound stream. Firstly, this outlook focuses our efforts on the extraction of semantically important information (e.g. melodies rather than frequency spectra); and secondly it directs us towards a vocabulary with which to converse technically about problems and solutions. The problem we presently address is the creation of cross-media performances based on music. From now on, for brevity and clarity, we will use ‘music’ to mean the main sonic aspect of such a performance and the term ‘resonant’ for the non-
musical part, such as an animated visualisation piece.

For the resonant side to be convincingly associated with a music performance, it must reflect and reinforce the meaningful events and characteristic structure of the music. How well an auxiliary media resonates may be likened to the ease of identifying which piece of music it resonates with. For example, a basic frequency-spectrum or VU meter animation looks very similar for quite disparate music, whereas a bespoke, hand-animated visualisation for stage projection, such as Amon Tobin’s ICAM set, looks very different for each track of the performance. Our system achieves this goal, even when there is no human direction to modify and elevate the resonant output.

1.1 Related Work

When considering work in this field, a major section can be classified as track-level processing systems which basically comprise genre classification (e.g. by Cifrasoft (2013)) and music fingerprinting (e.g. Wang (2006)). In general the provision of track-level information provides little towards real-time choreographing of resonant performances. There is, of course, much work done in the related field of music information retrieval (e.g. Tzanetakis and Cook (2000), Dixon (2007), Lartillot and Toivainen (2007), Cannam et al. (2010)), some even for real-time deployment. However, most have heavy platform ties (e.g. Matlab, Java, Windows) and are not designed for real-time operation, or are academically orientated to the point where integration with other systems is difficult. So far, very little has been accomplished with these systems in the context of resonant performances.

Downie et al. (2005) describes MIREX competitions, which provide a forum where music information algorithms compete against each other and provides breadth of individual algorithms on a range of music analysis tasks. However, these are only the basis for a system with the capabilities we desire. Furthermore, they have not yet created a standardised data interchange format, instead using ad-hoc result logging formats for each competition.

Another major section is the range of existing "sound to light" software that claims to be able to create authentic light-show performances automatically (e.g. LightJams (2013)) or semi-automatically (e.g. SOH (2013)) but their performance, being based on trivial audio analysis, fails to produce light-shows that are significantly better than a random controller. Systems (e.g. Pope (2013)) that have more substantial capabilities rely on offline analysis which, again, is not suitable for live cross-media performances.

Notably, there is very little in the way of commercially available software or technology that brings together cutting edge music analysis and the ability to deploy it in a cross media performance context. Some companies are providing limited feature sets, such as just beat detection (in the case of SampleSumo (2013)), or beat detection, fundamental identification and note start/end (as with ZPlane (2013)), but aside from their narrow semantic vocabulary, they do not yet have suitable front-ends for integration into an artist-driven performance mechanism.

1.2 Perspectives and Requirements

The stated capabilities of our system come from considerations of practicality and the views of both cross-media performance artists and software designers. As discussed above, a problem suffered by many existing systems is developing the one aspect of the problem to the exclusion of the other: in our design, we have been careful to integrate the requirements both on the technical system integration side and on the performance side (to maximise the potential of artistic expression).

The system we present is for use in a live performance context and can be considered as another instrument alongside those creating the music, except that our system takes the music as an input alongside the directions of the person who is controlling our system, and rather than outputting sounds, creates a performance on the lights of a lighting rig. Note that the underlying Music Access Platform has been designed to service both this application, and other possible end users and platforms, such as projected graphics, dancing robots, computer games and more. As such, the internal system consideration for a semantic music description language must keep in mind the broad potential scope of the Music Access Platform. Nevertheless, all discussion is directly relevant to cross-media performance production and our real-time light-show authoring application.

1.2.1. Salient Events

The key to the creation of a successful resonant accompaniment to a musical piece is in the identification of and reflection on the salient aspects of the music. This was discussed in the introduction, with the comment that it is far from easy to name the full range of possible salient events in music. We have approached this problem from both the technical side and performance side, by intersecting what is theoretically achievable with the feedback we received from consultations with practitioners.

We identified three levels of detail; at the top there are track level descriptions, such as musical style
(rock, pop, jazz, &c.), overall mood (up-beat, calm, tense, &c.), and song identification. Such information is, at present, out of the scope of the system due to its hard ties to the musical format and circumspect nature.

At the middle level there are transient or interval based descriptions, which register events that necessarily occur over a period of time, albeit short, e.g. of the order of the length of a bar. This is the level at which event descriptions become useful for resonant performances, although these traits can only be known in a temporally non-localised way and therefore can effectively only tag the current description stream. The events that have been identified to be at the top of the list are as follows:

**Tempo** the current beats-per-minute and time signature.
**Dynamics** crescendos, fade-outs and intended silence.
**Timbre** current mood indication and instrument identification.
**Structural descriptions** identification of a pre-programmed section of music (verse, chorus, bridge, build, coda, &c.) or the repetition of a previously-heard section of the current stream.

These characteristics serve two main purposes: they allow for the ability to ‘flavour’ the resonant performance; and they can help trigger the synchronisation of motif-based performance overlays.

At the event level, where salient events can be temporally localised very accurately, a more well-defined list has clearly emerged as important:

- Bar phase information (e.g. bar start events).
- Sudden silence events.
- Foreground note events (with timbre information).
- Drum onset events (including character information, e.g. snare or bass).
- Background events (such as backing instruments starting or stopping).

Importantly, the system design is such that this list can be expanded with new features as technical advancements in music analysis are made and artistic ambition in this area develops. The current implementation includes the most salient of these features, with the incorporation of the others being the subject of our future work.

### 2. DESIGN OVERVIEW

The Music Access Platform (MAP) is designed to sit between some audio source and some other media-generation process. The audio source may be a pre-existing digital recording, a microphone input or perhaps a multi-channel live performance; the platform’s design makes this largely immaterial. By media-generation process, we mean a process that may be utilise either human or machine-readable information to generate media. We mean to include most artistic endeavours in this, but particularly those which may be set to music; dance, lighting, motion-control systems, animatronics, animation, and abstract visualisation to name but a few. In some (e.g. dance), a one or many humans may necessarily be part of the generation process, others (e.g. abstract visualisation) the process may be entirely machine-authored.

We now come to describe the format of data exchange that forms the core of the MAP. We will then describe some techniques we use at present to determine this data from audio content.

#### 2.1 Data Interchange Format

Our approach to the data format errs, where necessary, on the side of generality and reuse; over-specification risks a data format unsuitable for later algorithms and/or applications on either in the input or output side. As such, we opted for a relatively well-understood base format of discrete time-specific items, called ‘events’ (much like MIDI).

As a data-interchange format for characterising streams of musical audio to be used for generation of other media, we had two options to maximise fairness over types of media, and thus future-proof the format. One would be to tie the format to music, allowing the output process to determine appropriate relationships with musical phenomena. The other would be to position the format away from music and attempt to create a wholly abstract representation.

The first, in the purest sense, reduces the problem of the MAP to music transcription. Though there are several advantages, this has three key drawbacks: difficulty, over-specification and under-specification. Over-specification since there is information necessarily required by transcription (such as the pitch of a particular instrument’s note-onset) which in many cross-media exercises is considered largely meaningless. Under-specification since vital information (e.g. the timbre of an instrument, or the interaction of distinct instruments or structures within a track) is ignored. In short, music notation is not an effective way to communicate cross-media concepts.

We took the second approach, attempting to, where possible, describe the music as it would be interpreted or perceived by a supposed listener away from its audio format. The notion of splitting the essence of an art form from its material format has long been discussed by philosophers (e.g. Wittgenstein (1921)), and we could not wish to contribute to such a fundamental puzzle. However,
the work may be seen as an approach to formulate an art-essence-language for a particular use-case.

2.1.1. Events
The format may be specified as being a 'stream' of discrete 'events'. Each event has a particular type and is associated with a particular time. A causal temporal relationship is assumed; an event within a stream affects its state until a relevant later event. For example, an event of type PeriodSet will alter the tempo information accordingly until, e.g. some later PeriodReset event.

In addition to type, events have a number of other basic meta-properties; its nominal position within the present periodic-structure, its predictability, and its salience or certainty of existence. These meta-properties are:

type One of a number of event types. Event types include Attack, Sustain, Decay and Release for specifying ongoing phenomena; PeriodSet, PeriodTweak and PeriodReset for specifying apparent periodicity.
position Unknown or [0, 63] for first 16th note in a 4-bar cycle, second 16th note, &c.
strength Quantity [-1, 1] describing the certainty and/or power of the event. Negative numbers describe expected events that didn't happen.
surprise Quantity [0, 1] describing the unexpectedness of the event; predictable events have low surprise.

The data is designed to be a fixed-size format allowing for trivial serialisation to a byte-stream and platform-neutrality. This is a boon for implementing real-time data throughput in embedded systems (of the ilk that may be used during a performance) as serialisation of more complex types can introduce considerable complexity.

They also have a series of properties to describe the nature of the event. For events of type PeriodSet/Tweak, time information is included in two fields: period and phase. These events are used to describe the predominant repetitive structure. The structure is abstract and it is left undefined how it may manifest itself in terms of other events. The phase property specifies its phase at the time of the event.

The types Attack, Decay and Sustain are used to describe abstract events in the stream; Decay and Sustain are alternatives depending on whether the event is momentary or long-lived. It is assumed that the same underlying phenomenon is tracked by neighbouring Decay and Sustain events. Events of type Attack may be used effectively as a separator, noting that the underlying phenomenon is different (e.g. a new beat or note onset). A single note with a pitch-slide may be described as a series of Sustain events, where as a set of individual notes could be described by the same events interspersed with Attack events. They each have several properties:

type Binary triplet to roughly classify event's 'personality'. An event may be either peaceful or aggressive, either chaotic or structured, and either disparate or pointed. For each combination of these three possibilities, we have a moniker (e.g. the character of peaceful, structured and disparate is named 'harmonious' whereas its opposite 'implosive' is the character aggressive, chaotic and pointed).

temperature Abstract quantity [0, 1] to describe the content of the event in terms of 'heat'. Tonally, a high, shrill sound corresponds to 0, and low, bassy sound to 1; chromatically, 0 corresponds to indigo and blue shades, and 1 to red, orange and yellow shades; visually, low temperature corresponds to relaxation, high to alertness.

jitter Abstract quantity [0, 1] to describe the content of the event in terms of how much random or chaotic behaviour it exhibits during its lifetime. A higher jitter may correspond to a voice, which while gestalt, seems unpredictable to the listener.

constancy Abstract quantity [0, 1] to describe the content of the event in terms of the slowness of change through which it transitions. A high constancy may correspond, e.g. to a lower decay rate of an instrument's ADSR envelope.

Events of type Release have no properties and are instead used to nullify previous Attack/Decay/Sustain events and denote that a stream is inactive.

We have specified events both as an XML datum, a C or C++ structure and as a raw byte-serialisation; we anticipate being able to describe it in other data-specification languages trivially.

2.1.2. Event Compilers
An event compiler is an algorithm for reducing some input audio into an event stream. It may be causal (and as such potentially function in a real-time situation) or acausal (and be suitable only for off-line processing). It has a very simple interface, consisting of two definitions: initialisation and iteration. Iterations happen at a fixed rate, and the procedure takes as input the new portion of audio data (a fixed number of samples) and a spectral representation calculated by a windowed short-time Fourier transform. It outputs a set of stream events, potentially empty, to update its interpretation of the state of the music.

The event compiler may be seen as a basic MIR component; the simplicity of the input/output formats and the minimalism of the interface makes interoperability and language neutrality a realistic possibility. Though the C++ interface is the primary platform, we have implemented C, Matlab and textual Unix shell-based compilers with no difficulty.
2.1.3. Channels
To allow for the possibility of merging multiple incoming streams of information while retaining their provenance, each event may have an associated channel. Channels would typically be used to distinguish between the outputs of different event compilers given the same source audio or different source audio when used with the same event compiler.

Aside from the added simplicity of having a single, time-multiplexed data stream for all events regardless of provenance, it trivialises the implementation of more advanced routing scenarios. We define a special ‘broadcast’ channel identifier, allowing an event compiler to put specific events into all outputs; periodicity information typically falls into this category; one event compiler alone may reasonably determine the periodicity of the entire piece.

2.2 Example Event Compilers
For the present prototype we have implemented three event compilers: a basic percussion detector, a melodic analyser, and a timbre analyser.

The percussion detector determines individual percussive events and will inform of any apparent periodicity. The latter is expressed through PeriodSet and PeriodReset events and is determined through an auto-correlation of a beat likelihood function similar to that of Brown (1993). All possible periodicities (around 128 in our prototype’s case) are tested via the auto-correlation function to determine a probability density function of the period; this is multiplied into a prior, smoothed and normalised. The highest peak is determined and an event is emitted if necessary. The beat likelihood function is a simple high-frequency-biased spectral magnitude as described by Bello et al. (2005).

The individual percussive events (i.e. beats) are detected by thresholding the beat likelihood function. To reduce multiple detections, the threshold is elevated to the function’s value and decays with a fixed half-life. An Attack event is emitted when a beat is detected; after a small period, a Decay event is emitted along with an updated temperature and strength according to the spectral profile. When no beats are detected for a short period of time, the stream is labelled inactive with a Release event.

The melodic analyser is designed to emit a Sustain event for each note played. The temperature of the event is to be related to the pitch of the note; notes of equivalent pitch should give events of equivalent temperatures. The implementation relies on a property of the short-time (windowed) Fourier transform: for a continuous tone, modelled by a sine-wave, the phase-difference of neighbouring spectra is proportional to the modulo remainder of the period of the wave divided by the window-hop distance. The spectral band with the greatest magnitude has its phase-change reduced to a temperature using the aforementioned relation and, if the band is sufficiently different from the previous a new Sustain event is emitted. After a sufficient period of no significant peak in the spectrum, a Release event is emitted allowing discrete notes to be detected.

Inspired by techniques of Tzanetakis and Cook (2000), timbre analysing compiler emits events based upon statistical analysis of the audio spectra. In particular, it emits Sustain events whose strength is an appropriately scaled magnitude of the spectrum and whose temperature is a scaled centroid. When the magnitude of the spectrum is sufficiently low, a Release event is emitted. The compiler reduces its output by delaying the events that are only minor changes on the previous event.

3. CASE STUDY
Through the case study we will demonstrate that the MAP facilitates the development of useful tools for cross-media art performance. We will illustrate how the MAP can be used by a system that fits into a traditional performance framework and leverage it to allow powerfully cross-media pieces.

Our case study revolves around a novel device (the ‘system’) used by an operator to control a set of computer-controllable lighting fixtures. The device takes as input a real-time audio feed of the music. It has a sophisticated method for automatically controlling arbitrary sets of lighting fixtures utilising an event stream. The device provides no way for a human to control the lighting fixtures directly; instead, a specially developed operator-interface—the first of its kind—may be used, in real-time, to edit, tweak and bias the event stream’s content as it is generated by the event compiler.

The system operator may, in this sense, be called a light-jockey. Much like a disc-jockey combines and alters existing audio (CDs and LPs) at a high-level with mixers, equalisers and turntables to create a final result that builds upon them, a light-jockey performs a similar job with relation to lighting. They have the opportunity to ‘mix’ event streams onto different lighting groups, to bias aspects of particular stream events, and to alter the character of particular streams. As such they can use very high-level concepts to shape, in real-time, the overall form of the lighting show while building
upon the intrinsic meaning of the music - its rhythm, dynamics and melodic content.

### 3.1 Design

Figure 1 shows how the system integrates with a normal music performance’s pipeline. Prior to mixing and amplification, the various instruments’ audio feeds are digitised and analysed by event compilers. The event streams are altered by the human operator before being given to inform an automated lighting control framework.

![Figure 1: A basic schematic of the system](image)

The operator may alter the event streams in several manners. Each event stream may individually have the strength, temperature, jitter and constancy of its events biased. Furthermore, each such stream may have a Boolean mask applied to its events’ characters, forcing one or more of the character traits to one of the two polarities.

In addition to altering the incoming events, the lighting control program may also be configured directly; the prominence (or ‘visual loudness’) of each event stream may be set, allowing a greater share of the available lighting fixtures to be assigned to a given channel in much the same way a mixing engineer may reduce the apparance of sundry instruments during a guitar solo. By default, all active channels (i.e. channels whose previous stream event was not ‘Release’) are of equal prominence.

In addition to these basic mechanisms, the controller may be placed in one of four modes; Hold, Immediate, Reclaim and Free. These determine the point at which the aforementioned settings take effect and also when and how the lighting fixtures may be repartitioned. The ‘Free’ mode allows the controlling program full freedom to repartition and any changes to stream events are immediate. ‘Reclaim’ and ‘Immediate’ are similar but the system may only repartition in the case of having newly active event streams or unused groups of lights (‘Immediate’ only). ‘Hold’, as the name suggests, halts all changes to event streams and the configuration until the mode is changed.

These are designed to allow the operator the ability to perform synchronised step-changes to the event stream biases and prominence ratios. Changes to these parameters may be stored up in ‘Hold’ mode and take effect all at once with a quick change to the ‘Immediate’ or ‘Free’ modes, depending on whether the operator wishes to reconfigure the groups entirely (potentially mapping different input channels to lighting fixtures) or merely allow the new biases to take effect.

### 3.2 Implementation

The prototype implementation is relatively simple. For the hardware, off-the-shelf components were utilised. A low-power x86-64-based computer serves as the central unit on which all software runs. For audio input, we use a 4-channel USB sound card and for interfacing to the lighting fixtures a simple USB to DMX converter suffices. Rather than building custom hardware for the light jockey, we utilise a programmable USB MIDI control surface.

![Figure 2: The Novation Nocturn, which we utilise as our controller interface](image)

Aside from the MAP, the significant software of the project is the user interface and the automatic lighting system. The latter is based upon a driver framework with a hardware-abstraction layer. This allows each lighting fixture, regardless of manufacturer or model, to have its capabilities (number of independent lights, motion, colour, brightness, gobo &c.), inspect and be controlled in the same manner. It is this portion of the software that drives the DMX interface and, being self-contained, is out of the scope of the present work.

The rest of the lighting control software is to use said hardware-abstraction layer to set the state of each of the available lighting fixtures from moment to moment (at a rate of around 43 Hz due to the DMX specification), given the incoming event stream.

#### 3.2.1. Operator Interface

The operator interface is the portion of the device that the operator (i.e. light-jockey) uses directly; we used the Novation Nocturn as the hardware for this prototype, reprogramming the functionality so that it received stream events from the MAP and
delivered them, transformed, to the lighting control system, as well as the other parameters such as mode and each channel’s prominence. This device has a total of nine rotary encoders and 16 push-button encoders.

Figure 3: How we map each of the Nocturn’s encoders to functions of the label

Figure 3 depicts the Nocturn’s basic mapping. The relative prominence of each of the four input channels is determined by the value of each of the four rotary encoders on the left. The mode is changed temporarily by holding down an encoder and permanently through two brief pushes. The push-button of the current mode is back-lit. The temporary hold functionality allows a mode to quickly take effect without having to switch back to the previous mode; typically the ‘Hold’ mode will be permanently selected with the other three being quickly tapped in order to execute the step-changes to the configuration.

The other four push-button encoders act as toggles, one for each of the four input channels. Pushing one alternates between selected (and back-lit) and de-selected (dark). The right side of the controls apply to each of the channels selected in this way.

On the right side, the four rotary encoders allow the operator to bias incoming events’ properties; strength, temperature, jitter and constancy may each be altered. Temperature is biased through a specially-crafted sigma-like function whereas the others through a simple scaling factor. In pairs, six of the push-button encoders can be used to mask the selected channels’ characters. Each encoder of each pair acts as a toggle that is mutually exclusive with its alternative. The characters of events in each selected channel are masked according to the selected character components; e.g. should Disparate and Violent be selected, an incoming event of character Chaotic/Disparate/Peaceful would be transformed to an equivalent event of character Chaotic/Disparate/Peaceful.

3.2.2. Lighting Control

The main lighting control program can divide lights into groups in order that each active events channel is expressed by a group of lighting fixtures. The groups’ contents are determined according to the prominence associated with the channel (set by the operator) and the character of the events in the channel (biased by the operator). Each group is assigned a sub-program which is directed by only that channel’s events.

We implemented several sub-programs, most for controlling multiple homogeneous basic RGB lights since they are the most common fixtures, but a few also for controlling motion-capable fixtures such as scanners.

The former group of sub-programs are all based around a feedback loop, whereby the lights’ intensities are attenuated over time according to either a reciprocal curve or a sinusoid (depending on whether the last stream event was an Attack/Decay or a Sustain). The intensity is also attenuated by a smoothed noise function; the jitter determines the amplitude of the noise function, the constancy determines the smoothing factor. The hue is set according to the temperature.

The different sub-programs each have a preference over particular characters. The ‘Machine-Gun’ sub-program which, when given an Attack event, brings to full intensity the dimmest block (of size proportional to the strength) of contiguous lights, for instance, prefers events of character Aggressive, Chaotic and Pointed, reflecting its visual character.

The ‘Spread’ sub-program, which prefers Peaceful, Chaotic and Disparate events, places a smoothing function over neighbouring lights each instant, thereby giving the effect of a single light’s intensity peacefully spreading into neighbouring lights.

The ‘Dance’ sub-program (which controls motion) moves the scanner’s light-beam in a smooth, pseudo-random path. The randomness of the path is parameterised on the jitter and constancy of the previous event, and the speed is controlled by strength of the previous event; in the case of an Attack/Decay event, it is reduced according to a reciprocal curve, in the case of a Sustain event it remains uniform.

4. DISCUSSION & CONCLUSION

Creating truly resonant cross-media material is both difficult and time-consuming and creating it in real-time with little or no preparation is, in general, impractical. Through the use of an intermediate description language, it is possible to facilitate the transformation of perceptual events, though musical in sensation, to other non-musical media. Such a language can be leveraged in tools for the creation of cross-media art and in its performance.
We have demonstrated that the intermediate language need not be based strictly upon musical concepts for it to enable useful media transitioning. Furthermore, even a reasonably general language can natively express the information required to produce a notably resonant cross-media performance that would otherwise be comparable to a manually-designed show.

Through generalising the intermediate language, new avenues are opened with regard to human interfacing; rather than the operator of a light show dealing purely in low-level terms, either of audio (e.g. loudness and spectral envelope) or of lighting (red/green/blue values, gobos, shutter levels), they become free to deal with higher-level, abstract concepts, elevating the artistic possibilities of the performance accordingly and bringing them into the role of light-jockey.

In concrete terms, the main features of our presented system are:
- real-time processing;
- controllable by a user;
- automatically expressive of the music;
- clear and complex lighting suggestive of proper authoring.

The Music Access Platform is a software framework to facilitate the above aims. The client software can utilise the information to, either automatically or in a human-directed manner, direct a second process; e.g. to choreograph a dancing model, to generate abstract visuals or (as at present) direct a light-show. In summary, we have presented a cross-media performance aid with a powerful and extensible capability range and with general and useful output options.

4.1 Further Work

The scope of the present work is severely limited and affords only the most loose of conclusions. The intermediate language, as already stated, is almost certainly over-generalised and as such under-expressive. Further work is needed to determine what extensions can be made to add to the richness of the language without sacrificing its neutrality.

In our case study, we presented only a single cross-media application of the MAP. There are many more obvious examples of where the MAP could be applied with regards to both fine art and media creation in general. The act of designing and implementing such systems in terms of the MAP would help us to better understand and codify analogies between differing art forms.

Working with more powerful music information retrieval techniques would improve the fidelity of the cross-media translation. This would in turn benefits any and all applications of the MAP. Perhaps through additions to the intermediate language, work in this direction may also be used to guide any language extension, improving its expressivity.

5. REFERENCES


LightJams (2013) URL: goo.gl/zH6IF.

SOH. (2013) URL goo.gl/h5VEc.


SampleSumo (2013) URL: samplesumo.com

ZPlane (2013) URL: zplane.de

