Web Browser as Platform for Audiovisual Performances

Nuno N. Correia
Goldsmiths, University of London
n.correia@gold.ac.uk

Jari Kleimola
Aalto University, Dept. of Media Technology
jari.kleimola@aalto.fi

ABSTRACT
The present study aims to address the following research question: how to create a tool for audiovisual performance, allowing for real-time usage of shared online visual resources, which can be customizable, and used across a variety of different hardware platforms? To address this issue, we have developed AVVX (AudioVisual Vector eXchange), a novel application for audiovisual performances, based on open web technologies such as HTML5, JavaScript and SVG (Scalable Vector Graphics). This paper contextualizes AVVX with related work and technologies, and then presents the design and development of the software. Taking as a starting point a workshop conducted with AVVX, the project has been evaluated by means of a questionnaire and user tests. The results of the tests indicate that the web browser, together with open web technologies, can provide a foundation for a customizable, content-sharing and multi-platform approach to audiovisual performance.

KEYWORDS
Audiovisual; interaction; performing arts; media art; web browser; web technologies; vector graphics.

INTRODUCTION
The preparation of visual content for audiovisual performance and VJing (Video Jockey performances) is time and resource consuming, usually relying on commercial software and video files. Such tools often consist of proprietary software, limiting the possibilities for customization and content sharing. Additionally, most commonly used software for this purpose is not available across different platforms, particularly emerging platforms with potential for performance (due to portability and interactive capabilities) such as tablets. These issues raise the following research question: how to create a tool for audiovisual performance, allowing for real-time usage of shared online visual resources, which can be customizable, and used across a variety of different hardware platforms? Our hypothesis is that the web browser, together with open web technologies such as HTML5, JavaScript and SVG (Scalable Vector Graphics), can provide a foundation for a customizable, content-sharing and multi-platform approach.

To address this issue, we have developed AVVX (AudioVisual Vector eXchange)[1], an open source[2] novel application for audiovisual performances, based on open web technologies – HTML5, SVG, CSS and related JavaScript APIs (Application Programming Interfaces). Open web technologies have become sufficiently powerful to handle complex audio and visual manipulation. The usage of vector graphics files favors a 'less is more' approach, instead of relying on the manipulation of video files, as most tools for VJing do. Vector graphics are lightweight and easy to include in any project (particularly important in a web context and in resource-constrained devices). They are also scalable, and adaptable to any screen resolution without quality loss. An initial version of AVVX was released in 2012 [2], implementing a vector graphics approach. However, it was built on top of a closed source software platform. In February 2014, we released a new version of AVVX adopting a browser-based approach. This paper focuses on the new version of AVVX.

[1] http://www.avvx.org

[2] https://github.com/nunocorreia/avvx

RELATED WORK, CONCEPTS AND TECHNOLOGIES

Related Work and Concepts

There is a long tradition of using abstract geometrical shapes for audiovisual art. These artworks have demonstrated the potential of creating engaging content combining music with simple graphical elements. For example, Oskar Fischinger (1900–1967), one of the visual music pioneers, pursued a purely abstract approach in his animations. Fischinger was inspired by Bernhard Diebold, who called for "new artists, *Bildmusikers* [visual musicians]" to achieve Wagner's ideal of gesamtkunstwerk (total artwork), "preferably abstract in nature" [7]. This approach was also pursued by a new generation of artists influenced by Fischinger, such as John Whitney, a pioneer in the use of computer graphics for animation [4].

As personal computers became more powerful in the 1990s, real-time video manipulation became easier. In that decade, the term VJ became more common in the context of live visual performance – Golo Föllmer and Julia Gerlach place the emergence of VJing in the clubbing scene of the 1990s [3]. Chris Salter describes the emergence of "screen-based performance" in the 1990s, adopting "a long litany of names such as audiovisual performance, real-time video, live cinema, performance cinema, and VJ culture" as the result of two branches of techno-cultural development [9]. According to Salter, this emergence can be explained, on the one hand, by advances in digital computation, "particularly the development of hardware and software components for the capture, processing, and manipulation of image and sound" and on the other hand, due to "the international rise of the techno/club scene", which quickly exploited these technologies.

VJs have been using not only commercial VJ software such as ArKaos[3], Modul8[4] and Resolume[5], but also custom-made systems with software such as Processing[6], openFrameworks[7] and VVVV[8]. AVVX fits within the latter group, but adopting open web standards. Some audiovisual artistic projects have tentatively adopted web technologies, but mostly they have consisted of systems with limited visual customization capabilities, such as FMOL [5] and AV Clash [1]. Important advances have been made on using open web technologies to create tools for music making, such as [8], but there are fewer such explorations in the audiovisual field. More recently, efforts have been made to adopt tangible and multi-touch user interfaces for audiovisual performances [6]. Tablets have lately been explored for VJing, with the release of commercial applications targeting these platforms (such as Vjay[9] for the Apple iPad). AVVX also aims to be multi-touch and tablet compatible.

Open Web Technologies

Browser-based web technologies are standardized in the World Wide Web Consortium (W3C), whose latest efforts materialized in the recent release of HTML5 and its related techniques. The work of the consortium is continuing actively with diverse JavaScript API specifications. The canonical sources for these APIs are the W3C recommendations [10]. The recommendations relevant for this work, i.e., the audiovisual and interaction related APIs, are described briefly below.

The visual techniques comprise vector graphics and native video playback. The Scalable Vector Graphics (SVG) specification was released in HTML4, and the current browser implementations conform to version 1.1 of the specification [11]. SVG defines a declarative language for form and appearance markup, and an imperative JavaScript API for dynamic manipulation. AVVX uses the

---

[3] http://www.arkaos.net/

[4] http://www.modul8.ch/

[5] http://resolume.com/

[6] http://www.processing.org/

[7] http://www.openframeworks.cc/

[8] http://vvvv.org/

[9] http://www.algoriddim.com/vjay

declarative part as an interchange format, while the imperative part is utilized in the interactive animations (*behaviors*). Preliminary support for HTML5 video [12] is also included.

HTML5 enables native audio playback. The audio features of HTML5 are further refined in the Web Audio API specification [13], which describes a set of audio source and processor nodes, and their interconnection into stream-based node-link graphs. AVVX uses the media element and microphone input nodes as audio sources, and the spectral analysis processor node for audio-reactive graphics implementation.

Finally, HTML5 addresses interaction related topics with touch and mobile sensor APIs. The most widely supported multi-touch specification is [14], which offers low level access to the touch points on the surface. Gestures are excluded from the specification. AVVX user interface employs touch-aware buttons and sliders, and tracks horizontal and vertical swipe gestures using the Hammer.js[10] library.

PROJECT DEVELOPMENT AND DESCRIPTION

Project in Use

The target audience for AVVX is mainly artists and designers with little or no VJing experience; therefore ease of use is a priority. Preparing visuals for AVVX involves creating the SVGs and indexing these in a XML file, aggregating them into *groups*. These groups (up to 100 are supported) can be different frames that will compose an animated sequence, or merely related graphics that the user decides to join together. The XML file also supports metadata for the images, such as authorship information. Upon launching AVVX, the XML file and associated SVG files are loaded. At the moment, four animation *behaviors* are available in AVVX: *slide* (horizontal or vertical translations); *zoom* and *zoom out* (scaling with rotation); and *trail* (random movement of graphics across the screen). These behaviors take advantage of basic bi-dimensional vector graphics transformations: scale, translate and rotate. Users can manipulate different parameters of these animations, such as speed and direction (see Figure 1).

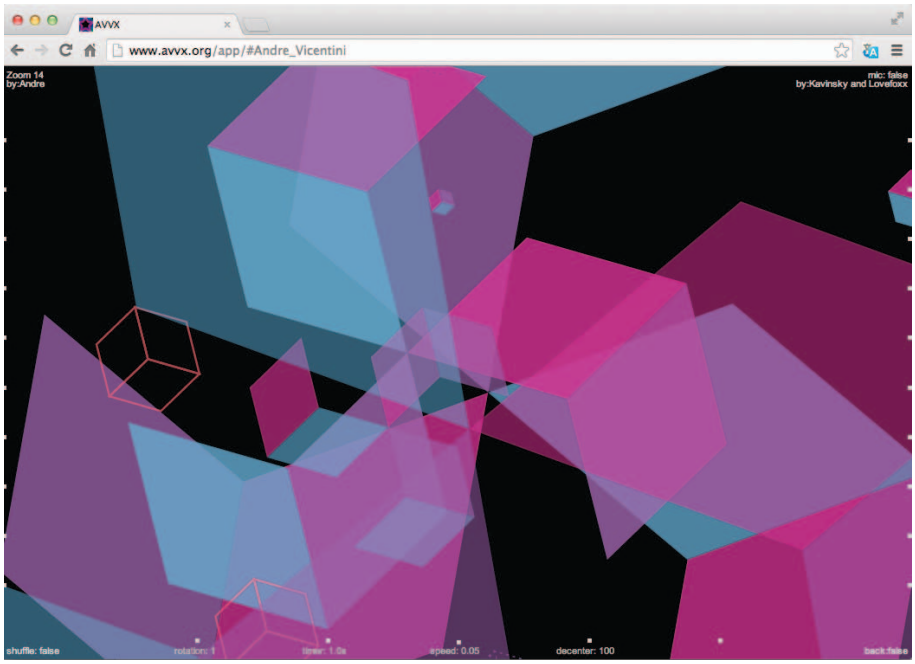

Figure 1: Screenshot of AVVX. The minimal AVVX GUI with textual info, buttons and sliders is shown on top of graphics.

Behaviors and their parameters can be changed using the keyboard interface or a GUI (Graphical User Interface). The GUI is minimal – small buttons, sliders and text positioned along the edges of the

[10] http://eightmedia.github.io/hammer.js/

screen, in order to be unobtrusive and not interfere with the graphical content. Users can also choose which SVG group to select. All functionalities are available in both keyboard and GUI interfaces. An on-screen display gives feedback about the current behavior, parameters and groups, in addition to author information (name and URL of the author of the visuals can be added to the XML). This textual information and can be hidden with a key press. Sound controls are also available, for MP3 playback or microphone input. The animations are audio-reactive – the reactivity consists of scaling the animations in proportion with the sound amplitude (in addition to other scaling animation taking place).

Architecture and Technical Development

An initial version of AVVX was originally developed in 2012 [2] using Adobe Flash, as a stand-alone application. Four workshops took place between 2012 and 2013 using this version. However, it became clear that the approach of using Flash was limiting, because: 1) Flash is not well supported across multiple platforms, notably in Apple's iOS devices; 2) being a stand-alone app instead of browser-based limits its portability and the possibility of loading content directly from the AVVX library; and 3) Flash is not free and open source, which creates restrictions, namely for artistic or teaching purposes.

The implementation of the present AVVX version consisted of porting, redesign, and deployment phases. The existing Flash ActionScript implementation was first ported into JavaScript as is, in order to ensure that the performance of the browser is on par with the Flash runtime. Since ActionScript and JavaScript are both dialects of the ECMAScript language, the porting phase was trivial. As expected, the SVG implementation required most effort. The Flash version employed a dedicated SVG library, but after exploring several JavaScript alternatives, we decided to implement the SVG functionality directly on top of the W3C API. Since the vector graphics are crafted outside the browser, and since their dynamic manipulation is algorithmically straightforward, the benefits of the higher level libraries were overshadowed by their processing overhead. XML parsing and audio playback/analysis functionality was written from scratch on top of the W3C APIs as well. The direct port ensured that the browser-based implementation was feasible, and after minor optimizations, the performance of the browser (Chrome v32) was comparable to that of the Flash version.

We then progressed into the second phase to redesign the AVVX framework. Our design goals were: 1) to make behaviors extensible, 2) to converge input modalities, and 3) to find an optimal latency vs. performance ratio for audio-reactive graphics. A simplified block diagram of the framework is shown in Figure 2.

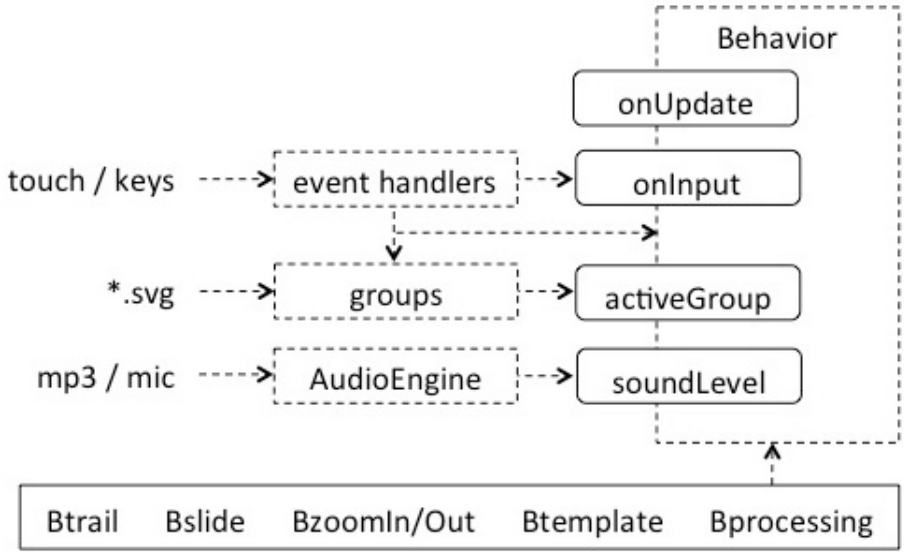

Figure 2: AVVX framework.

To achieve extensible behaviors, we first analyzed the existing behavior implementations. Common functionality was converged into the abstract *Behavior* superclass, which defines an interface that all

individual behaviors implement. The interface includes *onUpdate* (invoked at 60 fps) and *onInput* (invoked in response to user interactions) callback functions, as documented in the *Btemplate* class. *Btrail*, *Bslide* and *Bzoom* behaviors were re-implemented to conform to the interface. The extensibility was assessed with a new *Bprocessing* class, which implements preliminary support for bitmapped graphics.

The existing AVVX keyboard interface was augmented with touch-aware buttons, sliders and gestures. The interaction events are first interpreted in low-level event handlers, and then routed to the active behavior's *onInput* function. Key presses and button taps are routed as triggers with a unified command code as a parameter. The remaining touch interactions are routed as sliders and swipes, the first with sliderID/value pair, and the latter with the swipe direction as a parameter. The framework interprets behavior and image group swaps internally, but otherwise, the behaviors may respond to the interactions independently.

The AudioEngine class enables mp3 file and/or microphone playback, and provides a spectrum analyzer to approximate the instant spectral energy of the audio sources. The analyzer divides the frequency range (0…24 kHz) into 1024 bins, and computes an average magnitude across all the bins in real time (2048 samples of audio are processed for each computed average). The worst-case latency for the instant spectral energy acquisition is therefore 60 ms, including the latency caused by the 60 Hz screen refresh rate. Although smaller latency would allow more responsive audio-reactive graphics, it would also reduce the performance of the vector graphics rendering engine. We found the 2048 buffer size optimal for our implementation. The instant value is available in the active behavior as *soundLevel* property.

Finally, the deployment phase of the present implementation addressed both local and cloud-based configurations. AVVX requires an HTTP server to access the media configuration file (media.xml) and the actual media (SVG, audio and video) files. In local configuration, AVVX runtime is downloaded on a laptop, and a localhost server is set up to serve the environment. The cloud-based configuration avoids local server and AVVX runtime installation, but requires the SVG files to be uploaded to the cloud server before starting the application. AVVX may also be launched from the avvx.org website using predefined graphic and media.xml setups[11].

EVALUATION AND RESULTS

Methodology

A workshop was conducted between the 11th and 13th February 2014 in Helsinki, to disseminate and test the new version of AVVX. During the workshop, participants learned how to use AVVX, and prepared an audiovisual performance for the last day of the workshop. Preparation involved creating vector graphics (SVGs) to be used in AVVX, and code customization. 21 persons registered for the practice part of the workshop, of which 18 concluded and performed (see Figure 3). For the final performance, participants were asked to select one music track, and create visuals for that track. During the performance, each participant created live visuals with AVVX, based on the graphics created beforehand. The performance took place in a bar in Helsinki (Bar Sandro), a “real world” setting for a live visuals performance (see Figure 4). In terms of further outcomes of the workshop, 12 of the 18 performing participants gave permission for their graphics to be released online, at a specially created repository of SVGs[12], under a Creative Commons attribution license.

[11] For example: http://www.avvx.org/app/#Tatiana_Toutikian

[12] http://www.avvx.org/library/

Figure 3: AVVX workshop, February 2014.

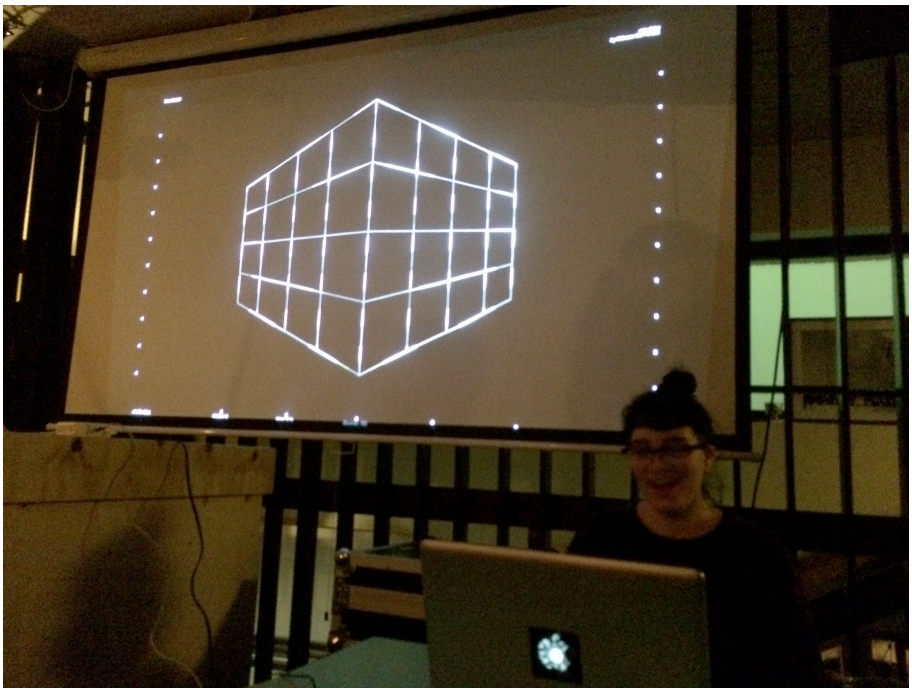

Figure 4: AVVX workshop – final performance.

A questionnaire[13] and user tests were conducted after the workshop to evaluate how effective AVVX was as a tool for live visuals. In particular, the user tests aimed to assess the multi-platform capabilities of AVVX, and the appeal of accessing online content. The questionnaire was answered by 12 out of the 18 workshop attendees. Three of the workshop attendees volunteered for user tests. The user tests focused on using tablets (Apple iPads) and accessing online content. Each of the three users spent 10 minutes playing with their own visuals in an iPad (see Figure 5). Then they spent 10

[13] https://docs.google.com/spreadsheet/viewform?formkey=dEFUdnZvencwcFBfUnpMV0ZKRUxLc1E6MA

additional minutes accessing, and playing with, other visuals from the online library. This was followed by a 30-minute interview with the three volunteers.

Results - Questionnaire

The respondents were all students (of interactive media, design, music or media arts), with the exception of 2, which were professional designers. Only one had previous experience as VJ. The majority (9 out of the 12) of the respondents answered that they would use AVVX again, and 11 would recommend the tool to someone interested in live visuals. Answering using a 5-point Likert scale, 6 out of the 12 respondents considered the audiovisual end result in the final performance engaging and 3 very engaging, with the remaining 3 being neutral about the result. Most of the test users (8 out of 12) considered the release of AVVX as open source as being very important, with 2 considering it important, and 2 neutral. The majority of the respondents consider AVVX to be easy (5 out of 12) or very easy (2 out of 12) to use. Nevertheless, 3 respondents still consider it difficult to use. The dual-interface approach of AVVX seems to have been successful – 7 out of the 12 test users consider it very useful to have two types of interface in AVVX (keyboard and GUI), while 3 respondents consider it useful (see Table 1). Answering a multiple option question, most (7 out of 12) prefer to use the keyboard, while 5 use equally both keyboard and GUI.

Figure 5: User testing

Likert scale	1	2	3	4	5
Audiovisual result - engaging?	0	0	3	6	3
Open source - important?	0	0	2	2	8
Ease of use - easy?	0	3	2	5	2
Dual interface - useful?	0	1	1	3	7

Table 1: Number of responses obtained per point in the Likert scale, from 1 – 'not at all' to 5 – 'very'

Results - User Tests

When asked if AVVX is a good tool for live visuals, and what are its strengths and weaknesses, all three interviewees were pleased with the usage of SVG graphics. One of the interviewees mentioned "I think the highlight of this tool that makes it very different, or the signature of this tool, is the usage

of SVG files", which "gives you the possibility to always have high quality with smooth edges, no matter like where the display would be". Another test user confirmed that "the input from the SVG file is really, really important" and "quite nice". The third interviewee highlighted ease of use: "the ease to use is the most interesting aspect, and also the vector graphics".

The GUI, and to a certain extent the way content is organized in AVVX, is what the interviewees are most displeased with: one test user mentioned that the interactive elements "are quite small in a way that it may distract me" and suggests that AVVX should have "two different screens", one for the GUI and another for the graphical output. Another interviewee considers that "the hierarchy of the program still needs a lot to be worked on" and that navigating through content is "difficult unless you have mastered it, and then you have memorized it pretty well". The last interviewee agreed with this observation, mentioning "you don't rely on anything besides your fingers or your memory", since the information in the corner of the screen "is too small to see", and added "I'm getting the feedback straight from the animation". In addition to these negative factors, one of the interviewees was also displeased with the occasional slowness of the software when dealing with more complex graphics.

The interviewees were then asked about multi-platform issues: is it relevant to have the tool available across platforms, such as laptops and tablets, and is it better suited for some platforms than others? All users agree that the laptop is preferable for preparing performances, with one preferring the laptop for live performances and the remaining two the iPad. Nevertheless, the latter two stressed that the user interface in the iPad should be further developed. One of the test users considers that both laptop and iPad "worked out for me", although he would rather use a laptop in a performance because he could "move around and bring a lot of different things at the same time". Another interviewee stated that the computer was better suited to build and test the visuals, but that she would prefer to use the iPad for performances since "it's more intuitive" and allows to communicate better with the audience, adding that the laptop "blocks" this communication. The third test user agrees with these views, adding that he would prefer the iPad more if the iPad interface would be better implemented.

Regarding preference for type of user interface with the laptop – keyboard or GUI – one interviewee stated that he uses both, with the remaining two expressing preference for the keyboard. The first test user mentioned that he uses the "right hand on the mouse and the left hand on the keyboard", but that the keyboard gives him further reassurance: it "would kind of assure me that I had pressed something and something will happen". The remaining two interviewees stated that using the mouse is slower, therefore they prefer the keyboard interface.

Finally, interviewees were asked if it is relevant to have access to online visual resources, and if the access to those resources is well implemented. One of the test users does not consider it very relevant as he is not "a fan of using other's visuals", while the other two interviewees consider this to be an important feature. One of the test users considers this to be relevant in an open source project as AVVX, as it favors remixing: "one thing can be kind of like replicated in a million different ways". She added that it would be important to allow for tagging of content, in order to find specific visuals more easily in the online database. The third interviewee agreed with these views, emphasizing that the online visuals reinforce the open source nature of the project, and adding that it would be appealing to search visuals "either by tag cloud or by looking at thumbnails".

CONCLUSIONS

The development and testing of AVVX have confirmed the hypothesis that the web browser, together with open web technologies, can provide a foundation for a customizable, content-sharing and multi-platform approach to audiovisual performance. The evaluation has shown that test users were satisfied with the end result when performing with the tool; that they value its open source approach; as well as its multi-platform, dual-interface approach (despite a preference for keyboard, and limitations in the tablet implementation). The interviews in particular revealed that the laptop is preferred for preparation, while tablet is best suited for performances; and that the adoption of an SVG-based vector graphics approach had been successful.

The study also reveals that work remains to be done in order to make AVVX easier to use, particularly regarding tablet usage and GUI. The GUI should be further developed to provide better feedback and

content previewing, eliminating the need to rely excessively on memory and practice. Additionally, further work should be done to transform the AVVX library[14] into a more useful repository for SVGs. Functionalities for uploading, tagging and previewing visuals should be added. Visuals could then be loaded into AVVX according to tag, and not just to author.

In addition to improvements arising from results of the current study, our future work aims to increase the expressivity and integration aspects of the AVVX platform. The expressivity can be increased in the sonic domain by inserting parameterized audio processing nodes – such as low pass filters and phaser effects – into the audio processing chain. The visual domain may be extended with custom behaviors, such as those based on 2D bitmap graphics and 3D WebGL models. The interactive properties could be expanded with touchless gestures acquired through W3C mobile sensor and pointer APIs. Finally, integration with existing audio streaming services would enable engaging active music listening scenarios.

The present study confirms that open web technologies are currently powerful enough to be used in audiovisual performances. Browser-based web technologies have the potential to open up new creative possibilities in the audiovisual domain, allowing for the usage of new types of devices and for sharing online content, enabling collaboration between artists and the reuse of materials.

ACKNOWLEDGMENTS

We would like to thank Aalto Media Factory for their support regarding the software development and workshop. We also thank the participants. This research was supported by a Marie Curie Intra European Fellowship within the 7th European Community Framework Programme.

REFERENCES

1. Correia, N.N. AV Clash, Online Audiovisual Project: A Case Study of Evaluation in New Media Art. *Proc. ACE 2011*, ACM Press (2011).
2. Correia, N.N. AVVX: A Vector Graphics Tool for Audiovisual Performances. *Leonardo Electronic Almanac 19*, 3 (2013), 134–147.
3. Föllmer, G. and Gerlach, J. Audiovisions. Music as an Intermedia Art Form. *Media Art Net*, 2005. http://www.mediaartnet.org/themes/image-sound_relations/audiovisions/.
4. Goodman, C. *Digital Visions: Computers and Art*. Harry N. Abrams, New York, 1987.
5. Jordà, S. FMOL Toward User-Friendly, Sophisticated New Musical Instruments. *Computer Music Journal 26*, (2002), pp. 23–39.
6. Lew, M. Live Cinema: Designing an Instrument for Cinema Editing as a Live Performance. *Proc. NIME 2004*, NIME (2004).
7. Moritz, W. *Optical Poetry: The Life and Work of Oskar Fischinger*. John Libbey Publishing, Eastleigh, 2004.
8. Roberts, C., Wakefield, G., and Wright, M. The Web Browser As Synthesizer And Interface. *Proc. NIME 2013*, NIME (2013).
9. Salter, C. Entangled: Technology and the Transformation of Performance. MIT Press, Massachusetts, 2010.
10. W3C. All Standards and Drafts. http://www.w3.org/TR/.
11. W3C. Scalable Vector Graphics (SVG) 1.1, Second Edition. http://www.w3.org/TR/SVG11/.
12. W3C. HTML5: Edition for Web Authors – The Video Element. http://www.w3.org/TR/html5-author/the-video-element.html.
13. W3C. Web Audio API. http://www.w3.org/TR/webaudio/.
14. W3C. Touch Events. http://www.w3.org/TR/touch-events.

[14] http://www.avvx.org/library