Synesthesia Add-on: a Tool for HTML Sonification

Roberto Piassi Passos Bodo[1*], **Flávio Luiz Schiavoni**[2]

[1] Institute of Mathematics and Statistics (IME)
University of São Paulo (USP)
São Paulo, São Paulo, Brazil

[2]Department of Computer Science (DCOMP)
Federal University of São João Del Rei (UFSJ)
São João Del Rei, Minas Gerais, Brazil

rppbodo@ime.usp.br, fls@ufsj.edu.br

Abstract

Web browsers using HTML5 and WebAudio have been widely used as real-time audio environment and brought up new possibilities for web art. In this paper, we present an investigation on HTML sonification. In our approach, HTML pages are read as musical scores and page elements are played as a sequencer. We developed a tool for website sonification which can be used to explore musical creativity and to create new sound contexts based on the web pages. We present the tags and attributes that are mapped to sound parameters. In addition, we show how was created sounds and visual feedback in this tool.

1. Introduction

For long, Internet has been used to distribute art pieces through virtual Museums and virtual Art Galleries. Although these are a great relationship between art and this technology, there are more possibilities to art on the web than being a place to art commerce, release or distribution.

From the very first moment that images went out over the Web, artists have been using the web as an art medium and not just a new way to publish information. It can be used with artistic purpose in a response to the digitalization of cultural forms and the information technology revolution [1] [2]. This art instance, called **web art**, is about art works made specifically for the web, available all the time, every place and everywhere to several distinct participants [2].

Historically, Web Art is probably a continuation of Media Art where the viewer is not only part of the audience, but a participant in that experience.

An important aspect on Web Art is about the processes and tools used to this art instance. Web Art is relatively inexpensive to produce because HTML is a free language, HTTP is a free protocol and web browsers are free tools. It makes this art format very accessible to digital artists[1].

Since the web browser is the web art medium, a basic construction on Web Art usually runs over the Hyper Text Markup Language (HTML). HTML is a fast evolving language, supported by a considerable community of developers, which evolution demanded the incorporation of new tags and routines in its last version, namely HTML5. Apart from several news, HTML5 brought to the web a sound engine, called WebAudio API[3]. Before HTML5 it was already possible to play audio files in a web page. The WebAudio API brought the possibility to create and process real-time audio in the web browser. The browser is now a real-time audio rendering tool and it gave new possibilities of interaction and feedback to Web Art.

In this paper, we are using this APT to extend the browser capability to artistically sonify every HTML page using the WebAudio API. Classically, a goal of sonification is to transform complex multidimensional data in intuitive audio [4] [5] (as in text-to-speech or web accessibility tools).

Alternatively to this approach for sonification, our goal here is to create purely aesthetic sounds

*Supported by CAPES.

that can be used to inspire compositional process or just be enjoyed by the user.

The remainder of the paper is organized as follows: Section 2 presents the idea of Web pages sonification, Section 3 presents the project main implementation, Section 4 presents initial results, and Section 5 presents Conclusion and Future Works.

2. About Web Pages

In the beginning of the Internet era, a web page was a plain text, black on white, unformatted and very informative document. In fact, there are several old websites which the content is presented in this format even today. Despite the fact that websites changed a lot from this initial layout, this format keeps the content easily rendered by a text-to-speech converter or a Braille device (and accessibility is the most common goal in web sonification [6]).

Such tools will read only the text content and will ignore the visual aspects of the website, namely a set of invisible tags and attributes whose settings will define the page style.

Nowadays, we consider a traditional web page as a triple of HTML, CSS, and JavaScript resources. Basically, HTML is responsible for the page content, page structure, CSS for styling (positions, sizes, colors, etc), while JavaScript is used for dynamic actions in client side (interactions, animations, etc).

In our project, we are ignoring the page content. We approached a web page as a music sheet and, with that in mind, the first idea we had was to use page structure as music structure and page styling as some kind of flavor for synthesis.

3. Implementation

Since the genesis of this project, we wish to allow users to sonify any site on the web. Naturally, host our code together with all the sites around the web is not possible, so we have decided to extend the browser adding to it a sonification capability.

Browsers can be extensible by an add-on, a kind of plugin written in JavaScript, and users can install add-ons adding more functionalities to the browser.

To ensure cross-browser compatibility we choose the WebExtensions system [1] which is compatible with Google Chrome and Opera natively, and with Firefox and Microsoft Edge after a few modifications.

With the add-on we could run our JavaScript code in any website, allowing users to explore sonification ideas around the web. The new add-on added our functionality to the web browser, as presented on Figure 1.

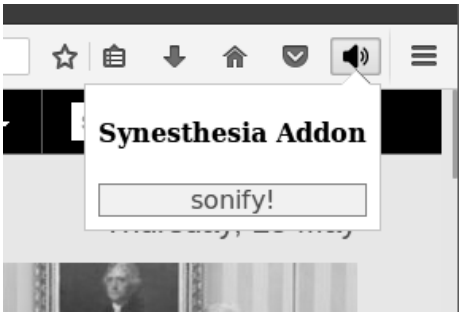

Figure 1: Screenshot of the add-on

3.1. Collecting information about websites

The HTML language has the peculiarity of enabling two sites with completely different code structures to have the same visual, and two sites with very similar source code to have completely different visual. Furthermore, it is not easy to determine which HTML tags are used in a website just by observing the rendered page.

To have a better view about which tags and attributes are common and how we could sonify it, we decide to first collect information about sites. We analyzed the HTML structure of a few websites generating statistics of all the information we were interested in.

More specifically, we made a JavaScript code that goes down recursively on the page structure from the <body> element gathering all useful information and saving it in an array (making a linearization of the data).

For instance, this code calculates a histogram of tag occurrences. We selected the top 10 tags

[1] Available on `https://developer.mozilla.org/en-US/Add-ons/WebExtensions`

for `http://bbc.com` and `http://cnn.com`, and presented them on Figure 2 and 3.

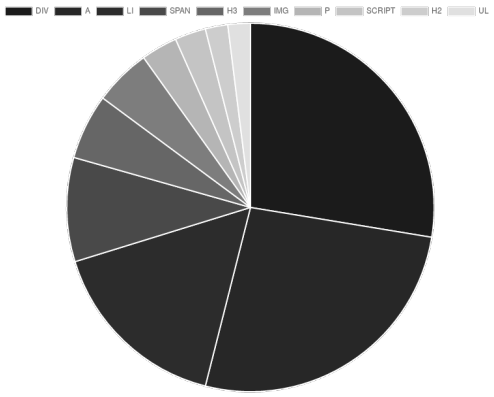

Figure 2: Most used tags of BBC site and the proportion of their occurrences.

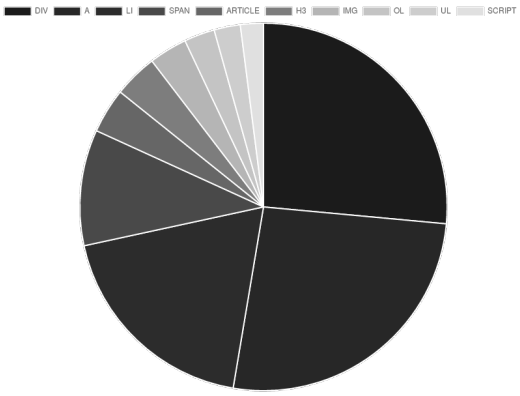

Figure 3: Most used tags of CNN site and the proportion of their occurrences.

According to these statistics, the most used tags in these websites are DIV, A, LI, and SPAN (other tags appears more rarely). It gave us a good tip about which HTML tags we could use in the sonification process.

When an element is selected to be sonified, we only have its name and where it belongs on the HTML tree, and we need more information about it to map to the synthesizers parameters. So, we decided to investigate all of the style attributes that are associated with the HTML element.

Analyzing the CSS sources is an arduous task, because we can have more than one class by element, more than one file defining these classes, and rule overwriting due to inheritance (the latter is the worst case).

So we decided to work with computed style in DOM. For this, there is a JavaScript function called *getComputedStyle()* that returns the values assigned to hundreds of style attributes. This large number was, to us, a new challenge due to the need to select a subset of all of this information. We considerate the so-called box model to solve this problem.

Every modern browser has in its developer tools a graphic visualization of an element by the terms of the box model. It considers that every element behave like a box. When you inspect an element, this graphic shows the computed dimensions (width and height) beside other attributes like padding (internal gap), margin (external gap) and border. Some browsers also show the top and left positions of the element.

With this in hand and with the hypothesis that these are the most used attributes in a page, we move on to the step of fetching elements from real sites and taking statistics from their styles. Tables 1 and 2 present the statistics of the same two websites mentioned above.

Table 1: BBC global statistics of styles

attribute	avg	std	min	max
width	343.83	820.57	0	25000
height	138.57	420.53	0	7589
top	1407.89	4013.39	-100000	7563
left	124.13	3014.81	-100000	1600
padding	19.38	56.80	0	702
borderWidth	0.11	0.84	0	18
margin	3.88	13.46	-224	88

Table 2: CNN global statistics of styles

attribute	avg	std	min	max
width	198.98	300.08	0	1348
height	53.89	120.86	0	1899
top	543.99	1051.76	-10000	2844
left	147.54	289.62	-100	1348
padding	4.97	29.78	0	639
borderWidth	0.12	0.75	0	8
margin	-0.66	30.05	-626	100

3.2. The HTML Score

HTML is a hierarchical document where tags are organized in a defined order. We can look for a complete HTML structure as a tree with nodes and their children, having the <body> tag as the

root. In our sonification project, the order of the elements defines the order of notes reproduction and the attributes of the elements define the notes configurations.

For instance, the pitch of the note can be defined by one attribute; the duration by other; the amplitude by yet another (specific information about mapping can be found in subsection 3.3).

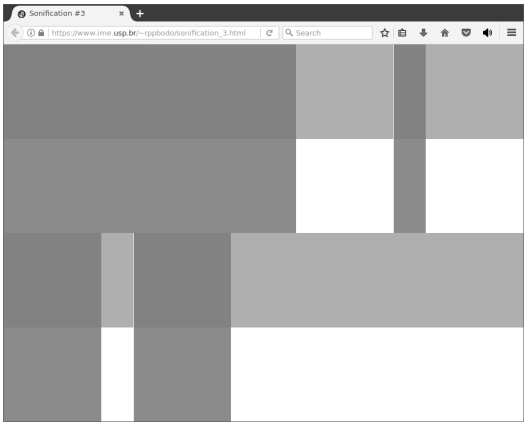

Figure 4: Example page where each block is mapped to a note in our sonification add-on.

What is important here is that was created a voice for each tag. Let's imagine a synthesizer playing notes for each rectangle on the web page shown in Figure 4 (implemented only with divs). When we map width to pitch and height to duration, we will have notes such as those in Figure 5. If we had a more complex page with more tags, we calculate the most frequent ones and sonify each tag using a different synthesizer.

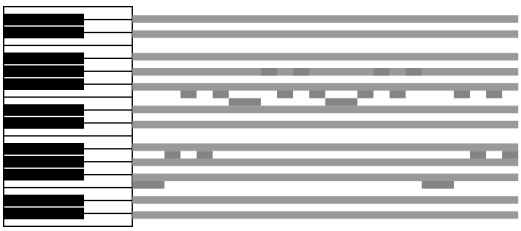

Figure 5: Visualization of notes in a piano roll after the sonification of the page presented in Figure 4

Observe that the notes in the piano roll do not have rests between them. This happened because when the time of a note offset is reached, the next one was started immediately. But this behavior is not mandatory. We can map any attribute to notes' starting times and with this we can produce rests between notes. This will work because the starting time of a note, added with its duration, not necessarily will reach the starting time of the next one.

Even more, with this onset time mapping, we can produce polyphony also. Depending of which attribute is mapped to onset time, we can have several notes beginning at the same time. The reader is already able to perceive the numerous behaviors that the add-on will have according to all possible mappings.

The notes' scheduling was implemented using a JavaScript method called *setTimeout()* that receives a callback function and a time value. With this method we were able to schedule all of the notes onsets. The offsets were handled by the synthesizers themselves (we passed the duration to *startNote()* prodecure).

This scheduling is totally dependent of a parameter that we called "virtual clock relative speed". The concept of virtual clock was presented by Roger Dannenberg in 1984 [7] in the context of Automated Musical Accompaniment. Basically, it is a clock that evolutes by a rate of the real clock. This rate is defined by the parameter mentioned above, which is a floating point number that will determine the speed of the execution. For instance, if it is set to 0.5, the speed it will be half of the original, if it is set to 2, it will be the double, and so on.

The most exciting thing about this parameter is not the capability of changing speed, but the capability of changing speed by attribute mapping. We could map body's background color to the speed and have darker pages slower than brighter ones. Or we could have variable speed throughout the sonification. This can be achieved mapping one attribute from the current sounding element to dynamically change speed for the next one. This can lead to totally unanticipated behavior, but this is considered a positive feature of the add-on for us.

3.3. Mapping HTML attributes to sound information

Initially, we did not know which attribute would be mapped to which synthesizer parame-

ter. So, using the extracted statistics, we detected the minimum and maximum values for each attribute and mapped the actual range of values to [0, 1].

With all the values between 0 and 1, we could map any of them to any synthesis parameter using any linear function (to adjust the value ranges). In our first tests we realized that some style attributes are most commonly used than others, even when we work with the set of the box model attributes.

For instance, borders are not frequently used (lots of elements had borderWidth equals to zero). Paddings and margins are the next ones in number of zero occurrences, followed by top and left. The most frequent computed style from our selection are, definitely, width and height.

When we use width and height in a parameter we had a greater variety; when we use borderWidth in this same parameter we had lots of equal behaviors. So we had two options: combine parameters (and, thus, have the less variable one working as a harmless modulator) or simply avoid setting these less frequent attributes to core synthesizers parameters (such as, pitch or duration).

3.4. Building Synthesizers

For the audio to be synthesized in real-time on the browser itself, we used WebAudio to implement the synthesizers. We tried to select techniques that cover various different timbres. Six instruments were developed for our add-on: Block, Drum, Guitar, Harpsichord, Maraca and a Pong.

All of them were implemented with the same interface so that they could be easily alternated. The pitch parameter is a MIDI note number, the duration is in milliseconds and amplitude is a value between 0 and 1 that determines the maximum in the note envelope. For illustration, the current mappings are presented on Table 3.

Table 3: Current properties mapping to sound attributes.

Synth	pitch	duration	amplitude
1	32 + W * 32	500 + H * 500	T
2	32 + L * 32	500 + P * 500	T
3	32 + H * 32	500 + W * 500	L
4	65 + L * 65	1000 * P * 5	M
5	32 + L * 32	500 + P * 500	B
6	44 + W * 32	500 + H * 500	L

W = width / H = height / T = top / L = left
P = padding / M = margin / B = borderWidth

3.5. Visual Feedback

In our first implementation we realized that there were no visual feedback from the add-on. Because of it, it was almost impossible to precisely affirm which element was being sonified at each time. To solve it, a CSS class was created to visually highlight the elements that were emitting sounds at a moment.

First, we tried to change colors, but some elements did not accept that (the concept of background color doesn't match with an image tag, for instance). After that, we tried to add a very noticeable border, but that broke the layout.

Borders has width and some pixels were added in the overall width of the elements. With this, some side-by-side elements became misaligned or, even worse, did not fit at all on their original positions.

The solution was to work with 3D transformations that did not change the original layout and can be applied in any element. We implemented an animation that shakes the element using small translations from side-to-side, making one element quite prominent in the page.

With this visual feedback we found another issue: elements present in the HTML but invisible on the page were being sonified. For instance, the page can have an image carousel that has several images but only one is visible or it can have a dropdown menu that has its items hidden. We found several instances of this problem and we found very difficult to came up with a solution suitable to all cases.

Just to soften this issue, we implemented a heuristic to check if an element is visible. It has 3 steps: a) calculate the top-left and bottom-right corners of the element; b) find out what are the visible elements of the page on these points; c) check if both are the same original element.

We know that this heuristic is not perfect, be-

cause it does not solve, for instance, the case of a N, M object on top of a $N + 1, M + 1$ object with 1px offset (we consider that the one below can not be seen), but it solves most of the rough cases.

With all of these visual issues surpassed, we ran the sonification add-on on the same websites that we extracted the statistics, and we were really satisfied with the audible results.

4. Results

Our first result is a statistics add-on, which can generate all statistics mentioned in Section 3.1 to any website, and help musicians and sonifiers to collect information about web pages. This add-on generates images for some statistics and for the piano roll, and can generate LaTeX tabular code also. This add-on is available on `https://github.com/rppbodo/statistics-addon`.

The second and main result is the sonification add-on itself, a sequencer that plays HTML pages as musical scores and allow users to listen page structure. It has 6 instruments in JavaScript / WebAudio with a common interface. These instruments uses different synthesis techniques like FM, AM, and additive. The add-on source code is available on `https://github.com/rppbodo/synesthesia-addon`.

5. Conclusion

In this paper, we presented a novel form to create music based on HTML pages by the means of a sonification add-on. For this, we have chosen to use the structure of HTML websites, instead of the content, to control production through synthesizers.

We presented our strategies to sonification, the statistics used to create the strategy and several considerations about our development process. Certainly, our strategies included subjective choices and the process counted on a previous experience to create synthesizers and choose good parameters mapping the page elements to sound.

As future work, we intend to explore websites musically and research how it can collaborate to music creativity on composition and improvisation. In the present implementation we are consider only the page structure but it could be interesting to sonify text nodes too. We are considering to create a third tool to achieve text sonification.

5.1. Acknowledgment

We would like to thank the Computer Music Research Group[2] and the Sonology Research Center (NuSom)[3] at the University of São Paulo, and the support from CAPES[4].

References

[1] Mark Tribe, Reena Jana, and Uta Grosenick. *New media art*. Taschen London and Cologne, 2006.

[2] Annette Weintraub. Art on the web, the web as art. *Commun. ACM*, 40(10):97–102, October 1997.

[3] Chris Rogers. W3c webaudio api. `https://www.w3.org/TR/webaudio/`, 2015. Accessed: 2017-07-01.

[4] Oded Ben-Tal and Jonathan Berger. Creative aspects of sonification. *Leonardo*, 37(3):229–233, 2004.

[5] Thomas Hermann, Andy Hunt, and John G Neuhoff. *The sonification handbook*. Logos Verlag Berlin, 2011.

[6] Lori Stefano Petrucci, Eric Harth, Patrick Roth, André Assimacopoulos, and Thierry Pun. Websound: a generic web sonification tool, and its application to an auditory web browser for blind and visually impaired users. *Proceedings of ICAD 2000*, pages 6–9, 2000.

[7] Roger B. Dannenberg. An on-line algorithm for real-time accompaniment. In *ICMC*, pages 193–198. Michigan Publishing, 1984.

[2] http://compmus.ime.usp.br/

[3] http://www.eca.usp.br/nusom/

[4] http://www.capes.gov.br/