Web Sonification with Synesthesia Tools

Roberto Piassi Passos Bodo (University of São Paulo, São Paulo, São Paulo, Brazil)
rppbodo@ime.usp.br
Flávio Luiz Schiavoni (Federal University of São João del-Rei,São João del-Rei, Minas Gerais, Brazil)
fls@ufsj.edu.br

Abstract: In this paper, we present an investigation on sonification of web pages involving three media: HTML structure, page text and images. In our approach, HTML pages are read as musical scores; text elements are also read as musical scores following the idea present in notation systems like ABC; and images are read as wave files. We developed a set of tools for website sonification which can be used to explore musical creativity and to create new sounds based on the web pages content. In addition, we show how to create sounds and visual feedback using this tool.
Keywords: Sonification. HTML5. Web Audio. Aesthetics in sonification.

Sonificação da Web com o Add-On Synesthesia

Resumo: Neste artigo, apresentamos uma investigação de sonificação de páginas web utilizando para isto três mídias: estruturas HTML, textos de páginas e imagens. Em nossa abordagem, as páginas HTML são lidas como partituras musicais; os elementos textuais também são lidos como partituras seguindo a ideia presente em notação textuais, como o formato ABC; e imagens são lidas como arquivos de som. Um conjunto de ferramentas foi desenvolvido para explorar a criatividade musical e criar novos sons baseados no conteúdo de páginas web. Além disto, apresentamos como criar sons e o feedback visual utilizando estas ferramentas.
Palavras-chaves: Sonificação. HTML5. Web Audio. Sonificação estética.

1. Introduction

From the very first moment that images went out over the web, artists have been using the web as an art medium. It can be used for artistic purposes as a response to the digitalization of cultural forms (JANA; TRIBE, 2006; WEINTRAUB, 1997). This art instance, called web art, features art works made specifically for the web, available on web browsers by web servers.

Historically, web art can probably be a continuation of media art mixed with interactive art where the viewer is not only part of the audience, but a participant in that experience (SEEVINCK, 2017). This experience is possible because of the fact that the web is an interactive media and users can react to a programmed environment, such as an installation, with the advantage of running it on their personal computer.

Web art is relatively inexpensive to produce because HTML is a free language, HTTP is a free protocol and web browsers are free tools. The main cost of a Web art project is about the time involved in developing a well-designed project. This makes this art format very accessible to digital artists (JANA; TRIBE, 2006). Also, most file formats running over the web are open, like HTML, JPG, BMP or WAVE. Thus, it is possible to use these file formats as interchangeable data between several different applications.

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. HTML5 brought to the web a sound engine, called Web Audio API (ROGERS, 2015). Before HTML5 it was already possible to play audio files in a web page, but the Web Audio 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 this capability brought new possibilities for Web Art.

 Recebido em: 27/11/2017 - Aprovado em: 05/03/2018

These possibilities have been explored by several artists and there are now several initiatives to produce real-time audio and music on the web using the Web Audio API. Examples of these initiatives can be found on Csound on the web (LAZZARINI et al, 2012), FAUST for web (LETZ et al, 2015) and the web based language / programming environment Gibber (ROBERTS; KUCHERA-MORIN, 2012). Discussions about web possibilities can be found on (WYSE; SUBRAMANIAN, 2013) and (ALLISON et al, 2016).

Another fact that can be used in favor of web artists is that every mobile device has an embedded browser and that smartphones have interaction through touch-screen, and also have a camera and lots of sensors. Nowadays it is possible to access the user's geolocation, the device's acceleration and rotation, the proximity of objects, and even the battery level and ambient light. All these through modern browsers and JavaScript.

Specific APIs were created to improve usability of web pages, and can be used for aesthetics purposes. For instance, web sites can change all their color palette according to the user's ambient light level. For us this has an enormous potential for artistic use and we believe that any sensor data can be mapped to any sonification parameter.

In this paper, we present several techniques for web sonification which will extend the browser capability to artistically sonify HTML pages (and its images, and texts nodes) using the Web Audio API for sound synthesis. Classically, a goal of sonification is to transform complex multidimensional data in intuitive audio (BEN-TAL; BERGER, 2004; HERMANN; HUNT; NEUHOFF, 2011) (as in text-to-speech or web accessibility tools). Alternatively to this approach for sonification, our goal here is to create purely aesthetic sounds that can be used to inspire compositional process or just to be enjoyed by the user. Certainly, one can find a relation between the page content and the sonified sounds, and use it to create a method to analyze web layout. It may be possible to use pages to represent music scores as a method for music notation by employing web pages. Indeed, the possibilities are enormous since we can awake the creativity of users with our sounds.

The remainder of the paper is organized as follows: Section 2 presents related works, Section 3 discusses Web pages sonification. Section 4 presents the First Sonification: HTML structure and styles. Section 5 presents the Second Sonification: Text to music. Section 6 presents the Third Sonification: Image to music. Section 7 presents the implementation, Section 8 presents Conclusion and Future Works.

2. Related Works

The human beings have an extraordinary auditory system, capable of distinguish the voice of a particular person in a chaotic environment (as a party or event), or the sound of a specific instrument in a complex scenario (as in a symphonic orchestra presentation). This ability is explored by auditory displays that use sounds as the primary mean of communication (HERMANN; HUNT; NEUHOFF, 2011).

According to Hermann, Hunt and Neuhoff (2011), an auditory display can be assigned to four categories:

1. alarms, alerts, and warnings;
2. status, process, and monitoring messages;
3. data exploration;
4. art, entertainment, sports, and exercise.

In the first category we have simple examples as doorbells, beeps from microwaves, smoke detectors, etc. From the second category we have more complex examples that explores variations on a single sound, as in cardiac monitors (where the beeps follows the heartbeat in real-time), pedestrian traffic lights (where each stage has a different beep pat-

tern or the absence of one), telephone tones (to indicate status as “calling”, “busy line”, “on hold”, etc).

The third category has several works with auditory graphs (FLOWERS; HAUER, 1993; BROWN; BREWSTER, 2003), and sonifications techniques based on models (HERMANN; HUNT, 2005). The fourth category, which is the one closest to the work presented in this paper, has applications in entertainment (for instance, audio-only versions of games) (WINBERG; HELLSTROM, 2001; MCCRINDLE; SYMONS, 2000), and in art (for instance, musical compositions made through data sonification) (QUINN, 2001; QUINN; QUINN; HATCHER, 2003).

Sonification is one of the main components of auditory systems, and consists of the generation of sounds to convey data to inform the users. Besides the fact that humans are able to recognize various different sounds and subtle variations in them, sonification is pursued when users are unable to observe the data (for instance, visual displays overloaded with information or with temporary busy with alternative data) or even unable to see the data (for instance, blind users or due to ambient factors as smoke or a tree branch) (HERMANN; HUNT; NEUHOFF, 2011).

The situation when users are unable to see the data includes researches on computers and accessibility. It is normal to think about screen readers reading text content in several platforms, such as text editors, spreadsheets, web browsers, and others. This to give content access to visually impaired computer users. Since web content is not only text, several approaches arose to sonify web content beyond text and give a sonic feedback about the whole web page.

Websound (PETRUCCI et al, 2000) is a web sonification tool created to increase web accessibility to blind and visually impaired computer users. This tool dynamically create alternative access modalities on top of a standard visual Web browser (Internet Explorer 5.0) using HTML structure do to add spacialization and permit individuals to explore spatial information by the way of audio-haptic interface.

A Web-based image sonification platform based on Javascript (SCHAUERTE; WÖRTWEIN; STIEFELHAGEN, 2015) allows easy access to graphical information for blind users. This application uses computer vision to extract informations from images like maps, graphs, charts, diagrams, and pictures of art and can be expanded to support further sonification.

Audio Enriched Links (PARENTE, 2003) is an extension for the JAWS screen reader for Windows and targets the Internet Explorer web browser. This tool uses text-to-speech algorithms to provide previews of linked web pages to users with visual impairments.

Although these related tools are also based on web sonification, this paper have a different main goal. Rather than accessibility, we are focused on aesthetical sonification, with the purpose to give sonic feedback to every user, regardless of his/her physical condition.

3. About Web Pages sonification

In the beginning of the Internet era, a web page was a plain text, black on white, unformatted document. Several old websites use this format even today. Their content is easily rendered by a text-to-speech converter or a Braille device (PETRUCCI et al, 2000). Such tools will read only the text content of the page and will ignore the visual aspects of the website. Even images will be ignored by the page reader, and an alternate text (alt ima-

ge tag property) will be read instead if the page developer provided one.

For practical matters, we can consider a default web page as a quadruple of HTML, CSS, JavaScript resources, and the content itself. Basically, HTML is responsible for the page structure, CSS for styling (positions, sizes, colors, etc), JavaScript for dynamic actions on the client side (content variation, animations, user interactions, etc), and the page contents are texts, images, videos, etc. We have several approaches for web sonification, and the only item from the quadruple mentioned above that we are not currently using is the JavaScript code (although <script> tags can be sonified as texts nodes).

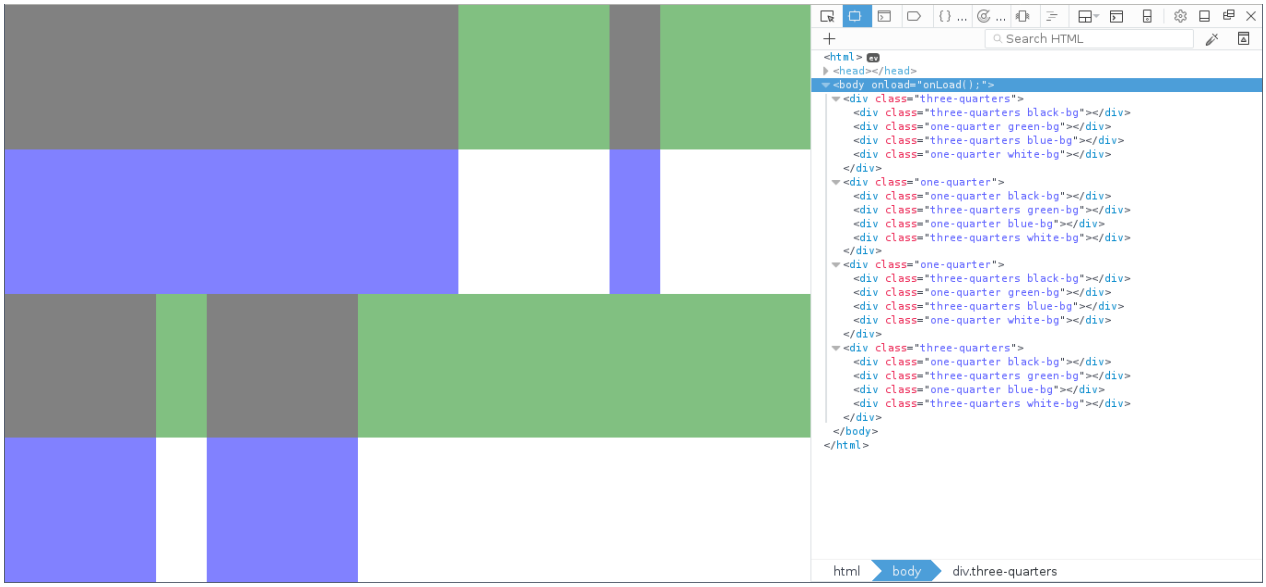

Figure 1: An example HTML page and its structure.

Firstly, we can consider a web page as a music sheet. Our first idea was to use the page structure depicted in Figure 1: An example HTML page as a musical structure, and the page styling (CSS) as a synthesis parameter. Therefore, identical HTML tags are sonified differently according to their formatting styles. This type of music sheet can be compared to MusicXML file format (GOOD, 2001) (Figure 3). Similarly to XML, HTML is a hierarchical file format based on tags and properties. Despite this resemblance, they were created for different purposes. Lastly, like MusicXML, our approach of HTML does not produce sound and depends on a synthesizers to play the score.

```
X: 1 % start of header

K: C % scale: C majorw

e1 d1 a1 c1 | e c e a | d c g e
```

Box 1: Example of an ABC score.

Secondly, our system interpret letters from texts as musical notes or commands. Here we have similarities with textual musical notation systems like ABC (WALSH, 1995) and LilyPond (NIENHUYS; NIEUWENHUIZEN, 2003). These textual notations use normal text representation, which is mono-dimensional and map chars to time and note representation, like presented in Box 1. Like ABC and LilyPond files, our text sonification uses a synthesizer to create sounds.

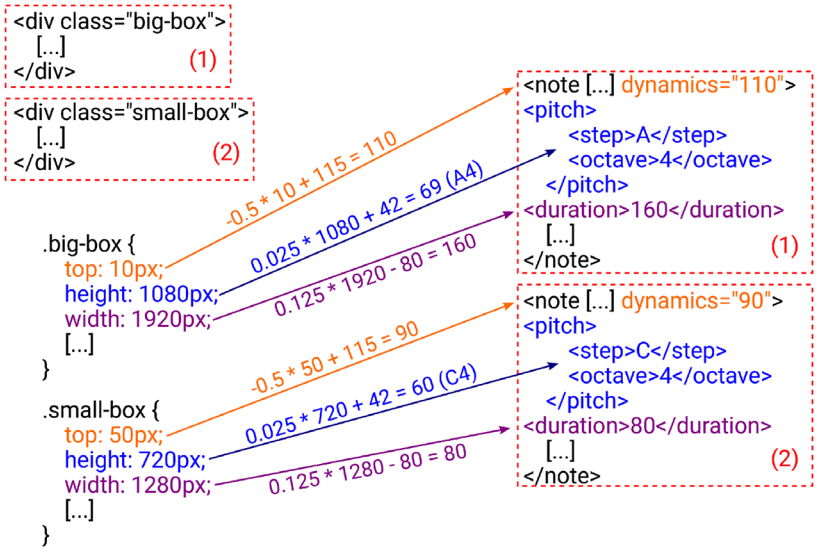

Figure 2: MusicXML versus HTML (together with a equivalent HTML sonification).

Thirdly, we are sonifying images present on the web page. The conversion is based on raw data. Our strategy here is to pick up pixels from the images and convert them directly to audio samples (Figure 3).

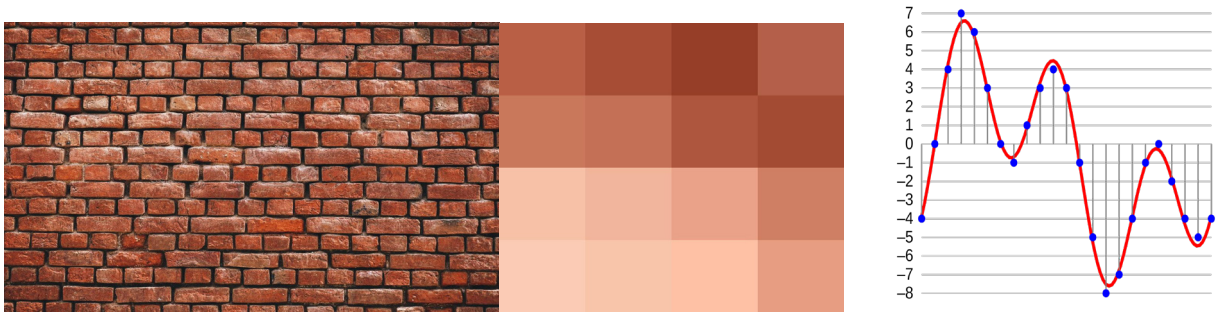

Figure 3: Converting RGB values from a figure to audio samples mapping each pixel value to the sample magnitude.

4. First sonification: HTML structure and styles

The HTML language has the peculiarity of enabling two sites with completely different code structures to have the same visual rendition, and sites with very similar source code to have completely different visual output. 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. To understand which tag could be used on the sonification, we selected the top 10 tags from http://bbc.com and http://cnn.com, and presented them on Figure 4. 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.

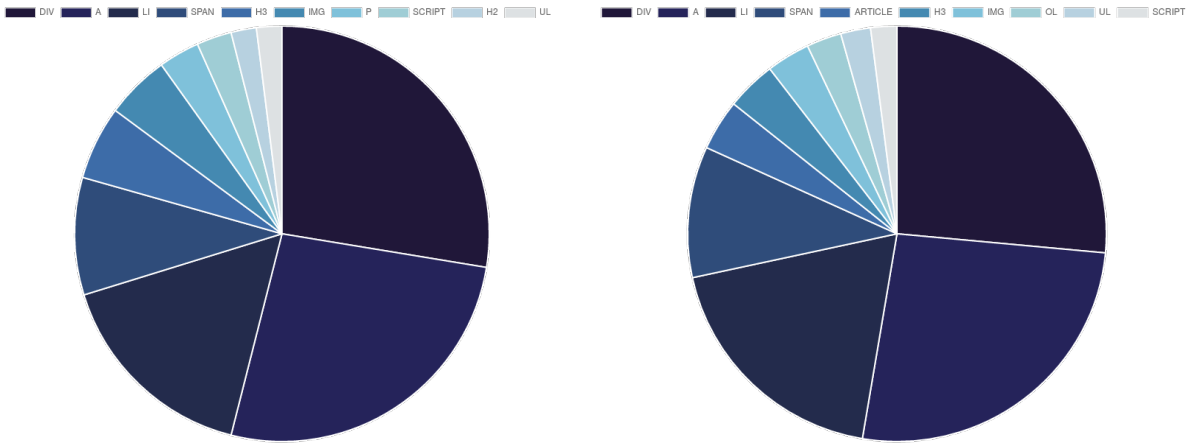

Figure 4: Most used tags of BBC and CNN sites, and the proportion of their occurrences.

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 style attributes using the CSS sources is an arduous task, because we can have more than one class by element, more than one file defining the classes, and rule overwriting due to inheritance. 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 applied the so-called box model to solve this problem.

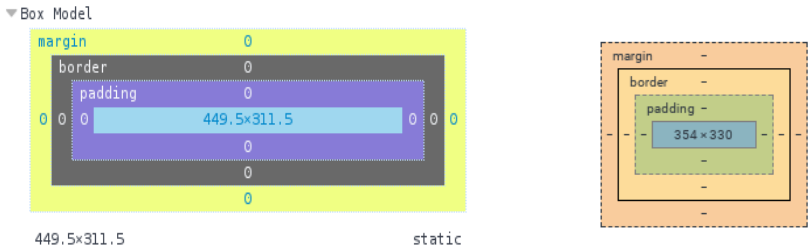

Figure 5: Box model in Firefox, and Chromium browsers.

Every modern browser has in its developer tools a graphic visualization of an element by the terms of the box model (as we can see in Figure 5). It considers that every element behave like a box. When you inspect an element, this graphic shows the computed dimensions (width and height) and 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 moved 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.

The width and height values are positive, as expected, even with some very high values. The highest height is probably from the <body> tag, which contains the entire page; the largest width is most likely from a sliding element that contains several sub-elements, and it is not fully visible. What caught our attention is top, left and margin minimum values that are negative in both websites. This may be a sign of the use of a worka-

round usually made to move elements far away (vertically or horizontally) to get them out of the user's view. Another observation is the presence of minimum values equal to zero. The impact of these zero-valued attributes is discussed further below.

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 1: BBC 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

Table 2: CNN global statistics of styles.

4.1. The HTML Score

HTML is a hierarchical document. 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 pitch execution, and the attributes of the elements define the notes configurations. For instance, the pitch of a note can be defined by one HTML attribute; the duration by another attribute; the amplitude by yet another (specific information about mappings can be found in subsection 3.2). For example, a synthesizer plays notes for each rectangle on the web page shown in Figure 6 (implemented with div tags). When we map width to pitch and height to duration, we will have notes such as those in the same Figure. If we had a more complex page with more tags, we calculate the most frequent ones and sonify each tag using a different synthesizer (each tag is a different voice and may have a different timbre as well).

In Figure 6 we have 20 elements: 16 are clearly visible by their colors, and the other 4 are just wrappers around groups of 4 elements (highlighted in red). The piano roll shows 20 notes relative to these 20 elements, where the 4 longest ones are relative to these wrappers (the durations were mapped from height). The other 16 notes have all the same duration, because all the sub-elements have the same height. The pitches are mapped from widths, so larger elements will produce lower pitches, and so on. We can verify 5 different widths being mapped to 5 different pitches. The order of the note reproduction

is the same as the one in which the recursive algorithm walked through the nodes of the HTML hierarchical tree (the first wrapper, then its 4 children, then the second wrapper, etc).

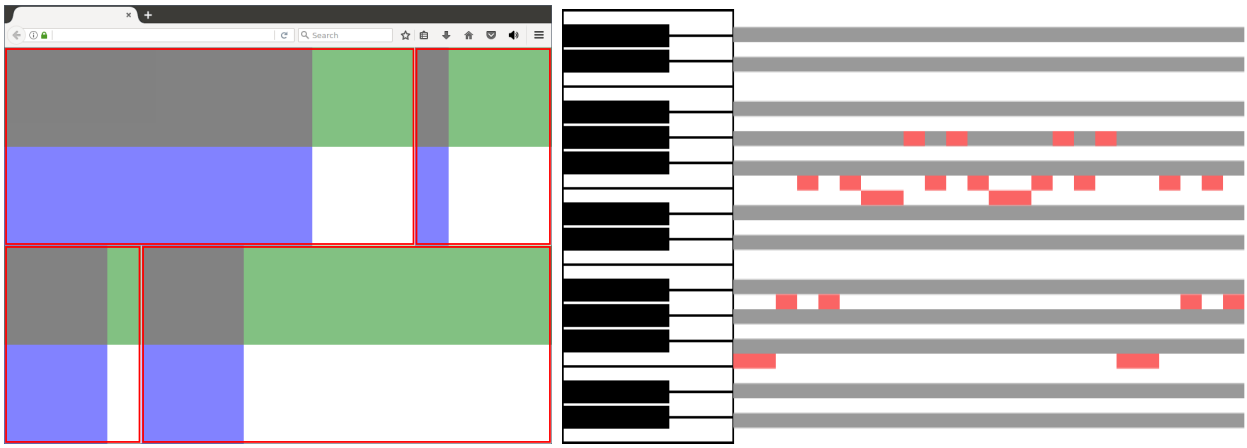

Figure 6: Example page where each block is mapped to a note, and the visualization of these notes in a piano roll (Sound Example 1).

The notes on the piano roll do not have rests between them. This happens because when the time of a note offset is reached, the next one starts immediately. But this behavior is not mandatory. We can map any attribute to the notes' starting times, like width, height, or position, and with this we can produce rests between notes. This works because the starting time of a note, added with its duration, not necessarily will reach the starting time of the next one. With this onset time mapping, we can produce polyphony and several notes beginning at the same time.

4.2. Mapping HTML attributes to sound information

We mapped the 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). Clearly, some mapping is senseless while other mapping can sound very interesting. To know what is really feasible it was necessary to test some parameters. In our first test 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 as a synthesis parameter we get a greater variety than when we use borderWidth. So we had two options: combine parameters or simply avoid setting the least frequent attributes to core synthesizers parameters (such as pitch or duration).

The actual mapping of the style attributes to synthesizer parameters is left to the user. We create an interface that allows the user to choose how many tags (and, consequently, how many voices) he want to sonify, and the computation of the parameters values. The latter is done using equations of the format ax+b, where the x variable is one style attribute and the coefficients a and b are user defined. Figure 9 shows how the user chooses the instrument and the mappings for some of the most frequent HTML tags.

4.3. Visual Feedback

After our first implementation we realized that there were no visual feedback for the sonification so it was almost impossible to determine which element was being sonified. A CSS class was created to visually highlight the elements that were emitting sounds.

First, we tried to use colors, but some elements did not accept that (the concept of background color is not applicable to an image tag, for instance). After that, we tried to add a very noticeable border, but that disrupted the layout. Some side-by-side elements became misaligned or, even worse, did not fit at their original positions.

The solution was to work with 3D transformations that did not change the original layout and that could be applied on any element. We implemented an animation that shakes the element using small translations from side-to-side, making the element quite prominent. 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. It can also have a drop-down menu that has hidden items. Several instances of this problem were very difficult to solve.

To minimize 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 the visible elements of the page on these points; c) check if both correspond to the same element. We know that this heuristic is not perfect, because 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 cases. Subsequently, we did the sonification on the same websites that we extracted the statistics. The sound results are discussed in Section 8.

5. Second Sonification: Text to music

We also worked with text sonification based on textual score formats, like ABC. An ABC file starts with a header part, to define title, composer, time, time signature, and key signature. Letters represent pitch values (c, d, e, …, b; C, D, E, …, B), z represents a pause followed by a number representing the duration. ABC has also special characters to change octaves, write chords and lyrics.

So, the first idea we had was to read a web page text as an ABC file directly, using the ABC valid characters and discarding the ones not used in the notation. We noticed that this approach does not work since it discards several characters frequently used in English texts The original text, with 106 chars is mapped to a 12-note score, as presented in Box 2.

Lorem ipsum dolor sit amet consectetur adipiscing elit.

~~Lor~~ e ~~m ipsum~~ d ~~olor sit~~ a ~~m~~ e ~~t~~ c ~~ons~~ e - c ~~t~~ e ~~tur~~ a - d ~~ipis~~ c ~~in~~ g - e ~~lit.~~

Box 2: First attempt on text sonification: Using chars present on ABC notation.

To get a better mapping, we tried extracting statistics on the usage of letters, just like we did with HTML tags. Samuel Morse carried out a study on the frequency of letter usage in English words (as can be seen on Table 4). His motivation was to map the simplest Morse codes to the most frequent letters. His analysis serves our purposes.

letter	percentage	letter	percentage
E	11.1607%	M	3.0129%
A	8.4966%	H	3.0034%
R	7.5809%	G	2.4705%
I	7.5448%	B	2.0720%
O	7.1635%	F	1.8121%
T	6.9509%	Y	1.7779%
N	6.6544%	W	1.2899%
S	5.7351%	K	1.1016%
L	5.4893%	V	1.0074%
C	4.5388%	X	0.2902%
U	3.6308%	Z	0.2722%
D	3.3844%	J	0.1965%
P	3.1671%	Q	0.1962%

Table 4: The letter frequency in the Oxford Dictionary.

With this information, we looked for each letter and mapped its value for some musical command. For instance, ABC has letters that represent pitches, special characters that select the octave, and sequences for note durations and other traditional music sheet notations.

5.1. From text to melody: mapping letters to notes

We choose the twelve most frequent (capital and lowercase) letters - E, A, R, I, O, T, N, S, L, C, U, D - to be mapped to pitches in the chromatic scale. This mapping could use the original pitch order, or alternative pitch sequences, like a cycle of fifths or a cycle of fourths, as presented on Table 5.

Letter	Original order	Cycles of Fifths	Cycles of Fourths
E	C	C	C
A	C# / Db	G	F
R	D	D	Bb / A#
I	D# / Eb	A	Eb / D#
O	E	E	Ab / G#
T	F	B	Db / C#
N	F# / Gb	F# / Gb	Gb / F#
S	G	C# / Db	B
L	G# / Ab	G# / Ab	E
C	A	D# / Eb	A
U	A# / Bb	A# / Bb	D
D	B	F	G

Table 5: Mapping letters to pitches.

The octaves of these pitches are defined by the next letter. We got the decimal value of the ASCII character and perform a modulo operation to return a number between 0 and 9 (the octaves used in the MIDI protocol). Table 6 presents some of these mappings.

letter	ascii value	modulo (octave)	letter	ascii value	modulo (octave)
a	97	7	A	65	5
b	98	8	B	66	6
c	99	9	C	67	7
d	100	0	D	68	8
e	101	1	E	69	9
f	102	2	F	70	0
g	103	3	G	71	1
…			…		

Table 6: Mapping letters to octaves.

The next eight most frequent letters - P, M, H, G, B, F, Y, W - were mapped to durations from breve to hemidemisemiquaver, as presented on Table 7. We tried to assign the less frequent ones to the extreme durations (the longest, and the shortest) to avoid radical timings. For instance, a double-whole note in 120 BPM lasts 4 seconds, and a 64th note lasts 31.25ms, and we want them to appear less often.

Letter	Duration (british)	Duration (american)
Y	breve	double whole note
B	semibreve	whole note
H	minim	half note
P	crotchet	quarter note
M	quaver	eighth note
G	semiquaver	sixteenth note
F	demisemiquaver	thirty-second note
W	hemidemisemiquaver	sixty-fourth note

Table 7: Mapping letters to durations.

This text sonification starts with a 120 BPM tempo and event duration equals to a quaver note. The complete process is composed by a series of verifications, followed by actions: if the letter represents a pitch, then we compute the octave and play it right away; if the letter represents a duration change, then we change the base duration. Using this modification, the same sonification of our first attempt got a more interesting result, as presented on Box 3.

Starts with 120 BPM and ♩ (500 ms)

L G#1 (500ms)
o E4 (500ms)
r D1 (500ms)
e C9 (500ms)
m ♪

i D#2 (250ms)
p ♩
s G7 (500ms)
u A#9 (500ms)
m ♪

d B1 (250ms)
o E8 (250ms)

l G#1 (250ms)
o E4 (250ms)
r D2 (250ms)

s G5 (250ms)
i D#6 (250ms)
t F2 (250ms)

a C#9 (250ms)
m ♪
e C6 (250ms)
t F4 (250ms)
,

c A1 (250ms)
o E0 (250ms)
n F#5 (250ms)
s G1 (250ms)
e C9 (250ms)
c A6 (250ms)
t F1 (250ms)
e C6 (250ms)
t F7 (250ms)
u A#4 (250ms)
r D2 (250ms)

a C#0 (250ms)
d B5 (250ms)

i D#2 (250ms)
p ♩
i D#5 (500ms)
s G9 (500ms)
c A5 (500ms)
i D#0 (500ms)
n F#3 (500ms)
g ♪

e C8 (125ms)
l G#5 (125ms)
i D#6 (125ms)
t F6 (125ms)
.

Box 3: Second attempt on text sonification: merging char statistics and modulo to calculate the tempo.

5.2. From text to harmony: mapping words to chords

Another sonification approach is the use of words as entities, instead of the letters individually. We implemented a second text sonification tool that plays a chord for each word following the mappings explained below.

First, we need to determine the major key and consequently, the major scale that will be used to make the chord. At this moment, for the sake of simplicity, we are not using minor scales in our process. For this task, we rely on the Dave Carlton's study in Popular

Song Statistics (CARLTON, 2012), where he studied the chords of 1300 songs and found out which keys are used most (as we can see in Table 10). We are using the letter frequencies from Morse code sequence to map frequent letters (belonging to the words) to frequent keys. We mapped the twelve middle letters from the Table 4 to twelve major keys, and we define that the first letter occurring in the word is the responsible for the key to the word (as more than one letter may appear in the word).

Letter	key	appearance
S	C	26%
L	G	12%
C	Eb	10%
U	F	9%
D	D	8%
P	A	8%
M	E	7%
H	Db	7%
G	Bb	5%
B	Ab	4%
F	B	3%
Y	F#	3%

Table 10: Mapping letters to the most frequent keys.

Next, we need to decide which notes will appear in the chord. For this, we considered the seven notes of the major scale of the previous selected key. We will work with two octaves to create more sound density. The letters are assigned to relative intervals based on the chosen key. The combination of all intervals will form the complete chord. The actual mappings are shown in Table 11.

letter	number of semitones	letter	number of semitones
E	0	W	12
A	2	K	14
R	4	V	16
I	5	X	17
O	7	Z	19
T	9	J	21
N	11	Q	23

Table 11: Interval mappings.

We will present the sonification of the word LOREM to exemplify our steps. We can use Table 10 to define the key, so the letter L gives us the G major key. This key will be used to choose the notes in the chord. Notice that the fundamental G it is not in the chord yet. Next letter, O will be used, according to Table 11, giving us a perfect fifth, 7 semitones from the fundamental or 62 on MIDI value. Letter R is 4 semitones from the fundamental, 59 in MIDI. Letter E is the fundamental itself, 55 in MIDI. Letter M has no mapping value and it will be ignored by the sonification process.

Lastly, we had to define the chords durations. We did this by taking the word length and mapping it directly to a value in milliseconds, 250ms per char. Although this seems

too simple it has a practical advantage that long words will sound longer, and the user can perceive that visually. The complete process for this type of text sonification is a bit different. Instead of sweeping the text letter-by-letter, it splits the text by spaces, and sonifies each word as a chord. An example of this sonification is present on Box 4.

```
Lorem  [55, 59, 62] (1250ms)
ipsum  [62] (1250ms)
dolor  [54, 57] (1250ms)
sit  [53, 57] (750ms)
amet,  [52, 54, 61] (1250ms)
consectetur  [51, 55, 58, 60, 62] (2750ms)
adipiscing  [52, 55, 61] (2500ms)
elit.  [55, 60, 64] (1250ms)
```

Box 4: Mapping words to chords (Sound Example 2).

The first versions of both text sonifiers walked through the complete hierarchical tree of the HTML tags and started the individual processes for each text node. This generated a chaotic result, because all nodes produced sounds at the same time. The current versions of the text sonifiers ask the user to select with the mouse the desired text to be sonified. After the text selection the user can press a button to start the process for that snippet. Several snippets can be sonified at the same time, producing polyphony just like with HTML sonification, but with a much better control of the number of simultaneous sounding voices, and their global durations (to use this text sonifier in a performance a short process spans just a few words, or a long one spans several paragraphs).

6. Third Sonification: Image to music

The third type of sonification is the image sonification. In the HTML sonification we approached the <img> tag by its styles, so we could map the width, height, and other parameters to synthesizers' parameters. As an alternative method, we propose a direct mapping from the pixels of the image to the samples of the audio signal.

Our approach for image sonification starts with an algorithm that receives a BMP file as input and generates a WAVE file as output (Figure 7). With this algorithm we can choose the sample rate and bit depth of the WAVE file. A digital image, like a BMP file, is a sequence of bytes representing color information. Image values are read from the left-bottom corner to right top corner, using a 24 bits value to represent a RGB color, 1 byte per color channel. The data that defines these pixels is linear just like if the images were unidimensional or with 1 pixel height. The "line break" used to trace the images is based on the image width information present on the file header. The header includes information like the resolution, the height, the sample size and the RGB format. High-quality audio is stored using a PCM format. Sound files are unidimensional arrays with a sample by sample information. The file header contains sample rate, bit depth, sample format, channels, and others.

Using this information, it is possible to make an abstraction and look for the chunk of bytes from an image file as the same chunk of bytes from an audio file, and vice-versa. They are only interpreted in different ways in different contexts. The context can be the file extension or the file metadata (usually in the file header). Considering only the conversion

from image to audio: the samples of the audio file, generated from the pixels of the image file, can still be read with different sample rates and bit depths, so we considered several interpretations that we explain below.

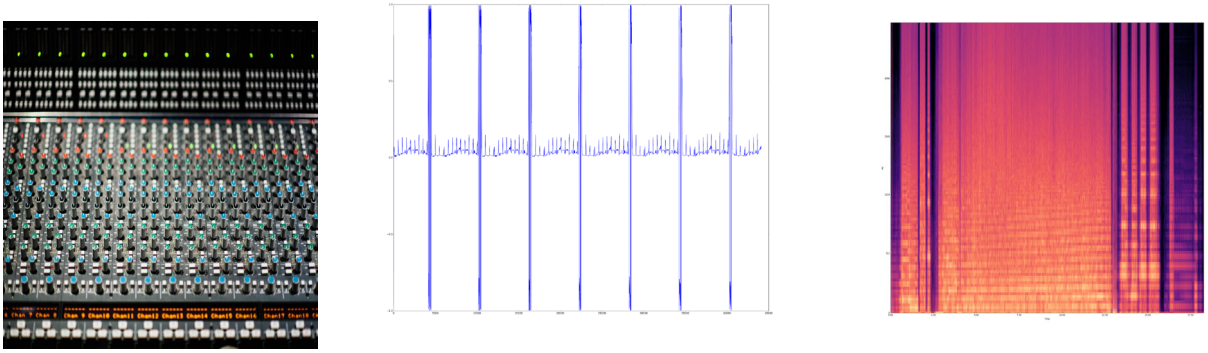

Figure 7: A Mixing console, the sonification of the Mixing console (just a few samples), and the mel-spectrogram (Sound Example 3).

This direct interpretation can solve our pixel-to-sample conversion problem, but it has some issues. One of them, clearly, is the mixture of the bytes of each pixel in different samples. This can lead to the loss of periodic patterns from the original image. We considered some alternatives: the first was to work with 24-bit samples; the second was to generate one WAVE file for each color channel, and work with 8-bit samples; the third was to convert the image to gray scale, and also work with 8-bit samples. Currently, the only option that is not contemplated explicitly is the second one, because the user can him/herself split the images into three (one for each color channel), and generate 8-bit WAVE files.

For the sonification in real-time of images from web pages we did a JavaScript code that gets the samples from the pixels and plays them in real-time using the WebAudio API (instead of saving the result to an audio file). Beyond images already present in a web page we did a web page that can sonify images from the user's camera. Therefore, we provided a portable tool for anything-you-can-see sonification. One can take a picture of his dog, or her favorite painting, or a beautiful Persian carpet (as we can see on Figure 8), and all of these pictures can be sonified.

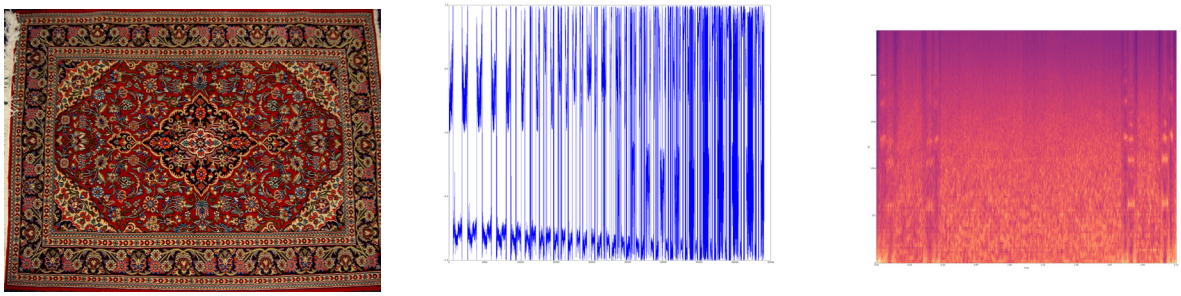

Figure 8: A Persian carpet, the sonification of the Persian carpet (just a few samples), and the relative mel-spectrogram (Sound Example 4).

7 Implementation

Browsers can be extensible by extensions, a type of plugin written in JavaScript, and users can install these extensions adding more functionalities to the browser. To ensure cross-browser compatibility we choose the WebExtensions system 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. Our extension running on the web browser is presented on Figure 9.

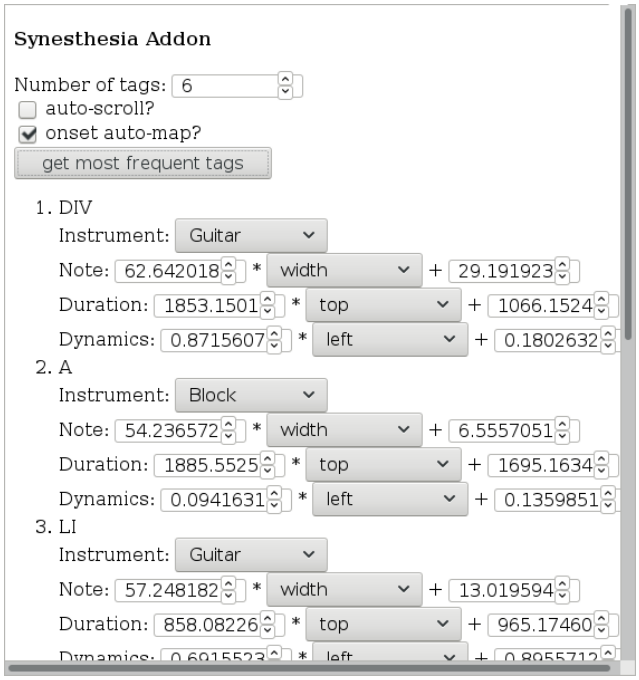

Figure 9: Screenshot of the sonification tool showing how users can map parameters.

The HTML sonification is completely inserted in the add-on mentioned above. The add-on source code is available on https://github.com/rppbodo/synesthesia-addon. The text sonification is available on http://rppbodo.github.io/text2melody.html. Text to harmony sonification is available on http://rppbodo.github.io/text2harmony.html. The image sonification is available on http://rppbodo.github.io/image2audio.html.

The onset scheduling is implemented using a JavaScript method called setTimeout that receives a callback function and a time value. With this method we are able to schedule all of the notes onsets. The offsets were handled by the synthesizers themselves (we passed the duration to the procedure called startNote). The scheduling is totally dependent on a parameter that we called "virtual clock relative speed". The concept of virtual clock was presented by Roger Dannenberg in 1984 (DANNENBERG, 1984) in the context of automated musical accompaniment. Basically, it is a clock that follows the rate of the operating system 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, it will be half of the original, if it is set to 2, it will be double, and so on.

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

For the audio to be synthesized in real-time on the browser itself, we used WebAudio to implement a synthesizer. We tried to select techniques that produce different timbres. Six instruments were developed for our extension: Block, Drum, Guitar, Harpsichord, Maraca and Pong. All of them were implemented with the same interface so that they could

be easily alternated. The pitch parameter is the MIDI note number, the duration is in milliseconds and the amplitude is a value between 0 and 1.

8. Conclusion

In this paper, we presented a novel form to create music based on HTML pages, text, and images by the means of a set of sonification tools. We chose to use the structure of HTML websites to control synthesizers. We used text nodes of web pages as musical notation. The pixels from image files were mapped to the samples of the audio signals.

We presented our strategies to sonification, the statistics used to create these strategies and made several considerations about the development process. The sonic results were diverse and we achieved a very wide range of sounds. We believe that these tools could be used in live performance, they could help a composers to bootstrap musical creations, or they could be used for entertainment purposes. As future work, we intend to explore how websites can boost musical creativity in composition and improvisation.

8.1. Acknowledgment

We would like to thank the Computer Music Research Group (http://compmus.ime.usp.br/) at the University of São Paulo and the Sonology Research Center (NuSom - http://www.eca.usp.br/nusom/) at the University of São Paulo, Interdisciplinary Postgraduate Program in Arts, Urbanities and Sustainability – PIPAUS from the Federal University of São João del-Rei and the support from CAPES.

Notes

1 Sounds Examples also available on http://rppbodo.github.io/musica-hodie/

References

ALLISON, Jesse; HOLMES, Daniel; BERKOWITZ, Zachary; PFALZ, Andrew; CONLIN, William; HWANG, Nick; TAYLOR, Benjamin. Programming Music Camp: Using Web Audio to Teach Creative Coding. In: WEB AUDIO CONFERENCE, 2, 2016, Atlanta, Georgia. *Proceedings of the 2nd Web Audio Conference*. Georgia Institute of Technology, 2016.

BEN-TAL, Oded; BERGER, Jonathan. Creative aspects of sonification. *Leonardo*, Volume 37, Issue 3, p. 229-233, June 2004.

BROWN, Lorna; BREWSTER, Stephen. Drawing by ear: Interpreting sonified line graphs. In: INTERNATIONAL CONFERENCE ON AUDITORY DISPLAY, 9, 2003, Boston, Massachusetts. *Proceedings of the 9th International Conference on Auditory Display*. Georgia Institute of Technology, 2003.

CARLTON, Dave. *I analyzed the chords of 1300 popular songs for patterns. This is what I found.* 2012. Available on: <http://www.hooktheory.com/blog/i-analyzed-the-chords-of-1300-popular-songs-for-patterns-this-is-what-i-found/>. Accessed: 2017-07-01.

DANNENBERG, Roger. An on-line algorithm for real-time accompaniment. In: INTERNATIONAL COMPUTER MUSIC CONFERENCE, 1984, France. *Proceedings of the 1984 International Computer Music Conference*. Computer Music Association, 1985, p. 193-198.

FLOWERS, John; HAUER, Terry. "Sound" alternatives to visual graphics for exploratory data analysis. *Behavior Research Methods, Instruments & Computers*, Volume 25, Issue 2, p. 242-

249, June 1993.

GOOD, Michael. MusicXML for Notation and Analysis. In: HEWLETT, Walter; SELFRIDGE--FIELD, Eleanor. *The Virtual Score: Representation, Retrieval, Restoration*, MIT Press, Cambridge, MA, 2001, p. 113-124.

HERMANN, Thomas; HUNT, Andy. An Introduction to Interactive Sonification. *IEEE MultiMedia,* Volume 12, Issue 2, p.20-24, April-June 2005.

HERMANN, Thomas; HUNT, Andy; NEUHOFF, John. *The Sonification Handbook*. Berlin: Logos Publishing House, 2011.

JANA, Reena; TRIBE, Mark. *New Media Art*. London: Taschen, 2006. 96p.

LAZZARINI, Victor; YI, Steven; TIMONEY, Joseph; KELLER, Damian; PIMENTA, Marcelo. The Mobile CSound Platform. In: INTERNATIONAL COMPUTER MUSIC CONFERENCE, 2012, Slovenia. *Proceedings of the 2012 International Computer Music Conference*. Computer Music Association, 2012, p. 163-167.

LETZ, Stephane; DENOUX, Sarah; ORLAREY, Yann; FOBER, Dominique. Faust audio DSP language in the Web. In: *Proceedings of the Linux Audio Conference.* Mainz, Germany: Johannes Gutenberg University (JGU), *201*5, p. 29–36.

MCCRINDLE, Rachel; SYMONS, David. Audio space invaders. In: INTERNATIONAL CONFERENCE ON DISABILITY, VIRTUAL REALITY AND ASSOCIATED TECHNOLOGIES, 3, 2000, Alghero, Italy. *Proceedings of the 3rd International Conference on Disability, Virtual Reality and Associated Technologies*. 2000, p.59-65.

NIENHUYS, Han-Wen; NIEUWENHUIZEN, Jan. Lilypond, a system for automated music engraving. In: XIV COLLOQUIUM ON MUSICAL INFORMATICS, 2003, Firenze, Italy. *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003).* Tempo Reale, 2003, p.167-172.

PARENTE, Peter. Audio enriched links: web page previews for blind users. In: INTERNATIONAL ACM SIGACCESS CONFERENCE ON COMPUTERS AND ACCESSIBILITY, 6, 2004, Atlanta, GA, USA. *Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility*. New York, NY, USA, ACM, 2004, p.2-8.

PETRUCCI, Lori Stefano; HARTH, Eric; ROTH, Patrick; ASSIMACOPOULOS, Andre; PUN, Thierry. Websound: A generic web sonification tool, and its application to an auditory web browser for blind and visually impaired users. In: INTERNATIONAL CONFERENCE ON AUDITORY DISPLAY, 6, 2000, Atlanta, Georgia. *Proceedings of the 6th International Conference on Auditory Display (ICAD 2000)*. Georgia Institute of Technology, 2000, p.6-9.

QUINN, Marty. Research set to music: The climate symphony and other sonifications of ice core, radar, DNA, seismic and solar wind data. In: INTERNATIONAL CONFERENCE ON AUDITORY DISPLAY, 7, 2001, Espoo, Finland. *Proceedings of the 7th International Conference on Auditory Display*. Georgia Institute of Technology, 2001.

QUINN, Marty; QUINN, Wendy; HATCHER, Ben. For those who died: A 9/11 tribute. In: INTERNATIONAL CONFERENCE ON AUDITORY DISPLAY, 9, 2003, Boston, Massachusetts. *Proceedings of the 9th International Conference on Auditory Display*. Georgia Institute of Technology, 2003.

ROBERTS, Charles; KUCHERA-MORIN, JoAnn. Gibber: Live coding audio in the browser. In: *Proceedings of the 2012 International Computer Music Conference*. Computer Music Association, 2012, p. 64-69.

ROGERS, Chris. *W3C Web Audio API*. Available on <https://www.w3.org/TR/webaudio/>. Accessed: 2017-07-01.

SEEVINCK, Jennifer. *Emergence in Interactive Art*. Switzerland: Springer International Publishing, 2017. 182p.

SCHAUERTE, Boris; WÖRTWEIN, Torsten; STIEFELHAGEN, Rainer. A Web-based Platform for Interactive Image Sonification. In: HUMANS AND COMPUTERS WORKSHOP, 2015. *Proceedings of the 2015 Humans and Computers Workshop*. De Gruyter, 2015, p. 399-403.

WALSHAW, Chris. *The ABC Music Notation*. Available on <http://abcnotation.com>. Accessed: 2017-07-01.

WEINTRAUB, Annette. Art on the Web, the Web as art. *Communications of the ACM*. New York, NY, USA, Volume 40, Issue 10, p.97-102, October 1997.

WINBERG, Fredrik; HELLSTROM, Sten. Qualitative aspects of auditory direct manipulation: A case study of the Towers of Hanoi. In: INTERNATIONAL CONFERENCE ON AUDITORY DISPLAY, 7, 2001, Espoo, Finland. *Proceedings of the 7th International Conference on Auditory Display*. Georgia Institute of Technology, 2001.

WYSE, Lonce; SUBRAMANIAN, Srikumar. The viability of the web browser as a computer music platform. *Computer Music Journal*, Cambridge, MA, USA, Volume 37, Issue 4, p.10-23, 2013.

Roberto Piassi Bodo: Possui graduação em Ciência da Computação pela Universidade de São Paulo(2011) e mestrado em Ciências da Computação pela Universidade de São Paulo(2015). Tem experiência na área de Ciência da Computação.

Flávio Luiz Schiavoni: possui graduação em Ciência da Computação pela Universidade Estadual de Maringá (1999), mestrado em Ciência da Computação pela Universidade Estadual de Maringá (2007) e doutorado em Ciências da Computação pela Universidade de São Paulo (2013). Atualmente é Professor Adjunto da Universidade Federal de São João Del-Rei. Tem experiência na área de Ciência da Computação, com ênfase em Metodologia e Técnicas da Computação. Atuando principalmente nos seguintes temas: música em rede, áudio, MIDI, ambiente musical em rede, performance musical em rede e comunicação musical síncrona.