AUDIOGRAPH: AUDITORY GRAPHS FOR GOOGLE SHEETS WITH FAUST

Andrés Aparicio

Center for Research in Audio Technologies
Music Institute
Faculty of Arts

Pontificia Universidad Católica de Chile
aaparicio@uc.cl

Rodrigo F. Cádiz

Center for Research in Audio Technologies
Music Institute, Faculty of Arts
Department of Electrical Engineering
School of Engineering
Pontificia Universidad Católica de Chile
rcadiz@uc.cl

ABSTRACT

In this article we present Audiograph, a web-based auditory graph application that allows users to sonify numerical data directly from a spreadsheet, using Google Sheets. One or more data series can be converted into sound at the same time. Our tool is implemented as an add-on inside Google Sheets using the powerful sound synthesis engine FAUST. We have chosen FAUST as the sonification platform due to its highly efficient rendering of digital sound processing algorithms and its integration with WebAssembly, which allows to use any modern web browser in a much more powerful way that with previous JavaScript-based technologies.

1. INTRODUCTION

People with visual impairments or blindness have serious difficulties to access and comprehend the information contained in graphs, which are widely utilized to describe numerical data, especially in scientific or mathematical fields. Text-to-speech or screen-reader software available in most modern operating systems today may help them to understand the context of a graph, including information such as the title, axis labels, and what the visual display is about, but not the actual data itself. This is especially the case in spreadsheets environments such as Microsoft Excel or Google Sheets.

Understanding data in visual graphs is not an easy task for sighted people and, in the case of visually-impaired people, the difficulty increases. As data becomes more complex, multi-dimensional or bigger in size, the issue becomes more difficult [1]. In the United States, coordinate graphs and number lines are an essential part of mathematics education [2], and are often part of required standards from kindergarten to high school, while in Chile, one of the objectives of the primary and secondary school curriculum is the usage of graphs to interpret and access different kinds of information [1]. Graphical data is used from 1st to 10th grades, starting first with pictographs, and later introducing other kinds, such as bar, line or pie graphs. In the case of blind children, currently there are no established methodologies for teaching the understanding of graphs except via tactile-graphs, when available. This turns out to be a problem later in life for visually-impaired people pursuing careers in the natural or social sciences, where the use of graphs is essential and implementation of accessible options is not widespread.

There is significant evidence that auditory graphs allow for a better understanding of numerical data contained in graphs, especially for visually-impaired people [3] [4] [5] [6]. This approach has many advantages over the standard method of tactile-graphs as they are easy to implement with any general purpose computer equipment available today for educational purposes. Also, in the specific case of educational settings, auditory graphs are a satisfactory alternative to tactile-graphs, in terms of curriculum compatibility, students performance and teachers satisfaction [7].

Our main goal is to implement graphical data sonification strategies that would allow blind or visually-impaired people to access quantitative information contained in visual graphs, especially in educational, professional or academic contexts. This is the reason why we have focused on a free widely used tool for numeric calculation and plotting on a spreadsheet, such as Google Sheets. Spreadsheets are a very common and useful tool for the storage and analysis of data, and they are used throughout industry, commerce, government and education [8]. We would like to add the capability of not only seeing the data in a spreadsheet environment, but also to hear it.

This article is structured as follows. In section 2, the general sonification strategy is presented. Section 3 discusses the usage of FAUST as the engine for sound generation and processing and provided one example of the type of synthesis methods that can be used. Section 4 presents our application in detail. Finally, conclusions and future work are discusses.

2. SONIFICATION STRATEGY

There are two main ways of using auditory graphs: to estimate tendencies in the data or to estimate exact values. In the first case, the idea is to determine patterns in the data while the latter tries to

At the time this manuscript was written and Audiograph was developed, Rodrigo F. Cádiz was a Tinker Visiting Professor at the Center for Latin American Studies (CLAS) and Center for Computer Research in Music and Acoustics (CCRMA), Stanford University.

[1]Retrieved on 4/20/2018 from http://www.curriculumnacional.cl

determine with the most possible precision the sound of a particular point in the data space [6]. At the current state of our research, we are focusing in pattern estimation for the case of line graphs of one of more series.

In the particular case of spreadsheet sonification, there has been significant work that has been reported in the literature. Stockman [8] proposed an integrated approach to sound-based access to spreadsheets. For this purpose, he developed a macro for Microsoft's Excel written in Visual Basic, while the sound rendering was done in CSound. This macro first identifies important features of the data such as global maximum and minimum numeric values, offset and scaling parameters. The system is able to sonify a selected range of cells either row-wise or column-wise or partial portions of the selected range.

Hetzler and Tardiff [9] developed a spreadsheet-based tool that uses sonification to support and reinforce calculus instruction, by means of graphical and numeric representations of mathematical functions. Their objective was to dispose of a tool that "students could use to create auditory graphs, to explore and learn about limits, derivatives, concavity, and extreme values". The implementation was also embedded into Microsoft's Excel using Visual Basic and ActiveX.

In the case of line plots, auditory graphs are typically sonified by converting values in the y-axis to pitches or frequencies and the x-axis to time, as done by [10] [11]. The way that the y-axis data is converted into sound may vary, such as in the case of [9] where they used a discrete number scale, mapping each value logarithmically into to the fundamental frequency of a chord. Major chords were used to represent non-negative values, and minor chords to represent negative values.

The handling of time can also vary, by means of a continuous or discrete presentation of sound events. Harrar and Stockman conducted an experiment where the continuous presentation mode generated more accurate results when the participants were asked to draw the audio graph in comparison to the discrete mode of presentation. But for point estimation tasks, discrete sound presentation generated a better accuracy than the continuous sound presentation [12].

In our approach, data series are converted to an audio stream in a sequential manner, and we map the value of each cell to a particular pitch. There are two fundamental modes of presentation: as a discrete series of audible events and as a continuous frequency-varying sound. Intensity is coupled with frequency, as they have been proven to be integral dimensions [13]. The duration of the whole sonification process is in the range of 3-7 seconds, as it has been shown to be the most effective time range [12].

It is important to emphasize that studies of auditory graphs with more than one data stream are fewer than the ones that focus on only one data series. This is the reason why we designed our tool to be able to sonify more than one data stream at the same time, which leads to interesting research questions such as how many sound parameters should be used, which ones should be used in all series and which ones separately [13]. In concrete, when more than one data series are present we also incorporate stereo mixing as a strategy to separate the different data series in the stereo field.

Our tool can be run on any modern web browser that supports WebAssembly and sonification parameters can be easily customized and changed in real-time and provide almost immediate feedback on the visual information to the listener.

3. SOUND SYNTHESIS

We have chosen FAUST [14] as the sound synthesis platform for our tool. FAUST is a powerful and expressive programming language that provides a purely functional approach to audio signal processing while offering a high level of performance comparable to C/C++ based audio libraries.

The semantic of FAUST is simple and well defined, allowing the FAUST compiler to be semantically driven. This means that instead of compiling a program literally, it compiles the mathematical function it denotes. A FAUST program denotes a signal processor, a mathematical function that transforms input signals into output signals.

Contrary to most JavaScript-based audio libraries, which provide usually a very simple and limited subset of sound synthesis algorithms and data control, FAUST is a complete digital audio processing engine, comprising hundreds of sound analyzers, generators, filters and effects [2].

However, one of the most important facts of using FAUST for web-based auditory graphs, is its recent support for WebAssembly. [15]. According to the official website, "WebAssembly (abbreviated Wasm) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server applications". In practice, audio that runs via WebAssembly on a web browser is far more accurate and efficient than JavaScript-based solutions.

Figure 1 presents an example of the type of sound sources that can be generated in FAUST. One of the nice features of FAUST is that it provides block diagrams of all the digital signal processing subroutines being generated in the code. In this example, the instrument consists on three sawtooth oscillators connected to an amplitude envelope packed in a group called voice.

We now proceed to describe our web-based auditory graph solution in more detail.

4. AUDITORY GRAPHS IN GOOGLE SHEETS

We believe that the most important factor that differentiates our proposal from the previous ones [8] [16] [9], is the usage of open-source and state of the art platforms such as Google sheets, Faust and WebAssembly, that provide a much better performance and flexibility for the design of different sonifications. Anyone with a Google account can create a spreadsheet, install our Audiograph tool and use it. The previous proposals were based on Microsoft Excel, relying on obscure and not well-maintained development environments such as Visual Basic (deprecated in 2008), or CSound, which is not being used actively in the computer music community. Excel is surely still widely used, but it remains a commercial tool. Our add-on tool is free and open-source. While FAUST is a relatively new tool, it is gaining momentum and it is very powerful. Our solution is also available as a JavaScript API, meaning that it could be incorporated into any web-based application.

To accomplish our goal, we designed a web system with three components[3] (Figure 2). First, the *FAUST Layer* provides an instrument DSP (Figure 1) with a set of parameters that is compiled into a WebAssembly module. This module, in turn, is exposed

[2]`http://faust.grame.fr/library.html`

[3]Available for download at `https://github.com/rcadiz/web-audiographs` and as a add-on inside Google Sheets.

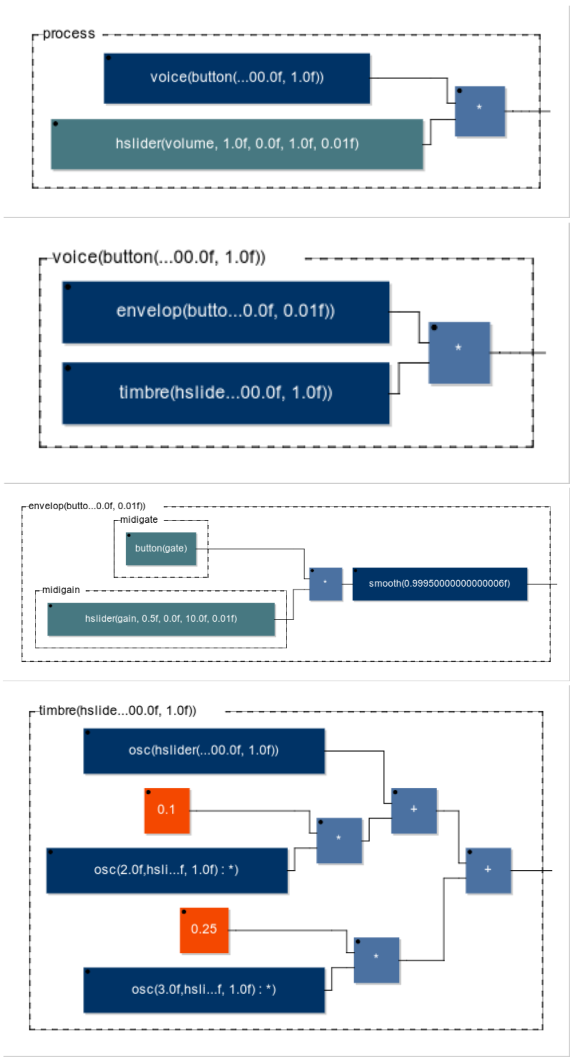

process
voice(button(...00.0f, 1.0f))
hslider(volume, 1.0f, 0.0f, 1.0f, 0.01f)
*
voice(button(...00.0f, 1.0f))
envelop(butto...0.0f, 0.01f))
timbre(hslide...00.0f, 1.0f))
*
envelop(butto...0.0f, 0.01f))
midigate
button(gate)
midigain
hslider(gain, 0.5f, 0.0f, 10.0f, 0.01f)
*
smooth(0.99950000000000006f)
timbre(hslide...00.0f, 1.0f))
osc(hslider(...00.0f, 1.0f))
0.1
+
*
osc(2.0f,hsli...f, 1.0f) : *)
+
0.25
*
osc(3.0f,hsli...f, 1.0f) : *)

Figure 1: FAUST block diagrams of a simple organ-like instrument.

via a Javascript API packaged as a dynamic module that may be loaded asynchronously. This abstracts the sound synthesis details and allows changing the sonification strategy by updating only the WebAssembly module when the instrument DSP is changed. Second, the *Audiograph Module* provides a set of operations that are used to both to handle incoming data and its sonification process. The sonification process may be controlled either programmatically by code, or through a standard set of controls that may be manipulated by users to change sonification parameters. Finally, the *Client Interface* uses API calls to the *Audiograph Module* both to render a user interface (UI) and to manage the sonification process. The implementation of this component depends on where the system is to be deployed: as part of a webpage, an extension of another web system, or as part of a Node application, for example.

We implemented a *Client Interface* for Google Sheets, a part of Google suite of office web applications. These applications may be enhanced with add-ons implemented in Google App Scripts, an scripting language based on JavaScript. These add-ons can be used to extend the application UI via menus or sidebars and, in this case,

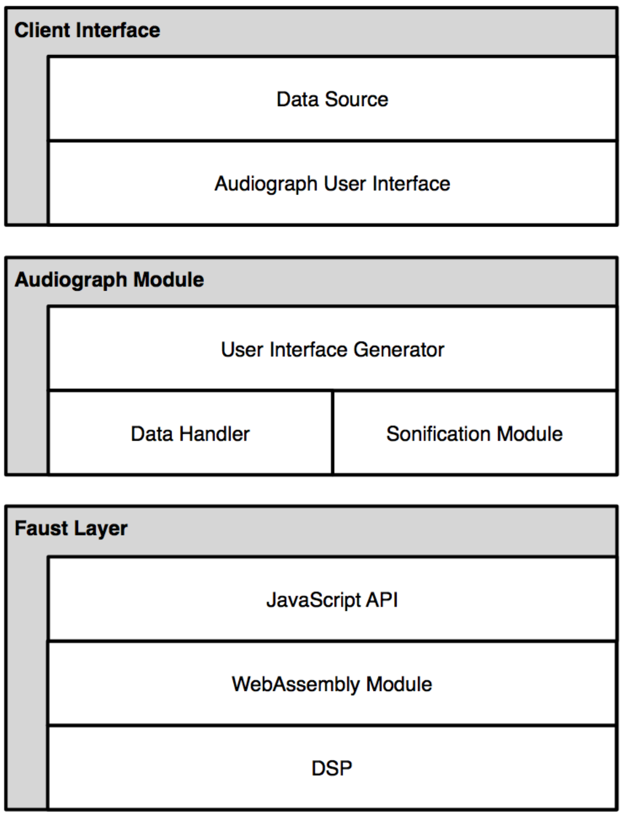
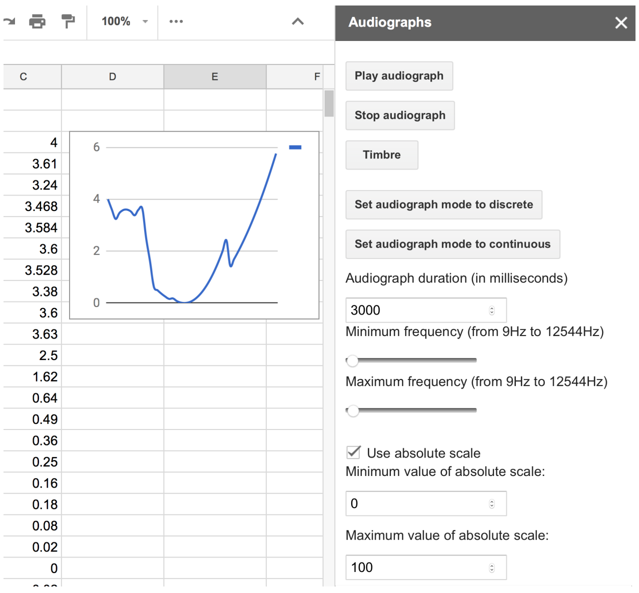


Figure 2: Architecture of the proposed solution.

operate on numeric data stored in the spreadsheet. Each add-on requires a UI and a script that handles communication between the spreadsheet and the add-on UI. In our case, the add-on UI contains the user controls provided by the Audiograph Module (Figure 3).

Once installed and enabled, the add-on is used by selecting a group of cells in the spreadsheet and activating the add-on. This loads the Audiograph Module, which renders the UI, connects the user controls to the appropriate browser events, then reads the selected data from the spreadsheets and sends it to the sonification process, starting automatically as soon as possible. Initially, the sonification uses its default settings but they may be changed with the user controls, and the process restarted. Available settings are:

1. Mode: whether the sound is discrete or continuous.
2. Duration: time, in milliseconds, for the audiograph to play.
3. Frequency: possible range used when mapping values to frequencies.
4. Scale type: controls whether the mapping from values to frequencies maps the minimum and maximum values from the data series to the minimum and maximum frequencies (the relative type), or whether the mapping uses a fixed range of values provided by the user (the absolute type).
5. Absolute scale: minimum and maximum values used in an absolute scale.
6. Timbre: type of sounds and synthesis algorithms used for the sonification.


100%
Audiographs
Play audiograph
Stop audiograph
Timbre
Set audiograph mode to discrete
Set audiograph mode to continuous
C
D
E
F
6
4
2
0

C
4
3.61
3.24
3.468
3.584
3.6
3.528
3.38
3.6
3.63
2.5
1.62
0.64
0.49
0.36
0.25
0.16
0.18
0.08
0.02
0

Audiograph duration (in milliseconds): 3000
Minimum frequency (from 9Hz to 12544Hz)
Maximum frequency (from 9Hz to 12544Hz)
☑ Use absolute scale
Minimum value of absolute scale: 0
Maximum value of absolute scale: 100

Figure 3: Example of Google Sheets Audiograph add-on sidebar with basic options displayed.

5. CONCLUSIONS AND FUTURE WORK

We have shown that it is possible to implement a web-based auditory graph application to sonify numerical data from Google Sheets that allows users to hear the data in addition to manipulating and seeing it. We have used FAUST, an advance digital sound processing engine, which allows the use of complex sonification strategies in a web-based context. Our architecture enables changing sonification strategies easily while keeping a standardized interface for the end users.

In terms of future work, we are planning to conduct tests with end-users in order to validate the design decisions we have made with the application. We are also considering to add sonification for context information (tick marks, for example), as a separate layer of sound that the user can select at bill. We are looking into the addition of binaural sound or ambisonics in order to better position audio signals in the listening space. A final application that we think could be useful is the sonification of analytical data functions instead of data series.

6. ACKNOWLEDGMENTS

This research was funded by Google via a 2017 Latin American Research Award (LARA) and by Fondecyt Grant #1161328, Government of Chile.

7. REFERENCES

[1] B. N. Walker and L. M. Mauney, "Universal design of auditory graphs: A comparison of sonification mappings for visually impaired and sighted listeners," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 2, no. 3, p. 12, 2010.

[2] B. J. Tomlinson, J. Batterman, Y. Chieh Chew, A. Henry, and B. N. Walker, "Exploring Auditory Graphing Software in Classroom: The Effect of Auditory Graphs on the Classroom Environment," *ACM Trans. Access. Comput. Article*, vol. 9, no. 3, 2016.

[3] M. A. Alonso-Arevalo, S. Shelley, D. Hermes, J. Hollowood, M. Pettitt, S. Sharples, and A. Kohlrausch, "Curve shape and curvature perception through interactive sonification," *ACM Trans. Appl. Percept.*, vol. 9, no. 4, pp. 17:1–17:19, Oct. 2012.

[4] B. Brown, L.M., Brewster, S.A., Ramloll, S.A., Burton, R., and Riedel, "Design Guidelines for Audio Presentation of Graphs and Tables," *9th International Conference on Auditory Display*, no. July, pp. 284–287, 2003.

[5] F. Delogu, M. Palmiero, S. Federici, C. Plaisant, H. Zhao, and O. Belardinelli, "Non-visual exploration of geographic maps: does sonification help?" *Disability & Rehabilitation: Assistive Technology*, vol. 5, no. 3, pp. 164–174, 2010.

[6] B. N. Walker, L. M. Mauney, and J. Godfrey, "The Audio Abacus: Representing a Wide Range of Values with Accuracy and Precision." in *Proceedings of the International Conference on Auditory Display*, 2004.

[7] B. K. Davison, "Evaluating auditory graphs with blind students in a classroom," *ACM SIGACCESS Accessibility and Computing*, no. 102, pp. 4–7, 2012.

[8] T. Stockman, "The design and evaluation of auditory access to spreadsheets," in *Proceedings of the 10th International Conference on Auditory Display (ICAD2004)*, 2004.

[9] S. M. Hetzler and R. M. Tardiff, "Two Tools for Integrating Sonification into Calculus Instruction." in *International Conference on Auditory Display*. Georgia Institute of Technology, 2006, pp. 281–284.

[10] D. L. Mansur, M. M. Blattner, and K. I. Joy, "Sound graphs: A numerical data analysis method for the blind," *Journal of Medical Systems*, vol. 9, no. 3, pp. 163–174, jun 1985.

[11] L. M. Brown and S. A. Brewster, "Drawing by ear: Interpreting sonified line graphs," in *Proceedings of the International Conference on Auditory Display*, 2003.

[12] L. Harrar and T. Stockman, "Designing Auditory Graph Overviews: An Examination of Discrete vs. Continuous Sound and the Influence of Presentation Speed," *Proceedings of the 13th International Conference on Auditory Display (ICAD2007)*, pp. 299–305, 2007.

[13] S. C. Peres and D. M. Lane, "Auditory graphs: The effects of redundant dimensions and divided attention," in *Proceedings of the International Conference on Auditory Display*, 2005, pp. 169–174.

[14] Y. Orlarey, D. Fober, and S. Letz, "Faust: an efficient functional approach to dsp programming," *New Computational Paradigms for Computer Music*, vol. 290, p. 14, 2009.

[15] S. Letz, Y. Orlarey, and D. Fober, "Compiling faust audio dsp code to webassembly," in *Web Audio Conference*, London, UK, 2017.

[16] T. Stockman, L. Nickerson, and G. Hind, "Auditory graphs: A summary of current experience and towards a research agenda," in *International Conference on Auditory Display*. Georgia Institute of Technology, 2005, pp. 420–422.