SIREN: A CASE STUDY IN WEB AUDIO BASED SONIFICATION

Tristan Peng

CCRMA
Stanford University
Stanford, CA
tristan.peng@stanford.edu

Hongchan Choi

CCRMA
Stanford University
Stanford, CA
hongchan@ccrma.stanford.edu

ABSTRACT

SIREN is an open-source, web-based sonification workstation that provides an accessible entry point to data mapping sonification and aims to demonstrate a use case for the Web Audio API. With plug-and-play functionality, and numerous methods to customize and create meaningful auditory display, *SIREN* provides useful features for pedagogy, methods for exploratory data sonification, and an extensible, open-ended development platform. Inspired by common digital audio workstation (DAW) workflows, *SIREN* provides a familiar and intuitive layout based upon data matrices as *tracks* that can be chained together as *channels*, thus allowing values in one data sequence to control a parameter of another data array. This paper describes the application's design philosophy and provides a case study as usage examples. We place the program in a comparative context with other data-mapping front-ends, and describe future goals.

1. INTRODUCTION

The Web Audio API offers a robust and flexible system for creating collaborative and interactive audio tools [1]. Developers can adapt and implement a broad range of audio synthesis and processing methods (such as integrate spatial and filtering effects), and add visualizations. The web platform invites a democratized, open-ended basis that places both development and application in the hands of anyone with a web browser.

In this paper we present *SIREN*[1], a sonification mapping paradigm that uses the Web Audio API to create an extensible, modular, and easily adaptable sonification framework. *SIREN*'s user interface is inspired by digital audio workstations such as Logic Pro or Ableton Live, and built with web technologies including React and the Web Audio API. *SIREN* aims to simplify the process of creating meaningful data sonification with an intuitive, DAW-based interface with plug-and-play functionality.

2. RELATED WORK

Numerous sonification workstations have preceded *SIREN*, In this section we consider some key features of earlier frameworks—including *Sonification Sandbox*, *Rotator*, *xSonify*, *Sonification Workstation*, *Sonification Toolkit*, and *libmapper*—and cite possible places for innovation through *SIREN*.

2.1. Handling multidimensional data

A key objective of data sonification is the ability to seek and explore relationships between multiple sets of data. To effectively implement parameter mapping sonification, the app must be able to map multidimensional data to a myriad of different sound synthesis parameters [2]. This facility is implemented in a variety of ways in different sonification applications. *Sonification Sandbox*, designed by Georgia Tech's School of Psychology's Sonification Lab, uses a table, imported from a CSV file, combined with a versatile mapping window to enable multidimensional sonification [3]. The features list of *Sonification Sandbox* also includes methods to view data graphically, with screenshots shown on the download website [4]. Designed by MIT's Media Lab, *Rotator* takes a different approach to visualization and user-interaction. In an example provided by the author *Rotator* implements a visual display of various data points on a data node map, wherein the user can drag a box labeled "Auditory" to box the data points to be sonified, and "Visual" for a graphical representation of the data points [5]. Other sonification tools lack this level of flexibility with datasets. *xSonify*, for example, can only handle data of a single dimension, relegating additional dimensions to future work [6]. With *Sonification Workstation*, data is handled through unique variables, where they can then be manipulated together to synthesize an output [7]. *Sonification Toolkit* handles multidimensional data in a way similar to *SIREN* through a table where users can add data sources, which then can be sonified together polyphonically [8]. *libmapper*, a mapping paradigm for creating digital musical instruments, has extensive support for various input types and data sources, and can easily handle different inputs through a graph [9]. *SIREN* presents original methods with regards to its handling of multidimensional data, which will be explored further in section 4.

2.2. Parameter mapping options

Many of the existing sonification tools provide a wide array of parameters for users to control and personalize. *Sonification Sandbox* provides features that allow the user to map data to auditory dimensions such as timbre, pitch, volume, and panning position [3]. *Rotator* implements similar controls, as their interface contains a variety of sliders to adjust settings ranging from tempo, pitch, gain, modulation, and more. This flexibility is also mirrored in *xSonify* to a lesser extent, as the app to have three controls: pitch, loudness, and rhythm. *Sonification Workstation* achieves parameter

[1] Code available at **https://github.com/Kizjkre/siren**, app hosted at **https://kizjkre.github.io/siren/**

mapping through mathematical expressions, where the data tracks are assigned as variables that can be substituted into these expressions [7]. *Sonification Toolkit* has source-level parameter mapping options, as each data source has customizable options including oscillator type, frequency range, and basic effects such as panning [8]. As a digital music instrument mapper tool, *libmapper* offers a variety of controllable synthesis parameters in its synth output [9]. *SIREN* approaches parameter mapping in a novel manner by implementing different settings at different levels of scope. Furthermore, the innovative "channels" concept, described in section 5.3, details the approach that *SIREN* implements.

2.3. Universality and Extensibility

In developing sonification tools, it is important to consider the app's *universality* (that is, the app's ease of access and amenability for collaboration) and *extensibility* (the ease of continued development of the app, and for affording the user updates as new features are implemented).

Most of these applications can be sorted into two categories: native apps and web apps. Most of the apps examined in this section are native apps, including *Sonification Sandbox*, *xSonify*, *libmapper*, and *Sonification Workstation* [3, 6, 9, 7]. Conversely, *Rotator*, *Sonification Toolkit*, and *SIREN* are web apps [5, 8]. Web apps offer greater accessibility and easier navigation in comparison to a native app. Furthermore, web apps can receive instant updates, as refreshing the page would get the newest version. This makes *SIREN* especially easy to access and easy to use.

3. GOALS

SIREN strives for ease of use in data sonification through an intuitive interface and using web technologies that have farther reach than traditional applications. Paired with the Web Audio API, *SIREN* explores some of the use cases for web technologies in sonification in addition to demonstrating how the web can revolutionize data sonification. Furthermore, the *SIREN*'s unique design philosophy allows for the realization of more nuanced models for sonification.

4. DESIGN

SIREN is a single page web application with various control panels that facilitate experimentation with alternate possible mappings, and customization of a final sonification. The main workspace includes a sidebar on the left that acts as a file browser for all uploaded files provided by the user, and the track view on the right for visualizing and sonifying data. Within each track, settings windows and channel windows can be toggled with the "Sonification Settings (Advanced)" button. These windows provide further functionality in sound design. These windows can be organized to best fit within a browser window. The settings offered by *SIREN* for each individual track include volume control, pan, reverberation, discrete/continuous pitches, and functions for altering the data.

4.1. The DAW model

SIREN's design is inspired by the layout of popular digital audio workstations, and the fundamental features are named accordingly. The basic building blocks for *SIREN*'s workflow are *tracks* and *channels*, which provide layers of abstraction to build complex and meaningful sonifications. Other important features of DAWs, such as automation, are also implemented in *SIREN* through channels. *SIREN*'s interface mimics the DAW's track view, in which the audio content of a track is displayed with adjacent controls aligned to each track. Here, the waveforms of the DAW are instead replaced with visualizations of the data in each track. The graph acts as a display of the automation curve when connected to channels. Global controls are available for setting and varying overall tempo, and for selecting pitch organization (scales). These innovative features will be dissected in greater detail in part 5.

4.1.1. Tracks

The basic unit of sonification in *SIREN* is called a "track," and represents one dimension of the data. Multiple tracks can be layered to create a sonify multidimensional data polyphonicallym allowing users to explore the interaction of various aspects of the data. Like DAWs, each track can have different effects applied to it, such as volume, pan, and reverberation, and these can be adjusted specifically for every track. Similarly, track data can be altered to focus on parts of a dataset or to change the sonification mapping scheme from datum to pitches manually or automatically through pruning outliers. Furthermore, mappings between datum to frequencies can be adjusted to be either discrete pitch or a continuous pitch. By itself, the track serves as a starting point for interacting with data, where it can be combined in "channels" to create more complex sonifications.

To create a track, click on a dataset in the file browser and select the dimension from the dataset in the window that appears.

4.1.2. Channels

Channels provide another layer of abstraction on top of tracks, and their primary function is to group tracks together for more advanced sonification techniques. Similar to how automation works in DAWs, channels help automate different sound features using tracks as the automation curve. The motivation behind channels is for multiple tracks to be grouped essentially as one track, providing more possibilities and more unique ways to synthesize sound with the data sequence. These modulation mapping features currently include pitch, volume, pan, and reverb, and these features will alter the final sonification depending on the tracks chosen. Fig 1 provides a visual for channels. Tracks 0 and 1 will sonify normally with any settings applied to it by the user, while tracks 2 through 5 are connected to channel 0, with each track automating a sound parameter. These tracks do not directly influence the synthesized output, but will affect a sound parameter of the output that is synthesized by the channel. This feature allows for a different type of exploration into relationships between the different dimensions of the data aurally, so instead of the traditional approach of simply comparing two data sources pitchwise, datasets can be compared using other sonic qualities. More concrete examples of the usefulness of channels will be presented in section 5.

To create a channel, users can use the channel section of the advanced sonification settings to create channels and connect tracks to channels, as well as control various channel settings such as assigning tracks to automate different sound parameters.

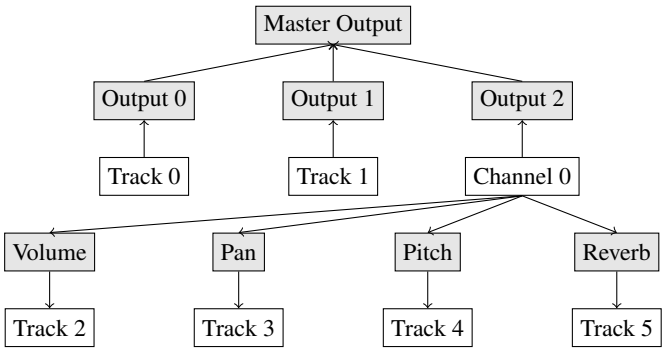


Figure 1: The relationship between tracks and channels

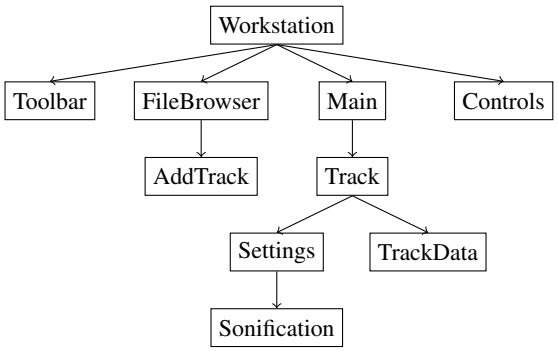


Figure 2: *SIREN*'s architecture

(a) Global state.

(b) Track settings.

(b) Channel settings.

Figure 3: State/Session management structure

4.2. Architecture

Currently, *SIREN* is a frontend single-page application built with React and Redux. Apart from these libraries, *SIREN* also currently uses D3.js[2] for data parsing and visualization, Halfmoon[3] for the UI layout and components, and the Web Audio API for building the sonification. As shown in Fig. 2, all of *SIREN*'s functionality are abstracted into components, encapsulated in a `Workstation` component that serves as the root of the application. From the `Workstation`, there are four subcomponents, with `Main` handling the bulk of the sonification UI, and the `Sonification` window handling most of the options.

To manage the various settings, the state is kept in JSON format, shown in Fig. 3. The global state is shown in (a), while the state for each track and channel is shown in (b) and (c), respectively. After creating a sonification, users can export their sessions to preserve it for future work. *SIREN* uses this JSON object model to manage sessions, and the import and export buttons are used to load and unload these states.

5. EXAMPLES

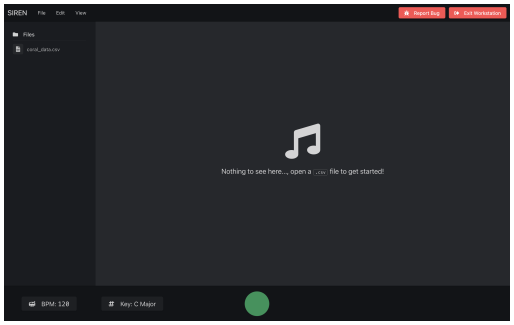


Figure 4: The home screen for SIREN with the default test data `coral_data.csv` loaded in the file browser.

Fig. 4 shows the layout of *SIREN*'s workstation. Users can open CSV files and load them into the file browser on the left,

[2] `https://d3js.org/`.
[3] `https://www.gethalfmoon.com/`.

which is then parsed into individually selectable tracks through a track addition window. The menu bar on top provides useful functions, including loading files, importing and exporting work sessions, editing tracks and channels, as well as adjusting the layout to make more room for the track view. The controls on the bottom provide necessary sonification features, such as tempo/speed, key, and controlling the playback.

5.1. Genome sequence

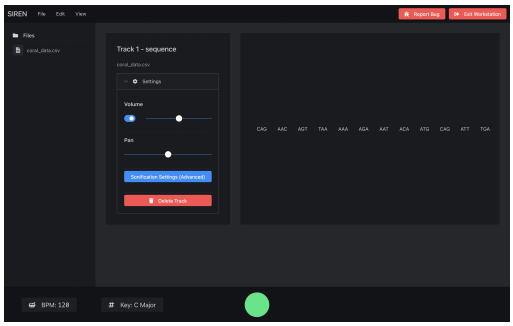


Figure 5: An example of using coral genome sequences as a track, segmented in groups of three nucleotides.

For non-numerical data, *SIREN* provides flexible options for alternative sonification methods. As can be seen in Fig. 5, coral DNA sequences are segmented into groups of three. *SIREN* maps these segments to one or more pitches depending on the segmentation size. In Fig. 5, a segmentation size of three would sonify the track as a triad, with each character mapping to one note. The data display of the track shows the segmented data to visualize the effects of segmentation. From the "Settings" dropdown menu, available for every track, users can adjust track settings, with basic controls such as volume and pan. More advanced settings are found in the "Sonification Settings" window displayed with the blue button. In Fig. 5, the volume is set to 50% while the other features are set to their default values.

5.2. Temperature data

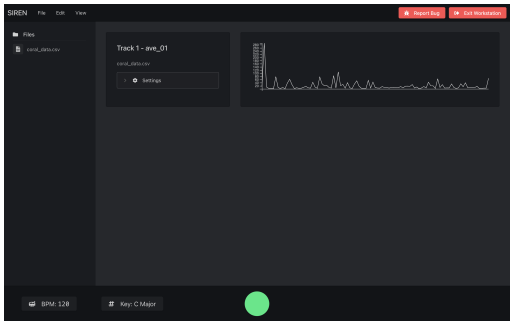
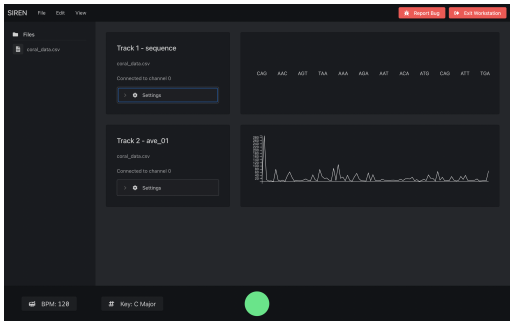


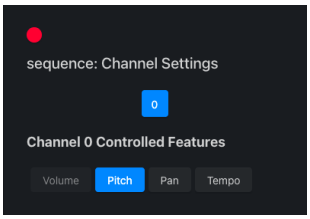
Figure 6: An example of using coral temperature data as a track displayed in a graph.

Numerical data such as temperature can also be easily sonified with *SIREN*. Fig. 6 demonstrates an example of what a numerical track would look like. Instead of a table of values, there is a scaled graphical display of the data in the track. The graph also acts as a reference for the changes in frequency for single-track sonifications, and as the automation curve when connected to a channel. Tracks like these can offer an aural perspective on changes throughout time, or to detect variations within different samples.

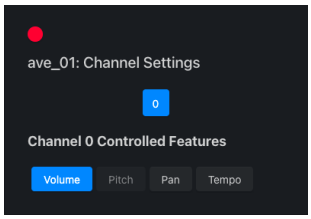
5.3. Channelizing the two



(a) The workstation interface after connecting tracks 1 and 2 to a channel.



(b) The channel settings window for track 1.



(c) The channel settings window for track 2.

Figure 7: An example of using coral temperature data and genome sequence in a channel.

By loading both examples from Fig. 5 and Fig. 6 into tracks simultaneously, the user can connect these tracks to channels, which are then denoted by the text on the information box for each track. In the example shown in Fig. 7, both tracks are connected to channel 0, with each track automating a different sound feature, as shown in the channel settings for each track in Fig. 7 (b) and (c). As seen in the channel settings, track 1 controls the pitch of the output, while track 2 controls the volume. This allows for a multidimensional aural exploration of the data, as tracks with the same dimensions or measurement units can be grouped into different channels, each automating the same sound synthesis parameter in different channels. Thus, the interactions between not only the same type of data can be sonified meaningfully, but also between different types of data, in this case in Fig. 7, between the genome sequence and temperature.

6. DISCUSSION

6.1. Use Cases

SIREN has a variety of use cases. Most notably, *SIREN* was featured in the workshop "Sound as ocean memory: ecoacoustics, audification and sonification" hosted by the Ocean Memory Project, where *SIREN* was used as a vehicle to explore data collected on

coral genomes and temperatures. Combining DNA sequences and coral temperatures in a channel demonstrated a meaningful use case for exploring the dataset in a unique method. Apart from oceanic data, *SIREN* has the potential to sonify other datasets in different areas of study. Through the original concept of channels, *SIREN* provides another perspective into parameter mapping sonification, and offers an easy tool for rapid sonification prototyping as well as an accessible introduction into parameter mapping sonification for beginners. Due to the low-friction onboarding and intuitive concepts of tracks, *SIREN* also serves as a potentially useful tool in an educational setting, where newcomers can test out data sonification through *SIREN* before diving deeper.

6.2. Limitations of *SIREN*

Although *SIREN* has plenty of innovative features that enable complex data sonification easily, there are limitations to *SIREN*'s functionality,, which will be addressed in section 7. One limitation of *SIREN* right now is the actual sonification output. As of the writing of this paper, *SIREN* only supports sine wave oscillators for each track and channel. Additionally, *SIREN* only supports well-formed, uploaded CSV files, which limits the scope of data that can be explored with *SIREN*. Most importantly, *SIREN*'s sonified output cannot be exported as a sound file, and can currently only be shared by means of sharing a *SIREN* session file and imported into the app.

7. FUTURE WORK

Many features, such as more sonification options, sound features, as well as improving the versatility of channels, have yet to be implemented, and future work on *SIREN* has been planned to extend its functionality. Following *SIREN*'s design philosophy, all work and development on *SIREN* will be rooted in the fundamental concepts of tracks and channels.

As stated in section 6, one of *SIREN*'s limitations is the lack of fully-customizable of the output timbre. Future work on *SIREN* includes supporting not only the other types of basic oscillators (square, sawtooth, and triangle), but also custom waves. Accompanying this will also be a beginner-friendly yet extensible user interface to craft custom oscillators as well as have the ability to upload samples that can personalize each track and channel. Beyond this, *SIREN* also aims to support a graphical programming interface aligned with the workflow of the Web Audio API to design custom synthesizers that can be applied to tracks and channels.

Another area of focus for *SIREN*'s development will be on incorporating other sonification methods, such as frequency modulation and granular synthesis, which will provide another layer of customization and extensibility for *SIREN*.

On the other end, making *SIREN* an accessible tool for everyone is another goal for this project, so improving the user interface will also be a priority for future work. Among these UI improvements will be restructuring the settings window to give adequate space for all of the functionality that *SIREN* provides, as well as other quality of life display changes that allow tracks to be displayed better and seeing the progression of what is played. Channels will also receive UI updates, as it is a core component of *SIREN*, and the interface for creating and managing different channels will be more graphical and intuitive. Furthermore, work is planned on easier dataset uploading capabilities as well as developing a "hub" for sharing datasets and sonification results.

8. REFERENCES

[1] W3C, "Web audio api," https://www.w3.org/TR/webaudio/, 2021.

[2] F. Grond and J. Berger, *Chapter 15: Parameter Mapping Sonification*. Berlin, Germany: Logos Publishing House, 2011, pp. 363–397.

[3] B. N. Walker and J. T. Cothran, "Sonification sandbox: A graphical toolkit for auditory graphs," in *Proceedings of the 2003 International Conference on Auditory Display*, 2003.

[4] B. Walker, "Sonification sandbox," http://sonify.psych.gatech.edu/research/sonification_sandbox/index.html, 2009.

[5] J. Cherston and J. A. Paradiso, "Rotator: Flexible distribution of data across sensory channels," in *The 23rd International Conference on Auditory Display*, 2017.

[6] R. M. Candey, A. M. Schertenleib, and W. L. D. Merced, "xsonify sonification tool for space physics," in *Proceedings of the 12th International Conference on Auditory Display*, 2006.

[7] S. Phillips and A. Cabrera, "Sonification workstation," in *The 25th International Conference on Auditory Display*, 2019.

[8] H. Lindetorp, "Webaudioxml sonification toolkit," https://github.com/hanslindetorp/SonificationToolkit/, 2021.

[9] R. M. Candey, A. M. Schertenleib, and W. L. D. Merced, "From controller to sound: Tools for collaborative development of digital musical instruments," in *Proceedings of the International Computer Music Conference*, 2007.

9. ACKNOWLEDGMENTS

Work on SIREN was supported by a grant from the Woods Institute for the Environment, Stanford University, and by the Ocean Memory Project.