SOURCE: a Freesound Community Music Sampler

Frederic Font

frederic.font@upf.edu

Music Technology Group, Universitat Pompeu Fabra

Spain

ABSTRACT

SOURCE is an open-source hardware music sampler powered by Freesound's collection of near 500k Creative Commons sounds contributed by a community of thousands of people around the world. SOURCE provides a hardware interface with Freesound and implements different methods to search and load sounds into the sampler. We see SOURCE as a proof of concept device that can be used as an extendable base system on top of which further research on the interaction between hardware devices and online large sound collections can be carried out. This paper describes the architecture of SOURCE and the different ways in which it interacts with Freesound. Even though we have not carried out a formal evaluation of SOURCE our informal tests show that SOURCE can be successfully integrated into a live music performance setup, and influence the creative process in interesting ways by being able to quickly generate new rich sound palettes that would otherwise be difficult to create with traditional hardware music samplers.

CCS CONCEPTS

• **Human-centered computing**; • **Information systems** → *Users and interactive retrieval*; *Digital libraries and archives*;

KEYWORDS

music sampler, audio collections, freesound, music information retrieval, smart musical instruments, internet of audio things, internet of musical things

ACM Reference Format:

Frederic Font. 2021. SOURCE: a Freesound Community Music Sampler. In *Proceedings of Audio Mostly (AM'21), September 1–3, 2021, Trento, Italy*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3478384.3478388>

1 INTRODUCTION

Among other things, the creative process of musicians and, more generally, *audio practitioners*, is highly influenced by the features of existing software and hardware creation tools, and by the audio content available for reuse. For example, the introduction of hardware *music samplers* that had the ability to sample existing recordings to make new music, greatly contributed to the definition of the *hip hop* music genre in the 1970s and 1980s. Nowadays, with the proliferation of the *remix culture* [8], the creation of new audio

content which reuses other existing sampled content is commonplace. In that context, resources such as Freesound¹ [4], an online sound sharing site currently hosting near 500k reusable sounds released under Creative Commons licenses, are incredibly useful for audio practitioners. However, the exploration of ways in which creators can interact with such a large collection of sounds and integrate them in their creative processes remains an open research question.

Within the Music Technology Group of Universitat Pompeu Fabra² (the research group where Freesound was started in 2005 and where it is still being developed and maintained), we have explored different ways of interacting with Freesound beyond the standard *text search* interaction style implemented in the usual web interface. For example, we have explored ways to organize the results of a textual research query in a 2-dimensional map with sounds projected according to some notion of acoustic similarity [3]. As another example, we explored ways to filter search results based on high-level audio perceptual qualities and to directly integrate such results into a digital audio workstation [2]. Similar works have also been carried out outside the Music Technology Group (see Section 2 for more details). However, and to the best of our knowledge, all existing works exploring novel interaction methods with large audio collections are based on the *computer keyboard and pointing devices* paradigm. However, for audio practitioners and, more specifically, in the case of music creation and live performance, it has been argued that having dedicated *hardware* interfaces that go beyond the aforementioned interaction paradigm can benefit the creative process and bring new and interesting elements to it [6]. We therefore believe that this is a path worth exploring, particularly in the context of Freesound thanks to the integration possibilities that it offers through its API.

In this paper we present SOURCE, a hardware music sampler powered by Freesound's collection of near 500k Creative Commons sounds contributed by a community of thousands of people around the world. SOURCE brings some ideas from previous research on interaction with large audio collections into a hardware device with a user interface that includes buttons, faders, a digital encoder and a small display. We describe the architecture of SOURCE, and explain the ways in which it interacts with Freesound. Rather than as a *finished* device, we see SOURCE as a proof of concept of a system that can be further extended to carry out research about the interaction with large audio collections beyond the computer keyboard and pointing devices interaction paradigm. Even though the current version of SOURCE has not yet been formally evaluated, it has been informally used by the authors in the context of a live music performance setup with successful results.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

AM'21, September 1–3, 2022, Trento, Italy

© 2021 Association for Computing Machinery.

ACM ISBN 978-1-4503-8569-5/21/09...\$15.00

<https://doi.org/10.1145/3478384.3478388>

¹<https://freesound.org>

²<https://mtg.upf.edu>

SOURCE is open source and the code can be found in a public source code repository³. Although it has been designed to work on a specific hardware solution based on a Raspberry Pi⁴ (see Section 3.1), SOURCE can also be run as an *audio plug-in* with somewhat limited functionality in desktop computers and laptops. A video demonstrating the most important features currently implemented in SOURCE is publicly available and linked in the source code repository.

The rest of the paper is organized as follows. In Section 2 we describe the related work and devices that inspired the development of SOURCE. In Section 3 we provide details about the design, implementation and features of SOURCE. Section 4 provides pointers into future research directions for extending SOURCE. We end the paper in Section 5 with a summary and final conclusions.

2 RELATED WORK

SOURCE draws inspiration from traditional music samplers and implements many *sample-playing* features commonly found in those devices. Music samplers are well-established hardware devices commonly used by audio practitioners. Nowadays, a plethora of music samplers is available commercially from many well-known brands including AKAI, Roland and Elektron⁵. It is a common feature of music samplers that these include advanced sequencing features which essentially turn them into complete digital audio workstations. However, the ways in which commercially available hardware music samplers interact with audio collections are rather limited and have remained mostly the same since the early mass-produced digital samplers.

Sounds are typically loaded into samplers either by live-recording them, or by using a computer to transfer audio files to a storage system (usually an SD card) that can be read by the device. Sounds stored in the storage system may be hierarchically organized in folders which can be navigated following a similar paradigm to that of desktop computer file systems, but adapted to the characteristics of the sampler interface (e.g. small display (if any), available buttons, etc.). To the best of our knowledge, none of the commercially available music samplers puts a special emphasis on providing innovative ways to browse the sound collection available in the storage system, and none features an internet connection that would allow fast interaction and integration with external audio collections. As a result, the process of loading sounds to the sampler is a manual and time-consuming one. For example, if we want to load an interesting sound we just found in Freesound to our sampler, we will most likely need to download the sound, possibly convert it to a supported audio format, extract the SD card from the sampler and connect it to a computer, copy the sound to the card, insert the card again in the sampler, locate the sound using the sampler's browsing interface, and finally select the sound for loading it. Even though this process can be streamlined with the aid of computer tools (e.g. see the *Volca-Freesound*⁶ project, which can transfer Freesound sounds to Korg's Volca Sample⁷ directly from a web browser), we believe that music samplers could boost their creative potential by

providing innovative ways of interacting with large audio collections and, in particular, with online sound collections which can be regularly updated such as Freesound.

Unlike with hardware music samplers, in the context of desktop computers (and sometimes mobile devices as well) there have been many research efforts exploring novel ways of interacting with large audio collections. For example, the Freesound website incorporates a *similarity search* feature that allows users to retrieve sounds which are acoustically similar to a given example. A similar notion of acoustic similarity has been used in *Freesound Explorer*, a sound exploration and music making tool that projects the results of a Freesound textual query into a 2-dimensional map where close sounds are more acoustically similar than distant ones. This last example follows from a tradition of works exploring the idea of projecting sound and music items into a map (e.g. [5, 7, 10]). Still in relation to Freesound, recent research has focused on the extraction of high-level perceptual audio features which automatically characterize sound samples with concepts such as *brightness*, *depth* or *hardness* [9]. Such features have been integrated into a *Max4Live*⁸ device which allows filtering Freesound sounds using perceptual features and directly integrating the results in Ableton Live⁹ [2]. It is also worth mentioning the case of the *Freesound Loop Generator*¹⁰, which features a step sequencer with 16 sample slots in which short sounds from Freesound are automatically loaded after the user introduces a textual query (or uses one of the available randomization methods). In this case, the interesting idea is that users don't choose exactly what sounds are loaded in the sampler but only provide a general guideline, and the exact sounds themselves are selected from the system among all sounds available in Freesound that match a specific criteria.

There also exist commercially available software tools that include advanced ways of navigating audio collections and loading sounds into a sampler. For instance, *XO* is an audio plugin by *XLN Audio*¹¹ which analyzes users' local audio collections and generates a map for navigating them (similarly to the map-based systems mentioned above). As another example, *Loopcloud* by *Loopmasters*¹² also analyzes users' local audio collections to extract semantic descriptions that can be used to navigate such collections beyond the classic hierarchical folder structure. Finally, there also exist commercial samplers integrated with online sound collections such as *AudioTexture* by *LeSound*¹³ (which is integrated with Freesound), and *Arcade* by *Output*¹⁴, which is connected to a big proprietary collection of sounds and loops that is updated regularly and that can be browsed within the sampler plugin.

SOURCE is a hardware music sampler that extends the classic idea of the hardware sampler by being able to search and download sounds from Freesound. Furthermore, SOURCE incorporates some innovative ways to retrieve content from Freesound inspired by the aforementioned works, and it can be extended to carry out further research in such directions. We believe the ideas behind

³<https://github.com/ffont/source>

⁴<https://www.raspberrypi.org/products/raspberry-pi-4-model-b>

⁵<https://www.akaipro.com>, <https://www.roland.com>, <https://www.elektron.se>

⁶<https://hisschemoller.github.io/volca-freesound>

⁷https://www.korg.com/us/products/dj/volca_sample

⁸<https://www.ableton.com/en/live/max-for-live>

⁹<https://www.ableton.com/en/live>

¹⁰<https://ffont.github.io/freesound-loop-generator>

¹¹<https://www.xlnaudio.com/products/xo>

¹²<https://www.loopcloud.com/cloud>

¹³<https://lesound.io/product/audiotexture-free>

¹⁴<https://output.com/products/arcade>

Figure 1: Block diagram

SOURCE are well aligned with the emergence of the *Internet of Musical/Audio Things* [13, 15] paradigms and the concept of *Smart Musical Instruments* [11]. From that perspective, SOURCE is very related to *cloud-smart* musical instruments such as the *Smart Mandolin* [12] and the *Smart Guitar* [16], in which modified physical instruments with an embedded sound processor and wireless connectivity can access online Creative Commons sound and music repositories to retrieve audio files to be used as accompaniment.

3 SYSTEM DESCRIPTION

SOURCE is composed of a number of software processes that run on a hardware solution based on a Raspberry Pi 4 Model B, the Elk Pi¹⁵ hat for the Raspberry Pi (which provides low-latency multi-channel audio I/O), and the Elk BLACKBOARD¹⁶ controller board (which provides the user interface elements including buttons, faders, a display, and the audio I/O connectors). All software processes run under Elk Audio OS¹⁷, an open source operating system optimized for low-latency and real-time audio systems [14]. The core of SOURCE is the *sampler engine* which is implemented as a VST plugin and is run by the *sushi* process (a plugin host bundled with Elk Audio OS). The communication with the sensors of the controller board is carried out by the *sensei* process, which is also part of the Elk Audio OS distribution. Finally, a *glue app* is responsible for connecting all the sub-systems together (mostly via *Open Sound Control*, OSC¹⁸), controlling the state of the user interface, exposing an HTTP endpoint that offers a complementary user interface, and, most importantly, communicating with Freesound to search and download sounds. Figure 1 shows a block diagram including all the aforementioned software processes and hardware elements. The following sections provide more details about the user interface (Section 3.1), the sampler engine (Section 3.2) and the glue app

¹⁵<https://elk.audio/extended-dev-kit>

¹⁶<https://elk.audio/blackboard>

¹⁷<https://elk.audio/audio-os>

¹⁸https://en.wikipedia.org/wiki/Open_Sound_Control

(Section 3.3), including details about the ways in which SOURCE communicates and interacts with Freesound (Section 3.3.1).

3.1 User interface

Figure 2: Photo of the device, showing the user interface

Figure 3: Alternative web-based user interface

The SOURCE user interface is provided by the controls and feedback elements available in the BLACKBOARD controller board (Figure 2). The controller board contains 9 buttons (each with a LED on top), 4 faders, 1 encoder with push button, 1 potentiometer, and a 128x64 OLED display. Additionally, the controller board includes all the audio I/O connections (from which the current release of SOURCE only uses a single stereo output). The encoder is generally used to navigate through menus and configure some parameters shown in the display. The faders are mainly used to adjust specific sound parameters (see Section 3.2), and the red buttons are used to trigger sounds or select sounds loaded in the sampler. The left-most button is used as a *shift* modifier which adds a second layer of functionality to the rest of user interface controls. Finally, the

Figure 4: Controller board simulator used in development

potentiometer is used to control the global output volume of the device.

Even though all of the features implemented in SOURCE can be accessed using these user interface elements, SOURCE also offers a complementary user interface that can be loaded in a web browser (as long as SOURCE and the device running the web browser are in the same local network). This complementary interface facilitates things like introducing text (e.g. to enter the query terms for a search query or the name of a preset), showing historic sound usage information (to facilitate attribution of Creative Commons sounds) and transferring sounds and presets to the device. The complementary interface can also be used as an alternative way to control SOURCE without using the hardware user interface, showing most of SOURCE controllable parameters in one single screen (Figure 3). Furthermore, the complementary interface also includes a simulator of the controller board (Figure 4) which can be specially useful if the controller board is not available, for example when performing development in a laptop computer.

3.2 Sampler engine

The sampler engine is implemented as a VST audio plugin written in C++ using the JUCE library¹⁹. We choose the VST plugin format because of it being a standard for audio development and because it can potentially run in multiple platforms and applications. Nevertheless, using the JUCE platform it is possible to export the sampler engine using other plugin formats. The sampler engine implements all the audio playback functionality and handles MIDI events (MIDI notes and *control change* messages). It is designed to run *headless* (without a graphical user interface), and be remote controlled using OSC messages. The sampler engine holds its own application state including *i*) a *volatile* part of the state which is expected to change very fast and stores things like the playhead position of the sounds that are currently being played and the *RMS* level of the output signal; *ii*) the *preset* part of the state which mainly includes a list of the sounds loaded, their metadata and their parameter settings; and *iii*) a *global* state which includes some application-level settings that persist between plugin runs and are not preset-specific. In order to show information about the engine state in the user interface, the sampler engine needs to share the state with the glue app. To that

end, the sampler engine and the glue app communicate in different ways. *Preset* and *global* state parameters are sent to the glue app using a specific endpoint of its HTTP server at regular intervals of 1 Hz. The *volatile* state is much smaller and is sent to the glue app via OSC upon request, normally updating at a rate of 15 Hz.

The sampler engine implements audio playback functionality which is common in many existing music samplers. In most cases, the implementation of such functionalities is very simplistic and can not be compared quality-wise with commercially available options. However, the innovation focus of SOURCE is not on the sampler engine but on the ways in which that engine is populated with sounds. When a sound is loaded into the engine, SOURCE can configure a number of sound parameters including the start/end and loop start/end positions, a classic ADSR amplitude envelope, the cutoff and resonance of a low pass filter with its ADSR envelope, the MIDI root note at which the sound should be played with normal speed, an overall pitch modifier, and a number of modulation settings that can be assigned to note velocity and aftertouch (including support for *polyphonic* aftertouch). Furthermore, some of these parameters can be mapped to MIDI control change messages on a per sound basis. Sounds can also be played in slices assigned to different MIDI notes. The engine is capable of loading any number of sounds (only limited by the amount of available RAM memory), and can do arbitrary mappings between the loaded sounds and MIDI notes (configured through the glue app). The SOURCE demonstration video linked in the source code repository showcases some of the possibilities of the engine.

Even though the sampler engine is designed as a VST plugin which is expected to run headlessly without a plugin user interface, we also include the possibility to compile SOURCE as a *stand alone* audio plugin which does not require the glue app and can be used in desktop computers. In that case, the sampler engine bundles an HTTP server and a *web view* component which serves the alternative SOURCE user interface shown in Figure 3 and is used as the plugin graphical user interface. As mentioned above, the communication with Freesound is normally handled by the glue app. However, minimal interaction capabilities have also been implemented in the sampler engine so that it can still carry out basic Freesound interaction and download sounds when running without the glue app. This facilitates the use of SOURCE without the original technology stack for which it was designed.

3.3 Glue app

The glue app is responsible for interconnecting all the sub-systems running in SOURCE and handling all the communication with Freesound. This is the component that incorporates the major differences between SOURCE and traditional hardware music samplers. It is written in Python and can be easily modified and extended. The glue app incorporates 5 main sub processes which are constantly running in parallel:

- **User interface controller:** this process is responsible for responding to user actions (e.g. a user pressing a button), for setting the state of the LEDs of the controller board and for generating the contents of the display. Also, the main application logic of the glue app is implemented in this process, including the communication with Freesound. This

¹⁹<https://juce.com>

process allows users to do things like searching for sounds, adjusting sound parameters (including, among many others, a visualization of the waveform to set sound slices and a keyboard-like interface to configure MIDI note mappings), saving and loading presets, and configuring global SOURCE settings. The different ways in which SOURCE interacts with Freesound to generate new presets and load sounds to the sampler are described below in Section 3.3.1.

- **Sound downloader:** downloading of sounds happens in its own process and is asynchronously reported to the sampler engine so that it knows when a sound has finished downloading and is ready to be loaded. Having the downloading process separated from the rest of the processes helps making sure that potential slow downloads and other server-side errors have a controllable impact on the user experience. Downloaded sounds are kept in a folder in the local storage of SOURCE so that, if requested again in the future, sounds already downloaded are not downloaded again.
- **OSC receiver and sender:** this process allows for a bidirectional communication via OSC with the sampler engine. This is used to remotely control the engine and to receive volatile state information (see Section 3.2).
- **HTTP server:** the HTTP server implements the necessary endpoints to serve the complementary SOURCE user interface described in Section 3.1. In some cases, it is also used by the sampler engine to post state information (see Section 3.2).
- **Sample engine state synchronizer:** this is a process that keeps asking the sampler engine to transmit state updates which are later received either through the OSC receiver or the HTTP server.

3.3.1 Interaction with Freesound. SOURCE features an integration with Freesound which allows it to load sounds from a growing collection of near 500k sounds. Essentially, SOURCE provides methods to generate new presets by searching and loading sounds from Freesound. When generating a new preset, SOURCE will ask the user to introduce some textual query terms, to define how many sounds should be retrieved from Freesound (N), and to indicate what should be their minimum and maximum duration. SOURCE will also ask what method should be used to map loaded sounds to MIDI notes: in *contiguous* mode the N retrieved sounds will be mapped to contiguous MIDI regions of $128/N$ notes, while in *interleaved* mode adjacent MIDI notes will be mapped to different sounds. In addition, the user is also able to set filters for *brightness*, *hardness*, *depth*, *roughness*, *boominess*, *warmth* and *sharpness*, the perceptual qualities developed in [9]. Finally, a license filter can also be applied which will allow to either include sounds from *all licenses*, include *only sounds under the public domain*, or *exclude sounds with restricted commercial usage*. Once the query is defined, SOURCE will try to get a maximum of 150 sounds from Freesound that match the query parameters, and then do a random selection of the N sounds that will be finally loaded in the sampler engine. This set of filters is quite powerful because it gives users a certain control about the output of the generated preset, but still keeps room for unexpected sounds that might influence the creative process in interesting ways. For instance, if a user wants to create a preset with percussive wood-like sounds to be played on a 4x4 pad

grid, she might search for 16 short sounds with the query term "wood" and map them using the *interleaved* mode so that adjacent pads trigger different sounds. If the user wants to create a playable pitched instrument, the number of loaded sounds can be limited to 1, the minimum duration set above 3 seconds, the query term set to concepts such as "violin single note", and the sound mapped using the "contiguous" mode so that playing keys in a MIDI keyboard will play the sample at the different expected pitches. If the user wants to search for dark ambient pads to bootstrap a new piece, the brightness and depth filters could be used together with the query term "pad" and a rather long duration filter.

Another interesting interaction method implemented in SOURCE is the *random* option. When generating a new preset, SOURCE also provides the option to not specify any query terms. In that case, only the above mentioned filters are applied, and SOURCE will select N sounds randomly from the whole set of search results (which can be huge if filters are not very restrictive). Using the random feature, the sounds loaded into the sampler will most likely be completely unexpected. Despite the simplicity of this idea, in our informal tests the random option has been shown to be a very useful feature for bringing new inspiration to the creative process and for suggesting the use of sounds that otherwise would have never been reached.

A third method with which SOURCE interacts with Freesound is by taking advantage of the notion of acoustic similarity as implemented in the *similarity search* feature of Freesound. Using this method, SOURCE can replace all the sounds of a preset by similar alternatives. Because SOURCE can search and download sounds without stopping the sampler engine, an interesting workflow can be applied in which the sounds of a preset can be iteratively replaced by similar ones while an externally generated sequence of MIDI notes keeps on triggering sounds. This allows the user to experiment with variants of the loaded sounds which keep on evolving to new directions when the *similarity jumps* accumulate. Replacing sounds by similarity can also be done at the individual sound level (i.e. only replacing one sound at a time). Similarly, the textual query and random methods mentioned above can also be applied at the individual sound level.

When sounds are downloaded from Freesound, SOURCE stores some metadata associated with them. This includes basic information such as the name of the sound, the name of the user who created it, and the license with which the sound is released. This is later used to generate sound usage logs that can be used to facilitate attribution requirements of Creative Commons licenses, and that can be accessed using the complementary user interface. Furthermore, when downloading sounds from Freesound, SOURCE also takes advantage of the fact that Freesound analyzes all the sounds with the audio analysis library Essentia [1] and makes the resulting audio descriptors available. SOURCE retrieves information about the audio onset positions estimated by Essentia and can use this information to determine the position of the slices when playing a sound in *slice mode*.

4 EXTENDING SOURCE: FUTURE RESEARCH DIRECTIONS

SOURCE can be extended in different ways. Because of its open source nature, modifications can be introduced that meet the specific needs of researchers and audio practitioners. Even though SOURCE is designed to run on a specific hardware stack, we have seen that it can also be run on other platforms including desktop computers and without the controller board. For development purposes, the provided controller board simulator (Figure 4) can be used for quick prototyping. Furthermore, SOURCE can also run as a stand alone audio plugin on a desktop computer without the glue app. In that case, the methods of interacting with Freesound are limited because the interaction with Freesound is normally handled by the glue app, but SOURCE can still be run and controlled using the alternative interface bundled in the plugin (Figure 3). Therefore, the possibilities of adapting and extending SOURCE are not limited to using the original hardware stack that is described in this paper. Instructions for running SOURCE in different setups are outside the scope of this paper, but can be found in the source code repository. What follows is a list of some potential future research directions for extending SOURCE:

- An interesting research direction is to investigate how can we use a notion of acoustic similarity to browse and select sounds in a 2D space similarly to how it is done in some of the works mentioned in Section 2. A big challenge when bringing this concept to SOURCE is that the hardware user interface does not provide a big enough display to show such a space. However, the alternative user interface loaded in a mobile device or laptop computer could be very useful to that end.
- Another interesting research direction is the automatic generation of new *rhythmic* presets of 16 sounds for the sampler. The idea is that the 16-sound sets mapped to a 4x4 grid are a de facto standard in music production tools that generate rhythms, and similar types of sounds (e.g. kick drums, snares) are typically mapped to the same positions in the grid. If we are able to characterize acoustically how rhythmic sounds are typically mapped in a 4x4 grid, we could then use this information to map sounds from Freesound which feature similar acoustic characteristics and create rhythmic presets more intelligently than with the current pseudo-random assignment of sounds to the grid (i.e. the *interleaved* mode).
- Similarly to the previous idea, a use case also exists for the generation of *pitched instrument* presets that can be played with a MIDI keyboard. In that case, Freesound could be queried to find multiple recordings of single-notes from a specific instrument with different pitches and even velocity layers. That would facilitate the generation of good quality *multisample* presets to be played with SOURCE.
- Even though the original inspiration for making SOURCE came from hardware music samples, it would be worth also exploring the idea of implementing a new interface for SOURCE to be used when running as an audio plugin in a desktop computer or laptop, and which could implement all the methods to interact with Freesound which are currently found in the hardware version of SOURCE.

5 CONCLUSIONS

In this paper we have introduced SOURCE, an open-source hardware music sampler which features a rich integration with Freesound. We have contextualized the idea of SOURCE and described its architecture, as well as pointed into some relevant future research directions. Even though we have not carried out any formal evaluation of the current release of SOURCE, our internal tests have shown that SOURCE can be successfully integrated into a live music performance setup and that the different methods with which it retrieves sounds from Freesound are useful to bring new sounds into the creative process in ways in which traditional hardware samplers are not capable of. We expect to continue developing SOURCE in the future and make it evolve into new directions. We are also open to external contributions and would like to see SOURCE evolving into a community project.

REFERENCES

- [1] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Zapata, and Xavier Serra. 2013. ESSENTIA: An open-source library for sound and music analysis. In *MM 2013 - Proceedings of the 2013 ACM Multimedia Conference*. 855–858. <https://doi.org/10.1145/2502081.2502229>
- [2] A. Correya. 2017. *Retrieving Ambiguous Sounds Using Perceptual Timbral Attributes in Audio Production Environments*. Technical Report. Universitat Pompeu Fabra. <https://doi.org/10.5281/ZENODO.1098587>
- [3] Frederic Font and Giuseppe Bandiera. 2016. Freesound Explorer: Make Music While Discovering Freesound!. In *Proceedings of the Web Audio Conference*. https://repositori.upf.edu/bitstream/handle/10230/32538/font_wac_freesound.pdf?sequence=1&isAllowed=y
- [4] Frederic Font, Gerard Roma, and Xavier Serra. 2013. Freesound technical demo. In *MM 2013 - Proceedings of the 2013 ACM Multimedia Conference*. 411–412. <https://doi.org/10.1145/2502081.2502245>
- [5] Sebastian Heise, Michael Hlatky, and Jörn Lovisach. 2008. Sound Torch: Quick browsing in large audio collections. In *Audio Engineering Society - 125th Audio Engineering Society Convention 2008*, Vol. 2. 819–826. <https://www.aes.org/e-lib/browse.cfm?elib=14696>
- [6] Sergi Jordà. 2008. On stage: The reactable and other musical tangibles go real. *International Journal of Arts and Technology* 1, 3-4 (2008), 268–287. <https://doi.org/10.1504/IJART.2008.022363>
- [7] Carles F. Julià and Sergi Jordà. 2009. Songexplorer: A tabletop application for exploring large collections of songs. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*. 675–680. <https://doi.org/10.5281/zenodo.1418177>
- [8] Lawrence Lessig. 2008. *Remix*. Penguin Press. <https://doi.org/10.5040/9781849662505>
- [9] Andy Pearce, Tim Brookes, and Russell Mason. 2017. Timbral attributes for sound effect library searching. In *Proceedings of the AES International Conference*, Vol. 22-24-June. 29–36. <http://www.aes.org/e-lib/browse.cfm?elib=18754>
- [10] Diemo Schwarz. 2003. The Caterpillar System for Data-Driven Concatenative Sound Synthesis. In *Proceedings of the 6th Int. Conference on Digital Audio Effects (DAFx-03)*. 1–6. <https://hal.archives-ouvertes.fr/hal-01161127>
- [11] Luca Turchet. 2019. Smart Musical Instruments: Vision, Design Principles, and Future Directions. *IEEE Access* 7 (2019), 8944–8963. <https://doi.org/10.1109/ACCESS.2018.2876891>
- [12] Luca Turchet and Mathieu Barthet. 2018. Jamming with a Smart Mandolin and Freesound-based Accompaniment. In *2018 23rd Conference of Open Innovations Association (FRUCT)*. 375–381. <https://doi.org/10.23919/FRUCT.2018.8588110>
- [13] Luca Turchet, Gyorgy Fazekas, Mathieu Lagrange, Hossein S. Ghadikolaei, and Carlo Fischione. 2020. The Internet of Audio Things: State of the Art, Vision, and Challenges. *IEEE Internet of Things Journal* 7, 10 (2020). <https://doi.org/10.1109/JIOT.2020.2997047>
- [14] Luca Turchet and Carlo Fischione. 2021. Elk Audio OS: An Open Source Operating System for the Internet of Musical Things. *ACM Trans. Internet Things* 2, 2, Article 12 (March 2021), 18 pages. <https://doi.org/10.1145/3446393>
- [15] Luca Turchet, Carlo Fischione, Georg Essl, Damián Keller, and Mathieu Barthet. 2018. Internet of Musical Things: Vision and Challenges. *IEEE Access* 6 (2018), 61994–62017. <https://doi.org/10.1109/ACCESS.2018.2872625>
- [16] Luca Turchet, Johan Pauwels, Carlo Fischione, and György Fazekas. 2020. Cloud-Smart Musical Instrument Interactions: Querying a Large Music Collection with a Smart Guitar. *ACM Transactions on Internet Technology* (06 2020). <https://doi.org/10.1145/3377881>