Composing and Improvising Using Sound Content-Based Descriptive Filtering

Austin Franklin
Louisiana State Univeristy
School of Music &
Center for Computation &
Technology
afran84@lsu.edu

Dylan Burchett
Louisiana State Univeristy
School of Music &
Center for Computation &
Technology
dburch6@lsu.edu

William Thompson IV
Louisiana State Univeristy
School of Music &
Center for Computation &
Technology
wthom53@lsu.edu

ABSTRACT

The Freesound Player is a digital instrument developed in Max MSP that uses the Freesound API to make requests for as many as 16 sound samples which are filtered based on sound content. Once the samples are returned they are loaded into buffers and can be performed using a MIDI controller and processed in a variety of ways. The filters implemented will be discussed and demonstrated using three music compositions by the authors, along with considerations for composing and improvising using sound content-based descriptive filtering.

1. INTRODUCTION

The popular online repository for creative commons audio samples known as Freesound.org has been an important resource for composers for many years, allowing them to browse, download, and use samples uploaded by its users for their own projects. The goal of *The Freesound Player* is to create a robust compositional tool that both utilizes this repository of collective commons audio samples while streamlining the compositional process by allowing composers to more quickly narrow down search results using the API and allowing the samples to be quickly recalled, processed, and performed within the Max MSP environment. The authors present 3 compositions using *The Freesound Player* as the primary musical tool, documenting both the creative process and search parameters used for each section of the piece.

2. RELATED WORK

There are several examples of tools that utilize Freeound's API, many of which can be found through Freesound Labs.[1] However, these are often designed to be tools for audio sample discovery or for generating soundscapes, such as the *Freesound Explorer* [1], rather than as compositional tools that are simultaneously intuitive, flexible, and multifunctional. We considered three works in particular.

(i) Playsound.space. Developed by Ariane Stolfi in collaboration with Alessia Milo, Miguel Ceriani, Fabio Viola and Mathieu Barthet, *playsound.space* is a web-based tool that allows users to search for and play creative commons-licensed audio samples from Freesound [5]. This is perhaps the project that is closest conceptually to *The Freesound Player*, but there are several important distinctions. While *playsound.space* can query Freesound samples with text, it cannot filter any of the samples based on the sounds content [6]. This means that a query that can be defined broadly will return a large number of samples and will make locating sounds with a particular characteristic extremely difficult. Additionally, once the samples are returned they can only be looped and layered together in various ways, while *The Freesound Player* is capable of scrubbing and processing audio samples using objects within the Max MSP environment. With a controller connected *The Freesound Player* becomes a sampler, while the additional knobs and faders can be configured to control filters, envelopes, playback speeds, or to achieve any other creative goal.

(ii) Elevator Music Generator. Another project that is conceptually similar is the *elevator music generator*, developed by Stefan Brunner. This project, created in Max MSP, randomly selects audio samples based on their ID, adjusts the playback speed, and mixes them together. The result is a constant stream of continuously transforming music.[2] While this project is not built, or even capable of being used to compose due to its indeterminate design, it does provide a unique framework which utilizes the API in an inherently musical way. What the *elevator music generator* lacks, *The Freesound Player* makes up for by being able to first specify the quality and source of the sounds to be manipulated. The result gives the user more agency by incorporating the filtering functionality of the API while maintaining the ability to transform the audio samples through various MSP objects.

(iii) Freesound Loop Generator. The *freesound loop generator*, developed by Frederic Font, is a 16-step sequencer that allows users to create loops using Freesound audio samples. Users can browse samples using the query and tags, by timeline or similarity, or randomly. The loops can be exported as a WAV file once they are created or the audio samples can be downloaded individually.[3] While this is a creative application of the API, it is better suited for non-musicians because

[1] https://labs.freesound.org/

[2] https://labs.freesound.org/apps/2014/12/14/elevatormusicgenerator.html

[3] https://labs.freesound.org/apps/2020/03/04/freesound-loop-generator.html

Web Audio Conference WAC-2021, July 5–7, 2021, Barcelona, Spain.

Figure 1: The Freesound Player digital instrument

of its simplistic interface and the indeterminate quality of its search results. *The Freesound Player* extends this functionality while allowing for customization in Max determined by skill level. Beginners can compose with the player by using the interface without modification, while more experienced users can implement their own modules and abstractions to make the compositional process more personal.

3. INSTRUMENT DESIGN

The instrument has three main components: a search interface where the user selects from a bank of filters, selects the filter value ranges, sample durations, and keyword(s) for the query (Figure 2), a series of waveform~ objects arranged in a 4x4 grid where the audio samples are visualized once they are returned (Figure 3), and a soundscape feature that randomizes the sample indexes and playback speeds to create continuously changing soundscapes. A demonstration video is available at: https://www.youtube.com/watch?v=APPheMWle0Q.

Conceptually, the design of the *The Freesound Player* is based on a sampler, where audio samples are gathered from Freesound.org and then performed with a controller. The instrument works best with a controller, but can also be used with keyboard keys using Max's key map assign function or by using events to time, sequence, and manipulate playback. When used with a controller such as the M-Audio Trigger Finger[4], each pad can be connected to a specific sample. Holding down a pad will cause the audio sample to loop until the pad is released, while a single press will cause the sample to play once. A double press will cause the waveform~ object to scrub through the audio sample with a randomized scrub speed and range.

3.1 Implemented Filters

All of the filters implemented were created through the AudioCommons initiative [2], which "aims at delivering an ecosystem which gathers audio content creators, content providers, consumers, data scientists and the music tech industry [3]." They were chosen based on their general efficacy while still being able to return at least 16 audio samples in most cases. The filters and their descriptions are listed as follows:

- *depth* - A deep sound is one that conveys the sense of having been made far down below the surface of its source.
- *hardness* - A hard sound is one that conveys the sense of having been made (i) by something solid, firm or rigid; or (ii) with a great deal of force.
- *brightness* - A bright sound is one that is clear/vibrant and/or contains significant high-pitched elements.
- *roughness* - A rough sound is one that has an uneven or irregular sonic texture.
- *boominess* - A boomy sound is one that conveys a sense of loudness, depth and resonance.
- *warmth* - A warm sound is one that promotes a sensation analogous to that caused by a physical increase in temperature.

[4]https://m-audio.com/microsites/tfp/

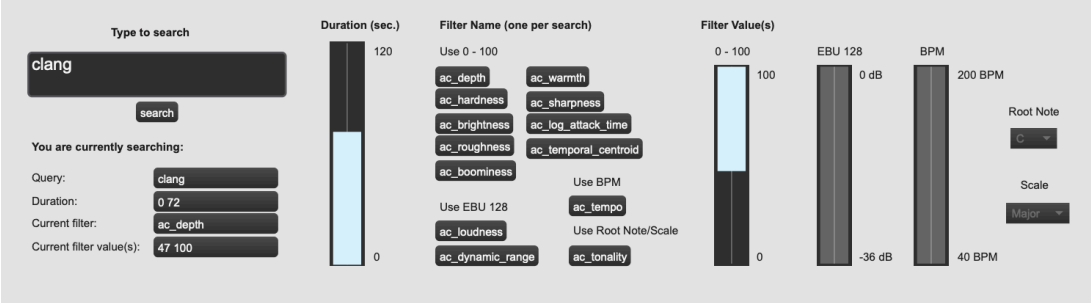

Figure 2: The Freesound Player search interface in Max MSP

- *sharpness* - A sharp sound is one that suggests it might cut if it were to take on physical form.
- *log_attack_time* - The attack time is defined as the time duration from when the sound becomes perceptually audible to when it reaches its maximum intensity.
- *temporal_centroid* - The point in time in a signal that is a temporal balancing point of the sound event energy.
- *loudness* - The integrated (overall) loudness (LUFS) measured using the EBU R128 standard.
- *dynamic_range* - Loudness range (dB, LU) measured using the EBU R128 standard.
- *tonality* - Key value estimated by key detection algorithm.
- *tempo* - BPM value estimated by beat tracking algorithm.

The filter values range from 0-100 in the case of the depth, hardness, brightness, roughness, boominess, warmth, and sharpness filters, while the others use time in seconds, the EBU R128 standard, messages formatted in root note/scale value pairs, and BPM. In Max MSP, rsliders are used to set both the minimum and maximum values for the selected filter. This allows the filters to be applied as broadly or narrowly as desired using a range. Without setting ranges the API will most often return no audio samples because it will only retrieve the ones that match the exact single value specified. Setting a low maximum filter value range (0-25) effectively filters out that descriptor and returns samples that do not have that aural quality, while a high minimum value does the opposite. The Freesound API documentation contains more information about search resources, as well as additional filters not implemented.[5]

A search term, duration range, filter, and filter value range must be chosen before making a request with the API. Figure 2 shows the current search parameters below the "You are currently searching" text on the search interface. Multiple search terms separated by spaces can be used to narrow down a request, but only one filter can be used at a time. The original design of *The Freesound Player* allowed the user to select multiple filters for a single search, but this led to requests that were far too narrow and often did not return any samples, or requests that did not aurally seem to represent the parameters of the search. For example, searching with both the *ac_boominess* and *ac_hardness* filters with a value range of 50-100 might return samples that do not sound exceptionally resonant or firm because these descriptors are often mutually exclusive.

[5]https://freesound.org/docs/api/index.html

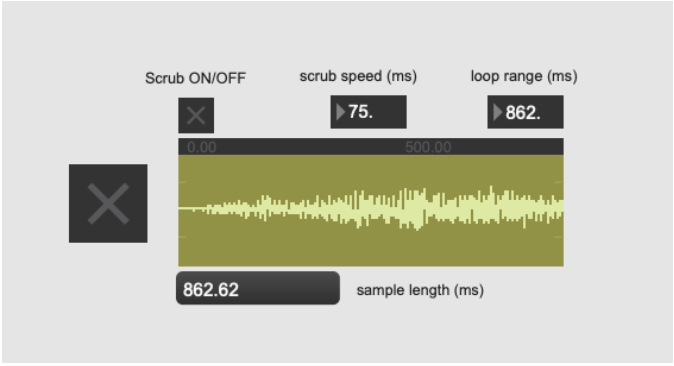

Figure 3: Waveform visualization on The Freesound Player

3.2 Waveform Visualization

The polybuffer~ object is used to load the returned audio samples and visualize the waveform of each sample along with the sample length in ms (Figure 3). The play start and end points on the waveform~ objects can be selected with a mouse to narrow the area of the sample that can be played, and a scrubbing function can be toggled on that randomizes the scrubbing speed and the playback range of the sample.

3.3 Soundscape Generator

The Freesound Player can also be used to create soundscapes, using a metro and a drunk object to constantly vary the interpolation time between samples. Only four samples can be heard at a time, and when a sample is selected the playback speed is randomized between 0.5 and 2. This creates variation each time a sample is played back, and in the context of the other three samples it can often sound like a brand new sound.

4. COMPOSITION TASKS

The authors present 3 compositions using *The Freesound Player* as the primary musical tool, documenting both the creative process and search parameters used for each section of the piece. Other programs, such as DAW's, as well as other Max objects and abstractions to process the audio samples were allowed to be used. In particular, the following

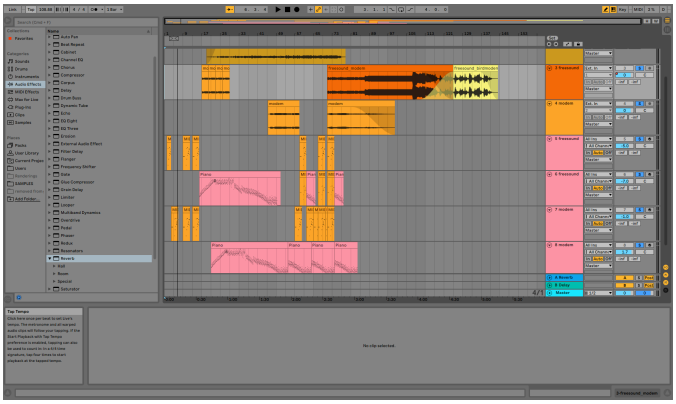

Figure 4: Arrangement view of Orithomodem in Ableton Live

information was documented by each composer:

- Query and filters used for each section of the piece.
- Compositional process (Max processing, DAW's, sensors, and controllers used).
- Musical characteristics of the composition in relation to the employed filters.
- Thoughts about the utility of *The Freesound Player* for music composition or improvisation.

The compositions have been compiled into a playlist on soundcloud. They can found at: https://soundcloud.com/user-563963786/sets/wac-2021-composing-and

4.1 Orinthomodem

Orinthomodem is a composition written by William Thompson IV using *The Freesound Player* that combines two groups of audio sample categories. One such category is a collection of dial-up modems. As a child of the 1980s and 90s, the composer found these sounds nostalgic and unmistakable. The other collection of sounds are of bird songs. This group of sounds were chosen as a way to contrast the digital sounds of modems with sounds from nature.

Using *The Freesound Player* the composer was able to initiate one query for each sound category. Once all 16 buffers were loaded in the Max patch the composer recorded live improvisations. In an effort to collect dial-up modem sounds the query term "modem" was used allowing for durations between 0 and 120 seconds. The filter *ac_warmth* was used with a value range of 54 and 100. With regards to the bird song sound category, the query term "bird song" was used allowing for durations between 0 and 120 seconds. The filter *ac_temporal_centroid* was used with a value range of 0 to 100. Interestingly, in a few of the audio buffers unexpected sounds appeared. This was creatively inspiring. For example, in one such instance human speech saying the names of several birds emerged. This was incorporated into the final composition.

During these improvisations, sounds loaded into buffers were performed via Max's key map assign function which allowed for momentary switch triggers from the computer keyboard. These recorded improvisations were then loaded

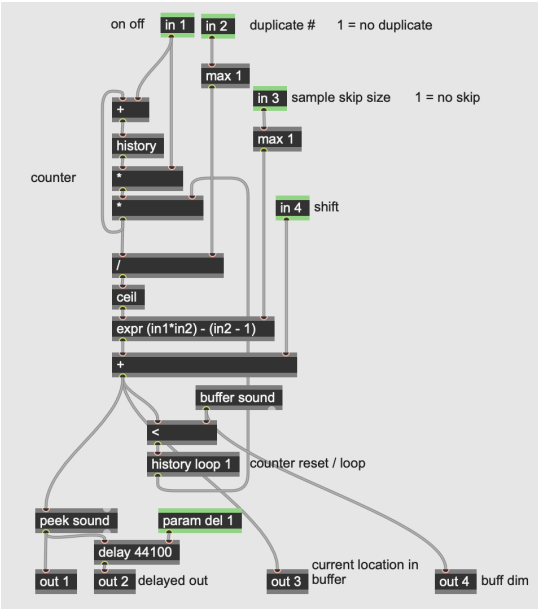

Figure 5: Gen patch added to The Freesound Player for Something in that Drawer

into the arrangement view of Ableton Live to create the arrangement (Figure 4). In Live the composer was able to extract melodic information from a few samples and create sample instruments that use the material. In an effort to comingle these two categories of sound the composer chose to drive samplers loaded with this bird song with the melodic MIDI extracted from modem sounds. Additionally, some of the samples were driven by MIDI that was originally composed and separate from melodic extractions in Live. All other sounds are ambient textures from modem and bird sounds collected using *The Freesound Player*.

4.2 Something in that Drawer

Something in that Drawer is a short composition written by Dylan Burchett that attempts to explore the interplay between sound and text. In order to generate samples that would support this inquiry, all queries were drawn from a Ron Padgett poem entitled "Nothing in That Drawer", the content of which repeats the line "Nothing in that drawer" fourteen times.

The word "nothing" was used with the *ac_depth* filter, the word "in" was used with the *ac_boominess* filter, the word "that" was used with the *ac_warmth* filter, and the word "drawer" was used with the *ac_sharpness* filter. All filters were used with filter values set from 0-100 and a duration between 0-18 seconds. Once all sixteen buffers had loaded samples, each sample was vetted for relevance. All relevant samples selected from each initial group of sixteen were then layered using the soundscape function set at an approximate speed of 250ms and recorded. These soundscape recordings were then loaded into a sample-based Max MSP patch (Figure 5) the composer has used previously for improvisation and performed until a satisfactory result was reached. Something in That Drawer was recorded live and is created entirely from sample soundscapes created with the Freesound Player.

4.3 Submerged

Submerged is a composition written by Austin Franklin that explores a single filter and its relationship to the imagery evoked by the samples returned with it. The piece was created by searching both "deep" and "harmonic" using a duration of 5-30 seconds, and only the *ac_depth* filter with a filter value range of 75-100. Using the *ac_depth* filter, the imagery of an underwater voyage in a submarine is evoked using only the deepest sounding samples for each search.

To construct the form of the piece, the output from Max MSP was sent into Studio One 4 and recorded (Figure 6). Two recordings were taken for each search, yielding four tracks total. The recordings were improvised using sounds that were similar in character for each return, such as longer drone-like samples or noisy samples that sounded similar to a radio, and layered on top of each other. Once the tracks were recorded, the volume was then appropriately balanced between them, since several samples were exceptionally loud inside the Max environment.

The piece was performed using an M-Audio Trigger Finger and the original sounds samples only, with no modification of *The Freesound Player* patch prior to composing. The soundscape function was used for the beginning drone-like section of the piece with a maximum speed of 3500ms, and the scrubbing function was used during a few of the noisier sounds to create chaotic rhythms.

5. CONCLUSIONS

The Freesound Player is a digital instrument built in Max MSP that utilizes the Freesound API to compose and improvise music. The instrument can be used with a controller or additional software such as DAW's, and the audio can be manipulated in Max to utilize various signal processing techniques. Overall, the authors feel that *The Freesound Player* makes composing using sound samples gathered from Freesound.org a much easier task, but the instrument is still inherently improvisational given the indeterminate nature of using the API. The addition of extraneous software makes dictating the structure of the composition a much easier task, whether by using it to record layers of sounds or to arrange improvisations created with the digital instrument.

The authors also found that when performing with the exact same search parameters at different times, the order in which the audio samples were returned had often changed. This particular challenge lends *The Freesound Player* to be used in an improvisational way (especially when using a controller to trigger samples) because the composer cannot guarantee that the location of the samples on the interface will be the same from performance to performance.

One of the authors remarked on finding unexpected samples in their requests that influenced the compositional process or found its way into the final composition. These samples were likely the result of a request with the API that was either too broad or mischaracterized by the owner upon upload to the repository.

6. FUTURE WORK

Currently, there is no CC attribution for samples gathered and performed with *The Freesound Player*. An immediate improvement would involve retrieving the licensing information along with each sound sample and allowing the composer to quickly access that information from within the patch.

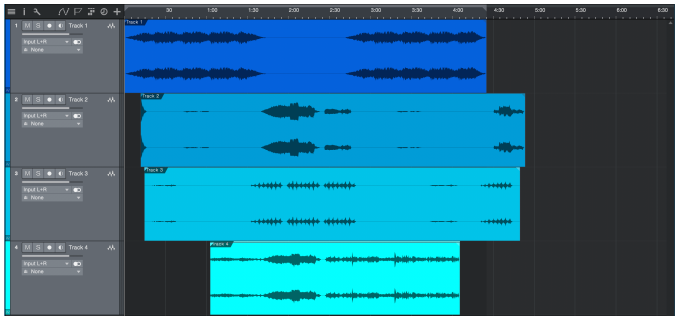

Figure 6: Submerged track recordings in Studio One 4

The authors are still exploring several other improvements to *The Freesound Player*. A Max for Live version of the instrument could allow composers to explore, record, and compose within a single program. This would allow for users to incorporate other M4L devices without needing to use the Max environment or having any experience programming in Max. Students and professionals in the film, television, radio, and music industries could use this instrument within Ableton to create or experiment with audio for film, or to create more accurate temp tracks for composers, making these samples much more likely to be used in films. Other uses include storing samples in packs that are uploaded to the Freesound.org repository and recalled for performances, as well as extending search parameters to include using live input to query samples used for improvisation, similiar to the work Gerard Roma and Xavier Serra have done using a microphone to query Freesound [4].

References

[1] F. Font and G. Bandiera. Freesound explorer: Make music while discovering freesound! In F. Thalmann and S. Ewert, editors, *Proceedings of the International Web Audio Conference*, WAC '17, London, United Kingdom, August 2017. Queen Mary University of London.

[2] F. Font, T. Brookes, G. Fazekas, M. Guerber, A. La Burthe, D. Plans, M. D. Plumbley, M. Shaashua, W. Wang, and X. Serra. Audio commons: Bringing creative commons audio content to the creative industries. In *Audio Engineering Society Conference: 61st International Conference: Audio for Games*, Feb 2016.

[3] A. Milo, M. Barthet, and G. Fazekas. The audio commons initiative. In *DMRN+ 13: Digital Music Research Network One-day Workshop 2018*.

[4] G. Roma and X. Serra. Querying freesound with a microphone. In S. Goldszmidt, N. Schnell, V. Saiz, and B. Matuszewski, editors, *Proceedings of the International Web Audio Conference*, WAC '15, Paris, France, January 2015. IRCAM.

[5] A. Stolfi, M. Ceriani, L. Turchet, and M. Barthet. Playsound.space: Inclusive free music improvisations using audio commons. In T. M. Luke Dahl, Douglas Bowman, editor, *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages

228–233, Blacksburg, Virginia, USA, June 2018. Virginia Tech.

[6] A. Stolfi, A. Milo, M. Ceriani, and M. Barthet. Participatory musical improvisations with playsound. space. In *Proceedings of the Web Audio Conference*, 2018.