# Report for Copy-Paste Sonification

Erik Natanael Gustafsson

2019-06-24

## Pre-compositional Research and Data-Mangling

After having examined the data.json file by hand to understand the structure of the data I parsed it in Rust and analysed the content in terms of word frequency, levels of depth (see depth_wave.txt and data_restyled_wave.txt). Looking up the meaning of some of the most common words in the JavaFX library gave me a better understanding of their function.

Early on in the process I decided that the most obvious and probably most rewarding aspect to use for sonification was how far away from the user, and also the programmer of the software, following the descent deep into library calls. data_restyled_wave.txt shows the wavelike movement up and down the node tree. To add some additional connection to the copy paste theme I gave each event a relevance score based on words appearing in their function names (e.g. "KeyCodeCombination").

To simplify the sonification process I created a score in csv format exporting only the values I would use in the sonification (depth, source and relevance).

## Difficulties I Faced

When trying to compile the actual text editor (to get a feel for the program execution) I ran into a library conflict where the version of javafx required did not seem to be available from Ubuntu repositories (the PPA that used to host it was discontinued). As I did not think it would have a great impact on the work compared to other things I did not pursue this any further.

In order to perform wavetable synthesis using the depth values as a signal I needed to reduce the 9918 values to a considerably smaller list. I first tried to implement the Douglas-Peucker line reduction algorithm, but this implementation failed (seemingly eating all my RAM and crashing the system) so I settled for a naive averaging reduction with the reasoning that at this scale small differences in

accuracy will be imperceptible and smoothing the curve would be beneficial to the sonic usefulness.

Reading the data set and following the functions I had trouble figuring out exactly how and where the copy paste actions were being carried out, if I had gotten the file without context I would have assumed instead that it was a program shutting down (because of the "App" functions having names such as "onCloseEvent" and "closeTabsAndExit()"). If I had had more time I would have done more research into stacktraces and how they work to get a deeper understanding of the data.

## Sonification Process

All of the sonification took place in SuperCollider where I adapted a few synthesis techniques that I have used previously to this specific case as well as exterimented with some things that were completely new to me. The main_v[1-7].scd files in the sc_src folder correspond roughly to the recordings 1-7 in the recordings folder.

I made seven versions of the sonification experimenting with pitch material, timbres, reverb, delay, and a few different mapping strategies. At the heart of all of these mapping strategies was the use of the depth parameter to signify distance from the listener. For the 4th prototype I added wavetable synthesis where the movement in depth levels was translated to an audio rate signal used in an oscillator so that the data would be present both on the micro (or maybe nano) and macro level. Because this sound was very high frequency heavy I had to shape it quite a lot for it to fit the sound image I envisioned. Letting these wavetable synthesised sounds depend heavily on the relevance score and the "source" of the function call gave the sonification a surprising amount of structure and interesting variation.

Introducing the harsher sound of the wavetable synthesis also required me to radically reduce the tempo of the sonification, with the side effect of bringing out significantly more, but different, structures. The events now happen in a more human time-scale which make them slightly less alien and more relatable musically, but potentially also reduces the invocation of the computational sublime in the listener.

Looking back at version 1 I think the approach I started with still has its merits, but it all very much depends on context. In certain context the simpler nature of the sounds and the shorter format are huge advantages because it is possible to take the whole piece in as one stretched out moment. The pitch treatment in the first version is also more true to the depth level input, treating frequencies as overtones of the first node (like a linear grid), and allows us an overview of the macro structure of the piece.

The last version, meanwhile, brings out a lot of musical qualities and patterns that were impossible to pick out when listening to the first version. It adds symbolism through the use of a very extended tonal macro structure and better reveals the complexity of the data. It would work much better in a context like a concert situation or similar where a deeper focus can be expected of the audience.

## Conclusion

Choosing to focus on a simple parameter of the dataset I was able to extract an amount of structure and variation that surprised and delighted me. I have provided all versions of the sonifications in the recordings folder (in 500 kbps Vorbis), the last one being my preferred and the most polished version. For more detailed info about the process I kept a log in notes.md.