FAUST Multiplatform toolbox for Body Brain Digital Musical Instruments

David Fierro
CICM-MUSIDANSE
Paris 8 University
david.fierro@mshparisnord.fr

Alain Bonardi
CICM-MUSIDANSE
Paris 8 University
alain.bonardi@univ-paris8.fr

Atau Tanaka
Embodied Audiovisual Interaction Group
Goldsmiths University
a.tanaka@gold.ac.u

ABSTRACT

This article presents new tools developed in the FAUST language to create musical interactions using electrophysiological signals as input. The tools developed are centered around signal processing and simulation of electrophysiological signals. These techniques are used to clean and process biosignals and subsequently provide real-time interactions to feed the control of sonic processes. The system provides modules that are musically expressive especially in the domain of spatial sound.

These tools also allow to set up a testing environment by replacing the need of electrophysiological capture devices.

The findings of this research provide a better understanding of how the FAUST language can be used in conjunction with physiological signals and brings to light interesting opportunities to explore further possibilities in music creation in an open source environment with the possibility of multitarget compilation, allowing our modules to be used either in such softwares as Max[10] or embedded in microcontrollers.

Author Keywords

FAUST, signal processing, sound synthesis, simulation, electroencephalogram (EEG), electromyogram (EMG), digital musical instruments (DMIs)

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)** → **Interactive systems and tools** → User interface toolkits; • **Applied computing** → **Arts and humanities** → Sound and music computing;

1. INTRODUCTION

In recent years, advances in technology have enabled researchers to develop new tools for electrophysiological signal processing and simulation. Although these tools are available and many are proposed as Open Source, there is a limitation while using electrophysiological signals on Digital Musical instruments that use brain electroencephalogram (EEG) and muscle electromyogram (EMG). Together these signals of the body can be called "ExG". Problems like the need of external devices and the format of electrophysiological recordings not being suitable for DAW[1]'s may prevent musicians from exploring the creation of music using ExG signals.

This paper discusses the development of new tools for electrophysiological signal processing, signal simulation and sound synthesis developed in the FAUST[5] programming language. It presents the advantages of using FAUST for this purpose, and the challenges that must be addressed in order to make the tools more effective.

Every module discussed in this article can be downloaded from our Gitlab repository: https://gitlab.huma-num.fr/bbdmi/bbdmi

1.1. Context

The developed tools were created as part of a research project called BBDMI[2]. This research project explores the potential of using electrophysiological signals to create digital musical instruments. It explores the potential implications of using electrophysiological signals for musical expression, and the challenges associated with designing and creating such devices. The overall goals of the project are already described in [7].

To achieve the goal of building digital instruments specifically for EMG and EEG sonification, the proposed tools were developed thinking in different parts of the electroacoustic chain. They can be divided into 3 main groups: signal preprocessing, signal simulation and sound treatments. The first section is focused on the preprocessing of electrophysiological signals to prepare them for further use as control signals. Signal simulation modules generate signals that can be used to replace real EMG and EEG capture devices. Sound treatment modules contain a set of objects made for sound synthesis and audio effects.

1.1.1. Sound spatialisation

Sound synthesis processes were developed with the intention of being compatible with the developments on the ambisonic field of the laboratory[1]. Research and teaching of ambisonic spatialisation for musical composition and sound design has been at the center of our work for years with such developments as the HOA[3] library and now the ABCLIB[4] library that takes over in FAUST. Thanks to this language, this work can be used on several devices.

NIME' 23, May 31–June, 2023, Mexico City, Mexico.

[1] Digital Audio Workstation
[2] https://bbdmi.nakala.fr/
[3] https://faustlibraries.grame.fr/libs/hoa/
[4] https://github.com/alainbonardi/abclib

1.1.2. ExG music hardware

We have developed music hardware to capture EMG and EEG signals[2]. The board is a micro-controller based signal processing running the OWL signal processing framework [8]. This framework cross-compiles common computer music languages such as Pure Data, Max gen~, and FAUST to low level micro-controller executables. The board is designed as a class compliant audio and MIDI device, allowing it to be used as a music interface with a host system without the need for special drivers. This board combines biosignals and audio signal in a single signal processing chain and is capable of streaming the recorded information by MIDI or raw audio making it an efficient interface for body signals for digital musical instrument applications. A new ExG update to the board will be demoed elsewhere in this conference.

1.2. The choice of FAUST

The software is written in FAUST (Functional Audio Stream) an open source functional language for signal processing with multiple compilation targets including Max, Pure Data[5], or VST on various platforms: computers, mobile devices, web browsers, and microcontroller based devices.

In addition to creating sound synthesis engines, FAUST allows users to create signal processing algorithms for treating control signals. Our library contains objects to clean and extract features from electrophysiological signals.

2. TOOLBOX ARCHITECTURE

The toolbox was developed using a modular logic separating primitive functions from more complex ones. Having a modular system allows us to design a toolbox that can be adapted to many user cases. Code for the modules are found in the "bbdmi.lib" library and every object makes a call to it.

In order to create the modules as compile ready objects, we created basic methods accessible from our library. Complex modules use these primitive methods. Final modules have a Graphical User Interface that allows users to control internal parameters.

As functions were separated into basic and simple types, some methods do not have a GUI but can be easily compiled to any platform by following the same logic we use for the complex objects.

Methods on our library offer the possibility to be compiled using multiple channels. For some objects, every channel has the same sound process while others make use of ambisonic encoding and decoding of signals to create musical effects by modifying independently every spheric harmonic generated by the ambisonic encoding process.

3. FAUST IN MAX

Here, we present the Max wrapper objects created to interact with our compiled FAUST primitives.

Objects in the Max library expose parameters for every channel allowing different output signals on every voice.

Figure 1 shows the "bpatcher"[6] created for our proposed EMG simulator granting the user access to parameters on every channel.

Max objects developed in our project can also be downloaded from our main repository.

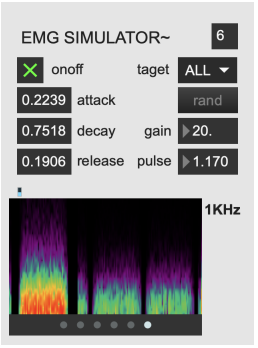

Figure 1. EMG simulator object on Max

4. FAUST FOR ELECTROPHYSIOLOGICAL SIGNAL PROCESSING

Signal preprocessing is an essential step in the analysis of any electrophysiological signal. Preprocessing includes filtering, segmentation and feature extraction, which reduce noise, detect relevant features and improve signal quality. Preprocessing also helps to identify useful data points for further analysis.

This preprocessing step is essential for obtaining accurate and meaningful signals from the capture device especially if the input source is contaminated with instrumental noises.

4.1. Noise filtering, envelope following

Preprocessing and noise filtering is also important in order to use biosignals as control. These processes can help to reduce noise and artifacts, improve signal-to-noise ratio, and reduce computational complexity. Additionally, preprocessing and noise filtering can help to make the signal more interpretable and easier to analyze, which can make it more suitable for use as a control.

Our toolbox includes a set of preprocessing objects that will do 50/60Hz notch filtering, DC-blocking, envelope following and RMS calculations. By using a single module capable of filtering and giving a usable signal for control parameters, this module offers an easy and fast way to start working with noisy electrophysiological signals.

4.2. Simulation of electrical noises

Every simulator developed in our project tries to emulate the real signal captured with EEG or EMG devices without instrumental noises. In order to be able to use the generated signal as a real one, we developed an electrical noise simulator. This 50 or 60 Hz generator adds some harmonics to the signal, simulating specific noises induced in the electric lines by devices connected to the network.

[5] https://puredata.info/

[6] Encapsulating technique allowing GUI to be exposed as an interface within a Max patch.

Figure 2 shows the Faust code generating the fundamental frequency with 5 harmonics added to the noise to simulate electronic devices connected to the network:

```
bbdmi_electric_noise(f) = 0.95 * os.osc(f), par(i,5,(1/(25*(i+1))) * os.osc((2*(i+1)+1)*f)):+,+,+:+,_:+;
```

Figure 2. Faust code of the electric noise simulator

5. SIMULATION OF EEG AND EMG ELECTROPHYSIOLOGICAL SIGNALS

Working with electrophysiological signals implies having an external device capable of capturing these signals from the human body or having a recording that could be used on a DAW. There are many situations where musicians or researchers need to test their tools and neither have an ExG device nor a recorded signal that can be used in their preferred DAW. Recordings of electrophysiological are generally made using tools made for laboratory testing.

The LSL[7] protocol is mostly used in laboratory environments because of its precision in time as it works on timestamps. It also offers the possibility of having multiple streams that can also be recorded into an XDF[8] file. Although LSL and XDF files are a standard in the world of laboratory tests, musicians have a hard time trying to sonify these signals or even trying to extract features from them. In order to make the process of working with electrophysiological signals easier, we propose a set of tools to simulate EMG and EEG signals. These signals can be used directly on the desired DAW and recorded as audio files allowing musicians and researchers to work on new platforms.

5.1. EMG simulation

Our proposed EMG simulator allows musicians and developers to quickly and accurately test their software tools without the need for expensive and time-consuming experiments. This could lead to a better understanding of how EMG signals can be used to create music, and could also lead to the development of new and improved musical software tools.

Before defining the spectrum and dynamics of the generated signal, a decision had to be made between which type of EMG signal would be simulated as there are mainly two types of techniques[9], surface electro-myography (sEMG) and intramuscular electromyography (IMG). Surface EMG uses electrodes placed on the surface of the skin to measure the electrical activity at that particular site. IMG, on the other hand, uses fine-needle electrodes that are directly inserted into the muscle to measure the electrical activity more accurately. Furthermore, IMG is often used in research and clinical settings, while sEMG is most widely used in musical performances, sports science and rehabilitation.

The simulated signal will simulate an sEMG like signal taking into account the spatial filtering of the skin, the size of the electrodes and instrumental noises and filtering. The simulator offers the possibility to modify the envelope of the signal while the spectrum stays the same to guarantee an EMG like signal. The included GUI allows the user to modify the intensity of the signal over time, as well as the attack, decay and release times of the envelope, enabling the simulation of different types of muscular efforts.

Figure 3 shows the spectrum and the time representation of the simulated EMG signals.

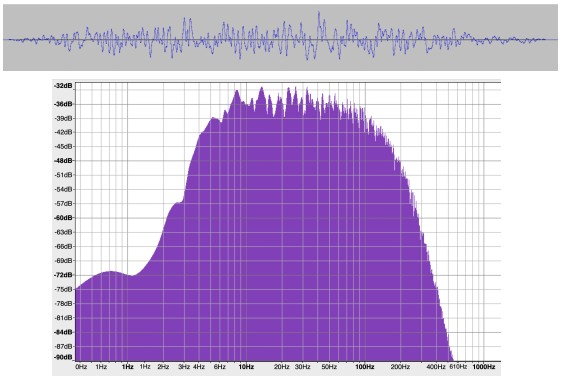

Figure 3. Simulated EMG signal

5.2. EEG simulation

As part of our toolbox, we propose an EEG simulator that generates a signal similar to the real captured signal. Many different recordings were used as examples including our own recordings made with a “Gtec Unicorn”[9] and a “Mentalab”[10] device using gel electrodes. Brainwaves are very complex and although we don’t claim to simulate any cognitive state, we consider it useful to have a tool that allows us to control the alpha (7.5Hz - 12.5Hz) and the beta (16.5 - 20 Hz) band powers.

The spectrum of the EEG signal depends on many factors. For the same person the spectrum of the captured signal will vary depending on where the electrodes are placed, cognitive state of the person and many other different factors. Depending on the objective of the EEG analysis the most important frequency bands will change, the proposed EEG simulator focuses on two specific brainwave bands, alpha and beta.

This simulator is a simple model that does not offer the possibility to work with complex BCI (Brain Computer Interfaces)[6] paradigms like motor lateralization. However, it is still a useful tool for understanding the effects of alpha waves on the tools that musicians and researchers are using. The simulator is able to accurately generate alpha waves in the frequency range of 8-12 Hz with a controllable envelope.

Figures number 4 and 5 show the comparison between the simulated signal and the one recorded with our own devices. The generated and simulated EEG signals have many properties that resemble the real EEG signals. The frequency range of the simulated EEG signals typically falls within the range of 1-30 Hz, and although it may not capture all of the nuances of a real EEG signal, it generally embodies the same key characteristics of the original signal.

[7] LabStreamLayer explanation link
[8] XDF files explanation

[9] https://www.unicorn-bi.com/fr/brain-interface-technology/
[10] We thank Mentalab (Mentalab GmbH, Munich) for providing us with their Explore EEG system for testing.

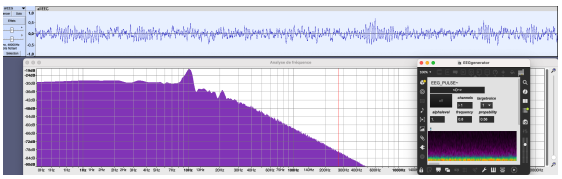

Figure 4. EEG Simulator

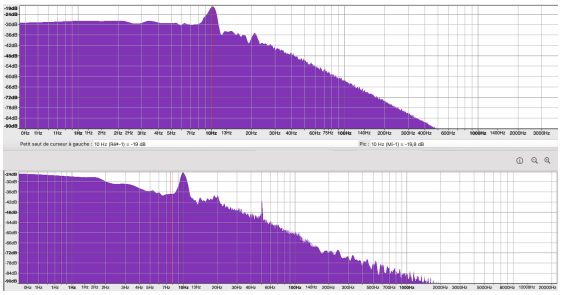

Figure 5. Simulated EEG vs real EEG

The amplitude of the simulated EEG signal is generally similar to the recorded data, and it contains features such as alpha waves and beta waves which are representative of real EEG signals. Additionally, the simulated alpha band power can be fine-tuned to match the behavior of the real EEG signal in terms of magnitude and envelope control.

5.3. Limited band pulse generator

Although electrophysiological signals are limited in spectrum and have specific dynamics, sometimes it comes to need a broader generator with controllable spectrum and dynamics. The "band_pulse" generator creates a limited band of noise with a controllable envelope.

The envelope control (ADSR) determines the length, attack, decay, sustain, and release of each "burst", while the limited bandwidth filter determines the frequency range that the sound will focus on. The frequency band of the produced signal can be modified by adjusting the depth and width of the filter. This allows users to create specific band limited signals. Additionally, the ADSR envelope can be manipulated to create more complex signals with interesting dynamic movements.

This pulse generator is available on our BBDMI FAUST library and it has been tested on the Max platform by adding some visual components to the controls. Figure 6 shows the compiled Max object.

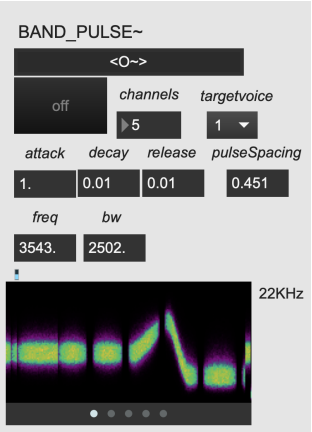

Figure 6. Compiled Faust object for Max

6. SPECTRAL AND SPATIAL SYNTHESIS AND FX

The ability to use electrophysiological signals to control music can open up a new world of possibilities for musicians and performers. Despite the fact that these signals have a highly expressive potential, we found ourselves often with very few input channels to work with. In general, connecting each EMG and EEG capturing devices demands not only time and preparation but also having all the required equipment. This limitation orients most electrophysiological music performances[3] to be done with few input control signals. Therefore it is imperative to create a sound engine that is very expressive and can be highly configurable even with few input control signals.

In order to give musicians the possibility to be highly expressive while creating sounds, we develop our modules leaving every minimal detail of the sound synthesis engine open for input control signals. So even when many variables stay constant, it is still possible to achieve a wide range of musical transformations just by modifying a few variables. Selecting, or "curating" parameters to be exposed to the musician in performance is a crucial part of the instrument design process.

6.1. Setting sound in space

In our research team, we have been working for more than ten years on the use of ambisonic spatialization techniques to enable the setting of sound in space, that is a deep intrication between sound processing or synthesis and its diffusion using the various spheric harmonics.

With the methods included in our own BBDMI library and our partners, we are able to provide a set of tools for ambisonic musical sound processes.

In addition to improving the realism and sense of immersion in virtual environments that can be achieved by ambisonic transformations of sound, these techniques can also be used to enhance soundscapes for creative and artistic applications and more specifically in our case, musical transformations.

The development of our "Live Granulator" is an example of the use of ambisonic treatments to achieve musical results. By setting a global *factor* parameter driving the individual parameters of the encoded ambisonic signal on each spheric harmonic, we create specific sound changes that will be interpreted by the decoder. A common transformation used in our laboratory is sound *decorrelation*[4], it is a technique used to create a diffuse sound field in sound spatialisation, particularly in an ambisonic context. The variations of the *factor of decorrelation* provide high musical expressivity in an efficient way for musical composition based on spatial processing of sound.

6.2. Live granulator

The live granulator works on live input signals and transforms them by applying a highly configurable envelope to each grain. The construction of the granulator comes as a proposition of a single object capable of doing many different sound processes to the sound. Having a single object capable of creating different sound textures allows us to use the processed electrophysiological signals in a very expressive way.

The live granulator has an internal buffer that allows the use of delayed signals on the input phase as well as at the final output section. Information on the audio buffers can be reinjected to the original input by changing the variable *feedback*. The variable *variability* will apply a randomness level to some of the internal parameters of the granulator making it more stochastic.

The specificity of the proposed granulator relies on the highly configurable envelopes, the distribution of internal parameters by each instantiated channel and the way the feedback loops are placed.

6.2.1. Ambisonic distributions

The live granulator was designed as a multichannel object in order to be able to work with the ambisonic spatialisation models developed in our laboratory. This approach allows us to modify some of the internal parameters of each channel following the logic of ambisonic effects. Each channel of the live granulator will modify one independent spheric harmonic generated by the ambisonic encoding phase. By changing variables like the delays applied to the sound or the frequency of the filters and modulating signals, we can achieve interesting sound effects like spatial decorrelation.

There are infinite possibilities of how the internal parameters of the granulator can be distributed from high level control parameters to each spheric harmonic. This distribution will modify the variables depending on the number of the harmonic. The relationship could be linear, by augmenting the value of every parameter by the number of the harmonic but we could also think of more complex distributions.

The variable *indexdistr* allows the user to choose from many different distributions using the transfer functions proposed by Alain Bonardi and Paul Goutman in the HOA library. Figure 7 shows the possible distributions as mathematical functions.

x		
x^2	composite1	x^5
sin	x^3	1-(1-x)^5
log(1+x)	1-(1-x)^3	composite4
sqrt(x)	composite2	2^(10(x-1))
1-cos(Pi/2*x)	x^4	composite5
(1-cos(Pi*x))/2	1-(1-x)^4	1-sqrt(1-x^2)
1-(1-x)^2	composite3	sqrt(1-(x-1)^2)

Figure 7. Functions for the ambisonic distribution

6.2.2. Feedback loops

There are two main feedback loops implemented on the granulator. They are both controlled by the *feedback*, *maxdelay* and *variability* variables. The *feedback* variable will control the amount of feedback applied while *maxdelay* and *variability* will control the amount of time delay applied to the input sound and to each grain.

The first feedback loop mixes the input signal with itself before being granulated. This allows the control of which sounds are being used to create the grains.

The second feedback loop is placed at the end of the granulation chain and serves as a mixer of grains. Every created grain will enter this feedback loop creating a sound texture composed with many different grains of size and texture.

Before entering the final loop, the *transpgrain* variable permits the pitch shift of each grain differently, that way the grains collected on the final feedback loop will be also different in tonality.

Finally, the last process applied to the sound is another pitch shift controlled by the variable *transpout*. This part of the module will allow the transposition of all the grains being produced.

6.2.3. Filtering

Variables *hpffreq* and *lpffreq* allow the user to filter the input signal before it enters the first feedback loop. This filtering phase will modify the spectrum of each grain and can be modified in real time.

6.2.4. Grain size and spacing

The size of each grain is calculated by adding the *grainsize* with a percentage of the *grainoffset* variable. The offset for each grain is a value between zero and the *grainoffset* multiplied by the *variability*. The distance between grains is configured with the *spacing* variable.

6.2.5. Grain envelope morphing

The granulator offers the possibility of changing the form of the envelope main signal. This can be achieved by tuning the variable *grainenvmorph*. This variable will morph the main envelope between a signal with an exponential behavior and one with a gaussian distribution. Although the change in the sound is very subtle it provides a sensitive variable to play with while creating each grain. The resulting envelope will be a morphed signal between the two main envelopes shown on Figure 8.

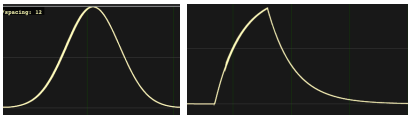

Figure 8. Envelope design

6.2.6. Envelope amplitude modulation

After each grain is created the module allows the user to apply an amplitude modulation effect to each grain using the variable *modfactor*. Figure 9 shows the envelope of a grain without modulation and another with 100% modulation.

By modulating the amplitude of the grains, it is possible to create subtle changes in the texture of a sound. This can be used to create a range of interesting effects, such as a pulsing or warbling sound.

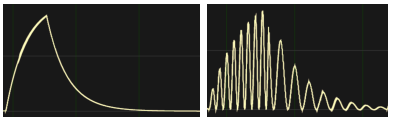

Figure 9: Envelope amplitude modulation

Note that the waveform used for the frequency modulation of each grain starts every time a grain is created. When the same signal is used to modulate every grain, it is possible to hear the frequency as a stationary tone over the generated sound, not a very interesting sound effect as it can be similar to a simple additive

synthesis process. To avoid stationary waves to set in, the phase of the envelope signals starts at zero every time a grain is created. The result of restarting the phase of the waveform every time a grain is created, is that it is almost impossible to have an exact spacing time between grains that will create a correlation between the modulating signals. Most of the time the result will be a superposition of grains with decorrelated modulating envelopes. By decorrelating the modulating signals from each grain we avoid the presence of a single tone and we obtain a more complex effect giving the sound many possible textures.

6.2.7. Modulated frequency of the amplitude modulation

In addition to modulating the amplitude of the envelope of each grain, the object also provides a variable called *modfreqmod* that will control the level of modulation of the amplitude modulation of the envelope. Figure 10 shows in yellow the control signal for the frequency modulation, the red signal is the resulting waveform of the envelope. This transformation generates an envelope with an amplitude modulation that can start at a high frequency and finish at a low frequency and vice versa. All this happens every time a grain is created.

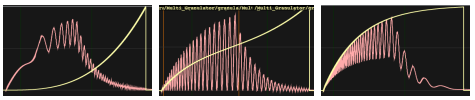

Figure 10. Modulated frequency of the amplitude modulation

Sound implications of modulating the frequency of an amplitude modulation can be quite complex. When the frequency of an amplitude modulation is changed, it can affect the timbre, or tone, of the sound. This is because the frequency of the modulation affects the harmonic content of the sound. For example, if the frequency of the modulation is increased, the sound will become "brighter" and more complex harmonically. Conversely, if the frequency of the modulation is decreased, the sound will become "darker".

In addition to affecting the timbre of the sound, modulating the frequency of an amplitude modulation can also affect the dynamics of the sound. When the frequency of the modulation is increased, the sound will become more dynamic and have more of a "punch" to it. Conversely, when the frequency of the modulation is decreased, the sound will become more mellow and less dynamic.

6.2.8. Morphing of the amplitude modulation

The ability to morph the waveform of the modulating signal in an amplitude modulation (AM) system can have a significant impact on the sound of the resulting signal. For example, a square wave modulating signal can create a bright, aggressive sound, while a sine wave modulating signal can create a smoother, more mellow sound. Additionally, the use of a modulating signal with a more complex waveform, such as a sawtooth or triangle wave, can create a unique sound that is not possible with simpler waveforms.

The variable *modmorph* of the granulator works as an interpolator controller between 4 signals: sinusoid, sawtooth, square and sinusoid again. Figure 11 shows the envelopes generated by different values of the *modmorph* variable. It is important to notice that integer values provide pure waves while decimal values will create a morphed shape between the two pure waveforms.

Having a sinusoid signal at the beginning and the end allows the user to create any possible morphing between the 3 basic waveforms.

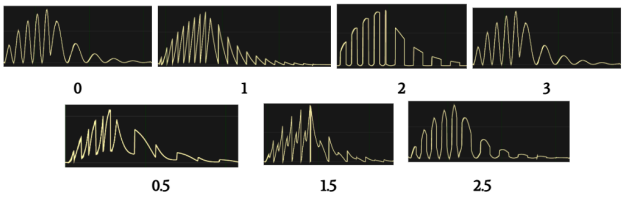

Figure 11. Morphing of the amplitude modulation

Morphing the waveform of the modulating signal can also be used to create interesting effects. By morphing between shapes at fast or slow speeds it is possible to achieve gradual transitions between sound textures. This can be used to create a range of interesting and unique sounds, from subtle changes to more dramatic shifts.

6.3. Lagrange and Bezier curves for sound synthesis

Lagrange and Bezier curves are a powerful and versatile tool for sound synthesis. They are used to generate smooth and continuous curves, which can be used to create synthesized sounds, from simple pitch variations to complex sound timbres. Additionally, the highly detailed curves allow for high-fidelity sound synthesis.

As part of our toolbox, we are developing a synthesis module using Lagrange and Bezier curves. We found that in order to achieve a complex and dynamic sound it is necessary to manipulate the waveform on a very short timescale. If the waveform takes too much time to change, the result will be a complex waveform with a sound similar to effects like ring modulation or FM synthesis.

6.3.1. Limitations of the Lagrange curve synthesis

Working with this type of synthesis is not without drawbacks. One of the main issues is that depending on the position of the points used to define the curve, the signal may explode, resulting in distorted or unusable sound. This is due to the fact that the curve is defined by a series of polynomials, and when the points are too close together, the polynomials can become unstable and cause the signal to explode.

To prevent this from happening, it is important to choose points that are well-spaced, so that the polynomials remain stable. Finally, it is important to ensure that the points are chosen in such a way that the resulting curve is smooth and does not contain any sharp corners. Something hard to achieve when working with Bezier curves for sound synthesis. Figure 12 shows an example of a simple Lagrange curve created with our synthesizer.

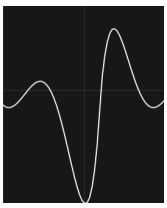

Figure 12. Lagrange curve synthesis

6.3.2. Limitations of the Bezier curve synthesis

Working with Bezier curves for sound synthesis can be problematic due to the fact that the first derivative of the resulting signal is not continuous when two waveforms are put together. This can lead to the generation of undesirable harmonics, which can lead to a distorted or otherwise unpleasant sound. To avoid this, it is important to ensure that the Bezier curve has a smooth transition while joining one curve to the next. Additionally, it is important to ensure that the control points used to define the curve are well-spaced, as this can help to reduce the amount of distortion caused by the discontinuous first derivative. Figure 13 shows an example of a simple Bezier curve created with our synthesizer.

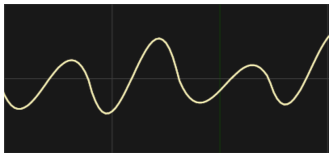

Figure 13. Bezier curve synthesis

6.3.3. Max object

Although the FAUST code of the live granulator already includes a GUI allowing the control of every internal parameter of the granulator, it is not exposed when the object is used in softwares like Max. As part of our project we propose a Max "bpatcher" that enhances the experience of musical composition by permitting the generation of random configurations and then the dumping of that configuration for further use.
Figure 14 shows the live granulator "bpatcher" for Max.

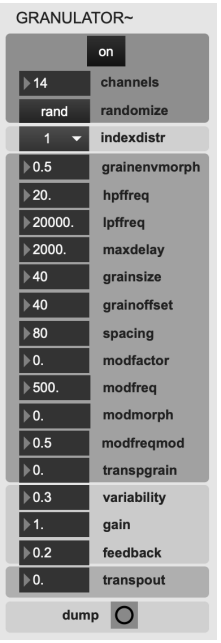

Figure 14. The Livegranulator object compiled for Max

7. MICROCONTROLLER EMBEDDED FAUST

We used the FAUST language on the EAVI board (reported elsewhere in this conference) to perform signal processing on electrophysiological signals. The FAUST code embedded on the microcontroller was used to do noise filtering, DC-blocking, envelope following and RMS (root-mean-squared). Doing this process directly on the board reduces the amount of work that needs to be done on the sound engine modules as the output signal can already be used as a control signal.

Noise filtering is done to remove some types of instrumental noise from the signal like 50/60Hz noise and its harmonics, while DC-blocking ensures that any DC drifts in the signal are decreased. Envelope following is performed to create an envelope around the signal so that the shapes of the signals can be determined. Finally, RMS is used to measure the power of the signals giving us a control signal that can be directly used on a sound engine.

Figure 15. EAVI board running FAUST codes

Figure 15 shows the EAVI board being used as a HCI (Human Computer Interface) to control sound processes modifying the sound of a guitar.

8. EVALUATION

Part of our research project includes the analysis of user study cases on body and brain musical digital musical interfaces. In that context, we've had the opportunity to test our tools on live performances[11] and workshops[12] giving us valuable feedback from users and the public.

8.1. Signal processing

Having clean electrophysiological signals as an input proved to be very useful especially when fine tuning the system to use it as a musical instrument. Undesired noises may break the experience of the musician and reduce the musical expression capabilities of the system.

The possibility of getting the envelope and RMS of the input signal directly from the capturing device facilitated the interaction between the user and the sound engine as no more processing was needed to use it as a control signal.

8.2. Signal simulation

The proposed simulators were used in different circumstances mostly in testing environments. These tools allowed us to test our system without the need of any external devices which is a big reliever if an installation is needed every time.

It is important to notice that specifically for our EEG simulator more work needs to be done in order to achieve a signal that can be similarly complex as the real EEG signals.

8.3. Musical expression

By using sensors attached to the body, our system is able to capture the subtle nuances of a person's movement and use that data to create music. Our users manifested that using the body as an interface enabled them to express themselves musically in a more natural and intuitive way.

8.4. Musical possibilities

Objects discussed in this paper were useful in the process of creating new interfaces for musical expression using physiological signals. Having the opportunity to configure every single part of our synthesis modules allowed us to transform the generated sound in many different ways. The resulting sounds can be very different from each other, allowing immense musical possibilities.

9. CONCLUSION AND FUTURE WORK

As part of the ongoing BBDMI project, the BBDMI Faust library proved to be useful in the creation of new musical interfaces for electrophysiological signals.

Sound engines created with our system offered musicians an expressive and highly configurable system.

It is part of our plan to port our modules into the EAVI board itself, taking advantage of its potential and the FAUST cross-platform language.

Our sets of tools are still in development and further progress will be uploaded to our main repository.

[11]https://www.mshparisnord.fr/event/concert-du-projet-bbdmi/

[12]https://www.mshparisnord.fr/event/journees-europeennes-du-patrimoine-a-la-msh-paris-nord-2022/

10. ETHICAL STANDARDS

The research presented in this paper has been supported by the French national research agency (ANR-21-CE38-0018). As part of the signing of the grant agreement, the project has been approved to conform with research ethics regulations of the Centre National de la Recherche Scientifique (CNRS). The user facing research in the project has obtained informed consent of participants.

We recognise expertise brought to the project by synthesizer designer Martin Klang of Rebel Technologies for the EAVI EMG hardware design (Horizon 2020 ERC 789825), and external advisor Robert Oostenveld.

11. REFERENCES

[1] Composer l'espace sonore: *https://revues.mshparisnord.fr/rfim/index.php?id=624*. Accessed: 2023-01-30.

[2] Di Donato, B. et al. 2019. EAVI EMG board. (Porto Alegre, Brazil, Jun. 2019).

[3] Donnarumma, M. 2012. Biotechnological Performance Practice. *Canadian Electroacoustic Community eContact!*. 14, 2 (2012).

[4] Goutmann, P. and Bonardi, A. 2022. Approaching Spatial Audio Processing by Means of Decorrelation and Ring Modulation in Ambisonics-Sound 2. (2022).

[5] Orlarey, Y. et al. 2009. FAUST : an Efficient Functional Approach to DSP Programming. *New Computational Paradigms For Computer Music*. Editions DELATOUR. 65–96.

[6] Rao, R.P.N. 2013. *Brain-computer interfacing: an introduction*. Cambridge University Press.

[7] Tanaka, A. et al. in review. Brain-Body Digital Musical Instrument Work-in-Progress. *ISEA2023* (in review).

[8] Webster, T. et al. 2014. The OWL Programmable Stage Effects Pedal: Revising the Concept of the Onstage Computer for Live Music Performance. *NIME 2014* (2014), 4.

[9] Zheng, M. et al. Surface Electromyography as a Natural Human-Machine Interface: A Review. 31.

[10] Zicarelli, D. 2002. How I Learned to Love a Program That Does Nothing. *Computer Music Journal*. 26, 4 (2002), 44–51.