Dynamic Music Creation on a Smartphone

Final Year Project Report - February 2013

Johan DELOUCHE & Paul TSNOBILADZE
Mobility & Security

2012-2013

Abstract

Our goal was to develop a mobile application aimed at dynamically creating music, depending on our electromagnetic environment. The original idea came from the principle of the theremin, which is the first (and only) “contactless” electronic music instrument.
We used an audio engine called Pure Data to create the sound. Depending on the different inputs (magnetic sensor, GSM signal strength, Wifi signal strength, etc.), different parameters of the music are modulated (tempo, rhythm, pitch, etc.). By walking around with your smartphone, you can then listen to a unique music composition that illustrates something you cannot usually “feel”: your electromagnetic environment.

Résumé

Notre objectif était de développer une application mobile capable de créer de la musique dynamiquement en fonction de notre environnement électromagnétique. L'idée d’origine est tirée du principe du theremin, qui est le premier (et unique) instrument de musique électronique "sans contact".
Nous avons utilisé un moteur audio appelé Pure Data pour créer le son. En fonction des différentes entrées (capteur magnétique, l'intensité du signal GSM, la puissance du signal Wifi, etc.), les différents paramètres de la musique sont modulés (le tempo, le rythme, la hauteur du son, etc.). En se promenant avec votre smartphone, vous pouvez alors écouter une composition musicale unique qui illustre quelque chose que vous ne pouvez pas habituellement ressentir : votre environnement électromagnétique.

Table of contents

I. Introduction

Context

To conclude our academic training at the École des Mines de Saint-Étienne, we had to carry out a one-month project. Several opportunities were offered to work in collaboration with people external to the school. One of them was a partnership with the École Supérieure d'Art d'Aix-en-Provence initiated by Mr Laurent Freund (École des Mines) and Mar. François Parra (Art School). We came to visit their school and exchange ideas about projects we had. After several meetings we started to work on a project with the sound teacher of the École Supérieure d'Art d'Aix-en-Provence, Mr François Parra. Our project is based on two very different programming languages that involve different skills. One is Pure Data, a sound-oriented dataflow programming language; the second one is Android, the well-known smartphone platform. Mr François Parra agreed on teaching us Pure Data, in exchange for which we would help him with Android development. After a couple of meetings in the Art School, we made a one week workshop in the Vasarely foundation and there, the project really started to take shape. We learned the bases of Pure Data programming and became more independent. We later carried on with developing the project at school by ourselves.

The idea

During one of our visits in the École Supérieure d'Art d'Aix-en-Provence, we heard about an earlier project that used the camera of a smartphone to produce sound. We liked the idea and started our search with a main goal: generate music by using the sensors of an Android smartphone. Letting the user create his own music by walking around in the streets or in a building. The theremin idea came through watching a video of its inventor. At first we picked up the electromagnetic field strength and set the frequency of a sine wave with it. The result was interesting but not pleasant to hear. We needed to improve the sound quality and thought about more sophisticated ways to generate nice sound effects. We decided to use more than one phone sensor. In the final version we use the electromagnetic field, accelerometer, GSM and wifi signal to set up different audio parameters. This will be explained more precisely later. At the end, we decided to give the user less freedom to control the music he creates, in order to have a more pleasant sound experience. A huge part of the sound effects are still totally determined by sensors and so can be managed by the user. That allows having a unique experience for each user, each time the application is used. It is a real-time artistic experiment.

The theremin

The theremin was invented in 1919 by a Russian physicist named Lev Termen (in the United States, his name was Léon Theremin).
Besides looking like no other instrument, the theremin is unique in that it is played without being touched.
The theremin has two antennas - one controlling pitch, and the other controlling volume. As a hand approaches the vertical antenna, the pitch gets higher. Approaching the horizontal antenna makes the volume softer. Because there is no physical contact with the instrument, playing the theremin in a precise melodic way requires practiced skill and keen attention to pitch. It is generally admitted, that

one needs a perfect pitch to play theremin. Theremin was particularly used in sci-fi movie themes and pop songs in the 50s and 60s, for example in “Good Vibrations” by the Beach Boys, or in Star Trek main theme.

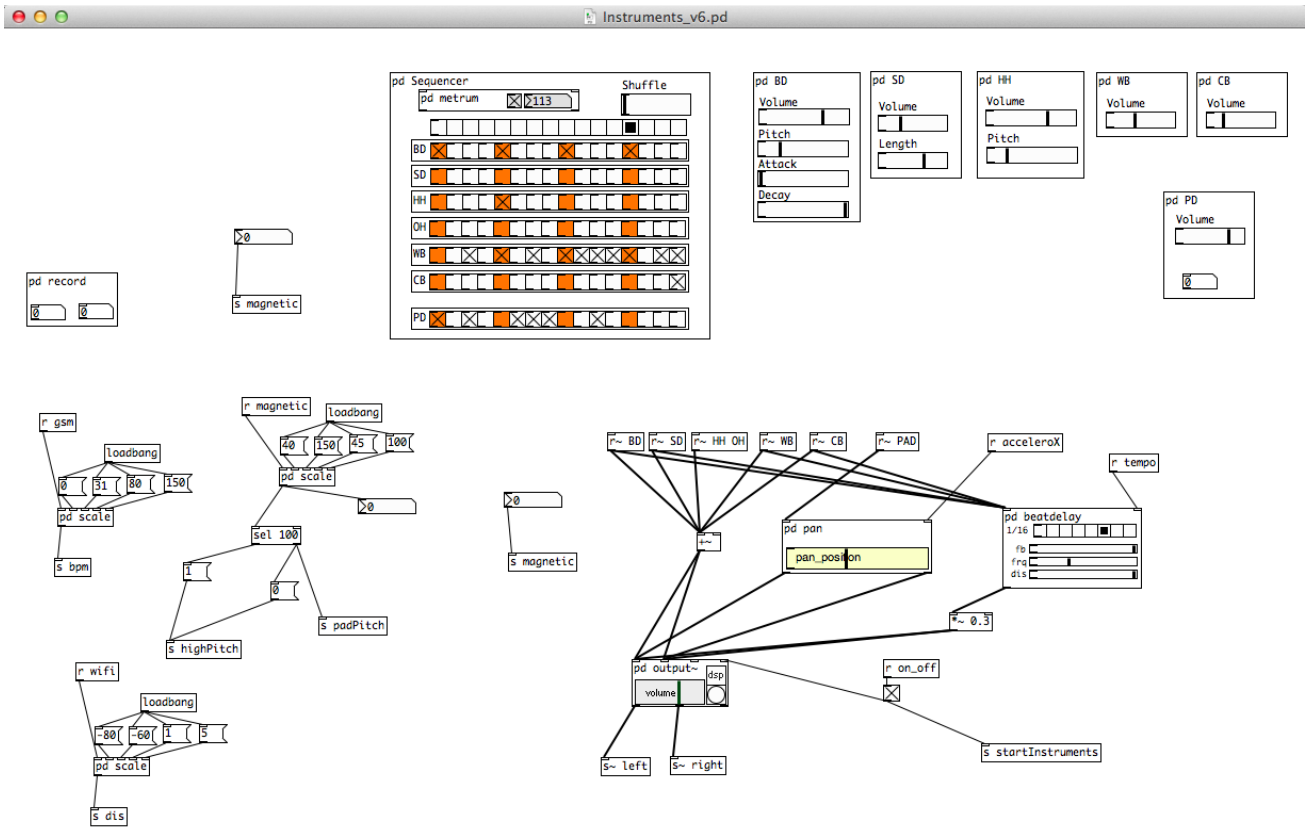
II. Sound Creation

1. Audio Engine

To create our application, we used a very powerful audio engine called Pure Data. It is an open source visual programming language aimed at creating multimedia works and interactive computer music. The principle of this tool is to link boxes that have specific functionalities to create the desired effect. You create a box, enter the name of the function you want to use, and link the top inlets of the box to the parameters of the function. You can then get the result by linking other boxes to the bottom inlets of the box. By combining hundreds of boxes, you can then create very complex functions and sounds. You can save your work in a file called patch, which allows you to use the same function later on.

1. Music Composition

Here is the Pure Data patch we used in our application.



The main part is the beat grid, which controls when the different instruments are played.
The different sensors of the phone interact with the parameters of the sound as following:

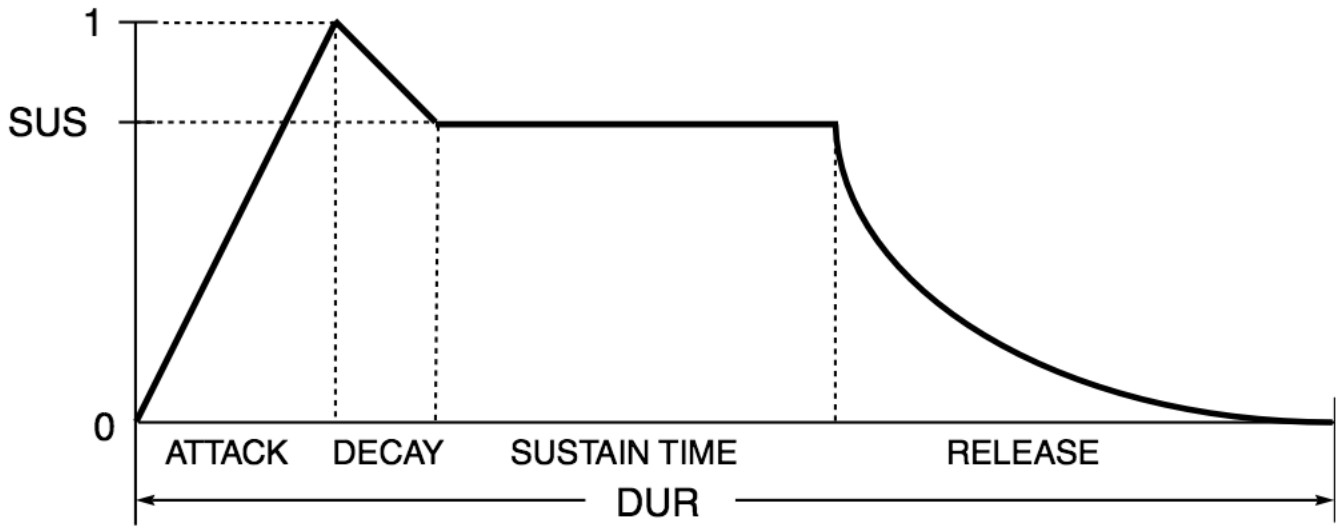
- The GSM signal strength controls the tempo of the music (BPM - Beat Per Minute).

- The magnetic field strength controls the pitch of the lead pad.
- When the magnetic field strength reaches a certain value, the pitch of the lead pad stops getting higher and an alternative rhythm with an opened hi-hat sequence.
- The way you rotate your phone to the left or to the right controls the panoramic of the lead pad sound, i.e. pans the sound to the left or to the right channel.
- The Wifi signal strength controls the amount of the beat delay effects that is added to the drum sounds.

2. Sound Synthesis

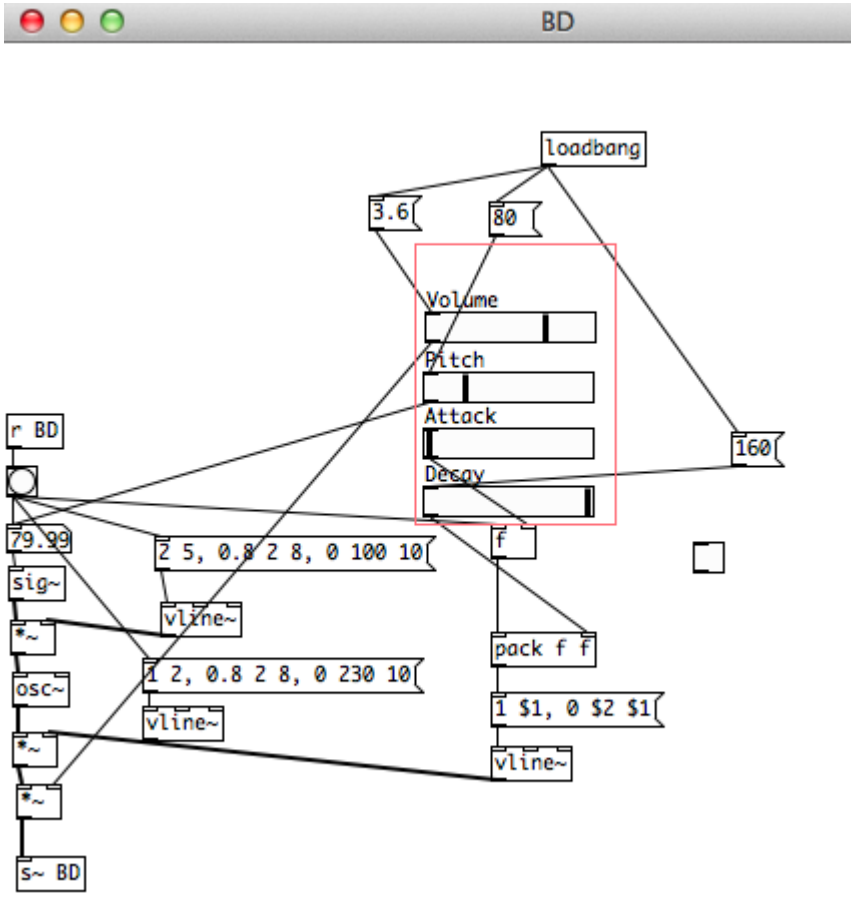
To create the sounds we wanted in our music composition, we used waveform synthesis. Among the most popular waveform synthesis techniques are subtractive synthesis, additive synthesis, wavetable synthesis, frequency modulation synthesis, phase distortion synthesis, physical modelling synthesis and sample-based synthesis.
Synthesizers usually include 4 main components that control the waveform (and consequently the resulting sound):

- The electronic oscillators, which create raw timbre from different basic waveforms: sine, triangle, square, etc.
- ADSR envelopes, which shape the volume or harmonic content of the produced note in the time domain with the principle parameters being attack, decay, sustain and release. Each of these words corresponds to a temporal phase of the sound:



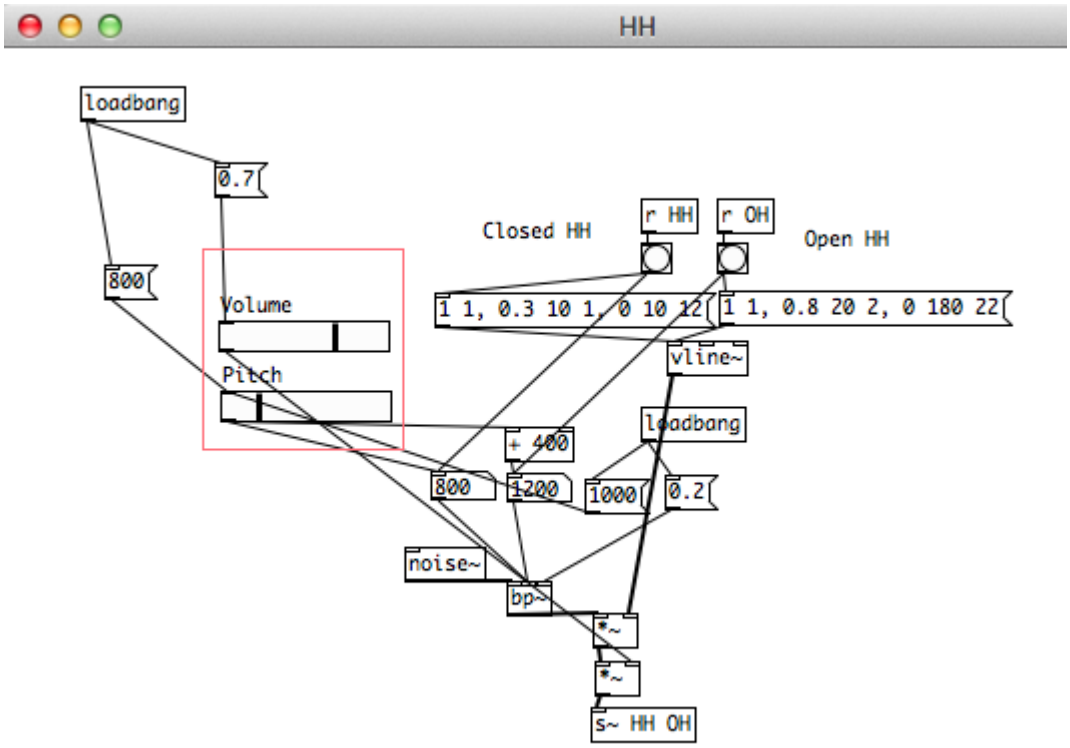
- Voltage Control Filters, which shape the sound, generated by the oscillators in the frequency domain, often under the control of an envelope or LFO. These are essential to subtractive synthesis.
- A Low Frequency Oscillator (LFO), which is an oscillator of adjustable frequency that can be used to modulate the sound rhythmically, for example to create tremolo or vibrato or to control a filter's operating frequency. LFOs are used in most forms of synthesis.

The way we synthesize the drum kit is based on the 1980s Rolland TR-808 rhythm box which is one of the most famous rhythm boxes of all time, since it was used in most house and electro tracks since the beginning of the 90s.
Here is how we synthesized the sound of the bass drum:



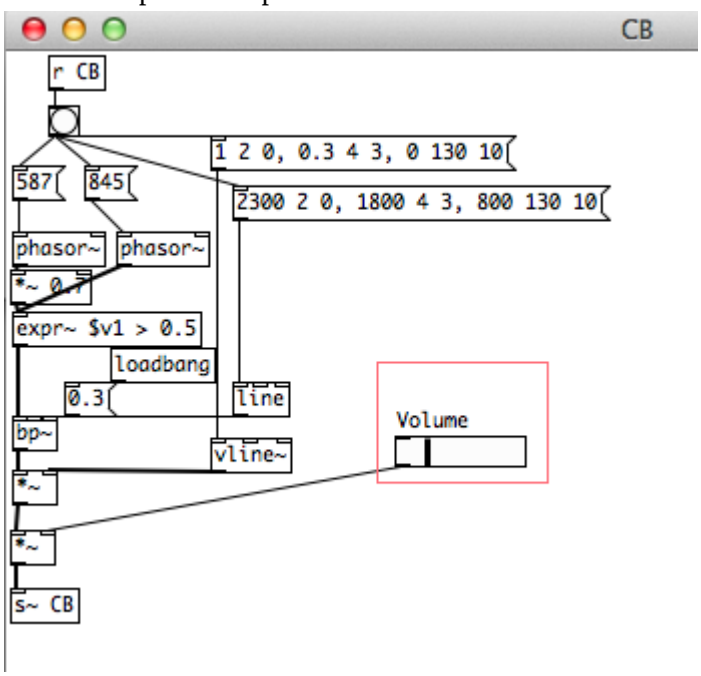
As you can see, we use the Attack Decay parameters that we saw earlier, plus the classic pitch and volume of the resulting sound. We use the vline~ function to create an envelope with the Attack and Decay parameters which are set by the sliders. This envelope modulates the base signal, which is a cosine signal whose frequency is set by the pitch slider. By combining those envelopes and the oscillator, you create a synthetic bass drum, which is typical of synthetic drum kits and especially the Rolland TR-808.

Here is how we synthesized the Hi-Hat:



This time, the base signal is not a cosine signal but a white noise, which simulates better the sound of a cymbal. You can modulate the pitch through a band pass filter (bop~). You select if it is a closed hi-hat or an opened hi-hat by changing the envelope.

Here is the patch that produces the Cow Bell sound:

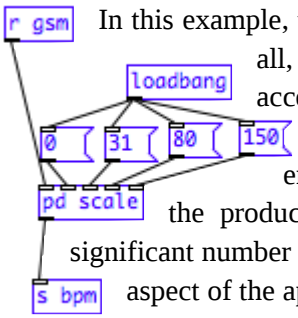


In that patch, you can see that the base signal is a phasor, i.e. a saw tooth signal. The rest of the patch is rather similar to the others.

The other patches we used included the same kind of components, i.e. cosine, saw tooth signal or white noise with several filters, usually band pass filters.

3. Inputs Management: the Scale Issue

One of the main problems in the sound creation phase was to make the sound change significantly depending on the inputs, i.e. the data sent by the sensors. To fix that problem, we used scale boxes that filter the range within which the values usually evolve and stretch it to the scale of values which is desired for the parameter the given input controls.



In this example, the GSM signal strength value evolves between 0 when there is no signal at all, and 31 when the signal is excellent. Since we want the music to remain at an acceptable tempo, we convert these values between 80 and 150 bpm. By changing the scale, you get a different impression of sensitivity to the environment. The problem is if you make the parameters change too quickly, the produced music will not be melodious. It is a precise tuning that requires a significant number of tests and on which we are still working. That is why working on the sound aspect of the application substantially damaged our hearing!

III. Android Development

The whole Android development part of the project has been done using Eclipse. It is a really powerful tool but unfortunately it is slightly unstable and not always easy to use. Eclipse is a software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. This free and open source software is written mostly in java and mainly used to develop applications in this language (Android is based on that language), but it can also be used to develop in other languages like Ada, C++ or PHP.

1. Libpd integration

In order to use Pure Data patches in an Android mobile, we had to use an open source project named Libpd. It is a complex library that brings into Android platform the Vanilla distribution of Pure Data. This is the original version of Pure Data integrating only the basic components. Once included in an Android project, the library provides functionality to use Pd patches embedded on the SD card of the phone and to communicate with them. From that moment on, you can load them and start exchanging messages with the patches. Libpd takes care of the audio synthesis in background and sends the audio signal to the mobile phone, which then plays it. Patches can be self-sufficient or wait for messages and commands. Ours are controlled by various messages and commands sent by the Android application. That is how we managed our patches to set some audio parameters.

1. The sensors

To follow the lead of the theremin instrument, we needed to pick up the electromagnetic field and calculate the strength of the signal. In order to do that, we developed a class that brings together various “SensorManager” objects. Those managers are provided by the Android API (Application Programming Interface). They allow us to handle the hardware of the phone. Because Android phones are very diverse and not all of them have the same hardware sensors, we chose four of the most common sensors, the electromagnetic sensor (used for the compass), the accelerometer sensor, the GSM sensor and the wifi sensor. The first two ones are used similarly and the other ones follow different protocols. The first step, for every one of them, is to ask android the permission to use them. In Android development, each time you want to access a specific component like a sensor, the Internet connection or the contact book, you have to ask for the appropriated permission. Therefore, when the user downloads your application on the market, he is warned and knows exactly what the application can or cannot do.

To use the electromagnetic sensor (it is the same with the accelerometer sensor), you have to ask Android for the unique instance of the SensorManager. We needed it to deal with this kind of sensors. This SensorManager granted us the access to the electromagnetic and accelerometer sensors. So we created a class able to listen to the sensor events and when an event occurred, we were able to filter which sensor it referred to and could get data about the strength of the signal. After all those steps, you can send a message to the Pd patch.

You handle GSM and Wifi sensors differently. In order to manage the GSM signal we needed the TelephonyManager. Then we initialized its listener to filter only the system message we wanted. In this case, we wanted to focus on changes in the signal strength. Once again, at that point, you can send a message to the Pd patch. The last one, the GSM sensor, needed a special broadcast listener that allowed us to listen to all the messages going through the Android system and filter, once again, the data we needed.

In appearance all those sensors can look similar but technically the system handles them differently and that is why you have to manage each one specifically. We will not go into deeper technical explanations.

2. The user interface

The term “User interface” refers to the graphical appearance of the application. To what it looks like. This is an important part of the development for mobile applications. Some studies have shown that the very first seconds are decisive. Most mobile applications are uninstalled few seconds after the first opening. That is why it is very important to provide the user with an enjoyable experience. A nice and original user interface helps to reach that goal.

At the beginning we mainly took care of developing the functionalities of the application. The user interface was developer oriented. Which means the application only displayed the information useful for the developer. After the workshop at the Vasarely Foundation, we agreed to create the first user-oriented version of our application, which was minimalist. We removed all the developer tools we used to debug the application and kept only a single white screen with a number representing the electromagnetic field strength.

60.022663

But at the end of the month, once all the technical functionalities were implemented, we started to think about the final user interface. We decided to keep four items on the screen: a VUmeter to show the electromagnetic field strength, an on/off button, a record button and a counter for the record time. We spent a non-negligible time on Photoshop and illustrator to process the images we used in the GUI. Indeed to put images on an Android application, we had to resize them several times to handle all the different screen sizes that can exist.

The VU meter was the most difficult one. It is a superposition of three different images moving, appearing and disappearing to give the impression of a real VUmeter moving. The motion of the needle was complex to manage, because it needed to make some geometrical rotation according to received data.

Both buttons are simply two images that exchange places according to the state of the button (pushed or not). The counter is one background image with numbers displayed over it in a font, which imitates the old digital alarm clocks.

In our user interface, there is a cinematographic reference. Will you find it?

IV. The final product

1. Description

The final product is a single view application, with a VU meter whose needle moves to indicate the strength of the electromagnetic field, an on/off button to start the music, a record button to record the played music on the SD card and a counter to display the record time.

When started, the application plays some music according to the data we receive as explained in the Music Composition section of the present document.

2. How to use it

This application is very easy to use. Once open, you push the on off button and the music starts or stops. The process is similar for the record button. Since we ran out of time to add an audio player to our application, you need to use a file explorer application, go into the SD card folder and find the pd_record folder to replay the recorded tracks. The application is optimally used with headphones and we highly recommend them to enjoy the full experience.

3. Analysis

The project almost reaches what we wanted in the first place. It is quite nice to hear and watch. To fully complete it, we would need at least one more week to fix some graphical bugs with the needle or the numbers of the counter that do not exactly move as wanted. But all those problems are minor issues.

We also wanted to add a second view page with an audio player to play the recorded tracks back but we ran out of time. Moreover, that would have been a more complex task to achieve. We would also like to fix some bugs in the sound generation to remove some parasites that can occur from time to time. We also think about improving the music generation by adding more interactions between the inputs and the musical elements. We would also like to add some new instruments, beats and transitions to the music patch. Of course a project can always be improved and we chose to stop here with a stable and complete application that already satisfies us.

4. Issues

Our first challenge was to understand how to use Pure Data on an Android phone. So we followed the instructions given by the creator. Eclipse is complex environment with many things to take care of. But we are used to it so we handled that part easily. Then we thought that the library would allow us

to generate sound with command lines. That was not even close to the truth. Actually the library simply makes it possible to communicate with existing Pd patches.

That leads to our second and major challenge: learn a new and unusual programming language, Pure Data.

To go through that issue in such short time, we followed training with François Parra. We completed that training with a workshop where we learnt a lot and became independent to continue our project. Moreover the distribution used in the Libpd library is lightened. We had to recreate some complex instrument using only the basic component of Pure Data.

As in many development projects, both of us were modifying the source code at the same time. We decided to use github, a famous versioning website. It is a powerful tool but too hard to use. We spent too much time trying to merge our project versions. We stopped using it and had to change the way we worked. As we worked on two different platforms with two different languages we split the work: Android development on the one hand and Pure Data Development on the other hand. We never had to merge anything again, and we saved a lot of time. We just needed to coordinate the communication between the two layers.

We encountered some others issues in the sound generation part. We had to manage some sound interference and saturation. Pure Data is not physically limited but speakers and GPU (Graphics Processing Unit) are. So we cut the high frequency off and added a sound limiter in order not to damage the phone or headphones components and the ears of the user.

V. Conclusion

This Final Year Project was a great opportunity for us to discover new fields and ways of working. Cooperating with the Aix School of Art was very interesting since our skills were really complementary. Learning Pure Data was truly enthralling and opened new vistas for us. Indeed, it forced us to understand sound synthesis from scratch, something that always interested us. We started the project with no clear objective, and our main goal at the beginning was to experiment the possibilities of combining a mobile phone with a sound engine. Therefore, we were very glad to be able to finally produce a complete and working application. We learned a lot about music composition, which is unexpected for an engineering project, but the artistic part of the project definitely broadened our horizons and we are thankful for that. The artistic atmosphere in which we worked clearly motivated us to try new things and gave us the freedom that was needed for such a project.

VI. Recommendations

Unfortunately, we think that having to deal with other projects or lessons during this one-month project has been counterproductive. Indeed, we should rather work full time on this project. Instead of having dedicated working hours in our timetable before the project actually starts, we would recommend starting directly with a month and a half time fully dedicated to the Final Year Project. Such an organization would have allowed us to take part from the beginning to the workshop organized by the École Supérieure d'Arts d'Aix-en-Provence.

We might regret a lake of structure in the organisation of the cooperation between both schools. Some projects should be prepared before and proposed to the students to motivate them to take part in those collaborative projects. Working with the Art School was profitable to us, so we would recommend more collaboration with other schools or universities.

Doing a workshop with the students and teachers of the Aix School of Arts made us realize how interesting and stimulating workshops can be, that is why we would like to have more workshops during our academic training, especially during the early stage of reflection.

VII. Bibliography

As any development project we mostly use the official Android documentation.

http://developer.android.com/index.html

All information we needed to use the Libpd library was here:

https://github.com/libpd/pd-for-android

We also used from time to time:

http://www.stackoverflow.com/
http://www.developpez.com/

Our inspiration came from this video:

http://www.youtube.com/watch?v=w5qf9O6c20o

VIII. Appendices

Lexicon

BPM: Beat Per Minute.

Dataflow programming language: In computer programming, dataflow programming is a programming paradigm that models a program as a directed graph of the data flowing between operations, thus implementing dataflow principles and architecture.

Eclipse: It is a multi-language software development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java.

Graphics Processing Unit: also occasionally called **visual processing unit** (**VPU**), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the building of images in a frame buffer intended for output to a display.

GUI: Graphical User Interface.

IDE: integrated development environment.

Pure Data: an open source visual programming language aimed at creating multimedia works and interactive computer music.

User interface: In the industrial design field of human–machine interaction, is the space where interaction between humans and machines occurs. In our project it is a graphical user interface.

4092 words.