POLITECNICO
DI MILANO

Computer Music System

CMLS Homework 3

Manuel Alejandro Jaramillo Rodríguez, 991288
Marcello Grati, 102707
Maria Gracia Fernandez, 990109
Silvia Pasin, 101090
Natasa Popovic, 991034

Milan, 2022

1. Introduction

In this third and last assignment, the task was the realization of a complete computer music system composed of three basic parts: interaction system unit, computer music unit and graphical feedback unit. Based on this idea, we came to the conclusion of making a kind of music video game from what we had learned during the course, the work done in the previous assignment and the hardware selected.

2. Development of the assignment

2.1 Hardware

For the development of our project we made an intensive research of all the hardware offered. Finally, we selected the one we thought would be the most interesting and that offered a great number of possibilities since, when we had to choose it, we still did not have a closed idea about what we were going to do.

In the end, this was our selection:

- **Electronic board:** we chose *Bela* because we were struck by its great versatility and specifications for a musical project such as the one we had to implement. In addition, we found it interesting to interact with an electronic board not as well known as Arduino or Raspberry.

- **Digital device:** although several attracted our attention, we finally chose the *conductivity switch sensor* because it offered a large number of possibilities for the user to interact directly with the computer music system.

- **Analog device:** we chose the *joystick* because it seemed to us to be the most versatile of all the options presented.

2.2 Idea

The process of choosing the idea to develop was a bit complicated due to the large number of possibilities that were presented. First, we focused more on ideas related to the conductivity sensor switch, such as the realization of a fruit piano or a mixer with the possibility of adding effects based on the button pressed, or fruit pressed in this case. Finally, we discarded a large number of these ideas because we did not think they were good enough,

we did not think they offered a good balance between originality and difficulty. We continued the creative process until we finally came to a conclusion.

We chose those interesting ideas mentioned in the development and tried to put them all together as compactly as possible. That's how *Postaccio Invaders* started.

It is a multiplayer music video game with an interface consisting of a piano at the top, a small selection screen at the bottom and in the center the spaceship's movement space.

The goal is that the player, driving the spaceship, must dodge the "bullets" launched by the keyboard, which is controlled by another player from a midi keyboard and without seeing the interface. In order to dodge the bullets, the ship will move through the interface changing the parameters that appear at the bottom, and he can also change some types of parameters through the conductivity switch sensor, making the notes played by the opposing player sound differently based on the position of the ship at each moment. The game ends the moment a bullet hits the ship.

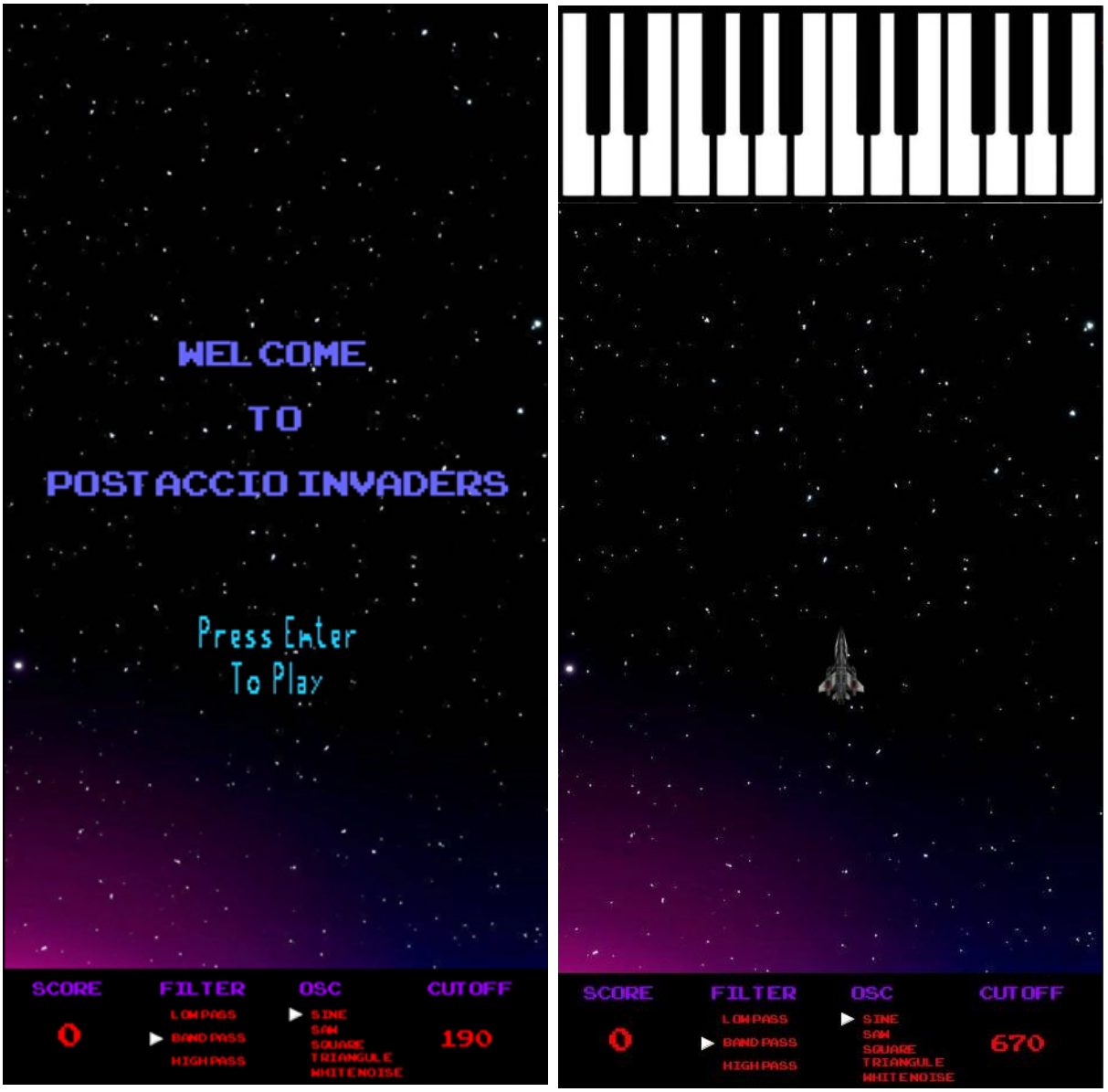

Image 1.- Video game Interface

2.3 Scheme

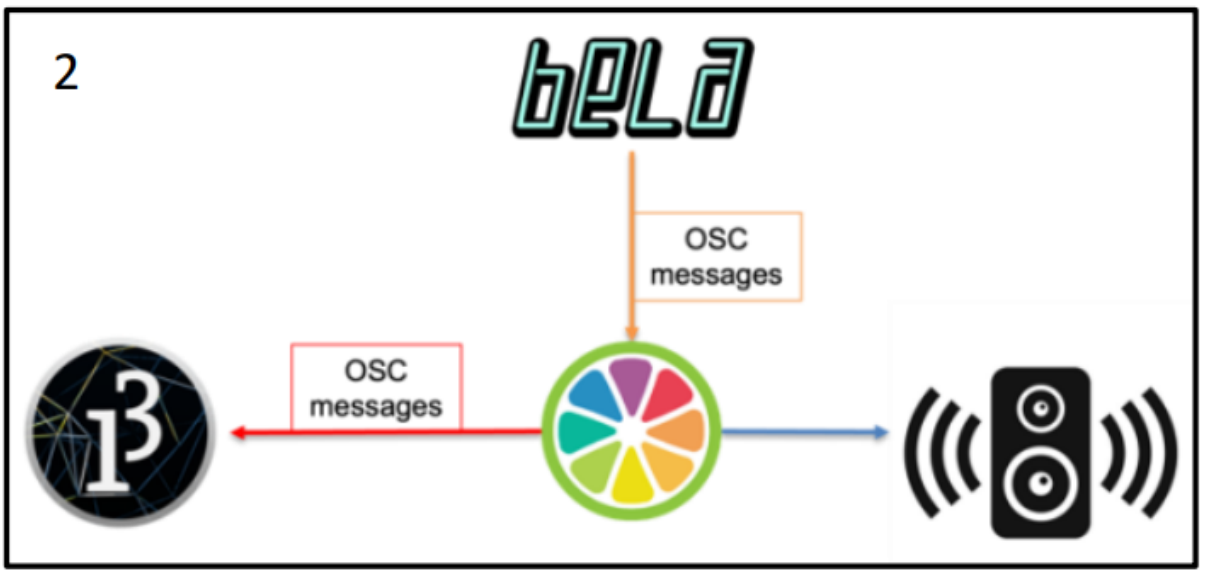

Image 2.- Scheme of the Computer Music System

As we chose Bela for the realization of our computer music system, we decided to use one of the schemes suggested in the statement of the assignment as a base and, based on the needs of the project, to change it if necessary at some point. This consisted of using Bela as the control device, SubSynthHW3 as a JUCE audio plugin with the purpose to generate sounds and manage the communications, and finally Processing for user interface and game design.

2.3.1 Bela

The electronic board Bela has the function to obtain the values from the analog and digital sensors and send them to JUCE plugin through OSC messages. In particular, the values X and Y of the joystick are read through the function *analogRead* from the analog input 0 and 1, respectively and the value of the conductivity switch sensor is read through the function *digitalRead.* These values are sent to JUCE through OSC messages. In order to do this, we used an object of the class OscSender. First, we have set up the IP address and the port to which the data has to be sent (IP for Windows: `192.168.6.1`, IP for MAC: `192.168.7.1`). Then we sent the actual 3 values, creating three different messages: "/joystick/x", "/joystick/y" and "/button".

2.3.2 Juce

We decided to start the project by using the code from HW2, which was the implementation of a subtractive synthesizer. Then, after some general improvements, we started working on the communication protocol based on OSC messages, connecting on two different ports the Bela device and Processing. This is made possible by using juce_osc module, that provides all the necessary classes and methods.

The JUCE plugin sound is controlled by midi, and takes as input from Bela three different messages:

1. /joystick/x : float value of x axis coordinate of the joystick
2. /joystick/y : float value of y axis coordinate of the joystick
3. /button: int value from 0 to 1, that represents if the button is connected or not

This 3 messages control respectively:

1. the cutoff frequency of the filter, scaled in a logarithmic way and band limited from 65 Hz (C2 midi note) to 523 Hz (C5 midi note), in order to achieve a better sound in the gameplay
2. the type of filter (bandpass, lowpass, highpass), dividing the height in 3 regions
3. the type of wave shape, using the button to increment a counter

The plugin take care of modifying this parameters and generates the sound based on the midi messages, then proceeds to send to Processing the following messages:

1. /joystick/x : same as before
2. /joystick/y : same as before
3. /wave : number id of the wave type
4. /key : number between 0 and 23 that identifies the note played by the midi keyboard

The main part of the code that implement these functions can be found inside the PluginEditor component, which inherits from OSCReceiver and implements the two methods oscMessageReceived() and showConnectionErrorMessage().

2.3.3 Processing

The graphical interface of the game is completely done in processing, to do so we created a basic background consisting of a space image, the piano keyboard image in the top, and the game state information in the bottom. Then we have three main windows, the home window which shows the game name waits until the user press “ENTER” to start the second window, which is the actual game, this is the most complex window since it should be updated in real time with all the user interactions i.e, moving the ship, changing the synth parameters and shooting the bullets.

The most complex part of the interface is the interaction with the player that is playing the MIDI keyboard, this because we need to make an accurate visual feedback of the notes played in order to let the player controlling the ship have enough time to respond to the bullets shooted from the keyboard, to so this we decided to create two main classes, one called “Bullet” and other called “KeyPressed”, the first creates an object to allocate the information of one bullet (int X position, int Y position, int velocity, boolean drawBullet), so every time a MIDI key is pressed drawBullet is triggered and the bullet object is rendered in the interface, the second class is “KeyPressed, this creates an object with the information of the key pressed by the keyboard player in order to put a visual mark on the interface keyboard images, everytime a key is pressed, the object is triggered and a mark is drawn in the corresponding position of the piano image.

Finally there is the game over window, everytime a bullet reaches the ship, the game is over and this window is shown, the user has the opportunity to press a key in order to play again.

2.4 Controls and Interface

Once explained the main idea and the different parts, in this section we are going to talk more in depth about the functioning of the implemented controls as well as other functionalities that can be observed in the interface.

First of all, the spaceship moves freely through the interface to escape from the bullets thrown by the piano. Its movement will be done through the joystick both in its X and Y axis; however, performing these actions not only moves the ship, but also allows us to play with other parameters of our system. When we move on the X axis (analogX) we will control the cutoff frequency of the filter we have at that moment. Meanwhile, with the Y axis (analogY) we can change the type of filter so that, when the spaceship goes up it will correspond to a high pass filter, in the middle it will be a band pass filter and, if it is the farthest away from the piano, it will be a low pass filter.

It should be noted that the frequency that is controlled with the analogX is mapped logarithmically. That is, the relationship between the range of joystick values (from 0 to 1) and the filter cutoff frequency range (from 65Hz to 523Hz), will be logarithmic in order to obtain a higher resolution in the low frequencies.

On the other hand, through the conductivity switch sensor we will be able to vary the oscillator waveform. You can choose between sine, saw, square, triangle and white noise while continuing to control the ship, so that we can look for a better sonority while continuing to dodge bullets, giving rise to another challenge for the player.

The score obtained is based on the number of bullets dodged, and is recorded and displayed to the player in the lower left corner of the screen along with the rest of the parameters mentioned above.

Image 3.- Parameters menu

3. Conclusions

The first thing we want to emphasize about the realization of this project is that we found it the most interesting we have done so far during the course. The freedom that we had to design an idea has seemed to us a key factor, as well as the introduction of hardware has seemed very interesting as it has allowed us to physically interact with our project making the process much more entertaining.

However, even with these advantages it does not mean that the complexity has been reduced compared to previous assignments in which we had to follow an idea and improve it but without losing a certain purpose. For example, the creative process and coming up with an idea was a bit complicated because we were not able to find a balance between a feasible idea and one that was good enough for the assignment. In addition, the first contact with Bela, the connection of the various components and checking its functionality, although it was rewarding we felt a little lost because of our little experience, but we managed to continue and focus quite solvent.

On the other hand, although our initial idea contemplated something more oriented to an instrument or to play with different effects, we are happy with the result and the implementation made. It has allowed us to create a multiplayer video game by developing some parts of the previous assignment and implementing many others, as well as deepening the possibilities offered by Bela and the different hardware we had.

Of course there are a lot of things to improve, we can add more features both to the game and to the synthesizer, however, our codes are organized in a modular object oriented way, which allows easy implementation of updates and adding more features.