Computer Music - Languages and Systems
Homework 3
Interaction Design: The FruitMachine

Group 6 - GELVS:

Del Moro Samuele, di Clerico Letizia, Negroni Viola, Perego Gabriele, Roncuzzi Enrico.

1 Goal

Implement a complete computer music system with interaction design principles. Use Arduino as control device, transmit gestural information to SuperCollider with Serial messages, provide the feedback events to Processing with OSC messages.

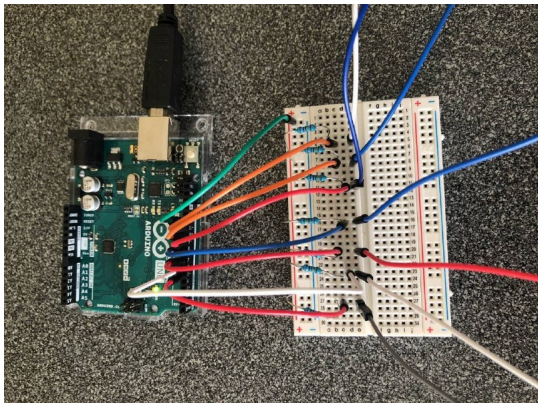
2 Concept

Is it really necessary to own a musical instrument or an expensive console to create beautiful pieces of music? Is really making music a privilege of few?
For this last homework of interaction design, we wanted to imagine a system that was able to bring anyone close to the creative process of making music, a tool that everyone could afford and built him/herself and that would be perceived as intuitive and simple to play with, by involving something that is not far at all from our everyday experience: **fruit**.

The Fruit Machine is the synthesis of these thoughts: a computer music system that allows the user to make music out of whichever kind fruit he desires, composing his/her personal keyboard as he/she pleases and making sounds simply by touching it.

3 Method

3.1 The control device: Arduino



Among the available control devices, it has been decided to operate with Arduino.
Here below the necessary components are listed:

- Arduino UNO
- Breadboard
- Resistors (7) 1MOhm
- Jumper Wires (7+1)
- F-M Dupont Wires (7)
- Power cable

As regards the programming side, as a first thing 7 Arduino pins (7 as the number of the notes in the diatonic scale) have been turned in Capacitive Sensors, importing the proper library. In this case we chose pins from 5 to 11, but of course other choices can be made, with the foresight to change the numbers in the code.
A capacitive sensor works basically as a capacitance, and in this case in particular it has been taught to detect any change in the pressure on its surface: when a contact is made the circuit closes and current flows within it.
This is why the resistors with the highest value of resistance have been chosen: the higher the resistance, the higher the readings we will get.

The setup part consists simply in turning off the autocalibration on the channels.

In the loop function the sensitivity of the sensors is set, then, for each channel, a "while" cycle has been implemented: when a sensor is touched a number is printed, a different one for each channel (to make them distinguishable), and the variable "stato" is updated.
The last couple of lines are required in order to set the "stato" variable equal to zero when no touch is applied, meaning that in that case no sound is going to be produced.

3.2 The input processing: SuperCollider

As going through the code, we first meet a section where some fundamental variables are declared, while right after the setup needed to link both Arduino and Processing is performed. We here remember that Arduino and SuperCollider communicate through Serial messages, while SuperCollider and Processing use the OSC protocol.

Then, the proper "input processing" section is met: here the SynthDefs that synthesize each kind of sound are defined.
In the order, we find:

The instrument types (each defines a specific timbre, but the fruits will always be associated, in this mode, at the seven notes C, D, E, F, G, A, B)

- Ring
- Bass Simple
- Bass Trance
- Piano

The percussive instruments (aka "Drums", one for each fruit)

- Kick: a typical element of a drum set
- Snare: a typical element of a drum set
- Clap: self explanatory

- Hit Hat: a typical element of a drum set
- BGlass: a "glass breaks" synthesizer
- Bell: self explanatory
- Laser: pretty much like the sound of a laser sword

Hence, a total of 11 different sounds can be appreciated so far.

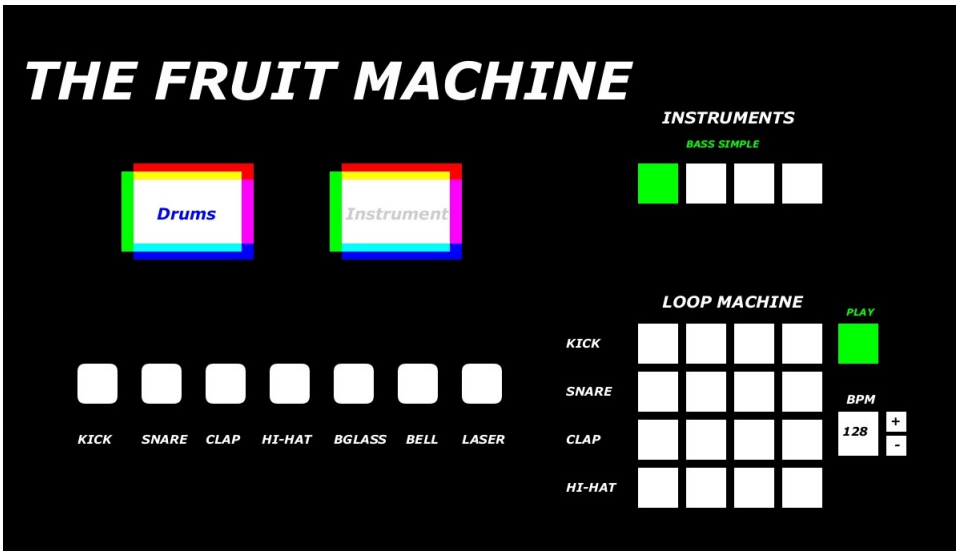
In the following section, the communication with both Arduino and Processing, where the GUI has been written, is properly set.
Four functions are implemented in the first place:

- funNote: sets the value of the fundamental note
- funDrumOnOff: discerns whether the Drums mode is set on or off by the user (of course, if Drums is off, Instrument is on)
- funPlayInstrument: activates the chosen instrument
- funSendOSCtoProc: the function that actually sends the communication to Processing by OSC messages

Then, as the Routine is started, the Serial messages are read from Arduino and each kind of OSC messages is defined, in order to interact with Processing: we have the messages to change the BPM when using the loop machine, the ones to switch between "Drums" and "Instrument", the ones that allow to browser between the different types of instruments and the ones that enhance the loop modality.

Finally, in the last part of the code, the connection is made: the registered input from Arduino is placed in the right playing mode and, therefore, every fruit is linked to a specific sound (a specific drum component or a given note). Information is then transmitted from/to Processing.

3.3 The GUI: Processing



The user interface has been implemented through the software Processing, who has been made able to communicate with SuperCollider, by sending and receiving OSC messages (the appropriate library "oscP5" has been imported).
This communication section can be found in the second part of the code: here, first we encounter the section dedicated to the OSC messages coming from SuperCollider to Processing, then below the one for the messages to be sent from Processing to SuperCollider.

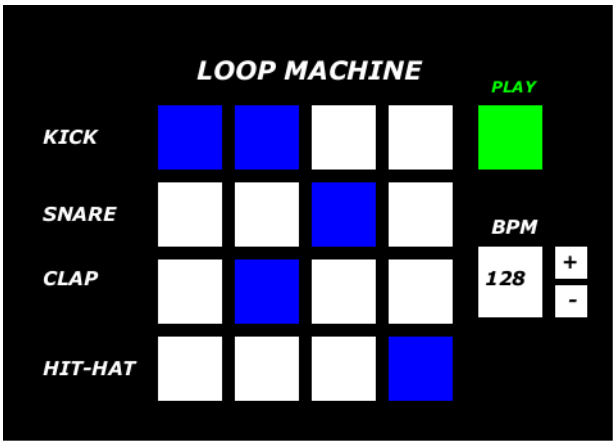
As regards the GUI, on the top left there are the 2 animated buttons that are related to the "playing modality". So far there are two available modes, the "Drums" one and the "Instrument" one: clicking on the former with the mouse will assign to each fruit of the set a drums component/percussive sound, while clicking on the latter will make the fruits correspond to the seven notes of the diatonic scale.

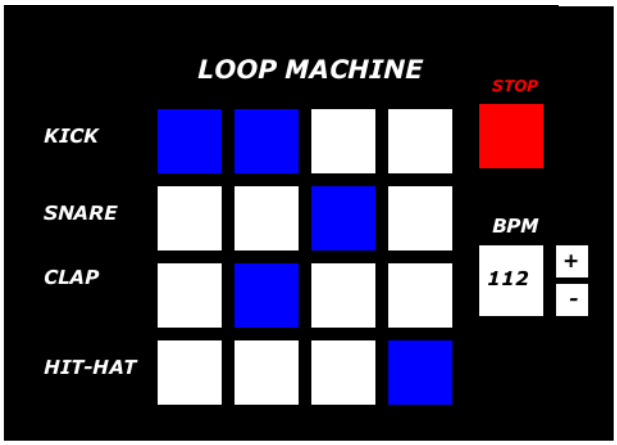
It is through the top right panel that will be possible to switch between the predefined types of musical instruments ("Piano", "Ring", "Bass Simple", "Bass Trance").

The bottom line of buttons has been introduced to give the user a visual real-time feedback of the fruit that he/she is playing: every time one fruit is pressed, the correspondent button will lighten up, showing a particular color and followed by some colorful random-positioned animations.

On the bottom right, we have an additional functionality implemented for the "Drums" mode: a loop station that basically works as a simple drum machine in beats of 4/4, through which it is possible to compose a basic rhythm with up to 4 kinds of percussive sounds ("Kick", "Snare", "Clap", "Hi-Hat").

Once created the rhythm, after pressing "Play" it will be possible for the user to use it as an underlying motif over which build a richer musical theme through the fruits.
A display showing the current BPM value has also been made available, and note that this value can be changed run-time if desired.



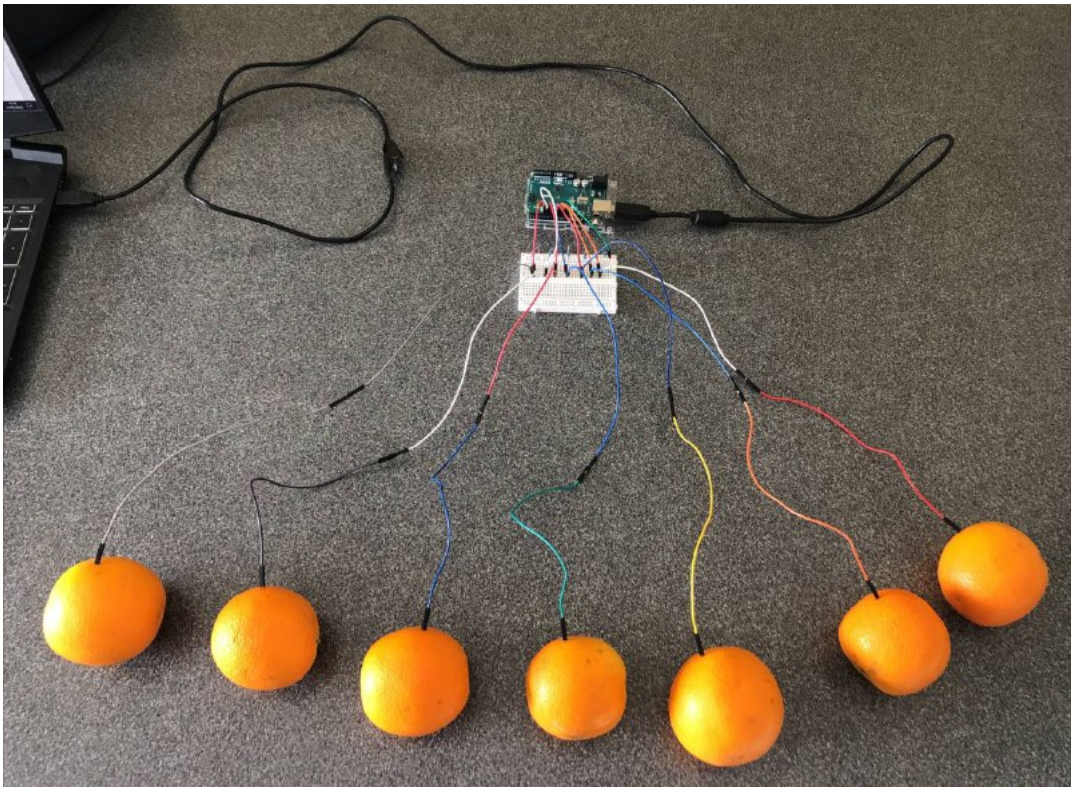


4 Results

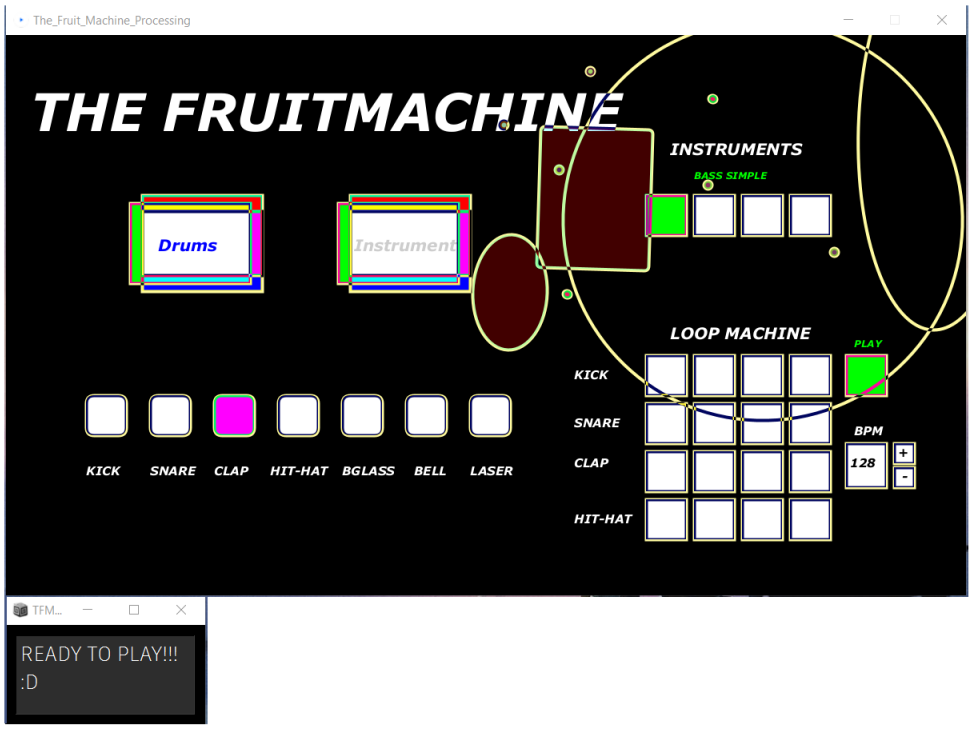
The Fruit Machine is a system that allows the user to create a very personal keyboard made of fruits.

In order to produce sounds, it is only necessary to touch the fruits, after having run the codes in Arduino, SuperCollider and Processing (this order is mandatory).

A feedback to the player is provided by a graphic interface, where it is possible to see the available modes (Drums/Instrument) and switch between them by clicking on the appropriate button.
The bottom line of buttons allows to have a visual reference of which fruit, and so which drums component/note is being played.



The right panels include additional functionalities: through the top one it is possible to switch the instrument that is being played in the "Instrument" mode (4 different types available for the time being) while the bottom one is a loop station designed for the "Drums" mode, that allows the user to create a basic rhythm on which to build a more complex mixture of sounds through the fruits, also changing the BPM value run-time if desired.



5 Conclusion and further improvements

Building a computer music system from scratch is, without any doubt, a thrilling challenge.

Hence, in order to keep the task inspiring without risking to focus on something too complex that would risk to become incomprehensible, an almost didactic goal has been set: the main driver behind this project proposal has been the clear aim of bringing anybody, even (and maybe especially) those who have absolutely no clue on how to play an instrument or using any kind of synthesizer, closer to music composition with something bizarre and surprising, but also absolutely friendly and affordable.

This, to be sure that our system would be sufficiently intuitive, and yet - nonetheless - it would still inevitably require a tricky implementation from the developers side.
As often happens, simplifying happens to be less easy than complexifying.

Of course, given the short amount of time to dispose, what we offer right now is just a glimpse of this intuition, whose elements have all been already included, but that for sure can be improved and extended.
Among the upgrades we would like to implement in the near future, as soon as there will be enough time there are:

- Allow to switch between the modes with an additional fruit, since there are still few pins available.
- For the same reason, expand the number of different possible inputs beyond 7.
- Arduino is not always very effective, so a more precise hardware could be used to detect the touch on the fruit.
- Increase the number of instruments and drums available and ameliorate the loop machine making it editable during the performance and adding a greater number of beats.