Computer Music Languages and Systems

Homework n° 3
FM synthesis

10574752

10751438

10612929

10486570

Master of Science in Music and Acoustic Engineering

POLITECNICO
MILANO 1863

Abstract

Our group worked on **Assignment 3**: Create an instrument based on FM synthesis and an interface for controlling it. The link containing the source code can be found on GitHub at: `https://github.com/EllDy96/CarlGang/tree/Homework3`

Chapter 1

1.1 SuperCollider

Synthesizer definition

MIDI control

We defined a global variable (*monoNote*) as an instance of the Synthesiser we defined, in order to easily have the access to the parameters of the synth and change them immediately. MIDI controls are composed by a *noteONFunc* that receives the velocity, the note, the MIDI channel and the MIDI port and set the frequency and the amplitude. It is also defined a *noteOffFunc* that receives the same parameters of the noteOnFunc and set the amplitude to zero.

OSC messages

The OSC communication is set to the localhost (id "127.0.0.1) at port 57120. The OSC message is composed by 4 parameters, respectively:

- msg[1] is the horizontal value of the centroid
- msg[2]is the vertical value of the centroid
- msg[3] is the palm-index distance
- msg[4] is the palm slope

The parameters received have all a value between the interval $[0, 1]$ so we mapped them in proper intervals in order to give to the user a pleasant effect and interaction when the hand is moved. After the mapping the hand's parameters are set to the corresponding synth's parameters:

$$\text{centroid x} \rightarrow \text{feedback}$$

centroid y → index
palm-index distance → indexScale
palm slope → panning

1.2 Interaction design

Modules and libraries used:

- Node.js
- Socket.io
- Express
- p5.js
- ml5.js
- osc.js

The synthesizer can be controlled through hand gestures captured from a webcam. For the hand pose recognition, we used a pre-trained ML model from ml5.js (a javascript framework for creative coding built on top of TensorFlow.js), which takes frame by frame the video stream and return the coordinates of 21 points of the hand (this process is GPU intensive, even though the model is lightweight, a system with a dedicated graphic card is advised for best results).

Figure 1.1: 21 hand points and control parameters

From these 21 points (x and y coordinates) we compute 3 parameters:

- the centroid (green dot) with coordinates:

$$x_c = \frac{\sum_{i=1}^{21} x_i}{21}$$

$$y_c = \frac{\sum_{i=1}^{21} y_i}{21}$$

- the distance between the tip of the middle finger (x_{mf}, y_{mf}) and the base of the palm (x_{pb}, y_{pb}) (length of the white line):

$$d = \sqrt{(x_{pb} - x_{mf})^2 + (y_{pb} - y_{mf})^2}$$

- the orientation of the hand (the slope of the white line) between $[0, \pi]$:

$$s = \left| \arctan\left(\frac{y_{mf} - y_{pb}}{x_{mf} - x_{pb}} \right) \right|$$

The user interface is hosted as a web page/application in an Express server, the connection is set up through the framework Socket.io. All the control parameter mentioned above are computed in the client and then sent to the server. From the server, the parameters are written in OSC messages and forwarded to SuperCollider. This last past is handled through the library osc.js, which can generate OSC messages from javascript objects and establish a connection with a receiver (i.e., SuperCollider through an UDP connection). The OSC message has only one path "/params" in which are contained all the parameters as floats.

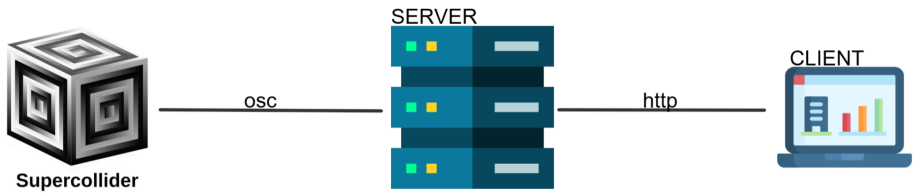

Figure 1.2: Application architecture

The user interface, as we just said, is a web application in which we imported the libraries ml5.js and p5.js. We set p5.js in Instance Mode in order to manage 4 different sketches which compose the main window. The bigger

p5 sketch at the top left is the one visualizing the webcam, the 21 points of the hand and the control parameters. The other three are a representation of the control parameters using psychedelic animations. At the bottom left we have a visualization for the hand orientation, at the top right for the x and y position of the centroid, and finally at the bottom right, for the distance between the middle finger and the palm base.

Chapter 2

2.1 How to use it?

In order to use the application, run the server using Node.js with the command `node .\server.js`. Then connect to the url `localhost:55123` in a browser (it may take some second to load the ML model). Open the synth in SuperCollider and run all the code, it will start to listen for OSC messages through the port 57120. Enjoy!

2.2 Improvements roadmap