Computer Music Languages and Systems

Homework n° 2
Flanger effect plugin with feedback

10574752

10751438

10612929

10486570

Master of Science in Music and Acoustic Engineering

POLITECNICO
MILANO 1863

Abstract

Our group worked on **Assignment 3**: implement a Flanger effect plugin with feedback. We divided our work in steps as presented in the sections of this document. We started from the block diagram of Figure 1.1, first implementing the Delay, then adding the LFO and the feedback. We designed the plug-in in JUCE, coding the audio processor and the Graphical User Interface with Projucer and Visual Studio. During testing, our results led us to set custom parameters' ranges that best suited the feel of the Flanger effect. As a reference, we used the "Material on delay lines" PDF and the related JUCE class laboratories. The link containing the source code can be found on GitHub at: `https://github.com/EllDy96/CarlGang/tree/Homework2`

Chapter 1

1.1 SuperCollider

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). 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 UPD connection). The OSC message has only one path "/params" in which are contained all the parameters as floats.

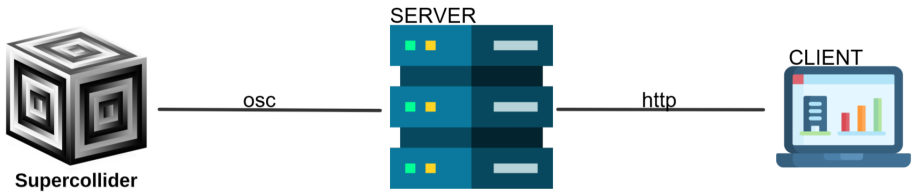

Figure 1.1: Application architecture

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 in 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.