Execution

Introduction

The following chapter gives an outline of the process that was undertaken to build out the final application. A description of early prototype work is given to give context to the construction of the final working prototype version. This is followed by a description of the technical architecture of the system as well as an outline of the sometimes tricky setup process of getting the live reload system working (as described in the previous chapter). Some detailed discussion of the the core functionality of the system is then given. The core of the system primarily consists of timeline events, which visually manifest themselves as strokes on the canvas and aurally as fm syntesized frequency modulated sounds. As will be discussed an important aspect of any well constructed software system is a good degree of seperation of concerns, a characteristic espoused to and evident in the resulting code. To this end, the code that brings the central functionality to life can conceptually divided into a data or entity layer, a business logic or use case layer and an output layer which in this case consists of the visual output into a html canvas element and the audio output through the web audio api. In this sense the architecture conforms to the principles of the Clean Architecture as presented by Bob Martin (???).

Early prototype work

Melodypainter

Melodypainter is an early protoype built out in Max MSP that allows users to draw freehand lines, which are converted into break point function data and used to generate a melodic profiles using Bach for Max MSP. Bach is a suite of composition tools that allow for a number of computer aided composition techniques (CAC) and provides similar functionality to IRCAM's Open Music system. These melodic profiles are then filtered to only includes notes from a pentatonic scale, to give reasonably pleasing aural results. Some notable flaws in the system include the following. It is limited to strictly western tonal music styles. It has no allowance for rhythm and plays only eight notes giving results a noticeably bland and predictable quality. The freeform nature of sketched input however was quite a pleasing means of inputting the control information.

Sonicshaper [0/1]

A separate application was created in Processing which allowed users to draw shapes, using either mouse or ideally, pen input and have a sound that is associated with each shape played back. As the sound of each shape plays back,

it is lit up using animation, creating a strong connection between the shape and it's resulting sound. The application uses the "gesture variation follower" system [@caramiaux_adaptive_2015], which while promising in principle, didn't have a high rate of accuracy in recognizing the shapes.

Web version of William Coleman's SonicPainter [0/1]

A potential starting point that was considered was using the code from William's SonicPainter and porting it to the web platform. This process proved to be quite straightforward. The processing code could more or less be embedded in a webpage as is using "processing.js", a web version of the Processing library that enables users to run processing sketches in the Web Browser. Some notable changes that had to be made were removing the OSC functionality as this is not technically possible to use in a browser. In addition, some other pieces of code had to be commented out and tweaked. As it's not possible to run Max MSP patches in the browser, the audio system was re-implemented using Tone.js.

Actual implementation

Setting up the architecture

Clojurescript and javascript npm modules

Paper.js and react.js (paper.js bindings)

Tone.js and react.js

Reagent and react.js paper.js bindings

Core functionality - timeline events (or notes)

1. Introduction
    - Describe the core functionality
    - Describe core entities
2. Add timeline event
    - Business logic
    - UI
    - Audio
3. Add vibrato
    - Business logic
    - UI

    - Audio
4. Remove note
5. Move note
6. Change sound (preset system)
7. Probability

Secondary functionality

1. Introduction
2. Transport controls
3. Animation (current play position & notes)
4. Undo and redo
5. Fullscreen
6. Outer UI
7. Save and load file

Performance issues

Conclusion