# Kevin Nolan MMT thesis 2017

Kevin Nolan

July 28, 2017

## Contents

# 1 Introduction

## 1.1 Motivations

Some primary motivations for the creation of the app are as follows (in no particular order):

- Draw attention to concept of more graphically oriented interfaces for synthesis (UPIC) to a wider audience by presenting the app in a very accessible format (PWA)
- Explore the advantages and limitations of prototyping an experimental interface using web technologies
- Frustration with the creative flow of commercial DAW applications
- Over representation of analog type metaphors in both DAW applications and in the web apps being developed (missed opportunity)
- Use functional programming techniques such as immutable data structures to ease the complexity of managing fundamental concerns such as undo.

## 1.2 Project goals

## 1.3 Structure of thesis

# 2 Background

## 2.1 DAW analog metaphors

One of the primary tools used by electronic musicians today for the production of music is DAW and it's inherent metaphors based on analog system still reign supreme in the field [?]. The familiar concepts of analog tape machines and mixers benefit the novice user by offering a network of familiar and tangible real world metaphors in which to carry out their creative work. However, as well as the benefits that these types of metaphors bring, they also impose some limitations and bring about certain biases. Musical ideas that are difficult to realise can be left unexplored.

A particular criticism of the DAW is the difficulty in maintaining and managing the editing of complex automation information. Automation is the term given to the continuous altering of aspects of the sound and is usually represented in lanes separate to the primary note pitch information. It may be recorded in or drawn in by the producer. Difficulties can arise, when multiple subtly interacting lines of automation, such as pitch bends and filter changes are being manipulated. William Coleman gives a particularly clear example of this and outlines the difficulty of representing "portamento time", the time it takes a note to slide from one to the next. The visual results can be jarring, unintuitive and not reflective of the audio results.

Duignan (2008) describes a similar problem in his study that monitored professional producers working in DAW environments [?, p. 156]. The particular problem identified by Duignan was that of processing one off effects for single musical events. A number of convoluted processes were observed, including bouncing the affected portion to audio, duplicating the track, setting up a particular auxiliary for the effect and controlling the effect with automation. In these cases, the hierarchy imposed by the DAW gets in the way, where it could be modeled quite elegantly in a more open program such as Max Msp. This, unfortunately, raises the issue of drifting into the area of analytic thinking and away from creative thinking, a combination that John Cage advises against: "Don't try to create and analyse at the same time. They're different processes." [?] The need to explore alternative metaphors is clear. A description of a promising alternative metaphor, that of drawing/sketching will now be discussed.

### 2.2 Legacy systems

- Oramics
- UPIC

### 2.3 Golan Levin etc

## 3 Similar work

### 3.1 Criteria

**Web based** Does the system work on a modern browser

**Symbolic rep** Does the system use symbolic representation or is it more or less a spectrum that you can draw on.

**Accessibility** From none to high, is the system accessible. An example of a no accessibility is a system like UPIC which is not accessible to something that is very accessible like web app. Paid software that needs to be installed is in the middle of this spectrum.

| System | Web based | Symbolic rep | Accessibility |
|---|---|---|---|
| SonicPainter | No (although ported by the author) | Yes | Low |
| UPIC | No | Yes | None |
| Oramics | No | Yes | None |

# 4 TODO My approach [0/2]

- □ Come up with a title
- □ Other stuff

## 4.1 Adding allowances for stylus

## 4.2 Relationship between control data and synths

Discuss Roger Dannenberg conceptualization of the two main paradigms of music software systems. Resource based vs instance [?]

## 4.3 Key specifications

## 4.4 Paper.js

## 4.5 NUI

Introduce the NUI

## 4.6 Tone.js

[?]

# 5 Execution

## 5.1 Early prototypes

### 5.1.1 Melodypainter

Thus far, some early test prototypes to establish possible directions for the application have been built. A Max Msp patch was created which allows the user to draw freehand lines, which are converted into break point function data and used as to generate a melodic profile in Bach. This is further processed into a pentatonic scale. Once input the system plays the resulting melody back. A notable flaw of the system was that it required users to draw shapes in a generally horizontal fashion for the data to be of use and to create a strong relationship between the visuals and the generated music.

### 5.1.2 Sonicsketch - shape recognition

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 used the "gesture variation follower" system [?], which while promising in principle, didn't have a high rate of accuracy in recognizing the shapes. It is for this reason that Microsoft's ink api is now being used for further prototyping.

## 5.2 The build out

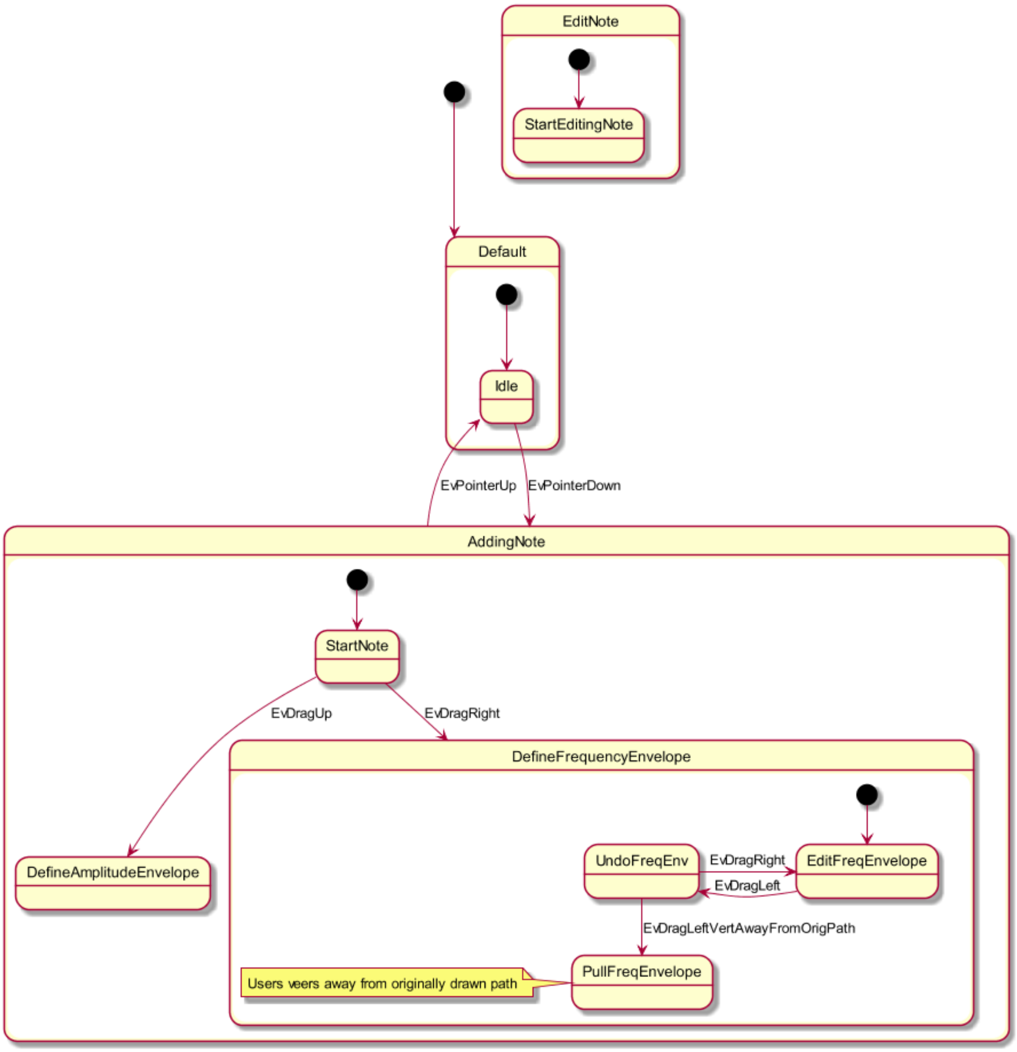
### 5.2.1 Building the framework

#### Advantages of the react.js model of UI programming

### 5.2.2 Stroke functionality

Describe how the stroke functionality works and was implemented.

#### Overall app



1. The relevant code

```
(defn pointer-move [{:keys [temp-obj active-preset]} evt]
(when-let [{:keys [path group] :as temp-obj} @temp-obj]
  (let [pointer-point (.. evt -point)
        rel-pos       (.. group (globalToLocal pointer-point))]
    (log (.. group -position) "pointer point rel")
    ;; Only add positive points relative to first
    ;; Remove points greater than pointer-points
    (when-let [last-seg (.. path getLastSegment)]
      (log last-seg)
      (let [last-point (-> last-seg .-point)
            pointer-x  (.-x rel-pos)]
        (if (>= pointer-x (.-x last-point))
            (-> path (.add rel-pos))
            (let [greater-segs (filter
                                 #(> (-> % .-point .-x) pointer-x)
                                 (.-segments path))]
              ;; Remove greater points
              (doseq [seg greater-segs]
                (.removeSegment path (.-index seg))))))))))
```

### Sketch walkthrough

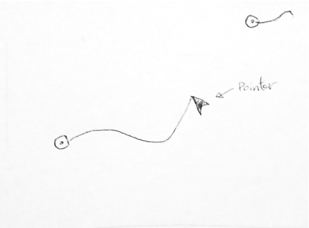
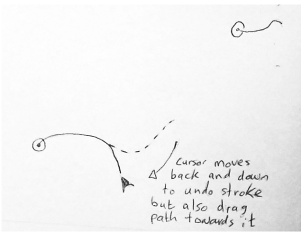
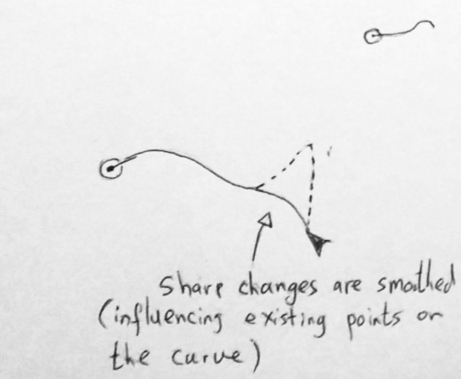
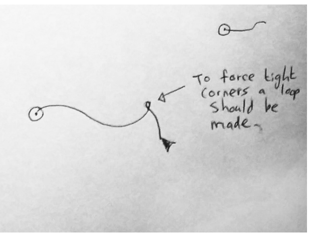


Figure 1: This is the caption for the next table (or link)







Please see [??]

### 5.2.3 Enabling undo and redo

### 5.2.4 Catering for additional input types

Describe issues encountered including a lack of support in paper.js and the small patch required to enable it.

# 6 Evaluation [0/1]

- □ Come up with a better title.

A reasonably functional application was built which allowed for some basic interactions. While very much a prototype the app does offer the following innovations:

- Using the web as a testing ground for new interaction paradigms
- Using react js and clojurescript to allow for realtime (or close to realtime) development
- Delivering a graphical synthesis oriented system in a very accessible format, a PWA
- Incorporation of alternative device inputs using the W3C pointer api to allow for more suitable and accurate input
- Trial of next generation react framework that extends it's reach beyond html markup which is the general limit of the original system
- The result of combining react, clojurescript and tone.js is a declaritive data DSL to describe audio processing. Potentially very useful as a beginner tool and as a prototyping tool.
- Makes the assertion that different tools can be used for different stages of the creative process and by providing integration into other platforms can be easily incorporated into the flow.
- Works as an idea generator. The resulting audio could be sampled and used in another app

Some of the shortcomings of the work:

- It's very easy to create a mess of frequencies and pitch bends and difficult or impossible to create standard musical material
- Suffers from performance issues and can get choked up when too much elements are added to the screen
- While some basic and standard useability features have been added such as undo and redo
- The sound quality isn't always perfect and some aliasing and other digital artifacts are in evidence

# 7 Conclusions and further work

## 7.1 Future work

While SonicSketch in it's current format is useable for further ease of use it would need work done on performance related issues, including but not limited to:

- Move more processing into web workers
- Look at compile to wasm based systems instead of Tone.js, including Csound
- Move graphics to use the GPU more heavily which would mean re-implementing a good deal of the functionality provided by paper.js

Incorporate more dimensions into the visualisation, such as distortion, delay effects, harmonic structure. Investigate how the visualisations could scale up to incorporate more information.

Allow for larger structures, perhaps by scrolling.

Allow for meta-strokes where a stroke would draw the contents of another scene.

# 8 References