Evaluation

Introduction

An evaluation of the application will now be given. This is based on self-assessment, user feedback and a usability questionnaire that polled a small number of participants. User feedback was garnered at an exhibition that showcased the application. More informal ongoing feedback also identified some issues that were fixed in the final version of the application, in addition to some that weren't.

Assessment

System architecture

Overall, the chosen set of technologies worked very well and supported the technical requirements of the project. The use of recently released and beta versions of software libraries involved a good deal of time investment. Once this was setup, however, the live coding workflow enabled easy experimentation and proved invaluable when implementing complex features such as the vibrato overlay. The transparent data oriented approach of ClojureScript aided in debugging and made it straightforward to add advanced features such as undo/redo and save/load.

Core functionality - sketching notes

Adding notes was quite an easy concept for users to grasp and were observed to quickly start making marks on the screen. Some users experienced confusion or weren't aware of how to play these back initially. Once they managed to play the audio back and understood the general nature of the sounds being created, a common request was to clear the screen to start over. After this, they would start experimenting with layering up different sounds on top of each other. Oftentimes abstract visual patterns emerged and it seemed the visual feedback played as much, if not more, of a role than the audio, in what was drawn on the screen. Perhaps some real-time audio feedback might balance this out more. The version that was tested required that notes are drawn from left to right, causing some initial confusion for users. A future version will remedy this and allow for both directions. The woodblock timbre only allows short lines to be drawn as the sound isn't sustained and quickly decays after its initial attack. This proved to be another source of confusion for users and indicates that it may be inconsistent with other parts of the UI.

Timbre selection

Timbre selection is provided by clicking on one of the four coloured symbols to the left of the screen. The regular geometric shapes represent harmonic shapes and the more chaotic shapes represent non-harmonic sounds. Users did not seem to make this connection though. After sketching some notes and hearing the various timbres, most users were able to make the connection and to develop an intuitive relationship between the visual appearance and the audio result. Some users felt that the integration of timbre selection was a little flawed. When a tool other than the sketch tool was in use, and a timbre selection was made, the expectation was that it would return to the sketch tool. Instead, it stayed on the same tool but just changed the timbre. This was updated in the final version to reflect this expected behaviour. Another suggestion was to use icons that represent the instrument rather than the more abstract icons present in the UI. It was decided though that this would lead to an expectation of more realistic sounds and, thus, was avoided, however.

Vibrato

Similar to tools other than the default sketch tool, the vibrato tool was not heavily used by users and it was rarely discovered without some instruction. Once discovered though, most users were able to operate it after one or two attempts. The overlaid envelope graphic, while visually appealing in itself, doesn't fit the sketching metaphor very well and doesn't depart very far from what would be found in a traditional DAW. The resulting visualisation on the notes helps to tie it back in with the sketching theme. Ideally, the user should not see an overlay and instead directly adjust the visualisations on the note. Unfortunately, these visualisations weren't executed perfectly either and are applied slightly unevenly along the path of the note. It is a definite improvement on those present in SonicPainter though.

Other tools

Of the other available tools, delete and move were the most used. An initial usability issue was the accurate click required to target the particular note to be adjusted. Adding a glow effect to the notes remedied this to some degree and gave a slightly larger area to click on. Apart from this, there weren't any notable usability issues with either the delete or the move tools. The resize and probability tools were the least seldom used and when they were used, posed some difficulties for the user. They both worked on the principle of clicking on the note in question and then dragging in or out to alter the value. The drag motion didn't factor in the current value but was, rather, an absolute value defined by the point that the user clicked and the distance to the user's mouse pointer. Depending on the previous value, this caused a visual jump. In the case of the probability tool, the change in saturation is quite subtle and the changing colours may be difficult for the user to notice. The resize tool, which adjusted the velocity, only changed the node size and not the overall size of the graphic. This made it slightly unintuitive and confusing for the user, perhaps contributing its low usage.

Performance issues

The application requires a reasonably fast and modern computer to run in a usable state. However, even with fast hardware, complex heavily populated sketches lead to audio and animation glitches. Some optimisations were made to improve this. The primary adjustment was to add a `should-component-update` to the `graphic-note` component. This gets called by the React system to check if the component should re-render and is normally not needed for Reagent apps as it supplies a default version. However based on performance testing with Google Chrome Developer tools it could be seen that a large amount of scripting was occurring on every frame (see figure fig:performance). This was the particularly the case when the note count went beyond a dozen or so. Introducing the custom `should-component-update` method reduced these renderings to an absolute minimum bringing a significant performance boost.

Even with this improvement performance is still an issue. This is largely due to the architectural decision to instantiate a new instrument for each note. Potential improvements to this could be to:

- Render content as audio at certain points by using an `AudioWebWorker`, a Web Audio API tool that allows rendering audio concurrently to a buffer. This would alleviate some of the burden on the CPU by decreasing the amount of audio generation it has to do.
- Use Web Assembly, a relatively recent Web technology that allows optimized C and C++ code to be compiled and run efficiently in the browser (Adenot, 2017). Current versions of both Csound and Faust (a functional audio programming language) can be compiled to run in the browser. The author was able to run a Csound version of John Chowning's "Stria" smoothly on a smartphone's web browser.
- Use hardware accelerated graphics to further ease the burden of the CPU and improve visual display and animation.

User testing and feedback

Usability questionnaire

The System Usability Scale (SUS) is usability questionnaire that uses a Likert scale to give an indication of the how easy the application is to use. It poses a number of questions designed to provoke extreme responses either in favour of or against the proposition. Some examples of these are:

- I think that I would like to use this system frequently
- I found the system very cumbersome to use

The original introduction of the SUS questionnaire stated that the individual questions are meaningless and the results must be taken as a whole to give a unidimensional usability scoring (Brook, 1995). Lewis and Sauro (2009) have shown that this can be broken into two dimensions, however, usability and learnability. This helps to gauge how beginner friendly the application in addition to how generally usable it is.

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

Factor	result
Learnability	92
Usability	83
Overall system satisfaction	85

The final score that the system got was 85 out of 100. Learnability scored higher than general usability, getting a total of 92 while general usability came in below that, with a score of 83. This gives a strong indication that the concepts and presentation of the app are easily grasped by novice users and that the general perceived usability is very high. Despite the fact that the number is scored out of 100 (with 100 being the highest score) it should not be interpreted as a percentage. Sauro (2011) has developed a grading system based on the results of over 500 tests, suggesting a grade of A to F, with A being the highest (figure fig:sus-grades). A score of 68 is average and would give a grade of C. Anything above 80.3 is an A grade and according to Sauro (2011), the point that users are more like to start sharing with family and friends. Therefore, by this metric, SonicSketch achieves a Grade A for overall system satisfaction, usability and for learnability.

Some caveats apply, however. The application was tested on a small group of participants (15), most of whom were quite familiar with working with audio and music applications. They were also colleagues of the author which may have caused a bias towards positive feedback. Nonetheless, the indication was that the application was straightforward and easy to start using, and provides a good basis for future work.

General feedback

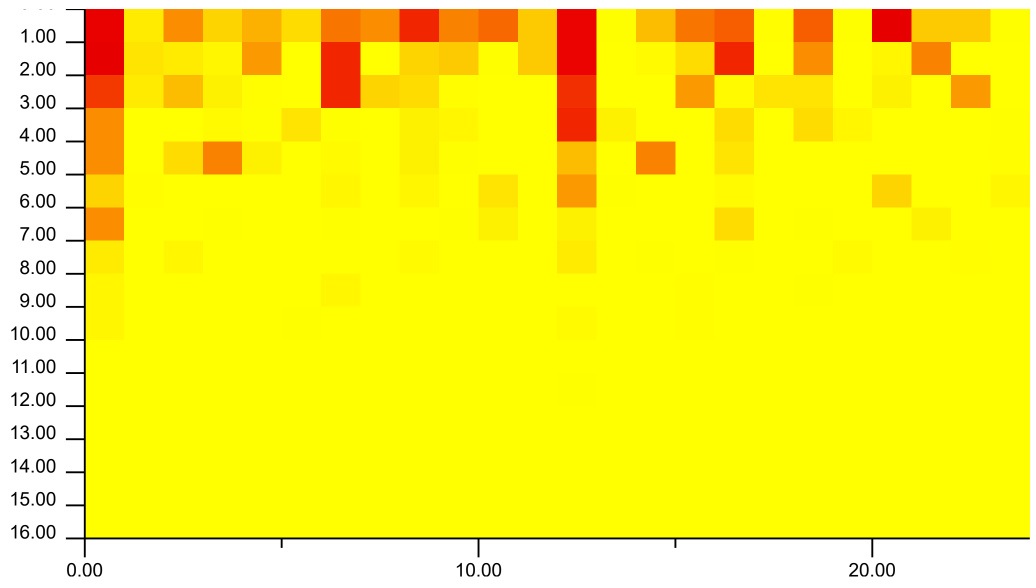

Figure 1: Heat graph displaying note start points

Overall the feedback from both the questionnaire and at the exhibition was positive

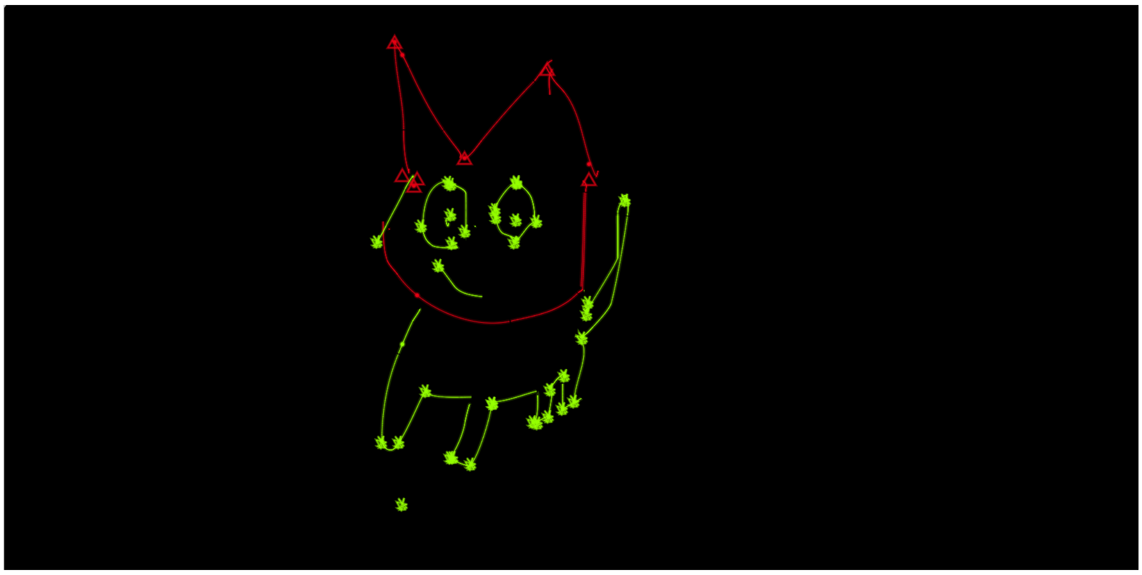
Figure 2: Some exhibit participants managed to draw figurative artwork!

with users reporting that it was a "fun and enjoyable experience" and that they "... could play with [it] for ages." A number of testers suggested that it would work well for sound design and cartoon sound effects in particular: "Really really fun! Can really see the benefit for sound design type scenarios, animation films and the like." Another recurring comment was that it would be interesting if you could sign your name and see what hear your sonic signature. Unfortunately, the app isn't able to give interesting results in this regard but it did show that users were engaging well with the concept of sonic sketching. Another user commented: "I enjoyed exploring how the different tools affected the audio and I liked trying to layer more and more sounds on top of one another." Again, this shows good engagement while at the same time pushing the prototype software slightly beyond its limits as it struggled to play back the ever increasing amount of audio material. A number of Sonic Sketches are presented that showcase the diverse approaches taken, with some users achieving figurative representations and another managing to (almost) sign her name (fig:user-sonic-sketches).

Conclusion

This chapter presented a critical assessment of the final application that factored in user testing and feedback. Each of the major features of the app were assessed in terms of their success in contributing to the overall application experience. As was discussed, some features worked well and didn't incur any major friction in usage whereas others leave room for improvement. Performance issues were discussed along with some potential remedies for these. Finally, the SUS usability survey was discussed along with general feedback received from users.

Adenot, P. (2017) *Web Audio API performance and debugging notes.* Available at: https://padenot.github.io/web-audio-perf/ (Accessed: 4 August 2017).

Brook, J. (1995) 'SUS - A quick and dirty usability scale'. Available at: https://pdfs.semanticscholar.org/c934/0d7584b8174df15f9296558c0441ca5ba045.pdf (Accessed: 22 August 2017).

Lewis, J. R. and Sauro, J. (2009) 'The factor structure of the system usability scale', in *International conference on human centered design.* Springer, pp. 94–103. Available at: https://link.springer.com/chapter/10.1007/978-3-642-02806-9_12 (Accessed: 22 August 2017).

Sauro, J. (2011) *MeasuringU: Measuring Usability with the System Usability Scale (SUS).* Available at: https://measuringu.com/sus/ (Accessed: 22 August 2017).