T-622-UROP

UNDERGRADUATE RESEARCH OPPORTUNITY

CREATING A MUSICAL INSTRUMENT BY TALKING TO LOADS OF PEOPLE

A GESTURE-BASED LOOPING APP FOR TABLETS

EIRÍKUR ORRI ÓLAFSSON

AUTUMN 2014
BSC COMPUTER SCIENCE
SUPERVISOR: MARTA LÁRUSDÓTTIR

Table of Contents

Goal

The goal of this research project was to make a musical app for the iPad, balancing exciting musical features with a low cognitive load on the user. To achieve this, user testing was an integral part of the process. In addition, the goal was to evaluate the benefits of user testing in the development process.

Motivation

The general motivation for this project was to satisfy a personal desire to combine music with programming. As a musician, arranger and improviser with a M.Mus in jazz trumpet performance, I have over fifteen years of experience playing concerts with bands in Iceland and around the world. I also use a lot of electronics in my work; synths, samplers, laptops and guitar pedals are all part and parcel of my music making practice.

Doing a B.Sc research project in Computer Science seemed like the perfect opportunity to deepen my knowledge of the intersection of these two subjects. Specifically, my idea was to develop a software musical instrument for the iPad.

Another, more concrete inspiration was the simplicity of hardware musical instruments in a software instrument-dominated world. I had recently bought a small hardware loop recorder machine called Kaleidoloop[1] with two knobs and four buttons. It was bright blue and about the size of a cigar box (fig. 1). Despite its limited feature set, it was a lot of fun to play with. I actually still use it during my own performances. In my opinion, this "funness" is a direct result of how these few controls are mapped to interesting audio processing parameters. Noticably, it has no screen.

There are other examples. One is the Yamaha VSS-30[2] "toy" sampling keyboard (fig. 2). This keyboard, from the 1980s, is used quite extensively in studios and stages around the world. It was originally sold as a toy, yet it routinely fetches double its original price on the second hand market. Again, it has relatively few controls and no screen.

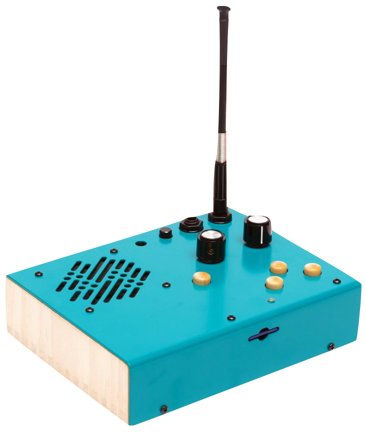


Fig. 1 Critter and Guitari Kaleidoloop

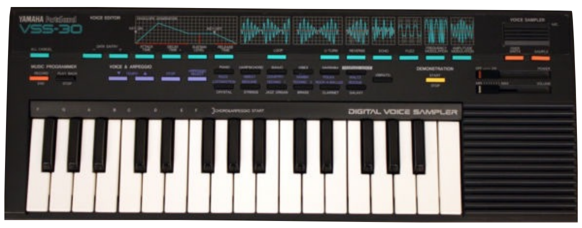


Fig. 2 Yamaha VSS-30

[1] http://www.critterandguitari.com/collections/instruments/products/kaleidoloop

[2] http://www.matrixsynth.com/2012/03/yamaha-portasound-vss-30-synthesizer.html

These two devices seem to enable a way of working that is the opposite of multitasking - single tasking, as it were. They stand out in a noticable way in an environment where music is routinely made on a laptop with endlessly extensible audio software at the user's fingertips. In my opinion, such a multitude of options in every step of the music making process can stunt creativity. It is my belief that with a simple interface and limited options, it is easier for the user to achieve a flow state when making music. Conversely, a complex interface with a high cognitive load can block the user's workflow, leading to needlessly abandoned music making sessions.

Concepts

Flow state: A subjective state where the person experiences the following (*Nakamura & Csikszentmihalyi, 2001*):

- Intense and focused concentration on what one is doing in the present moment
- Merging of action and awareness
- Loss of self-awareness in a social situation
- A sense of being in control of whatever happens next
- Distortion of temporal experience
- A feeling that the activity is rewarding in itself, rather than a perceived end goal

Cognitive load: The total amount of mental effort being used in a person's working memory.
The term was originally applied to learning new tasks[3] (Sweller, 1988), but has found use in software development, specifically concerning keeping contextual information in memory. In general, the more information you ask the user to remember, the harder it is to use the software.

Control Scheme: The mapping of audio processing parameters to user control elements.
An example of a control scheme is an on-screen slider to control the length of a reverberation combined with a start and a stop button to start and stop sound playback. The control elements often display the current state of the parameter being manipulated.

DAW: Digital Audio Workstation
Software to compose, record, edit and mix music at a professional level. Somewhat akin to Microsoft Word for audio. Examples of DAWs include Avid Pro Tools and Apple Logic Pro X.

Overview of existing tools

Software instruments for the iPad are plentiful. They range from disposable run-once toys all the way up to serious apps with extensive feature sets and similarly extensive learning curves. What follows is an overview of some of the best ones currently available.

Samplr
This is the app everybody tells me about when I tell them I'm making a musical app for the iPad. It is a tremendously powerful sampling workstation. The raw materials for sound production are live audio and user-imported sample files. It has eight different ways of playing the sound, ranging form vinyl scratching emulation, to granular synthesis, to arpeggiation. It has built-in effects to further manipulate the sound. Given the extensive feature set, the app does a great job of mapping it to a control scheme.

[3] Sweller, J. (1988), Cognitive Load During Problem Solving: Effects on Learning. Cognitive Science, 12: 257–285. doi: 10.1207/s15516709cog1202_4

The app's promotional copy places an emphasis on being able to interact directly with sound by touching the waveform. In reality, a drag movement doesn't really manipulate the waveform - it moves a playback head horizontally while also controlling volume with vertical movement, which is a similar thing, but not the same - you are directly manipulating "viewports" on the waveform, but not the waveform itself.

The UI design is flat and pragmatic (fig. 3), reminiscent of Teenage Engineering's OP-1 synth, with vector fonts. To my eyes, it looks great.

There might be six or more loops playing back at any time, and each loop has its own "tab", wherein the UI updates to display the current loop's settings. This means that it's quite easy to lose track of what is happening in which tab. This keeping of state in the user's memory is typical of what increases cognitive load.

The app has a loyal following and a vocal user base, and it does what it does very well. The learning curve and the cognitive load is a bit on the heavy side for my tastes, though, and the app has a different set of use cases in mind.

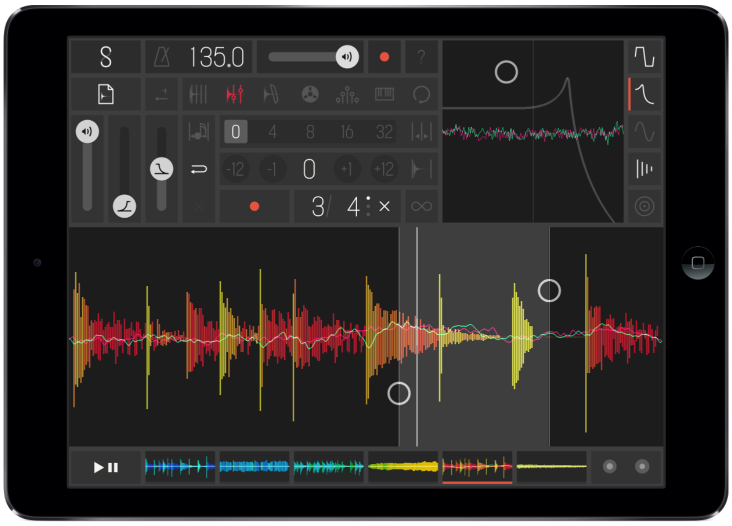


Fig. 3 - Samplr

Garage Band
Apple's Garage Band is one of the most fully-featured and ambitious iPad apps anywhere. It is a fully featured DAW, containing synths, samplers, multi-track recording and effects. Among its built-in instruments is a sampler. It has most all of the features needed for a simple sampler. However, it is a modal view on top of a DAW-style recording channel matrix. Therefore it feels like it should be dismissed at any point in order to get to the "real task at hand", namely to manipulate recordings of myself playing the instrument. I'd love it if the sampler were a standalone app.

SampleToy
This app has very limited scope, which is a good thing. It allows the user to record a small sample of herself and then play it back by holding the finger down on a 2D grid. By dragging the finger along the grid, the user can change the playback speed and effect intensity. The app looks rather dated, but the functionality is good.

iMPC Pro
This app is an attempt to continue the legacy of the Akai MPC sampling drum machine hardware on a software form. It is a whole production studio in an app and has all the necessary functionality to compose full songs, such as a sampling mode, mixing, sequencing and song structure editing. To me, this app is incredibly rich and I could foresee using it to create music for myself. But a "simple" instrument this is not.

Development process

The development process took place from August to December 2014. As can be seen on the timeline in figure 4, the process consisted of a requirements gathering phase, followed by four testing and refinement cycles. The testing methodology and the nature of the prototypes being tested reflected the status in the development process.

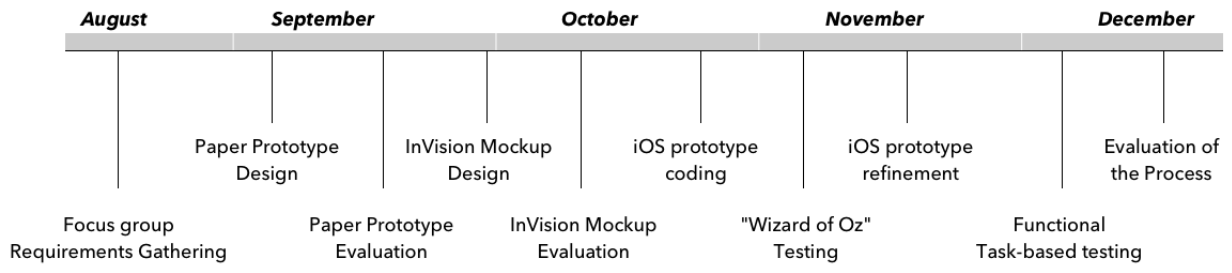


Fig. 4 - The Development Timeline

Original ideas and requirements

Originally, the idea was to make a prototype of a sampling app for the iPad. A sampler is one of the most common types of electronic instrument. Samplers first found common usage in the early eighties, mainly through hip hop music, but have since expanded to almost every kind of music being produced under the sun. Whereas synthesizers use basic building blocks such as sine waves as their sound source, samplers can use rich waveforms from real life recordings such as a violin, organ, a barking dog or a drum beat as a sound source. They are usually played with a keyboard, but can also be played through drum pads or sequencers.

In the summer, I had a focus group meeting with a few electronic musicians to talk hypothetically about making a sampler app for the iPad, and what the requirements for such an instrument might be. We had an interesting discussion, and I gathered a basic requirements list:

1. *Record a sound with the internal microphone*
2. *Play the sound with an on-screen keyboard*
3. *Add effects*
    a. *Reverse the sound*
    b. *Distortion*
    c. *Loop the sound*
        i. *set start and stop points of the loop*
        ii. *toggle between forwards looping and boomerang looping*
4. *Save the sound with a name*
5. *Load a previously saved sound*

Soon after, I got the idea that it would also be fun to make a looper. A looper, in its simplest form, is a device that enables sound on sound, like a tape loop that continually overdubs itself. It is often used by musicians who want to record multiple layers of themselves playing different parts, such as a guitar player first playing the bass line, then overdubbing the chords, and finally playing a melody on top.

Loopers have long been close to my heart, and I've been using them for at least ten years in my practice as a musician. I wanted to make one that closely resembled my current live electronics setup, consisting of a laptop running Ableton Live and a MIDI controller. I would need two looping slots, a record and overdub button, a reverse function, a filter effect and a way to switch octaves.

In addition to having traditional on-screen controls, I was also looking for a way to control the looper without looking at the screen. In effect, it would require a "blind mode", ie. a black screen, used as a canvas for gestures to control the app. The motivation for this is that it is often distracting to see musicians play with their heads buried into a screen. Additionally, as a performer, you lose contact with your fellow performers and the audience when your attention is focused on a screen. Thirdly, it is just plain exciting to see how far it's possible to take an iPad app that doesn't depend on the screen as the main user feedback mechanism.

In this scenario, the control scheme would need to be completely gestural. A friend of mine, Páll Ivan Pálsson, came up with the brilliant idea of using one finger gestures to control loop number one, and two finger gestures to control loop number two, and so forth. A fertile mind, he didn't think much of it, but I loved the idea and decided to continue on that path.

Paper prototype

Software is the world's most concrete and precise way of manifesting abstract ideas. You can achieve almost anything you want. This is great news, but despite the endless possibilities, the burden of actually having great ideas still rests on the person or the team creating the software.

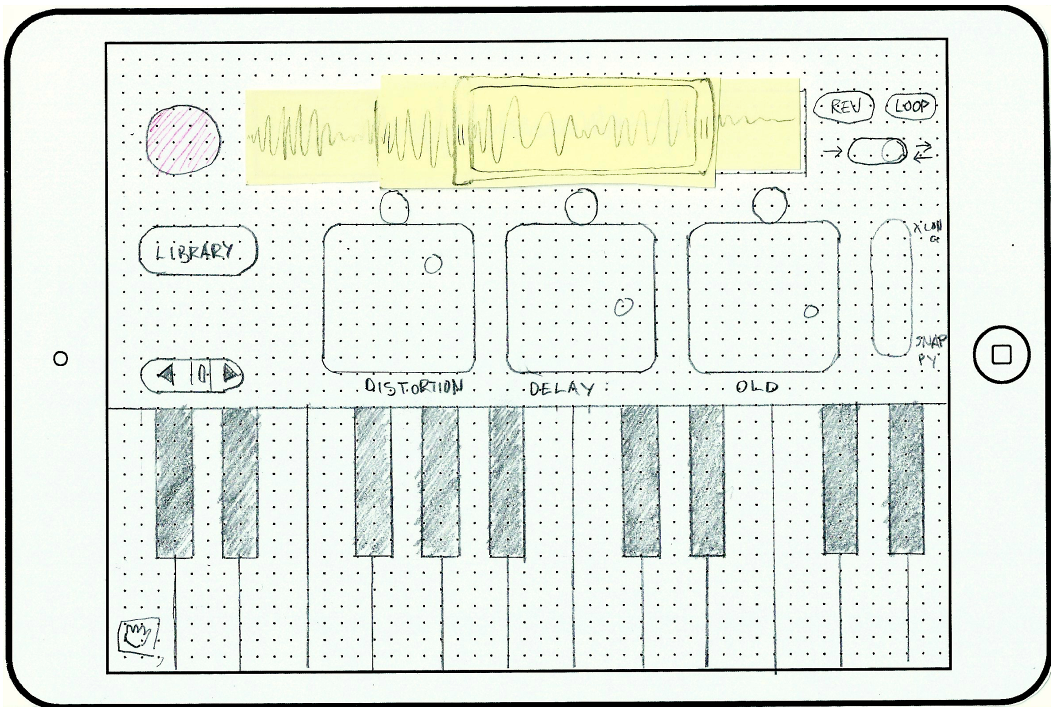


Fig. 5 - The sampler

Luckily, we can use methods and techniques to test our ideas in order to verify that they aren't totally vague and unrealistic. One of the cheapest, fastest tools to do so is by using pen and paper.

I sat down and drew some sketches, using an iPad mini stencil printout. The UI was an implementation of the requirements list. Since there were no precise UI demands to go by from the requirements gathering, I could basically put the UI elements wherever I wanted and in whichever configuration I deemed best.

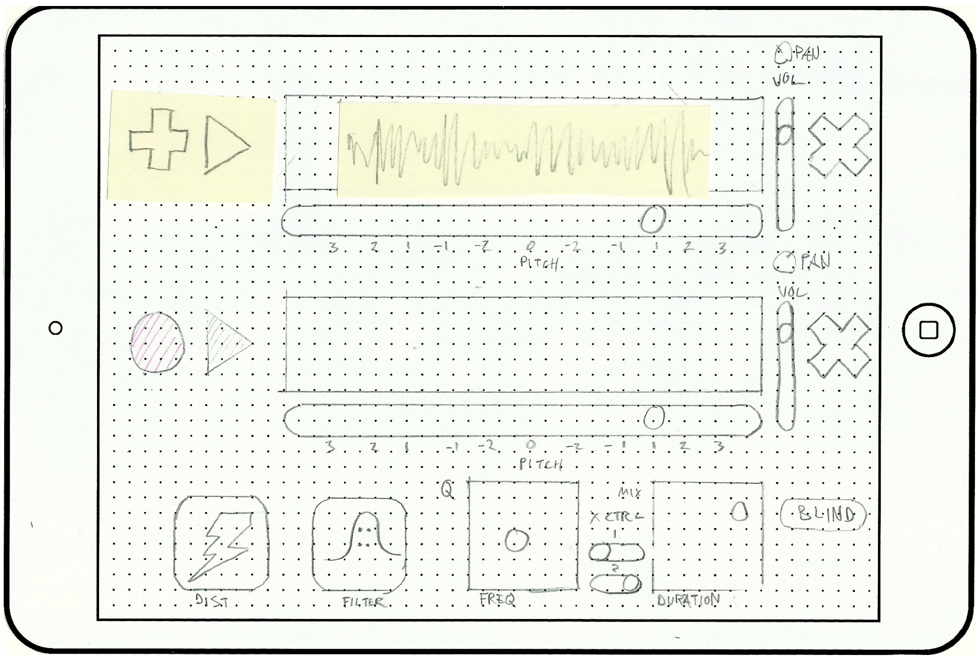


Fig. 6 - The looper

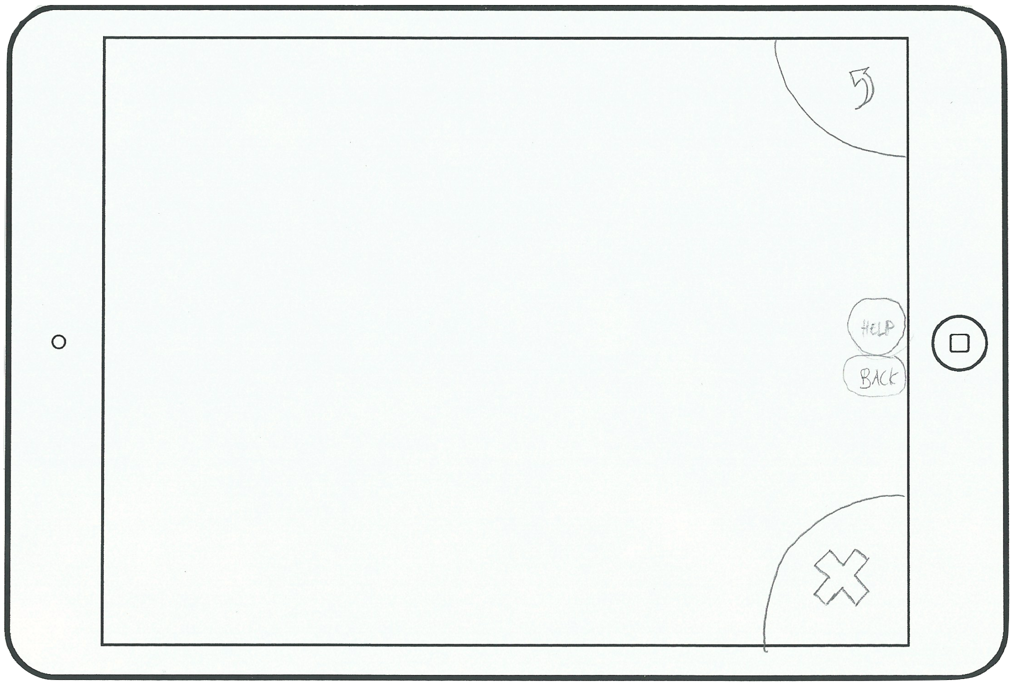


Fig 7 - Blind Mode

I created one for the sampler (fig. 5) and one for the looper (fig 6), simulating button presses and changes in states with post-its and cut-out dialog boxes. The sampler had a single screen UI with modal dialog boxes. The looper, however, had two screens - a traditional UI, and then a "blind mode" (fig. 7), that led the user to a black screen which acted as a gesture canvas.

Evaluating the paper prototypes

Having created the paper prototypes, I wanted to verify if the UI was indeed making sense, and that the feature set was still necessary and required.

I met up with two prominent Icelandic musicians to show them what I had come up with. Putting it bluntly, what I subconsciously wanted out of this first testing was to reinforce that my ideas were in fact a good starting point and maybe get some nudges in the right direction from them.

I started out by showing them the sampler prototype. I walked them through the workflow, and they had some tips about features and implementation. I indicated that this was only a prototype, but their comments tended to focus on the implementation as it was, and not so much about abstract possibilities beyond this particular implementation. In retrospect, I realize that my prototype was perhaps too fancy and finished, even though it was just drawings on an iPad prototype stencil. Scribbles on napkins might even have worked better as a way to fuel open discussions. Furthermore, I was tacitly eager to convince them that there was in fact nothing wrong with the prototype. The ego and the will to "succeed" is a strong force indeed.

The sampler is in many ways inspired by the Yamaha VSS-30. This sampler is one that's used extensively by these two musicians, so this framed the app's concept a bit. The only thing they really wanted to improve in the Yamaha keyboard was the ability to save sounds, and to extend the maximum envelope release time to extreme amounts. This would be no problem to implement, so they were pretty happy about the prospect of the app being developed.

Next up, I showed them the Looper prototype. The blind mode immediately caught their attention. After shortly explaining the gesture language to them, they were very excited about it. In short, they recommended going forward with the looper app. Key to the recommendation was that the features of the prototype sampler didn't add enough usefulness to replace the Yamaha VSS-30. Also, they felt like the blind mode was novel and exciting, especially considering that it used the iPad hardware in a much more meaningful way.

Results

When considering the methodology of the testing process, there are some key takeaways from this first test: If you want valuable insight and actionable results, rather than just feeling mildly good about yourself, you have to work hard in order to create a situation where people can talk freely without fearing that they are diminishing somebody's hard work. But it was still a very valuable test, and it got me enough feedback to continue.

After thinking about the user feedback, I decided to discard the sampler idea and continue with the looper prototype. Wanting to get further confirmation of the validity interface, I decided to make an interactive prototype of the basic navigational structure and recording functionality of the app.

Interactive Prototyping with InVision

After reading a glowing article[4] on Wired about the software prototyping tool InVision, I decided to try it out. It works as a highly context-sensitive slide show that can work on the iPad or as a web page. You can click buttons and thus simulate navigating through your app in a surprisingly good way. I spent half a day creating a nice mockup modelled after the earlier paper prototype (fig. 8). I had a wish to see the app running on a device, thus making it a bit more "real". Now I could test if the functionality and navigation were soundly implemented.

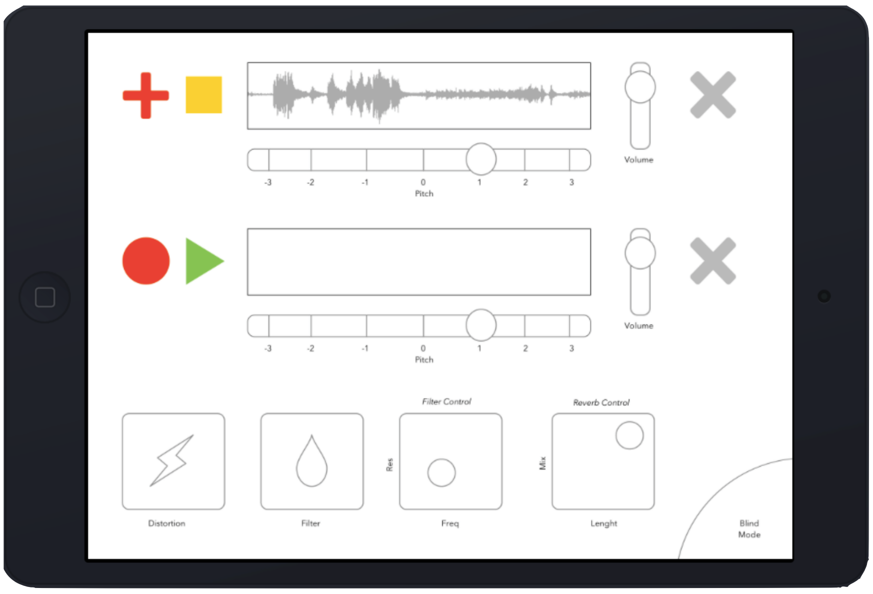


Fig. 8 - InVision prototype

Evaluation

I tested the prototype on a multi-disciplinary violinist by asking her to use the prototype freely while saying what she was thinking out loud. The testing was done in my living room. She spent some time navigating the prototype and observing the change in recording status and how to clear a loop. After the think-aloud testing, we had a discussion about the prototype. She was used to looping, and thus the intended functionality of the app. She didn't find any major problems with it.

Results

The test validated my ideas about the user interface, but I might have saved myself some time by just using pen and paper. It was good to get to know the tool, though. I imagine it could be useful in a situation where it's important to test the actual look and feel of the interface ("that button should be brighter!"), as opposed to rudimentary navigational structure.

[4] http://www.wired.com/2014/09/ux-app-thats-driving-design-everywhere-airbnb-zappos/

Not having discovered any drastic problems with the interface and functionality, I decided to start implementing the app in code.

iOS app prototype

As of July 2014, Apple supports two programming languages for creating apps for iOS: Objective-C and Swift, a brand new language. After watching Swift come out of beta in September, I decided to use it to develop the app. Apple also offers a new audio framework called AVAudioEngine that has high level support for many audio processing functions. It seemed to offer an easy way to implement many of my desired functions, so using it was perfect.

I used an Agile methodology for developing the prototype, keeping track of user stories and tasks on an online Scrum board. After two sprints, the prototype had playback functionality for two loops, as per the requirements. The loops could be controlled both from the normal mode and the "blind mode". In "blind mode", dragging along a vertical axis controlled the volume, and dragging along a horizontal axis changed the pitch+speed, or speed alone. I didn't have a set idea about wether these pitch parameters were the right choice for the vertical axis, but they seemed to be interesting parameters to control - they allowed people to get a vivid idea of what was being controlled. However, recording functionality proved to be a tough problem and was not ready at the time I decided to do user testing. Instead, the app launched with two preset loops.

Wizard of Oz Testing with iOS app prototype and Ableton Live

After having a functional prototype running on the iPad, I was excited to test it. Now that the validity of the user interface had been established by earlier tests, I was interested in the intricacies of the gesture language. Would it work to hold a finger on the iPad to record and then release it to start playing back? How about reversing the loop by swiping to the left?

With these questions in mind, I decided to conduct a Wizard of Oz-style testing. Normally, in a Wizard of Oz test, the tester (unknowingly) uses a "fake" app, with his actions being made by a secret operator behind the scenes. But this felt a bit dishonest and frankly unnecessary. I decided that I would actually tell the user that I was going to mimic his actions using a laptop, sitting next to him.

I contacted another electronic musician and we met at his house, where I set up my laptop and a small microphone. I gave him a short demo of the app's functionality and an explanation on the gesture language. I then asked him to put the iPad on the piano (with the screen asleep) and pretend to use the app in blind mode. I then did the equivalent sound processing in real time with my laptop, running Ableton Live.

Result

In reality, the Wizard of Oz testing didn't go so well - he had seen the app running and tried it out himself before the test. After that, it was kind of hard to mimick or pretend using the app. In any case, it obviously felt unnatural to him and we aborted the testing after a while.

Instead, we had a great discussion about how the parameters should be controlled and we came to a conclusion about the gesture language. In particular, I had intended recording to start when the user held his finger on the screen. He made the point that it's better to tap once and then have your hand free to play the instrument. Also, the pitch was continuously changing across the horizontal axis, like manually spinning a vinyl record. He pointed out that maybe it would be better to have the pitch move according to a musical scale, or even octaves.

I decided to take his advice to heart and so the next step was to implement recording functionality with the gesture language we discussed.

Refining the iOS prototype

I tried several approaches to implement recording functionality, and in the end I found a way that worked. It turned out that the new AVAudioEngine framework was perhaps too new to be completely ready. In any case, I could now record sound for both loops (but not overdubbing, alas). Because of the lack of overdubbing, the gesture language could not be fully realized. I decided that I would test it out regardless.

Task-based prototype testing with multiple users

I was curious about how the recording functionality would come into play. This was the first time that the app felt fully "connected" from input to output. I also had questions about the usability of the app, and the "blind mode" in particular. Was it too abstract? Would users get lost? Would it be necessary to involve visual feedback?

To get answers to these questions, I constructed three "tasks": one simple, one medium hard and one complex. I also made a list of questions about the app, ranging from UX specific ones to more market oriented questions. The description of the tasks and the questions can be found in the addendum. I then advertised for testers online, in a Facebook group for studio and electronic gear heads. The response was good, and I could get seven testers in total to come in to my house for coffee and to give the app a whirl, attempt to complete the three tasks and answer the questions.

Results

The test revealed that I essentially need to make three "blind modes" - one with a more "normal" effects palette, one with the pitch controls and more esoteric effects, and lastly a user-configurable blind mode, where the user can pick effects à la carte. Almost nobody cares about the other, normal visual control mode. The gestural language makes sense, but some visual feedback is necessary in order to diminish cognitive load on the user. Detailed results from this test are to be found in the addendum.

What I also learned from this final test in the project was that task-based functional testing can reveal problems you didn't anticipate. Somehow, discovering bugs like that feels delightful (for example, one user attempted to use the app in blind mode while the iPad was actually turned off). I don't know how to describe it, but it's as if the project has a personality of its own, revealing its face a little bit more with every bug and every corner case.

Additionally, I was caught off-guard by how much interest people were showing. By talking to them, they became less of an ambiguous, archetypal "user" and more of a stakeholder, or a real life person who *wanted* things from app and cared about how its development played out. My role subtly changed from sole developer to a steward of a shared vision. It was a beautiful feeling.

Next steps

For the next sprint, I want to involve visual feedback and implement an overdubbing feature, so that the gesture language can be fully tested (a FSM of the proposed gesture language can be seen in the addendum). I will then run a similar test to the last one, ie. a task-based functional test with plenty of room for discussion. I would also be interested in testing the app when people are explicitly looking away from the screen, as that was a big part of the motivation behind the app. Also, I will try to formulate good metrics to test for the cognitive load of the app, and if the app leads the user into a flow state.

After that, I will continue to refine the app, examine corner cases, and involve a graphic designer to design the look of the app. And of course, test some more. I will need to come up with a proper name for the app. Then I will do some more tests, and in the end I will publish the app on the Apple's App Store.

Evaluation of the method

User testing, when done right, is a great way to speed up development. It might sound counter-intuitive, but it should be valued as just another tool in the developer's toolbox to make him get to his goal, faster. I went into the project with a vague feeling that user testing was on the other side from the "hard" programming part - that user testing required "soft" skills. That might still be true, but it's a rather pointless distinction to draw when trying to create good software. Instead, it's more helpful to distinguish between techniques and methods that get the right thing to users in a shorter amount of time, and the ones that don't. Hard vs soft skills be damned, we need them all in order to create great software.

What I found most useful is to get a feel for what it really means to test. Testing is a skill in itself, and I think that I am better equipped to do better tests after this. First of all, you have to have a clear idea about what you are testing for. You must have a hypothesis. Secondly, you must create an atmosphere wherein the tester feels that she can speak frankly.

This is the hardest problem. I found that I myself got stressed and perhaps overly formal when conducting the tests, and that may have interfered with the atmosphere in some way. Maybe that is something that can be overcome with practice.

At this stage in the development, the things that were most valuable for me were the discussions after the test. As the app is still rather open for changes, discussion can be the most open and fluid form of feedback.

I know which forms of tests to avoid. For example, the InVision interactive prototype was not really needed. I was testing for basic app structure, not button hues, so that was an example of the wrong type of test for that specific time. Also, I made the mistake of having my paper prototypes too refined when all I wanted was general feedback on the viability of the app.

So, to summarize, I had a tendency to develop solutions first, and then test the viability of the problem. It should be the other way around, of course!

Conclusion

Going through a product and software development process with a healthy emphasis on user testing has convinced me that it is an immensely helpful approach. Without it, I would have been operating under my own unchallenged assumptions for a long time. Also, at this stage in the process, open discussion is an extremely fluid and valuable kind of feedback. Even in cases where my goal with the testing wasn't very clear, discussion always helped.

I look forward to deepen my knowledge of testing, so that I will be able to get even more informative results in the future.

References

Nakamura, J.; Csikszentmihalyi, M. (20 December 2001). "Flow Theory and Research". In C. R. Snyder Erik Wright, and Shane J. Lopez. *Handbook of Positive Psychology*. Oxford University Press

Addendum

Task-based prototype testing with multiple users

Description of tasks

1

- Record loop one and play it back
- Then record loop two and play it back
- Stop loop one
- Stop loop two

2

- Record loop one and play it back
- Speed it up
- Lower its volume
- Stop loop one

3

- Record loop one and play it back
- Record loop two and play it back
- Lower loop one's volume
- Raise loop two's volume
- Slow down loop one
- Speed up loop two
- Stop both loops

Results from the task-based test

	Tester 1	**Tester 2**	**Tester 3**	**Tester 4**	**Tester 5**	**Tester 6**	**Tester 7**
Task 1	✓	✓	✘	✓	✓	✓	✓
comments			Screen was off but tester kept trying to operate the app				
Task 2	✓	✓	✓	✓	✓	✓	✓
comments							
Task 3	✘	✓	✓	✓	✓	✓	✓
comments	Tester got confused about recording/playback state						

Direct questions

Did you feel like you had a clear mental model of what was going on while you were doing these tasks? Or when using the blind mode in general?
Almost everybody said that they had a clear mental model of what was happening.

Would you feel that visual feedback is necessary?
Testers 1, 2 and 7 felt like visual feedback would be necessary in some form, because keeping the state of the app in memory can be quite difficult if you're doing other things. Testers 4 and 5 explicitly like the lack of visual feedback. However, they also suggested that there be a "party mode" wherein the background changes in sync with the parameters changed - it would be impressive for the crowd to see when DJ-ing.

Do you use loopers in your work?
All subjects use loopers, except tester 6, who hardly uses any electronics at all in his work as an opera singer.

How would this fit into your workflow?
Tester 1 loves the idea of the blind mode. He would use the app in live situations for long, soundscape-y loops. Tester 5 would use this for sound design sessions. Others would use it for studio and live situations. Tester 7, noted that, as a violin player, she could not use the app while playing, because she needs both hands to play the instrument. She would like to use it for other instruments, such as the keyboard or while singing.

What would you like to control with the X-Y finger drag movements?
Testers 1, 2 and 5 were both clear about the current pitch controls not being useful. Delay time and feedback came up (also with tester 3), as well as having multiple effect parameters linked to one axis. Other testers rather liked the pitch controls.

How about swipe-ing?
Tester 1 would like to have some way of filtering the sound with a band pass filter, to make the loop fit into the middle of the frequency spectrum, without cluttering the top or the bottom. He suggested swipe-ing up as a way to do that.

Would you like to be able to create more loops?
Tester 2: "I used to think so, but now I can see that it's good to keep it down to two". Testers 4 and 5: Yes, maybe three?

Do you own an iPad?
Testers 1, 2, 4 and 5 don't own an iPad. Testers 3, 6 and 7 do.

Would you use this in your work?
All testers said yes. Tester 3: "Yes, definitely. Also, kids will love this."

Other feedback:
The pinch gesture came up twice. People want to timestretch the sound with that gesture. It might pose some difficulty, because it can break the connection between the number of fingers

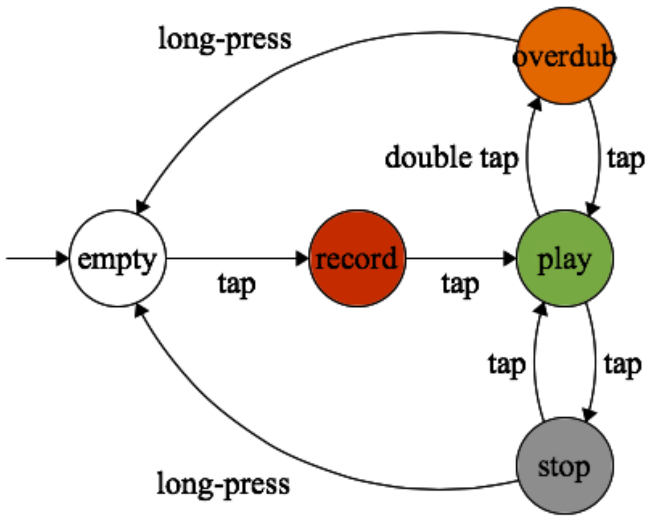
and the number of the loop. Ie. loop 1 would be controlled with one finger, but would need a second finger while pinching. This is intruding on loop 2's territory.

Background of testers

1: Double bass player with classical background but does a lot of work in experimental and electronic music.
2: Multi-instrumentalist and prominent electronic musician. Power user and virtuoso with electronic instruments. I would call him "the synth whisperer".
3: Sound engineer and software engineer. Sometimes takes on work in sound effect creation and post-production.
4. Composer and french horn player. Makes multimedia art.
5. Composer, trombone and clarinet player. Makes multimedia art. Is also a DJ and makes electronic dance music.
6. Opera singer. Likes electronic music production as a hobby.
7. Multidisciplinary violinist. Makes multimedia art, plays classical music, improvises and uses electronics.

Gesture language FSM diagram

This is the proposed gesture language as per the results of the task-based user testing. The colour of the states reflect the visual feedback that needs to be implemented in the next version. The current idea is to let each loop have a "neon light" underneath the left and right edges of the screen. The light will let the user know which state which loop is in. Notedly, this FSM lacks swipe movements. I will worry about them in the next iteration.


long-press
overdub
double tap
tap
empty
tap
record
tap
play
tap
tap
stop
long-press