(This document is written with the primary intention of describing the progress I've made so far to someone at RCOS who may take over this job in the future.)

My primary objective is to repeat the last callout that occurred when a user shakes their device. This involves two things:

1. Being able to trigger an event when the device is shaken
2. Repeating the last callout

Part 1 was pretty easy to implement. The program already has the relevant information to do something when the device is shaken. This most importantly involves the motionEnded function which is overridden (pretty standard from tutorials I've seen). On XCode, it's really easy to look for a particular keyword. On the left menu, which is commonly used for files, there is a tab which contains a search bar, where you can search for any keyword. Putting in motionEnded quickly led me to the appropriate section to put the code. I then put code to detect a motion shake and made a message get logged to the console whenever the device was shaken. Part 1 is completed - an event is triggered when the device is shaken.

Part 2 has been much trickier to implement, and, as of writing this, is currently not implemented. Implementing this part has required me to establish a thorough understanding of how sounds are processed in soundscape. While there are functions in Swift that can call tts audio, like .speak, Soundscape uses a far more complex approach to sounds, though it's this that allows it to broadcast more complex sounds, like 3D sounds. This 3D sound is significant - especially to someone who is blind. For example, when using headphones, using the app in Amos Eaton when Lally Hall is on the right will play the sound primarily on the right headphone. Moving the phone in real-time (for example, flipping the phone so Lally Hall is now on the left) primarily plays the sound on the left headphone.

The callout system also takes advantage of a queue, which ensures that only one sound is played at a time. All the text is put into a CalloutGroup, an object which is then passed to an AudioEngine at the last moment for rendering. The CalloutGroup has the potential to change while sounds are rendering, so by rendering the sound right before playing it, the most up to date information is presented to the user. Although many locations are presented, there appears to be a cap on how many locations are called out based on count and distance (count seems to be around 5, distance.. around a mile?) I don't know exactly where these numbers are stored.

The code also seems to utilize 'generators' as a part of the sound playback process, of which there are automatic and manual generators. Automatic generators generally queue up instead of interrupting and are updated based on things like user location (I'd guess that the sound moving to the other headphone based on how I adjusted my phone earlier was also a product of an automatic generator). Manual generators, represent user actions and tend to interrupt automatic generators. These generators are utilized in classes, which contain CalloutGroup objects.

My interpretation of the above request is to repeat the last callout even if current callouts are being called. For example, if callout1 was “Lally Hall around 100 feet”, callout2 was “Sage Lab around 200 feet”, and callout2 was speaking, shaking the device would interrupt callout2 and instead play callout1 (which played just before callout2), before playing callout2 from the beginning. Shaking while callout1 was playing would just restart callout1. If callout2 was the last callout, shaking should play callout2. Now note that callout1 is (I assume) based on an automatic generator. But it needs to interrupt callout2 before playing. In the AutomaticGenerator, there is canInterrupt variable, so before playing callout1, that should be set to true. This should allow callout1 to properly interrupt callout2.

How would I implement this? To start, CalloutGroup objects are generally located in classes that utilize generators, so by convention we should store callouts there. We could create a new variable that will store the information of the last CalloutGroup played that gets updated whenever a callout is finished playing. We then need to tell the audio generator to stop playing the current callout (if any) and play the last callout upon shaking the device.

While I haven’t been able to really implement something so far in this part, I do have a few ideas on things that might work. We could create a new variable that will store the information of the last CalloutGroup played. We then need to tell the audio generator that, when the device is shaken, to play the callout stored in that variable. The part that I’m struggling with is where to place that code and how to write that code.

Swift coding, despite being fairly high-level, is unlike traditional languages like C++, Java, or Python in its syntax and the things you can do with it, so stuff doesn’t stick out as easily. There are also lots of custom-defined objects which, most likely because of Swift and their complexity, doesn’t make it very clear to me what each line of code does.

However, one thing that has helped has been communicating with people in the Soundscape Slack. Here, RCOS students can communicate with other active developers. If Wes Turner is still at RCOS, ask him to add you and he should be able to honor that request. The expert on the sound side of Soundscape is Daniel Steinbrook, so if you work on this issue, send him a DM if you have any problems. I’ve been exchanging messages with him and he’s given me great advice on how the sound system works, much more information than me just poking around the code. If there’s one thing you should do after reading this, it should be joining the Soundscape Slack.