ShortSounds Code Review Analysis

In our code review, we looked over the code that implemented the effects. This included many different classes, including ReverbEffect.java, EffectController.java, EqEffect.java, Effect.java, parts of ShortSoundSQLHelper.java, parts of MainActivity.java, ReverbEffectController.java, EQEffectController.java, Fx_EQCanvas.java, and Fx_ReverbCanvas.java. These files were primarily written by the effects team (Seth, Nick, Casey), and Mattie reviewed them.

One of the things that we learned from the code review is that a lot of the classes were not very readable for a variety of reasons. Many methods within these classes had incomplete comments or no comments at all. This made the code really hard to understand without an outside explanation. Another thing that was prevalent throughout the files was the existence of leftover debugging techniques, such as logs and commented out code that is no longer necessary, which made the program look much more complicated and unclear. It was noted that the variable names that were not very descriptive. For example, in Fx_EQCanvas.java, there were points named lp, cp, and rp which were confusing. In addition, there were many "magic numbers" that should be made into constants throughout the different classes. There were inconsistencies in style, such as starting some method comments with the double slash, but using the multiline comment specifier for other method comments. These issues were fixed throughout the files, making the program much more readable.

In addition to increasing readability through various styling techniques, some of the code had to be restructured as well to make more sense. Inside of Fx_ReverbCanvas.java, there was a set of 3 or 4 lines that were repeated in three places and these lines were factored out into a method. In the EQEffectController.java and the ReverbEffectController.java, there was a method that always had to be called after the constructor, so that functionality was added to the constructor. There were also methods in these two files that simply repeated tasks that were already done in the constructor, so these were eliminated.

There were other issues brought up about the structure of how the effects were implemented and controlled. For example the effects UI is all controlled by a view animator that is used in main activity. This helped eliminate the problem of having multiple fragments with lifespans that could cause buggy audio code, but it also created its own problems with the architecture. MainActivity had knowledge of all the views instead of each view being in its own context. With this there was some code that had somewhat bad style of implementing a global controller that updated the effect models on the backend when the back button or cancel button was clicked. This was noted, but was agreed that it was the best design decision, given the view animator architecture we are working with & time constraints to change such architecture. This architecture also caused some poor style in the canvas classes because some point values could not be instantiated in the constructor. This was also because the two effect views are globally held by MainActivity and simply given different values when switching between track's effects. This means that these points should not be reset in all cases when switching views. This was also noted, but kept the same.