COSC345 Assignment 3 - Beta Report

August 13, 2018

Group Members

Anthony Dickson - 3348967
Rory Jackson - 2208377
Johnny Mann - 3891999

Progress

We have made some progress since the alpha version (assignment 2). For the beta version, we had aimed to finalise the UI, create the exercises, and add an exercise progression system. However, we are unable to deliver what we had planned. Not much progress has been made with the UI since the alpha version. We have made progress in creating a framework for the exercises, and we should have this working soon. As a result of the difficulty we have had to deal with implementing the exercise framework, we have also not been able to implement a progression system, since that would rely on the exercises working.

Once we get the exercises working, the rest of the application should be relatively easy to implement. The biggest roadblock we are currently working on is related to the way in which our application is processing audio input. The rate at which it gives readings for the pitch of the user's voice is not frequent enough for it to work with shorter notes (for example, if we wanted to process a second of audio where the user sings multiple notes, our application would currently only give a single reading for the entire section of audio instead of one reading per note). This is due to the way that the open-source FFT (Fast Fourier Transform) algorithm we are using works. Since we did not write this code ourselves, we are having difficulty figuring out how to modify it so that it does what we need it to. We estimate that it will take about two weeks to get back on schedule.

Issues and Solutions

We have encountered a few issues in regards to working as a group between the end of assignment 2 and now. The main issue is the time it took for us to begin working again. Most of the work took place within the last few weeks leading up to the due date of this assignment. This issue stems from multiple smaller issues as described below.

Distribution of Tasks

Firstly, because we decided to assign different aspects of the programming to different people (i.e. Anthony was meant to work on the UI, Johnny and Rory were meant to work on different aspects of the exercises), some of us had to wait on others to make progress before we could begin working on our allotted tasks. This meant that instead of working concurrently, which would have been far more efficient, we ended up each working at different times. The structure of our application is such that some parts of the code need to work before others can be written (for example, we need to get it to play notes and take in audio input before we can fully implement exercises, and we need to fully implement exercises before we can implement exercise progression, and so on). This makes it difficult to divide tasks among the team. A solution to this problem would be to figure out how to be less strict with our roles while

maintaining as even and fair of a split of the workload as possible. Since exercises need to be implemented before progression can be implemented, for example, we could all work on different aspects of the exercises, and then all work on different aspects of progression, etc, rather than one person working on exercises, and then one person working on progression.

Dealing With the Unexpected Absence of Group Members

Secondly, as a result of this compartmentalisation of roles, we ran into issues when group members were relying on another group member who did not have the ability to complete their work. For example, one member, who was responsible for adding some specific functionality, got quite sick and could not work for an extended period of time. The others were relying on this functionality to implement other parts of the application. As such, the other members of the group could not get any work done, because they needed this functionality implemented in order to begin working on their respective tasks. We did not have a plan in place to deal with any kinds of illnesses or emergencies, and as a result we just had to wait for these issues to pass before we could continue to work. We learned about this in last semester's lectures: it is important in the industry to have contingency plans in place and distribute the workload as much as possible to avoid these kinds of pitfalls, because they can be very costly.

A potential solution to this type of issue would be to distribute work within each class, or specific part of the application. This would minimise the damage caused by an illness or an emergency, as there would still be progress made on a particular task despite someone not being able to work. The only other option in this particular situation (in regards to the necessary linear progression of work) would be to redistribute the workload when these situations arise. The problem here would be the time it would take to become familiar with what the other person was working on. Our particular application makes this a tricky situation to deal with, but having now dealt with it, we have learned more about the responses that work and the responses that do not work. Another important issue this raises is our lack of a leadership role amongst the group. We have been fairly democratic in terms of our direction, and we have decided as a group what to work on. When we ran into these issues and delays, we could not decide how to solve them because not everyone could contribute to the discussion. One of us will act as more of a leader for the remainder of the year, so that the tasks and deadlines are more deterministic.

Agreeing on Implementation

Thirdly, we still had not completely come to an agreement in regards to the structure of the application. We had to have a couple of meetings to discuss further the way in which particular parts of the application would be implemented. This is due in part to some members of the group not completely understanding the application and also due in part to minor disagreements in terms of how specific parts should be implemented. Both of these issues may have stemmed from the difficulty of explaining code, and differences in design philosophies. We found that we would often think we had differing opinions as to how exercises should be implemented, but after fleshing these discussions out, we realized we were more or less arguing the same points. This could be fixed by using more diagrams and other visual methods of communication. When it comes to object-oriented programming, seeing visually how objects relate to each other makes it far easier to comprehend than trying to vocalize abstract concepts. When we began using diagrams, we discovered we actually agreed on the method we should use, so we know to use diagrams more often.

Getting Familiar With New Code

Finally, there was a steep learning curve when it came to some members familiarising themselves with the latest version of the application. This is due to the progression of code between assignments, and the fact that most of the code for assignment 2 was written by a single member of the group. Again, as a result of the compartmentalisation of tasks, some members of the group were waiting for further parts of the program to be implemented before they could begin their own work. When the above explained combination of factors led to delays, these members of the group were forced to wait for quite a long period of time before looking at the code again. This meant these members familiarizing themselves with significantly different code at the last minute. A solution to this would be for more regular communication between us as progress is made, keeping an eye out for new commits to the project repo, and to make sure each member knows about the new features. This would also be less of a problem if we solve the compartmentalisation issues, as there will be less time between tasks for each member.

Summary

There are several things we need to do to prevent these problems from coming up as we come to the end of these assignments. These solutions will be co-ordinated by Johnny, who will enter a leadership role. This will involve organizing a regular meeting time, checking each member's progress, setting deadlines, and making sure that there is enough work for each member to do regardless of the progress of the other members of the group. We recognize that we could have done several things differently and made more progress, and we fully intend to be more efficient.