Shawn Huntzberry

Artificial Instructor

Expected Demo

By December I expect the “Artificial Instructor” to have the majority of user input functionality completed. First, the program should be able to ensure that the user is connected to the system through some form of direct audio input. The available options are Built-in Audio Input or some form of external input. To do this the user will select the form of input through a series of buttons and the program will attempt to connect. If the connection is successful then it will notify the user and instruct them to tune their instrument, else it will prompt the user that the input is not valid and ask for another option from a list of available system resources. Next, the user will tune their instrument by following some instructions from the system to strum certain strings within a certain time frame. By doing this, the system is able to configure the expected values in order to accurately assign note values to the pitches detected. After this, it will allow the user to select a couple of simple lessons.

When a lesson is selected the user will first hear and see the “correct” attempt at playing the lesson. After the user is confident in attempting the lesson they will select the start lesson prompt. This will give a 5 second countdown in order to allow the user to get properly setup to attempt the lesson. While the user is attempting the lesson the system will give real-time note detection and show their rhythm on a graph. Additionally, while the program is executing it will be storing the notes in an array, in the order they are strummed, as well as saving a recording of the user’s attempt. The lesson is terminated by either the user selecting the “quit attempt” button or the time of the lesson as elapsed. If the user quits before the attempt is complete they will receive no credit for the attempt and be given no feedback, but if they make it through the entire

lesson they will be assigned a score on the amount of correct notes in order, the execution of notes at the right time, and the alignment of the correct notes at distinct time intervals.

After a score has been assigned to the lesson the user will have the ability to playback their attempt and the “correct” attempt they heard previously. I do not expect the algorithm for assigning score to an attempt to be completely working by the time of the demo but all other functionality is expected to be properly working. After displaying the functionality described above my demo will be complete and open for questions.

Schedule

- **Week of 10/19:**
    - **Progress:** Assign pitch ranges to note values for standard [E, A, D, G] tuning
    - **Tests:** After expected pitches have been assigned corresponding note values, test the output when different strings, and frets, are strummed. After recording results compare them to known, working, tuners and note trackers to check accuracy.
- **Week of 10/26:**
    - **Progress:** Get the tuning functionality working, or at least started, so that when a user is tuning the instrument the array that holds expected notes will be modified to be able to correctly match notes, even if they are not tuned 100 percent correctly.
    - **Tests:** Tune instrument properly and test by strumming several notes and seeing if actual output of note value is the same as expected value. Next tune the instrument so the notes are both “under-tuned” and over-tuned” and check if the

actual output of note value is the same as expected note value. For example, if the open fret on the first string is tuned to a pitch of 220 Hz when properly tuned the not value of E should be output when that pitch is detected. Next tune so the open fret on the first string outputs a pitch of 210 Hz and see if array is properly modified so it still outputs the note value of E when played.

- **Week of 11/2:**
    - **Progress:** Add the tuning of drop-D tuning[D, A, D, G] so the users have multiple tunings to practice with and the program has to differ between the pitches of the first string tuned to D and the third string tuned to D but outputs the same values as you go down the frets.
    - **Tests:** Tune the instrument to drop-D and strum each note on both the first and the third string making sure that the same notes are output.
- **Week of 11/9:**
    - **Progress:** Attempt to track the rhythm the user is playing at by looking at notes that are stummed over different time intervals. After time stamps can be assigned to the notes try to format them over a standard measure of time(four notes per measure).
    - **Tests:** Record input where 4 notes are played every measure and compare the time intervals. Play simple scale up and down, {A, B, C, D, E, F, G, A} and {A, G, F, E, D, C, B, A}, so I can check if actual time intervals are the same as expected. The time of the notes does not need to be played perfectly but the analysis of the program needs to be able to determine the correct times notes should be strummed in the measure and see how far off the actual strumming was from being correct.

- **Week of 11/16:**
    - **Progress:** Real time play-back of the users playing should be output while the user is playing. Adding the ability to hear the sound while playing will help the user be able to not only see their rhythm and notes while playing but get used to the expected sounds of notes and rate of the rhythm. Additionally, create a start and stop button so the program knows when to allow sound output and when to stop sound output.
    - **Tests:** To test this functionality I will have to simply play the bass after I have selected the start button. If it is working then you should be able to hear the sound while strumming the instrument and should not be able to hear it after the stop button has been pressed.
- **Week of 11/23:**
    - **Progress:** Code the first lesson, a simple song or scale will suffice. Not only does the values have to be coded for comparison but they song has to have the ability to output the values of the notes, a graph of the rhythm, and the ability to be played through the speakers before the user starts the lesson.
    - **Tests:** First select the lesson and preview the expected results to hear what should happen. Listen to the song or scale, watch the graph, and look at the notes that are expected to be played. Next start the attempt and see if the output can be compared and possibly assigned a score.
- **Week of 11/30:**
    - **Progress:** Finally, incorporate all elements into one working program so that everything described above can be done in the basic user interface. To do this, I

will combine all the elements into one program, or one package that interact with one another in order to produce overall functionality.

    - **Tests:** Test each individual component one by one and then all together. If they do not all work together, keep modifying and testing until the desired program is produced.

- **Demo Week**