CS 401 Semester Project

Guitar Teaching Application

Stephen Choi, Bradley Koch, Jason Michl, David Waszgis

Project Idea

Have you ever tried to play the guitar and gave up because you weren't getting it right away? Don't worry; many people looking to take up an instrument give up right away, because of how hard it is to learn. It is very frustrating when you can't figure out what to do or why you are doing it wrong. However, we have a solution.

Our application listens to you play and provides feedback based on your actions. You will be able to see on a screen the notes that you are playing right, as well as the notes that you are playing wrong. Naturally, it will also show you how to play the correct notes. You will be able to learn in the comfort of your own home. Just install our application on your computer, and play your music through a microphone hooked up to your computer. You will be able to do this anywhere you have a computer. Our application is a fun and easy way to learn how to play the guitar.

Currently, the market for guitar instruction consists mainly of personal instructors, instruction videos, and computer programs. While these three forms of instruction may help beginners to pick up the basics of playing the guitar, they are often expensive, or difficult to use.

Learning through a personal instructor is one of the most useful forms of instruction since someone of skill is directly interacting with the student in order to help them improve. While a great form of learning, quality instructors are often difficult to find and pay for.

Another option is learning from instructional videos. These can often be found on the internet or in standalone media formats. Instructional videos tend to be rather inefficient for teaching due to the fact that there is zero interaction between the instructor and the student, which can make it difficult to understand exactly what the instructor is doing as a result.

A third option for instruction can be found in computer instruction programs. Most computer programs attempt to instruct students by displaying instructions and fretwork on the screen. Although they often claim to be interactive, they lack true interaction – they take no input from the user and are more-or-less an enhanced form of video instruction.

Our program takes the positive aspects of these three formats and merges them into a single comprehensive application. It takes input from the user by means of a guitar and microphone, analyzes the input using pitch recognition, and compares the input to the notes the user was intending to play. Users will be given feedback based on how well their playing matched the expected notes, providing interactive learning that current computer based guitar instruction programs fail to give.

We are going to be promoting our application to beginner guitar players. People with little or no previous guitar experience will benefit from this the most, although skilled guitarists will be able to quickly learn new songs through this application. The same interface that teaches beginners how to play can quickly teach experts specific songs. The feedback this application provides will benefit beginners the most, however. It is a visual tool that beginners can use to see exactly what they are doing. It will show on the screen the notes that they played wrong during the course of the song that they just played.

Our group doesn’t have any previous experience with guitar training software, but we are all enthusiastic about the idea. We are up for the challenge, and know we can do it. None of our project members have extensive guitar experience – this gives us the beginner's perspective we need, since we will be seeing the project from our customer’s eyes. Since we will be using our application to learn, we will quickly discover what works and doesn't work; our training program will be better as a result.

Project Approach

Our computer has three major requirements: a computer, an audio input, and a guitar. The exact type of audio input depends on the type of guitar one is using. Acoustic guitars require a microphone for sound pickup, while electric guitars have can be connected directly to the computer, with the right plug adapter.

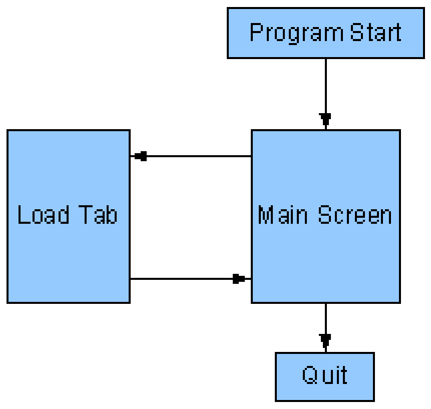
The foundation of our project is a method of detecting the pitch of an incoming sound wave. Our first major milestone is, therefore, developing pitch-detection code. We will test our code by loading audio files and trying to determine the pitch throughout. Once we have know our code works –

that it successfully detects pitch – we will map sounds to individual notes, and then notes to finger placement on a guitar neck. Furthermore, we intend to add real-time functionality to our code, so that notes can be detected even as they are being played.

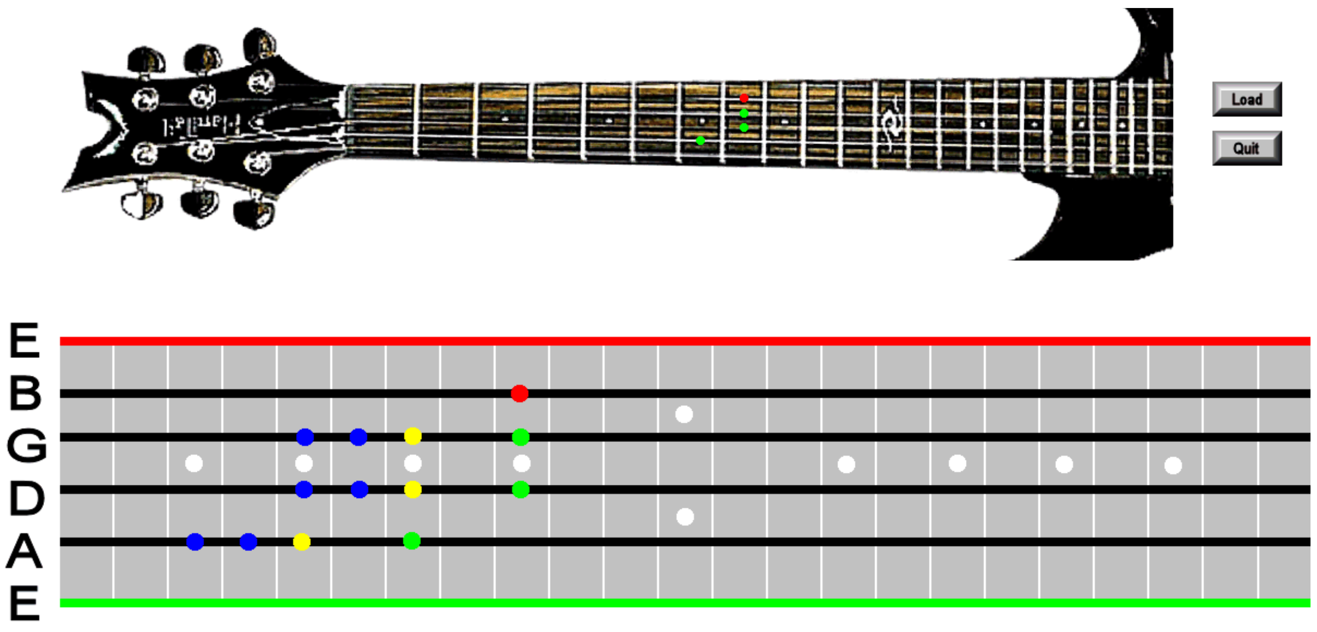
Our next major milestone is intelligent feedback based on incoming sound. To this end, we will construct a GUI that takes the data we've already gained – sounds, notes, finger placement – and display useful information to the user. For example, our program will help the user tune his or her guitar by displaying what note is being played, as well as show where the user's fingers would be placed to play that note on a properly tuned guitar.

The last major milestone applies the information we've gained, and uses it to teach the user how to play actual guitar songs. We intend to create a file format for guitar tablature. When our program reads a tablature file, it will show where the user should place his or her fingers to achieve the desired sound. When the user plays the correct strings, the program displays the next set of notes the user needs to play, teaching the user at his or her own pace.

We have three primary algorithms that will, when combined, allow us to map recorded sounds to fingerings. Pitch detection code is the first step in converting sound to fretwork. In order to do this, we will read a small burst of audio input, and determine its frequency. This will most likely be done through one of the many Fast Fourier Transform algorithm, or a similar method. Once we know the average frequency within a specific amount of time, we can map it to a musical note. For example, middle C is about 261.6 Hz. Other octaves are calculated by repeatedly doubling or halving this number. In this way, we can discover the pitch of any sound the program hears. Knowing the pitch allows us to output possible fingerings for that particular note. This is, again, a simple mapping, that can be adjusted depending on how the user's guitar is tuned.



This shows the general flow of the program. Once the program starts, it immediately goes to the main screen, which is where input is read, processed, and fed back to the user. The main screen has a button that brings up a prompt to load tablature for the user to play. It also has a button that allows the user to exit the program.



This is an example of what our main screen will look like. The fretboard image shows the current tuning for each string, as well as where the fingers should be placed on the neck of the guitar. Green circles and strings indicate that the fingers were placed correctly for those notes. Red circles and strings indicate that incorrect notes were strummed. Yellow circles indicate notes that will be placed next, while blue circles indicate notes that will be played soon. The guitar image, on the other hand, only displays the where the user currently has his or her fingers. The Load button allows the user to read another tablature file, and the Quit button obviously allows the user to quit the program.

Project Evaluation

Our project will be considered a success if we meet three goals:

1. Pitch recognition
2. Tablature translation
3. Proper feedback

First, our program must recognize the exact note the user is playing. This means the system will determine the pitch of a note by the given sound input. At the start of our project, we are concerned with single notes, instead of complicated techniques like chords, hammer-ons, pull-offs, sliding, bending, tapping, and harmonics. Since this project is focused on the beginning guitar player, we will assume that users will play simple melodies. If the note played by the user and the note recognized by the system are the same, then it is a success.

Second, the tablature should be precisely translated. Tabs provide the notes that our users are supposed to play. Therefore, the system should recognize what note should be played, and compare that to the note that actually was played. If the system recognizes correctly played notes, then this goal has been met.

Finally, our program needs to give useful feedback on the user’s playing. An overall result will be determined by calculating judging the skill of the user's playing. This means that user will get 100% if he played the correct note/pitch, and played the right rhythm. The feedback will show the overall percentage grade after it’s done. To define successful feedback, the system should compare the input and the tablature and output the right results. At the beginning of our project, the feedback feature will not be real-time, but it can be added as our program becomes more sophisticated. Since the feedback is not in real-time, it needs to be processed quickly so that users don't have to wait.

In order to test the software a combination of in-house testing and out-of-house testing will be used. The in-house testing will consist of the members of the team thoroughly testing the system to ensure that it is functioning properly. After in-house testing, we will have numerous volunteer test subjects of various skill levels play guitar with the program. These tests will not only help further determine that the software is functioning properly, but give a good measurement of the user friendliness of the interface, as well as how helpful the subjects felt the program was in teaching them to play the guitar.

Test conditions consist mainly of testing the pitch detection, and making sure that it is highly accurate. Additional conditions include ensuring that guitar tablature can be read accurately, comparing the pitch of user input to the expected pitch, testing how easy the program is to use, and demonstrating the effectiveness of the program in teaching someone to play guitar.

We hope to have a test pool of between ten and twenty test subjects. This figure includes team members as well as volunteer test subjects. This should allow for significant testing of core system functionality and user interaction, while still remaining a reasonable amount of test subjects to find.

We have two ways to evaluate our results. The first way is to use a MIDI file. Our results can be evaluated by comparing them with a 100% correct midi file. A working system should hear the MIDI file and judge it to be completely correct. If the system does not do this, then we need to make corrections to the system. We can also modify this MIDI to check various parts of the system, allowing us to build a sufficient number of test cases.

Another way we can evaluate our program is to allow people to play with it. In this way, we can see if the users agree with the results. This approach also allows us to demonstrate the improvements of the test subjects, which is this project's main goal. People can tell us their feelings about the effectiveness of feedback and teaching. This approach allows us to evaluate multiple aspects of our program at the same time.