The ChordMaster

21M.380

Sonic System Project

Author:

Andrew Sugaya

December 3, 2009

1 Introduction

There are many instrument apps on the iPhone, including the Pocket Guitar, MiniPiano, and Ocarina. In my recent case study of iPhone instruments, I discovered that small instruments, like the ocarina, were emulated well on the iPhone.[1] Along with this, I found that instruments larger than the iPhone, such as the guitar, were very difficult to effectively emulate on the iPhone. My goal with the Sonic System Project (SSP) was to create a new iPhone instrument and interface that would be able to emulate a large instrument well. In particular, I chose to emulate the guitar. In this paper, I present a short description of the system, evaluate the project result, and conclude with my plans for the future of the system.

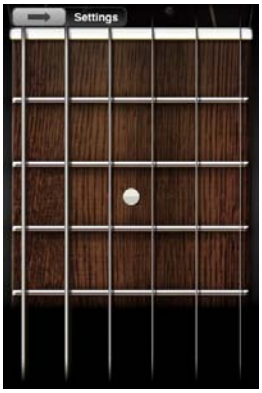
2 System Description

My SSP allows the user to select and play chords by using a strumming motion, similar to the motion used to play chords on a guitar. Fittingly, this system is named the ChordMaster. In the following sections, I discuss the motivation for the design of the ChordMaster, along with the design itself. Then, I present a short sample of my code, which highlights the process by which the application interprets accelerations of the device to detect a strumming motion.

2.1 Design Motivation and Results

My goal with this SSP was to effectively emulate an instrument larger than the iPhone. I chose to emulate the guitar, as it is a popular instrument and the previous incarnation of the guitar on the iPhone, the Pocket Guitar, was especially difficult to play. This difficulty was a result of trying to fit the guitar frets and strings onto a small screen, as shown in Fig 1.

[1] Sugaya



Courtesy of Shinya Kasatani. Used with permission.

Fig 1. This is the interface for the Pocket Guitar, a previous iPhone app that attempted to emulate the guitar. This is rather difficult to play, given the limited screen size of the iPhone.

Instead of simply emulating the physical attributes of a guitar, I decided to emulate the motion used to play the guitar. Specifically, I decided to emulate the strumming motion used in guitar-playing. One challenge was the selection of which notes would be played, as the user wouldn't be able to easily hold notes with one hand while the other hand moved the device. As strumming is generally used to play chords, I decided that the user would be able to select from 6 chords, which could be pre-constructed by the user (through the interface shown in Fig 2a). The user can choose which chord will be played through 6 buttons on the touch screen, each of which corresponds to and sets a chord to be played (Fig 2b). These buttons can easily be reached and touched by the pointer finger of the hand moving the device, which leads to ease of use.

In practical use, the user would first construct and set 6 chords, designating one to each button. This interface is shown below in Fig 2a. Then, the user can move the device in a strumming motion to produce the chord sounds. While performing this action, the user can choose to use a different interface, as shown in Fig 2b. In this interface, the user can tap any of the 6 buttons to begin playing the chord associated with that button. While this design strays from a physical emulation of the guitar, I believe that it effectively integrates the feeling of strumming a guitar in such a way that is easy to use. Further evaluation of this design is presented in a later section, under "Comparison and Evaluation."

Fig 2a. This is the interface that allows users to select chords.

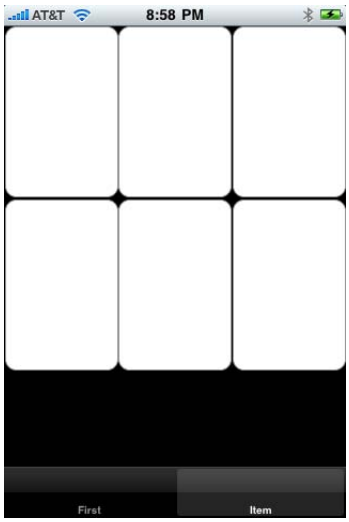


Fig 2b. This interface has larger buttons that can be tapped to select the next chord to be played. At the moment, the buttons are not labeled nor color-coded; these features will be added in a later version.

2.2 Sample Code

The most interesting part of the code is the function used to detect downward and upward strums. In Fig 3, we present the accelerometer readings for two different strums, with time moving from right to left and the red line representing readings for the x axis of the device. The x axis goes from top to bottom when the device is held sideways (that is, in a strumming position). The most important thing to notice about these readings is that the acceleration dips down below a certain threshold and then up above another threshold for a downward strum and vice versa for an upward strum. The ChordMaster uses this distinction to detect downward and upward strums.

At every time step, the ChordMaster receives an acceleration reading to determine if a strum has begun or ended. From observing the readings in Fig 3, the concept is simple—when the function detects a spike in the acceleration, a strum is detected. Specifically, the ChordMaster categorizes an acceleration reading for a given time step as above a threshold, below a threshold, or in the middle, while also keeping track of the reading in the previous time step. If the reading transitions from the middle to either above or below (the initial spike), the beginning of a strum is detected. Then, the next time the reading is in the opposite of the initial spike, the code detects that the strum has finished. The code for this is presented in Appendix A, with further comments.

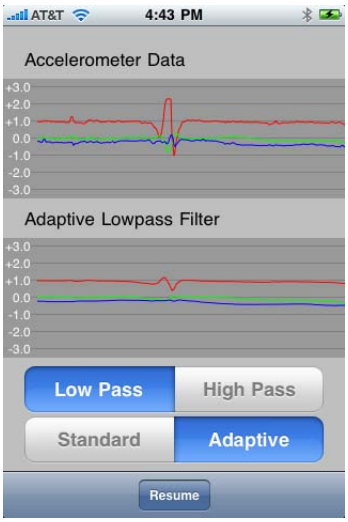


Fig 3a. This is the acceleration data for a downward strum. See below for more details.

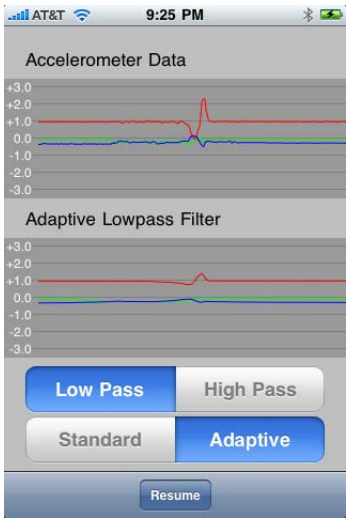


Fig 3b. This is the acceleration data for an upward strum. See below for more details.

Fig 3. This is the graph of accelerometer readings for one downward strum, with time on the x axis (going from right to left) and the g-force on the y axis. The various colors represent different axes of acceleration; the acceleration values from the x axis are used in the ChordMaster, shown in red. The x axis is the vertical axis when the device is in landscape mode (that is, in a strumming position), which allows for detection of acceleration due to strumming motion. The "adaptive low filter" readings are not relevant to this discussion; they are simply included as they were included in the AccelerometerGraph app.

3 Comparison and Evaluation

This application falls under the same category of other iPhone apps that emulate existing instruments. As mentioned earlier, such apps include the Pocket Guitar, MiniPiano, and Ocarina. The guitar and piano are large instruments that are integrated poorly into the iPhone format and are difficult to use, while the ocarina is a small instrument that is integrated effectively into the iPhone format. As of now, there are few, if any, implementations of large instruments on the iPhone that users find easy to use. The ChordMaster fills this niche, providing an easy to use interface for a large instrument—the guitar. The ChordMaster is still a prototype, but shows potential as an easy interface for a large instrument.

One way to evaluate this application is by seeing how well it accomplishes the original goal. The original goal was to create a new iPhone instrument and interface that would be able to emulate a large instrument well. As mentioned in the section "Design Motivation and Results," the ChordMaster only emulates the chord strumming aspect of the guitar and excludes the ability to play single notes. While this isn't a complete emulation of the guitar, I believe that it emulates one of the aspects of the guitar (playing chords) effectively.

Another way to evaluate this application is the evaluation used in my case study paper on iPhone instruments. In the paper, I evaluated iPhone instruments by their ease of learnability and ease of use.[2] With regard to learnability, the ChordMaster has a very simple interface for choosing a chords. Then, for playing chords, the ChordMaster uses the real world metaphor of strumming which allows users who have strummed before to easily adapt this previous knowledge to the ChordMaster and learn how to play with ease. In terms of ease of use, the ChordMaster has a simple interface and can easily reproduce intended sounds, suggesting that it is easy to use. Using this evaluation of ease of learnability and use, the ChordMaster is successful.

4 Future Work and Conclusion

There are many additions and extensions that can be made to the ChordMaster; I present a few of them here.

1. As of now, the ChordMaster does not play chords differently depending on whether it detects a downward strum or an upward strum. Given that it can currently differentiate between these two strums, it might be interesting to have the notes of a chord play in a different order depending on the direction of the strum.

2. In the future, the ChordMaster could allow for saving and loading of sets of chords, to allow for further ease of use.

3. The ChordMaster only allows for an octave of piano notes at the current time. In the future, it would be nice to incorporate notes from other instruments (including the guitar) and more octaves.

4.1 Conclusion

The ChordMaster has great potential and I plan to keep working on the system over winter break to incorporate these additions. I believe that the ChordMaster is successful in that it is easy to learn, easy to use, and emulates an aspect of the guitar effectively.

[2] Sugaya

This being said, the ChordMaster is still a prototype. There are many additions and modifications to be made before it can become a final product. I have two long-term goals in mind for this project. First, I feel that the ChordMaster will be truly successful when it becomes a source of entertainment—for example, when someone pulls out the ChordMaster around a camp fire and starts playing chords while singing along to entertain others. Second, I hope that this project inspires other iPhone instrument developers to free themselves of the need to emulate physical attributes of existing instruments. While some developers have already done so, I hope that more developers focus on emulating other attributes of instruments, such as the method by which they are played. This may inherently lead to emulation of the physical attributes of the instrument, but as seen with this project, this is not a necessity—emulation of physical attributes should be a derivative, not a goal.

5 Appendix A

```
ValuePosition position = MIDDLE; //Assume that the position is in the middle--this is updated later

//See if the acceleration.x value (trueVal) is above or
//below the thresholds. If neither, then the value is in the middle
if(trueVal > upperBound) position = ABOVE;
else if(trueVal < lowerBound)position = BELOW;

switch (position) {
    case ABOVE:
        //The value is above the high threshold
        switch (lastPosition) {
            case MIDDLE: //If the last position was in the middle
                if(!playingChord){//We'll play the chord if we're not already playing.
                    [ChordController playCurrentChord];//This is an upward strum
                    playingChord = true;//Make sure we're playing the code
                }
                break;
            case BELOW://The last position was below the threshold
                playingChord = false;//Then, we're no longer playing the code
                break;
            default:
                break;
        }
        lastPosition = position;//Update our "last" position
        break;
    case MIDDLE://This portion doesn't really matter as being in the middle doesn't make a difference

        if(!playingChord){//If we're not currently playing a chord
            lastPosition = position;//Update our "last" position
        }
        break;
    case BELOW://Similar to the "ABOVE" case, except reversed (it's a down strum)
        switch (lastPosition) {

            case MIDDLE:
                if(!playingChord){
                    [ChordController playCurrentChord];
                    playingChord = true;
                }
                break;
            case ABOVE:
                playingChord = false;
                break;
            default:
                break;
        }
        lastPosition = position;
        break;
    default:
        break;
}
```

References

[1] Sugaya, Andrew. *What Makes an iPhone Instrument Interface Successful?* 2009.

21M.380 Music and Technology (Contemporary History and Aesthetics)

Fall 2009