What Makes an iPhone Instrument Interface Successful?

21M.380

Semester Paper

A Case Study of Music and Technology

Author:

Andrew Sugaya

November 24, 2009

Contents

1 Introduction

Musical instruments and interfaces have existed for over 35,000 years.[1] In our current time, these instruments and their interfaces have found their way onto the iPhone platform, which has various restrictions. The most notable of these restrictions is its size, which is 2.4 in (62.1 mm) by 4.5 in (115.5 mm) by .48 in (12.3 mm)[2]. To put this into perspective, this is approximately the size of a stack of a dozen note cards. Given this constraint of limited size, the user interface becomes one of the most important design choices for developers. Some of these design choices for the user interface have proven successful, while others leave room for improvement. This leads us to the question—How does one design a successful iPhone instrument interface? To answer this, I first propose a couple criteria through which we can analyze the efficacy and success of an iPhone instrument interface. Then, I explore the sensors and input devices that make up the iPhone interface. Finally, I provide a study of various iPhone instruments that have been developed, using the aforementioned criteria to evaluate their levels of success.

2 Determining the Efficacy of an iPhone Instrument Interface

While the advent of the iPhone instrument is fairly new, user interface design is a challenge that stretches back much further in human history. Thus, to answer the question of what makes a good iPhone instrument interface, it helps to borrow a few ideas from the well-researched field of user interface design. One of the golden books of user interface design is "The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design," written by Deborah Mayhew. In this book, Mayhew states that the top two rules for a good user interface is that it should be easy to learn to use and easy to use[3]. To illustrate this

[1] Diaz
[2] Technical Specifications
[3] Mayhew, 511

distinction, here are two examples. A Dvorak keyboard is hard to learn and easy to use, as it places more commonly used keys in easy-to-reach places, leading to an irregular layout. This differs from a keyboard with keys laid out alphabetically, which would be easy to learn (key layout has a pattern) but difficult to use (letter frequency is disregarded). I propose ease of use and ease of learnability as the two main criteria for analyzing iPhone instrument interfaces. With these two metrics, I need to clarify what the term "use" entails when it comes to iPhone instruments. To do this, I resort once again to Mayhew. One of the points that Mayhew presents is that the use of an interface is dependent on its type of users[4]. In our case, iPhone instruments have two main communities of users. The first is the larger group of casual amateurs, who use the application as a source of entertainment. The second group is composed of professionals who use these iPhone instruments for live performances and recordings. In the following two sections, I discuss the implications of these different uses with the two metrics.

2.1 Ease of Learnability

For casual amateurs, ease of learnability is an important aspect because a hard-to-learn interface leads to frustration, after which users give up. In the case of musical instruments, one aspect in ease of learning is how quickly a user can learn how to produce an entertaining sound. Another aspect of learnability for amateurs may be the aesthetic of the application, which can serve to capture the user's interest and encourage the user to learn the interface. For professionals, ease of learnability is less important, as they may be more willing to spend more time and effort learning the interface. This being said, professionals are more likely to accept an interface that is easy to learn.

To go about creating an interface that is easy to learn, the simple solution is to develop a very simple interface with few controls. To implement a more complicated interface successfully, one can exploit the users' previous knowledge by developing an interface that is

[4]Mayhew, 1

similar to an already-popular interface. One example of using such a real-world metaphor is presented by Theo Mandel in his book, "The Elements of User Interface Design". Mandel describes his computer-based telephone system, which has a graphical interface that

> *...looks like a telephone answering machine! It didn't take me [Mandel] very long at all to figure out how to use the telephone answering system. I [Mandel] didn't even have to look at the brief documentation that came with the product.*[5]

In a later section of the paper, we will see that many of the current iPhone instruments use this idea of creating a real-world metaphor by emulating existing instruments.

2.2 Ease of Use

Ease of use for amateurs is defined by the ease with which the user can produce entertaining sounds and rhythms. For professionals, who use these instruments in performance, ease of use is determined by the ease with which they can produce intended sounds quickly and reliably. Reliability also includes repeatability, in the case of rehearsal-based performances (that is, not improvisational). Two factors that affect speed and reliability are the mapping of musical parameters, discussed in part below under "Strict and Relaxed Musical Parameters", and the resolution of the various input devices used in the interface, discussed later.

2.3 Strict and Relaxed Musical Parameters

Here, I present the terminology of strict and relaxed parameters, which are distinguished by the following. A difference in a *strict* parameter between two performances of the same composition would concern and be noticed by the audience, while a difference in a *relaxed* parameter would not necessarily be noticed or concerning. For instance, pitch is a strict parameter in that if a performer plays a D sharp at the beginning of Bach's Violin Sonata

[5]Mandel, Chap. 5, pg 18

#2 Fuga (which begins with a D), an audience would definitely notice and be concerned. An example of a relaxed parameter would be volume—if a performer began playing a piece in *pp* as opposed to *p*, an audience may not notice and/or care. For this discussion, the important distinction between strict and relaxed parameters is that strict parameters need to be more reliable than relaxed parameters.

3 iPhone: More than just a graphical interface.

When one hears the term "user interface," one commonly resorts to the notion of a graphical user interface. In this paper, however, we will consider all forms of input, graphical or not, as part of the user interface. This is especially important when considering the iPhone, as it houses many sensors and input devices, including a touch screen, a microphone, an accelerometer, a magnetometer, and a camera. In each of the following sections, I explore the technical details of each sensor, as well any associated musical metrics.

3.1 Touch Screen

Physically, the touch screen is the largest sensor in the iPhone. It is placed on top of the LCD display layers, spanning 3.5 inches diagonally.[6,7] It monitors changes in electrical current, allowing it to track multiple touches from the user.[8] Through experimentation, I have found that the touch screen can track up to a maximum of five touches simultaneously. From a practical standpoint, however, it must be noted that using more than three fingers at a given time makes it difficult for the user to see the screen.

One of the important details of the touch screen is the resolution of touches. That is, how precise are users' touches? For the developer, the precision of the touch inputs are

[6]Wilson
[7]"Tech Slavior"
[8]Wilson

given to the nearest pixel of the center of the touch. However, users cannot generally work with this level of precision, especially if they are using their fingers. More practically, if we assume the width of the average finger to be about 1 cm, then interface elements (such as buttons) should be no smaller than 1 cm[9]. This means that the iPhone's 5 cm x 7.5 cm screen has a practical resolution of approximately 5 x 7 touches. These measurements are relevant to both discrete interface elements (buttons) and continuous elements (sliders), with different implications, as discussed below.

3.1.1 Discrete Elements

Discrete elements inherently reduce the malicious effects of a misplaced touch by allowing for all touches within a certain range to map to a single input. So, as long as discrete elements are large enough (specifically, at least 1 cm x 1 cm), they are very reliable. This means that such discrete elements are effective at mapping to strict parameters, such as pitch. However, the practical resolution suggests that only 5 elements should be placed width-wise and 7 elements height-wise. In the case that an interface maps one button to one pitch, this suggests that we can only have 7 pitches if we use one dimension. This is pitiful, considering that even an octave on the piano includes 8 white notes. There are solutions to this challenge of limited space such as using combinations of buttons to determine notes, which is discussed in detail later in the section regarding Smule's Ocarina.

3.1.2 Continuous Elements

Continuous elements are interesting because their inputs can be continuous or discretized using a filter. To obtain the same level of reliability as a discrete element, multiple touches need to be mapped to a single input; that is, the inputs need to be discretized. If we do this, the practical resolution suggests that a slider that is N cm long should have at most N discrete values. Akin to the discrete elements, this leads to a maximum of 7 pitches

[9]Kewaley, 4-5

on one slider. If we don't discretize the inputs, we end up with less reliability. However, these continuous inputs map to relaxed musical parameters effectively, as relaxed musical parameters are more accepting of continuous inputs' inherently lower level of reliability. Continuous elements can also easily account for musical parameters that are not perceived linearly by the human ear. For instance, amplitude of a sound is perceived by the human ear on a logarithmic scale. This means that if one were to use a linear scale on the continuous slider normalized from 0 to 1, there would be a much greater difference in going from 0 to .5 than from .5 to 1. One can easily resolve this problem by implementing a fourth power-function (x^4) scale for the continuous inputs, which allows the ear to perceive the change in volume to be linear. Given the lower reliability and the ability to account for non-linearity, non-discretized elements are effective when mapped to relaxed musical parameters.

3.2 Microphone

The iPhone has a built-in microphone, which can pick up in a range from 30dB to approximately up to 105dB and in the newest model, has a frequency range of 100 to 19000 Hz.[10] This is fairly close to the range of human hearing, which is 20-20000 Hz[11] and includes all of the typical human vocal range, which ranges from 100-1000 Hz.[12] It is interesting to consider the mappings of these two input parameters, frequency and amplitude, discussed below.

3.2.1 Input Parameter One: Frequency

It must be noted that the input "frequency" is actually a collection of frequencies (the input is unlikely to be a perfect sine tone), which may need to be processed if being used as an input for uses other than recording. It is difficult to imagine frequency input being mapped to something other than frequency, especially if the iPhone instrument's output is in real time. However, when it comes to mapping an input to frequency, the frequencies from the

[10] faberacoustical
[11] Bullock, 85
[12] Luciani, 149

microphone are very effective in that what you hear is what you get. Assuming a continuous input, reliability of this input is determined by the user's voice, which may or may not be reliable. Later in this paper, I will discuss the application "I Am T-Pain" that discretizes the input for greater reliability and entertainment.

3.2.2 Input Parameter Two: Amplitude

As mentioned earlier, the input amplitude ranges from 30dB to approximately 105dB. The microphone returns values in dBFS ranging from -60 dBFS to 0 dBFS, which can be converted to a normalized linear scale ranging from 0 to 1.0, with values containing precision to the thousandth's place. This precision is a faux-precision, however. That is, it is difficult for users to work with such precision. Thus, I attempted to determine a more practical resolution by experimentally blowing into the device and attempting to "hold" an amplitude of 0.5 on the normalized linear scale. Specifically, I ran the SpeakHere app, which displays the amplitude level, and blew into the microphone while trying to maintain the amplitude level at the center. Through logging the results over a period of three seconds, I found that the inputs ranged from 0.4 to 0.6 on the normalized linear scale. Assuming that I am an average user, this means that the practical resolution of the amplitude input is 0.2 out of a total range of 1.0. Thus, this input would only be reliable if it were discretized into a maximum of five discrete values. This limited range in a discretized situation implies that this input is more suited for continuous inputs and lower reliability-situations—that is, relaxed parameters. This is not hard to believe, given that amplitude input from the microphone is designed to be mapped to amplitude, which is a relaxed parameter.

3.3 Accelerometer

The iPhone accelerometer returns values as follows—if one were to tie a brick to a string and attach it to the center of the bottom surface, the accelerometer would report the direction of

the string relative to the device (ie, the direction of gravity) and the strength of force in that direction. For the developer, the accelerometer returns values in terms of the g-force. That is, the force of gravity is 1.0G. By shaking the device, I was able to produce magnitudes of up to approximately 2.2G. Forces with magnitudes above 1G are not sustainable in a practical manner, while forces less than 1G can be sustained by changing the direction of gravity relative to the device; that is, by rotating the device. Therefore, to determine a practical resolution, I assume that the range is from -1.0G to 1.0G. It is important to note that this range is 180°, not a full circle, as all values (except 1.0G and -1.0G) are repeated on the circle. By using the AccelerometerGraph app, I experimentally determined that I could hold a given g-force at ± 0.02G. By adding a low-pass filter, I was able to hold at ± 0.01G. Using our range of -1.0G to 1.0G and practical resolution of 0.02G alone, one could theoretically discretize this continuous input into 100 discrete values, which is more than any other input device thus far. However, this resolution is challenged when we want to make this input reliable. One of the main notions of reliability is the aspect of quick repeatability. For instance, could one play a melody, put down the iPhone, pick it back up, and play the melody again? Given a randomly generated g-force value, I was only able to point the iPhone at the given direction ± 0.15G. This produces a big difference in our practical resolution, bringing the theoretically 100 discrete values down to approximately 6[13]. This new estimate of the practical resolution is within the same range as the previously mentioned inputs (the touch screen and the microphone). To increase the resolution, one could theoretically have a display that notifies the user of the current g-force value. While slow, this could allow for more accurate repeatability. However, the accelerometer is plagued with the challenge that the input value is directly associated with the rotational position of the device. In some of these positions, the display is either in a confusing position (at a diagonal) or has low visibility (facing the other direction). This suggests that using the display in conjunction with the accelerometer is difficult when using the full range of the accelerometer. In conclusion, the accelerometer can reliably produce 6 discrete values, which

[13]This is interesting, as six is half of the number of hours on a clock, suggesting that either my previous knowledge of clocks led to this value, or the number of hours on a clock was determined by an inherent desire for this specific spacing.

can be mapped to a strict parameter, or can produce continuous input that can be mapped to a relaxed parameter.

3.4 Magnetometer

In the newer versions of the iPhone, a magnetometer is included. Essentially, a magnetometer is a digital compass. So, this sensor returns readings about a surrounding magnetic field. The developer can grab the current heading of the device (from 0° to 360°), to an accuracy of about ± 2°. So, the practical resolution for the magnetometer before taking into account repeatability is about 4°, leading to 90 discrete values. Once we take repeatability into account, we can use our previous analysis for the accelerometer to determine that the practical resolution of the magnetometer drops down to about 30°, or about 12 discrete values. This value is double the number of discrete values for the accelerometer as it has a full 360° range, in comparison to the 180° range of the accelerometer. 12 discrete values is the most we have seen for any input (except perhaps the frequency input of the microphone), but the range of motion required to produce these values forces the user to turn the device around a full 360°, which is an awkward motion without extra props (such as a spinning plate or a turntable). As a result, the magnetometer can produce 12 discrete values, some of which may lack ease of use, or can produce continuous input to be mapped to a relaxed parameter. Along with the current heading, the magnetometer can also return the strength of the surrounding magnetic field, returned in microTeslas. For fear of destroying my device, I did not perform any experiments involving a magnet to determine the resolution of this input. However, it is definitely conceivable that such an input could be mapped to pitch, leading to a theremin-like device.

3.5 Camera

The iPhone hosts a 2 megapixel camera facing in the opposite direction of the touch screen.[14] The developer can access the camera and take images and videos, which can then be processed. Theoretically, this could be used as an input device for an instrument. However, this is difficult to use in real-time, as the images can only be acquired at a rate of about one per three seconds and the video must be recorded before it can be analyzed. Also, images and videos taken with the camera are very unreliable, as they are affected greatly by the surrounding environment. While this unreliability and difficulty of real-time use would make this input unfit for professionals, it is conceivable that the camera may be used as an input source for amateur entertainment. For instance, each pixel of a user's image has red, green, and blue values ranging from 0 to 255 each that may be mapped to three separate parameters like pitch, amplitude, and timbre. Then, each pixel can be played in order from left to right, top to bottom, leading to a composition derived from the image. In this example, the input is essentially a continuous input that has been discretized to values from 0 to 255.

4 iPhone Instruments: What works and what doesn't.

Since the release of the iPhone App Store, there have been many iPhone instruments. Some of these have great interfaces while others leave room for improvement. In the following sections, I present a discussion on a selection of these various iPhone instruments. Putting the proposed criteria into action, the two main questions that I ask about each instrument are "Is it easy to learn?" and "Is it easy to use?"

[14] Carr

4.1 MiniPiano

The MiniPiano is essentially a keyboard interface applied to the iPhone. It places an image of a keyboard on the screen and uses the multi-touch capabilities of the touch screen to emulate pressing of keys, of up to five keys.

4.1.1 Ease of Learnability - A Real-world Metaphor

The MiniPiano is perhaps the champion when it comes to ease of learnability. Not only is the interface very simple, it also uses the real-world metaphor of a piano. This makes it easy to learn for amateurs and professionals alike. However, in trying to emulate a large instrument such as the piano in such a small device, we see that there is a trade-off in the ease of use.

4.1.2 Ease of Use

One of the concerns related with the MiniPiano interface is that a large interface (a grand piano has 88 keys) is shrunk to the size of the iPhone, limiting the number of keys to 14.[15,16] The width of each key is approximately 1 cm, so the reliability of this interface is conserved. However, the number of notes that can be played is very limited (less than an octave), which makes this interface difficult to use in a practical setting. As a solution, developers of the application have added a second row of keys which doubles the note range. Unfortunately, this not only makes the keys smaller, but also decreases the ease of learnability, as most users are not acquainted with multiply-layered pianos. Another interesting remedy to the challenge of limited notes has been to use multiple iPhones, lined up next to each other.[17] Although this may be a creative solution, it is very expensive. Given the lack of a true solution to the problem of trying to shrink the piano into the iPhone, this interface, while

[15]The Piano
[16]TopTenREVIEWS - MiniPiano 2009
[17]iPhone Piano

easy to learn, is not easy to use.

4.2 Pocket Guitar

At the time of this writing, the pocket guitar is the seventh most popular paid music app.[18] As the name implies, it follows an interface close to that of a guitar. This application shows several strings on the screen and uses the multiple touch interface to detect fingers "holding down" frets, as well as "strums" and "plucks".[19] This interface also suffers from the same challenge as the MiniPiano in that it attempts to shrink a larger interface.

4.2.1 Ease of Learnability

With regard to ease of learnability, Pocket Guitar, like the MiniPiano, benefits from a real-world metaphor. Unlike the MiniPiano, the Pocket Guitar is only easy to learn if the user has previous experience with the real-world object. One user's recount of his experience with the Pocket Guitar illustrates this point.

> *I had never even touched a guitar outside of picking up my friend's "axe" (as the cool kids call them), strumming a few off-key strings, and promptly putting it down. I quickly found that Pocket Guitar's developers did such an accurate job recreating the guitar experience on the iPhone that I sounded just as horrible on it as on a real guitar.*[20]

From this we see that an interface does not become easy to learn by simply using a real-world metaphor. The more well-known the real-world object is, the more likely the interface is easy to learn.

[18] Top 50 Paid Music Apps
[19] PocketGuitar Review
[20] macenstein

4.2.2 Ease of Use

The Pocket Guitar suffers from the issue of not being able to emulate all of the notes on a real guitar. This is caused not only by the limited size of the iPhone, which only allows for a portion of the guitar to be visible, but is also limited by the number of touches the iPhone can handle. As the touch screen can only track up to five touches, the maximum number of frets that can be held down is four (one finger is needed to strum). On top of this, for chords with four fingers, the device becomes very difficult to hold. On a real guitar, the instrument is supported by the performer's thigh, arm, and/or a strap, so support is not an issue. With a hand-held device like the iPhone, the instrument is supported by the hand that is either holding down the frets or the hand strumming the notes, both of which is difficult. One solution to this is to place the iPhone on a flat surface and play it more like a koto. That is, hold down the frets with one hand as if playing the piano and strum the notes with the other hand in the same "piano position." While this leads to easier use, this leads to a harder-to-learn interface, which is undesirable. The Pocket Guitar is relatively easy to learn, especially with previous guitar experience, but is very difficult to use.

4.3 Ocarina

At the time of this writing, Smule's Ocarina is the sixth most popular paid music app[21]. This instrument, modeled after the ocarina, uses four buttons whose combination determines the pitch and the microphone's input amplitude to determine the amplitude of the instrument.

4.3.1 Ease of Learnability

As with the MiniPiano and the Pocket Guitar, the Ocarina uses a real-world metaphor. While users may have less experience with the ocarina than with a guitar, the interface is very simple (four buttons), which makes the ocarina easier to learn. Musical scores for the

[21] Top 50 Paid Music Apps

ocarina exist that show which holes should be covered; this allows users to learn and play songs very easily.

4.3.2 Ease of Use

The difference between the Ocarina and the MiniPiano/Pocket Guitar lies in its choice to emulate a small instrument. As founder of Smule, Ge Wang, writes,

> *We wanted to create an expressive musical instrument. However, instead of taking a larger instrument (e.g., a guitar or a piano) and "shrinking" the experience into a small mobile device, we started with a small and simple instrument, the ocarina (more specifically, the 4 hole English Pendant ocarina), and fleshed it out onto the form factor of the iPhone.*[22]

This idea of emulating small and simple historical instruments is very successful. As the emulated instrument fits well onto the iPhone, no notes are lost in the emulation process. The Ocarina also manages the interface's real estate very well by using combination of button presses to determine a note. Previously, we assumed that one button would be mapped to one pitch, as in the MiniPiano, which gives a range of N pitches for N buttons. However, using combinations of buttons, we get a much greater range of 2^N pitches for N buttons. With the Ocarina, which uses 4 buttons, we get a range of 16 notes, which is 4 times the number of values a one-button, one-pitch interface would have.

While other applications use the microphone to record sounds, Ocarina uses the microphone to determine when the user is blowing on the device. Specifically, the amplitude of the input (the breath) is mapped directly to the amplitude of the resulting note. [23]

[22] Wang
[23] Wang

4.4 Bebot

Bebot is an interesting application that uses up to four simultaneous touch locations to determine pitch, timbre, and/or volume of the sound.

4.4.1 Ease of Learnability - Progressive Disclosure

While Bebot does not use a real-world metaphor like the MiniPiano or Ocarina, it still has a very simple and easy-to-learn interface. This is in part due to its use of a technique known as "progressive disclosure." As Mandel writes,

> *Users should not be overwhelmed by what they can do in a product. You don't need to show users all of the functions the product offers. The best way to teach and guide users is to show them* what *they need,* when *they need it, and* where *they want it. This is the concept of* progressive disclosure.[24]

Bebot exemplifies progressive disclosure by providing the novice user with a starting screen that is very simple—touching the screen produces a sound. Bebot does an excellent job of allowing a novice user to quickly produce an entertaining sound. Continuing with the concept of progressive disclosure, Bebot allows curious users to pull up a secondary panel which has the option of loading other preset values for the timbre. Changing between these values is as simple as pressing a button and provides even more entertainment for the user. The next level of disclosure is revealed when a user decides to explore Saving Presets, Synth Controls, Effects, and Scale, which bring up more panels with numerous options. If the interface were to display all of these options to the novice user, the user would be overwhelmed and may give up learning the interface. However, through progressive disclosure, Bebot effectively allows for very easy learnability.

[24]Mandel, Chap. 5, pg 20

4.4.2 Ease of Use

Bebot uses the whole touch screen as an input device, mapping the two axes to two different musical parameters. In particular, the longer axis is mapped to pitch. Bebop allows the user to select the pitches to be either discretized or continuous. In the discretized scenario, we know that to ensure reliability, seven is the optimal maximum number of discrete values to use on this axis. To increase ease of use, Bebot provides the option to toggle a note grid that shows where notes are and an option to control the range of pitches covered by the axis, ranging from one octave to almost six octaves. For reliability, a more focused range is more optimal—so to determine if Bebot is reliable, this discussion focuses on the range of one octave. For one octave, about 14 discrete values are required, which is about 0.5 cm per value. While this breaks the rule of at least 1 cm per value, it should be noted that Apple also breaks this rule. In the standard iPhone keyboard, keys are allotted 0.5 cm in portrait mode[25]. This being said, the iPhone keyboard is "intelligent" in that it provides a visual confirmation of each letter as it is typed, uses pattern recognition of common sequences of letters, uses a dictionary, and has adaptive button sizes—that is, each button's hit area is determined by predicting the next letter to be typed[26]. Bebop is unable to be "intelligent" in any of these fashions, due to its inherent logistics. Specifically, notes in a composition do not necessarily follow a pattern so it is fairly unreasonable to use pattern recognition or adaptive button sizes. Visual confirmation and use of a dictionary assume that the output occurs when a touch is released, not when a touch begins. As Bebop produces sound when a touch begins, visual confirmation and use of a "dictionary" are not possible. Thus, Bebop does not have the most reliable of interfaces, as it only allots 0.5 cm per discrete value without any "intelligence." This being said, it seems that while difficult, this interface could be and is used by professionals like Jordan Rudess[27]. This may be due to the improv-like style of Rudess, which might not require repeatability to the extent that a rehearse-based performance would.

[25]Kewaley, 5
[26]Kewaley, 5-6
[27]Synthtopia

4.5 Cosmovox

Cosmovox has several timbres whose pitches can be controlled through various movements of the device detected by the accelerometer.

4.5.1 Ease of Learnability

Cosmovox, like Bebot, does not use a real-world metaphor. With regards to how quickly a user can make an entertaining sound, Cosmovox does a very good job—sound is output continuously when performance mode is selected (which is automatically selected). Cosmovox uses a two-step version of progressive disclosure that increases ease of learnability. While novice users can produce sounds immediately in performance mode, more curious users can alter options such as the range, fundamental, portamento, modulation, feedback, vibrato, and more. In comparison to Bebot, Cosmovox uses fewer levels of progressive disclosure, which makes ease of learning more difficult.

4.5.2 Ease of Use

Cosmovox relies solely on the use of the accelerometer for input, which is very unreliable, as seen in the previous section on the accelerometer. Cosmovox allows for continuous input as well as inputs discretized to notes in a scale, using preset scales such as Major, Minor, Pentatonic, etc. This idea of discretizing inputs to a scale is highly beneficial. A scale is the subset of notes in the entire octave (assuming the scale is not chromatic). By discretizing inputs to notes in a scale, unused notes are excluded from the mapping. This increases the range of the notes mapped by the discrete values almost two-fold (for instance, two octaves in the place of one). Increasing this range improves the usability of this interface for professionals. Like Bebot's ability to control the range of pitches covered by the axis, Cosmovox allows the user to change the range of notes mapped by the accelerometer's input values. Given this, it is important to note that as the accelerometer maps reliably to a

maximum of 6 discrete values, having more than one octave of a scale (8 notes) mapped by the accelerometer leads to a fairly unreliable interface. One pitfall of the Cosmovox interface is that to transition between two notes, all of the other notes in between must be played, due to the inherent continuity of the accelerometer. In conclusion, professionals may be able to use Cosmovox, as long as they don't need a large range of notes and don't mind playing transition notes.

4.6 I Am T-Pain

I Am T-Pain is an application by Smule that makes use of auto-tune. In effect, auto-tune maps input frequencies from the microphone to discretized pitches on a scale, much like Cosmovox.

4.6.1 Ease of Learnability - Aesthetic Value

To create an entertaining sound using I Am T-Pain, the user simply has to sing into the microphone. As such, the interface is very easy to learn. On top of this simple method of user input, I Am T-Pain improves upon learnability by using a graphical interface that contains aesthetic values designed specifically for the user. For instance, the title "I Am T-Pain" is in shiny gold "bling" and there are images of T-Pain that subconsciously and/or consciously motivate the user to want to learn to be like T-Pain. While this motivation does not necessarily make the interface easer to learn, it does make the user more interested, which leads to faster learning.

4.6.2 Ease of Use

Auto-tune is designed to make singing easier and as a direct result, I Am T-Pain is very easy to use. Due to the discretization that takes place, the reliability is greater than that

of the user's voice alone. The usable range of notes is not limited by the resolution of the input device. Instead, the usable range of notes is only limited by the range of notes of the user's voice. This dependency on the user's voice suggests that if the user has a very weak or otherwise hard-to-use voice, then this interface may be hard to use. However, for the average user, I Am T-Pain has an easy-to-use interface.

5 iPhone in Music: Conclusion

What makes an iPhone instrument interface successful? Through the exploration of user interface design, the quick answer was that an interface that is easy to learn and easy to use will be a successful interface. To answer the question of how to design such an interface on the iPhone, this paper studied and experimented with various iPhone sensors in depth to arrive at effective mappings between input devices and musical parameters. Finally, the paper critiqued current iPhone instruments, touching upon beneficial ideas such as the use of real-world metaphors, progressive disclosure, and aesthetic value. Whether you're a developer trying to develop the next best-selling app or a professional musician creating a new personal iPhone instrument, the explorations and ideas in this paper should be a valuable resource that guides the way to a successful iPhone instrument interface.

References

[1] Bullock, John, Joseph Boyle, and Michael Wang. *Physiology.* 4th ed. Baltimore: Lippincott Williams & Wilkins, 2001. 85. Print.

[2] Carr, Michael. "iPhone Camera Analysis of the Photo Capabilities and Features of the Apple iPhone." *About.com.* Web. Oct 25 2009. <http://cameras.about.com/od/cameraphonespdas/a/iPhonecamera.htm>.

[3] Diaz, Jesus. "35,000-year-old Flute Is First Instrument Ever - flute - Gizmodo." *Gizmodo.* 25 06 2009. Web. Oct 30 2009. <http://gizmodo.com/5302361/35000+year+old-flute-is-first-instrument-ever>.

[4] "iPhone Microphone Frequency Response Comparison." *faberacoustical.* Web. Oct 25 2009. <http://blog.faberacoustical.com/2009/iphone/iphone-microphone-frequency-response-comparison/>.

[5] "iPhone Piano." *YouTube.* Web. Oct 25 2009. <http://www.youtube.com/watch?v=y7IH71ioqc0>.

[6] Kewaley, Susheel. "The Effect of Multi-touch Technology on User Interface Design." *National Conference on Advances in Usability Engineering.* (2008): 4-5. Print.

[7] Luciani, Luigi. *Human Physiology.* Vol 3. London: Macmillan and Co., 1915. 149. Print.

[8] "AAARRGGHH!! Why do I keep buying iPhone instrument apps?!?!." *macenstein.* Web. Oct 25 2009. <http://macenstein.com/default/2008/12/aaarrgghh-why-do-i-keep-buying-iphone-instrument-apps/>.

[9] Mandel, Theo. *The elements of user interface design.* John Wiley , 1997. Print.

[10] Mayhew, D. J. *The usability engineering lifecycle.* Morgan Kaufmann, 1999. Print.

[11] "PocketGuitar Review." *MacWorld.* Web. Oct 25 2009. <http://www.macworld.com/appguide/app.html?id=69168>.

[12] "Rudess Meets Bebot – Synthtopia" *Synthtopia* 02 16 2009. Web. Nov 20 2009. <http://www.synthtopia.com/content/2009/02/16/rudess-meets-bebot/>

[13] "iPhone screen dimensions and comparisons." *Tech Slavior.* 09 01 2007. Web. Oct 25 2009. <http://slavior.wordpress.com/2007/01/09/iphone-screen-dimensions-and-comparisons/>.

[14] "Apple - iPhone - Technical Specifications." *Apple* 2009. Web. Nov 20 2009. <http://www.apple.com/iphone/specs.html>.

[15] "Top 50 Paid Music Apps." *appstoreapps.* Web. Oct 25 2009. <http://www.appstoreapps.com/top-50-paid-music-apps/>.

[16] "MiniPiano 2009." *TopTenREVIEWS.* Web. Oct 30 2009. <http://iphone-apps.toptenreviews.com/music/minipiano-review.html>.

[17] Wang, Ge. "Designing Smule's iPhone Ocarina." Print.

[18] "The Piano" *hyperphysics* Web. Nov 20 2009. < http://hyperphysics.phy-astr.gsu.edu/hbase/music/pianof.html>

[19] Wilson, Tracy. "HowStuffWorks "iPhone Touch Screen"." *HowStuffWorks.* Web. Oct 25 2009. <http://electronics.howstuffworks.com/iphone1.htm>.

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

Fall 2009