Design a Natural User Interface for Gesture Recognition Application

Zhaochen Liu

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-101
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-101.html

May 17, 2013

UNIVERSITY OF CALIFORNIA, BERKELEY

Design a Natural User Interface for Gesture Recognition Application

Zhaochen Liu

5/1/2013

Advisor: Prof. Björn Hartmann, Chun Ming Ching

Contents

1 Abstract

Gesture recognition allows users to interact with their electronic devices in a manner that mimics their interaction with the physical world, thus making devices easier to use for all users. Interacting with the software, using gesture and body motion is definitely a cool ideal. However, designing effective interactions is a challenge on a gesture-controlled application.

We partnered with the Microsoft Windows Phone Team to develop a Kinect physiotherapy application. This application enables the patients to perform recovery exercises remotely at home. It checks the patient’s body movements against doctors’ instructions and provides real-time instructions to the patients. In order to validate the market value for such an application and further enhance this prototype, we conducted market studies and user tests. In particular, I was in charge of the development of the user interface in this project. Also, I developed guidelines for designing the user interaction of gesture-controlled applications.

2 Introduction

2.1 Main Problem

The main problem with gesture recognition technology is that it is a relatively new technology and users are unfamiliar with gesture-controlled applications. Therefore, when it comes to the user interaction design of such applications, the developer may find it hard to find some textbook examples to learn from. The design community hasn't generalized rules of thumb of the user interaction design for such applications. In some context, it is a good thing because the user will try to learn everything that the application teaches him. The user does not have any pre-defined concept of "a good interaction". However, it is also a bad thing as it is the developers' responsibilities to ensure the user get to know the application as fast as possible and lower the learning curve for the user.

Moreover, designing the user interface for a gesture-controlled application is much harder than designing the user interface for a desktop application or a website. As a developer, you not only have to be careful about the layout, clarity or consistency, you also have to be very cautious about gesture design, combinations of input methods and real-time feedback to the user.

2.2 Why it is an interesting problem?

This problem is particularly interesting to solve because the gesture-controlled technology maybe become prevalent in people's lives. Making some prototypes in this area and trying to generalizing the design patterns and guidelines will be beneficial to future applications. Nowadays, most users have quite a lot experience with desktop software, web & mobile applications and formed some habits. Therefore, utilizing their usual practices to build a natural user interface for a gesture-controlled application is a good problem to solve.

2.3 Overview of this paper

This paper first goes over some of the background information of this problem. It talks about the core problem and why we are solving it. After that, the paper will focus on our steps to solve the problem. The result, followed by in-depth analysis of it will be discussed later. In particular, I highlight my contributions in the methodology section and talk about designing the natural interface for gestured-controlled applications. At last, I have list some recommendations and future improvements that can be done to this project.

3 Methodology

3.1 Overview

Our overall goal of this project is to develop a Kinect physiotherapy application. Our idea was to build and physiotherapy application to demonstrate the power of gesture recognition. The application will help some injured patients to perform exercises in order to recovery quickly. The application will provide real-time instructions to the users and tell the users how to move their body to the correct pose. Our application will also have an exercises manager that can help the user choose what exercise to do next. In addition, it can send the patient's information to his doctor.

During the course of this project, five team members are all involved. A lot of the tasks were an effort among all the team members. For example, we hosted sessions so that we could brainstorm ideas. Collectively, we also analyzed the market, made presentations to the Fung Institute, and worked together when we had a big decision to a make.

On an individual level, each team member has to take different tasks on certain occasions. When it made sense to divide the work, we usually tried to break a big problem into manageable small pieces. From there, each of us took one part of, based on our interests and skill sets, and worked in parallel.

3.2 Implementation

3.2.1 Technology

The main programing environment is .NET framework using C# programming languages. We also used Kinect SDK to handle some of the gesture recognition work.

3.2.2 .NET Framework and C#

.NET is a software framework developed by Microsoft that runs primarily on Microsoft Windows [1]. It provides large sets of libraries to support various tasks, thus reducing the tasks on the developers' side. C# is an object-oriented programming language that works well in the .NET framework. It is the ideal development language for software running in the Windows environment, such as Windows Phone and Microsoft Surface tablet.

3.2.3 Kinect SDK

Originally, Kinect is an add-on gadget for the Xbox game console. It is a special camera that is able to track human bodies. Microsoft saw the potential of Kinect and opened the development kit to the general public [2]. With this SDK, the engineers may be able to utilize the gesture recognition capabilities of the Kinect with ease. For example, in our project, with minimal setup, we are able to get the skeleton of a human body. It means that the SDK can provide us with the 3D coordinates of 20 joints of the user standing in front it [3].

3.3 My Contributions

As mentioned earlier, tasks such as marketing studies, strategy discussion and presentations are shared among all of the team members. Meanwhile, each team member has his specific task. My particular contributions include the exercise recognition, user interface and user interaction.

3.3.1 Exercise Recognition Engine

In the first semester, my teammate and I worked together on the exercise recognition and built a prototype to demonstrate the basic functionalities of the physiotherapy application. In particular, we built a parser that is able to take the exercise’s definition and convert it to C# objects. We defined the exercise in XML format and Then, Aarthi and I coded a state machine that is able to oversee the progress of the current progress. We utilized the Kinect SDK to get the user’s current gesture and compare it with the targeted gesture defined by the XML file. If the different is under a certain threshold, the state machine will recognize the current exercise step as completed and move to the next step. Aarthi improved this engine in the second semester.

Our preliminary application from the first semester was a success as it demonstrated a good use case of gesture recognition. However, that application was with minimal user interface elements which would not give user a good user experience. For example, the user had to click a button on the user interface with a mouse and then move away from the screen to perform the exercise. The interaction between the system and the user is nowhere optimal. To solve this problem, I focused on user interface development in the second semester.

3.3.2 User Interface Design

I adopted a rapid prototyping strategy while designing the user interface. I started off by sketching some user interfaces on a paper [Figure 3-1]. This is a quick way to brainstorm interfaces because it cost minimal time and effort. When coming up with those interfaces, I applied what I learned from the CS 160 User Interface Design class. Particularly, I used Jacob Nielson's 10 design heuristics [4] to evaluate and improve my design, which will be discussed in the next section.

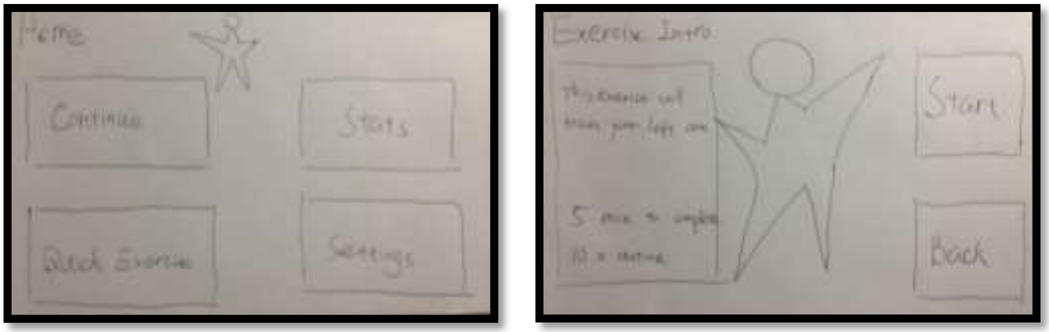

Figure 3-1 Early sketches on paper

After having the sample UI designs on the paper, I tested its effectiveness with some users with Wizard of OZ studies [5], an early-stage user study with the purpose to gather feedback without actually implementing the system. In addition to letting the subjects 'use' the system; I also asked them how they felt about this user interface. After gathering all the feedback from the users, I made a low-fidelity prototype consists of some mock-ups using Balsamiq, a widely-used mock-up software designed to create sample user interfaces quickly [6]. I repeated the steps of testing with real users. This time, because the low-fidelity prototype is closer to the final product, the subjects usually could get the idea of the application very quickly. In addition, they validated that the changes I have made after the paper prototype stages improved the overall user experience, namely, I made the buttons more spread out to reduce the chance of selecting the wrong buttons by mistake using just gestures.

The implementation of the User Interface utilizes the Windows Presentation Framework [7], a user interface library developed by Microsoft and compatible with Kinect SDK. The whole system is similar to a MVC (Model-View-Controller) where the view is the layout of user interface, defined by a XAML file [8]. Some elements are tied to an event handler, which will trigger an event when the user interacts with them. The event handler, which is the controller, will call out to the model to invoke the business logic, such as getting something from the database and transferring to another screen.

One central problem is enabling users to interact with our application solely through gestures. In the interface, a hand icon is displayed that tracks the movement of the user's hand [Figure 3-2]. Then, the user may push his hand forward to initiate a "select" action on a button. Or, he is able to hold his fist to start drag a scrollable view left or right [Figure 3-2].

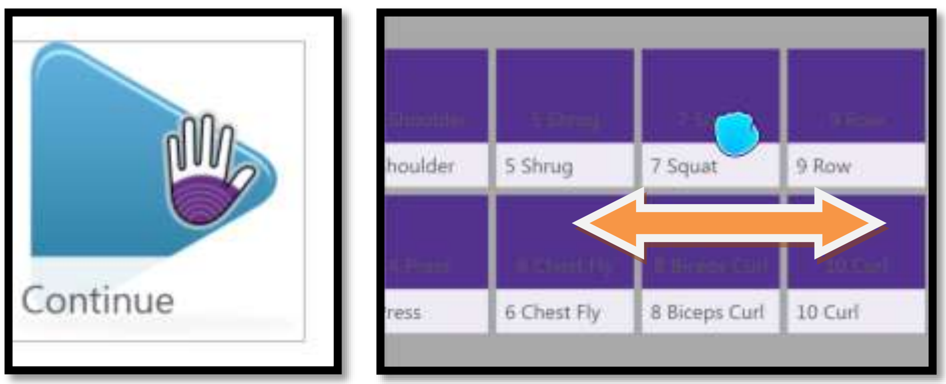

Figure 3-2 Demonstration of push button and scroll view

4 Discussion & Results

After implementing the system, we conducted series of user tests to fully evaluate the effectiveness of this physiotherapy application. Particularly, in my part, I was following a design-prototype-evaluate cycle so that it was essential to me to see how user reacts to the prototype I created.

4.1 Results

This physiotherapy application is able to let the users conduct a series of exercises and track his injury recovery progress on a daily basis. It also has the ability to package the patient's data and send it to his doctor.

There are four options on the home screen [Figure 4-1]: the user is able to continue the exercise routine the doctor created for him or he can freely select any available exercises in the system [Figure 4-2]. In addition, the user may push the 'Stats' button to track his current progress or send his information to his doctor [Figure 4-3].

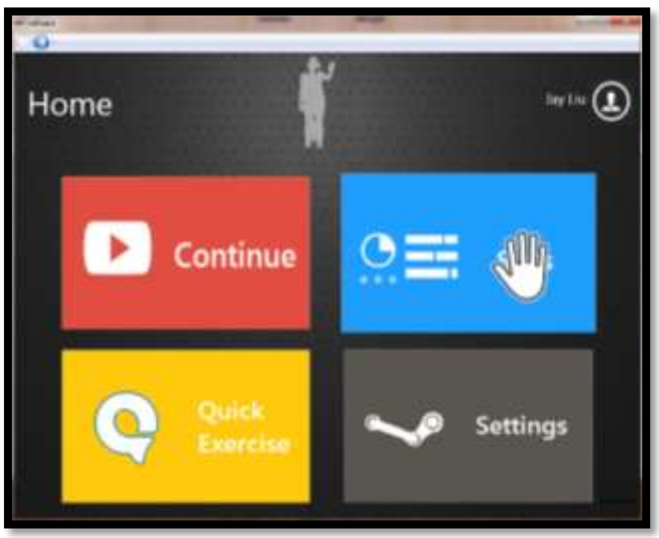

Figure 4-1 Home screen of the application

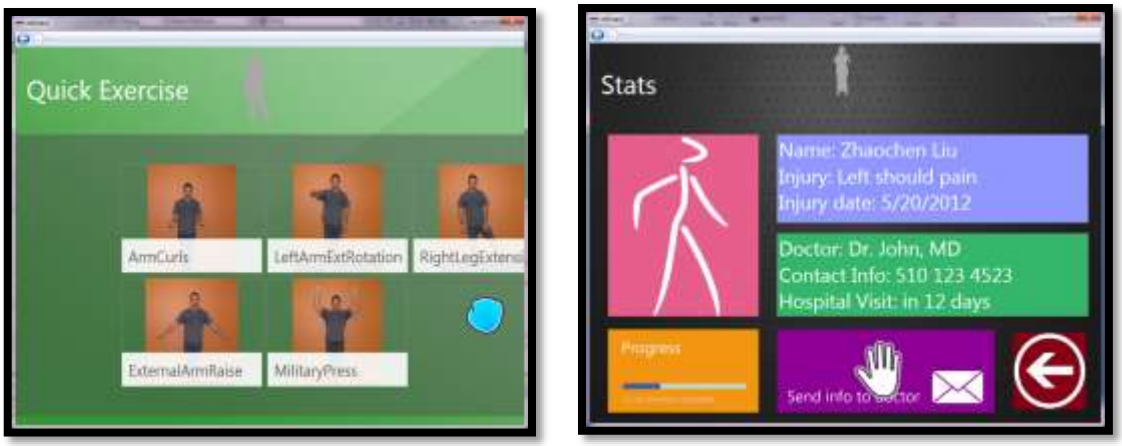

Figure 4-2 Exercise selection (left) & statistics (right)

Figure 4-3 Doctor receives the patient's info on his smart phone

During the exercise, the user is able to see his current body gesture as well as the target gesture. In addition, a 3D red arrow will tell the user what to do in order to move to the correct gesture. For example, if the user needs to move his left arm upward, an arrow pointing upward will be displayed [Figure 4-4].

Figure 4-4 A patient performing an exercise

4.2 Evaluation of the User Interface

The design process was iterative. I first adopted heuristic evaluation to identify any potential problem of the system. Heuristic evaluation is quick and cheap methods to judge the usability of the system based on some principles (the "heuristic"). Then, we did final rounds of user tests to access the overall quality of this application.

4.2.1 Nielsen's Heuristic Evaluation

I particularly used Jakob Nielsen's heuristic as my guidelines. Here are few samples to demonstrate my thought process and how I actually fixed the problems.

4.2.1.1 Sample Problem 1

Heuristic: Visibility of System Status

Violation: Originally, the system does not tell the user whether a Kinect was connected to the system. So, if the system goes un-responsive, the user might wonder what is going wrong, "did I click a wrong button?" or "how to fix this problem?"

Severity: Major usability problem: important to fix, so should be given high priority

Improvement: I added a Kinect status viewer that is able to show any problem the Kinect may be experience [Figure 4-5]. In addition, if the Kinect is working as expected, it will act as a mirror, showing the user’s gesture so the user knows exactly what his gesture looks like to the system [Figure 4-5].

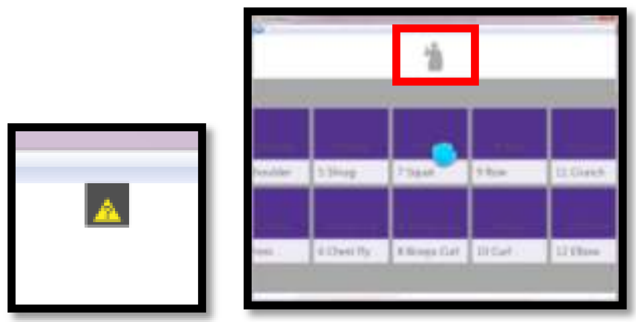

Figure 4-5 Kinect has a problem (left), show what the Kinect sees (right)

4.2.1.2 Sample Problem 2

Heuristic: Recognition rather than recall

Violation: while the user is doing an exercise, if he wants to pause the exercise, he can simply say “pause”. However, there is no way to for new users to know that this command even exists. Also, it is not a good idea to rely on user’s memory to let them remember a particular feature.

Severity: Major usability problem: important to fix, so should be given high priority

Improvement: I added an icon to indicate the user has the option to say the word “pause” if they want to take a rest for a while. This icon is in the corner so it does not take too much important space.

Figure 4-6 Added hint to reduce the user's memory load

4.3 Designing Gesture Recognition Application

The most important lesson I learned throughout this capstone project is some key principles of building user interfaces and creating good user experience. In particular, I learned a lot about gesture recognition applications and some rules of thumb when designing the user interaction for such applications.

4.3.1 Overall Design Principles

Choose the best input method in a scenario: some input methods are good at certain cases but bad at others. As a designer, you should consider which method will cost least overall effort to the user. For example, gesture recognition is not so good at text entry so you should not force the user to enter a piece of text by pushing the virtual keyboard with their gestures.

Let users know what the control options are: gesture recognition is a relatively new user interaction approach. Some users may get frustrated in the beginning if they don't know what to do. Be careful about this learning curve and provide some hints to help them get started.

Design-prototype-evaluate loop: it takes a lot of effort to make the interface and the user interactions just right. When I designed the UI for this capstone project, I applied different iterative design approaches (such as Wizard of Oz, User testing and heuristic evaluations) which I felt helped me to a great extent.

4.3.2 Feedback

Providing good feedback will make the user feel confident and help them understand what is happening in the application [9]. Unlike a keyboard and a mouse, gesturing is not that direct way to interact with the system. Especially, the fact that gesture recognition technology is still relatively new will bring the user more doubt while using a gesture-controlled application. The users will constantly wonder:

- Is the sensor working fine?
- Am I inside the view range of the sensor? What can the sensor see?
- What gestures can I perform now and when should I perform them?

Therefore, in order to make the user feel in control, applications should provide adequate cues and enough feedback. For example, in our Physiotherapy app, I added a viewer window that directly shows what Kinect sees. While the user is lifting his hand, a hand icon will be displayed according the user's movement. Plus, while the user is pushing down a button, the hand icon will gradually turn to purple to let the user know this is a valid action.

Figure 4-7 (a,b,c,d) hand indicator in different stages

Figure 4-7 (a) shows a hand icon when the user moves freely. Figure 4-7 (b) shows has some shadows and shows the user is engaging an intractable element. Figure 4-7 (c) shows the user is pushing a button. Figure 4-7 (d) shows a button has been pressed.

4.3.3 Gesture Design

Some gestures are innate and easy to figure out because the user intuitively knows them. For example, they user will try to point something to aim it or push a button to select it. However, some gestures such as performing a physiotherapy exercise do not make senses by themselves. Our system gives instructions to teach the users how to do them. It also provides real-time feedback by showing an arrow to let the user how exactly the user should adjust his gesture.

Moreover, it is critical to design appropriate user interactions in different situations. During an exercise, it is OK to ask the users to do some complex movement because we want have the user stretched. However, it is not a good idea to challenge the users outside the exercise (during the menu stages) because our goal is to let them start an exercise quickly.

4.3.4 Voice

In addition to gesturing, voice also makes the user interaction more natural. In our application, when a user records a new exercise, he can say ‘next’ to tell the recorder go to the next step. During a game, using voice commands also remove the ambiguity of whether the user wants to push the ‘pause’ button or wants to perform the exercise.

However, when design the voice command, one most important factor the designers have to think about is the confidence level. It is essential to find a good balance point between reducing false positive recognitions and making it difficult for the user to say the command clearly enough. Repeated testing with different users, especially at targeted environments where the application tends to run, will help the developers fine-tune the voice recognition system more easily.

5 Conclusions

We implemented a Kinect physiotherapy application using various technologies. We also performed market research and strategies studies to better examine the value of this technology on the mobile devices.

Particularly, I focused on designing and evaluating a user interface and tried to study various ways to improve the user experience of a gesture-controlled application. I really enjoyed the time working with my teammates as we helped each other a lot throughout the capstone project. More importantly, I am glad to work on UI&UX because I have strong interests in these areas. My 3-year goal is to become a product manager working on end customers facing mobile or web applications.

5.1 Future Work

During this project, we also investigated the possibilities of bringing the gesture recognition technology onto mobile devices. We have done ample research prove the market value of gesture recognition applications on a mobile device and we showed that such a capability will be helpful to users. Currently, this market is unoccupied but has a big potential. If Microsoft or other companies step into this market and come up with applications that have good use cases, it will sure benefit the company and the society to a great extend. Meanwhile it will propel the mobile industry and further introduce the computer vision technology to the general public.

In order to fully proving the value of gesture recognition technology on a mobile device, more prototypes are needed to cover more use cases. In addition, although the software part is ready for gesture recognition technology, the hardware might not be. This technology requires at least decent processing power, a durable battery and a special camera (infrared or dual cameras). Plus, to make a smart phone or tablet still portable, the size is still a very important factor. With the rapid evolving of the hardware industry, all of these can happen in the near future.

6 Bibliography

1. **Microsoft.** http://msdn.microsoft.com/en-us/library/zw4w595w.aspx. *MSDN.* [Online] [Cited: 2013 йил 12th-April.] http://msdn.microsoft.com/en-us/library/zw4w595w.aspx.

2. —. ‘Kinect Effect’ Magic Pushes Beyond the Living Room. *News Center, Microsoft.* [Online] [Cited: 2013 йил 12th-April.] http://www.microsoft.com/en-us/news/features/2011/oct11/10-31KinectEffect.aspx.

3. —. JointType Enumeration. *MSDN.* [Online] [Cited: 2013 йил 12th-April.] http://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx.

4. *Usability inspection methods.* **Nielsen, Jakob.** Chicago : CHI '95 Conference Companion on Human Factors in Computing Systems, 1995. pp. 377-378. 0-89791-755-3.

5. *Wizard of Oz studies — why and how.* **N. Dahlbäck, A. Jönsson, L. Ahrenberg.** 4, Linköping, Sweden : Natural Language Processing Laboratory, Department of Computer and Information Science, 1993 йил, Knowledge-Based Systems, Vol. 6.

6. Balsamiq Mockups. *Balsamiq.* [Online] Balsamiq Studios. [Cited: May 10th, 2012.] http://www.balsamiq.com/products/mockups.

7. Introduction to WPF. *MSDN.* [Online] Microsoft. [Cited: 2013 йил 11th-April.] http://msdn.microsoft.com/en-us/library/aa970268.aspx.

8. XAML Overview (WPF). *MSDN.* [Online] Microsoft. [Cited: 2013 йил 5th-April.] http://msdn.microsoft.com/en-us/library/ms752059.aspx.

9. Kinect for Windows Human Interface Guidelines v1.7.0. *MSDN.* [Online] 1.7. [Cited: 2013 йил 12th-April.] http://msdn.microsoft.com/en-us/library/jj663791.aspx.