March 2015

# Map My Trip: A Leap Motion Web App

Jeffrey Scott Signore
*Worcester Polytechnic Institute*

Lisa Anne Batbouta
*Worcester Polytechnic Institute*

Saraf Tasneem Rahman
*Worcester Polytechnic Institute*

**Project Number: MQP-GFP-1402**

# Map My Trip: A Leap Motion Web App

A Major Qualifying Project Report:

Submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Lisa Batbouta

Saraf Rahman

Jeffrey Signore

September 18, 2014

Approved by

**Professor Gary F. Pollice, Advisor**

## Abstract

Map My Trip is a travel application created using Leap Motion's motion sensing technologies to allow you to search for hotels with just a few circles of your finger.  Despite the initial objectives, the Leap Motion platform had not evolved to a point where the Web application support and support for this type of precision in a Web application was practical.  We were able, however, to use the Map My Trip application to research the strengths, weaknesses, and opportunities of the Leap Motion Technology in comparison with other websites and current Web technologies.

## Acknowledgements

We would like to thank our advisor, Professor Gary Pollice, for his guidance and expertise throughout this project. We would like to thank Professor David Brown for working with our team to identify the best interactive gestures for our application. Finally, we would like to thank all the participants in our user studies. Without all of their aid this project would not have been possible.

# Table of Contents

## Table of Figures

# Chapter 1: Introduction

In our everyday lives, people run into situations where they need hand-free technologies. For example, in the car while driving, users might need to be able to adjust the settings on his or her GPS without touching buttons. In other situations, users have found it more intuitive to make a gesture such as tapping the screen rather than typing on a keyboard[15]. In addition, these applications should be integrated with the Internet because of the websites we access as resources on a daily basis. Thus, Leap Motion, a motion sensor, offers users a new way for people to interact with computers. Leap Motion provides users with eight cubic feet of 3D touch that allows them to swipe through pictures on the Web, play air guitar, or travel the virtual world without touching their computer[9]. In addition, Leap Motion is progressing its technology by working with Mercedes-Benz to integrate these features in one of their upcoming cars[8].

## 1.1 Original Goals

Currently, in the Leap Motion App Store, there are no Web travel applications. However, it does have desktop travel applications, which use Leap Motion exclusively. A Web travel application using Leap Motion would give users more options for making travel accommodations. For example, users would be able to book a hotel or buy a plane ticket to their destination. Originally, we had set out to create a Web travel application using Leap Motion to improve the user's experience for planning a trip and booking hotels. We wanted to utilize Google Maps to take advantage of the Google Places API functionality, which can be used to find detailed information about hotels in a given vicinity. We planned to provide users with an easy-to-use interactive application that made users feel as if they were already at their destination. In addition, the idea was to create an interface from which users could easily plan trips without needing a mouse or keyboard, and have easy access to cost estimates, which would allow them to stay within their budget. Due to inherent issues with the current state of the technology, our project's focus shifted to an exploration of the strengths and weaknesses of Leap Motion as a Web technology through developing and testing a prototype application, user studies, and reviewing current Web technologies.

## 1.2 Original Requirements & Approach

In order to achieve our initial goal of creating a travel application using Web technologies with Leap Motion, we studied various motion gestures used on mobile applications. We developed our Web application that integrated Google Maps Application Programming Interface (API) and Leap Motion Software Development Kit (SDK) to display a Google map and utilized Leap Motion gestures. In order to create our Web applications, we used the technologies JavaScript, jQuery library and a local Node.js server.

## 1.3 Change of Focus Requirements & Approach

In order to resolve our subsequent problem of identifying the strengths and weaknesses of this application, we were required to:

1. Develop an additional Web application that did the same actions as the Leap Motion application, but used mouse click interaction instead of Leap Motion gestures.
2. Perform user studies to compare and evaluate the different Web application technologies.

## 1.4 Summary

The result of this Major Qualifying Project (MQP) is this report which chronicles the development of Map My Trip, our decision to move toward a research driven project, and the results of our user study analysis. Additionally, we have made future suggestions to advance the technologies of this product.

# Chapter 2: Literature Review

To understand the motivation behind Map My Trip using Leap Motion and research on the Leap Motion technology, this section describes what Leap Motion is, design choices for our application, and existing travel Web applications, as well as gesture decisions implemented in other applications.

## 2.1 Leap Motion

Leap Motion is a hardware sensor device that detects hand and finger movements, analogous to a mouse but requiring no touch. It provides a new way for users to interact with computers. Leap Motion provides users with 8 cubic feet of 3D touch that allows them to swipe through pictures on the Web, play air guitar, or travel the virtual world without touching the computer[9]. It is defined as a “motion capture” technology; it captures precise movements of parts of your hands during the motion and is able to display them on your computer[6]. This technology brings new innovations to the way we interact with computers, and the company plans to further improve on their product over time by implementing the product into future computers through connections with Hewlett-Packard and Asus[7]. In addition, Leap Motion is working with Mercedes-Benz to integrate motion gesture features into one of their upcoming cars[8]. These connections with these companies could progress Leap Motion technologies in the future.

### 2.1.1 Leap Motion Architecture

Leap Motion works on both Windows and Mac OS X operating systems and connects to the computer through a USB connection[15]. Leap Motion is able to retrieve and provide data through both a native interface and a Web socket. This lets developers have many options for programming languages including C# and Unity, Python, Java, Objective C, and JavaScript.

The native interface is provided through a dynamically loaded library that tracks data to your application. The architecture of the Leap Motion native interface can be seen in Figure 1 below.

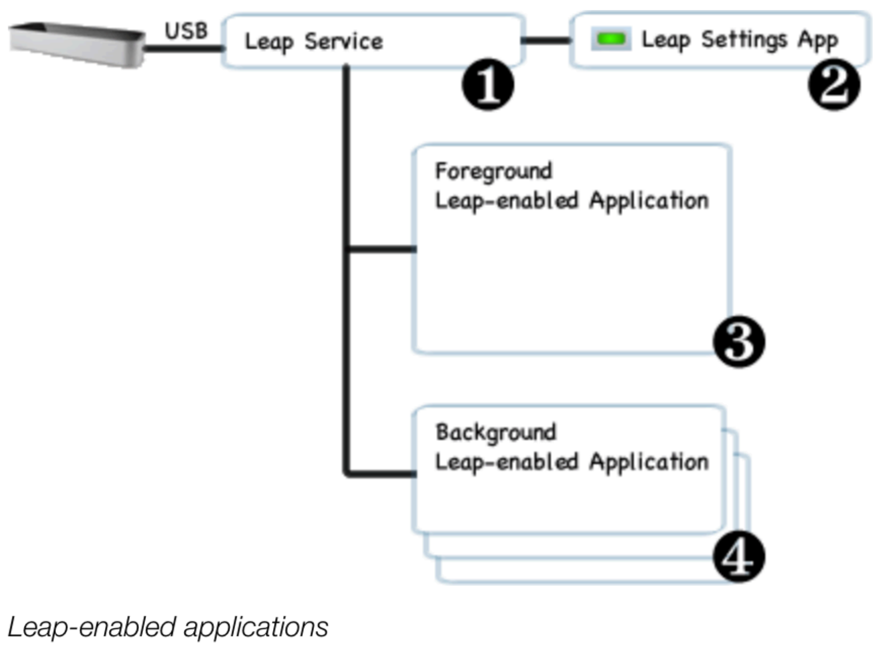


Figure 1: Leap Motion Native Interface[9]

As seen in Figure 1:

(1) The Leap Motion Controller sends gesture tracking data to the Leap service through the USB.

(2) A user can configure the Leap Motion installation process as well as access settings and other tools that are separate from Leap Motion applications.

(3) By Default, the application retrieves data when the application is open in the foreground of your desktop.

(4) However, you can set up your Leap Motion to also retrieve data when the application is running in the background and you are doing other work on your desktop.

The Leap Motion WebSocket server on the localhost domain at port 6437 provides tracking data in the form of JavaScript Object Notation (JSON) messages. The architecture of the Leap Motion WebSocket server can be seen in Figure 2 below.

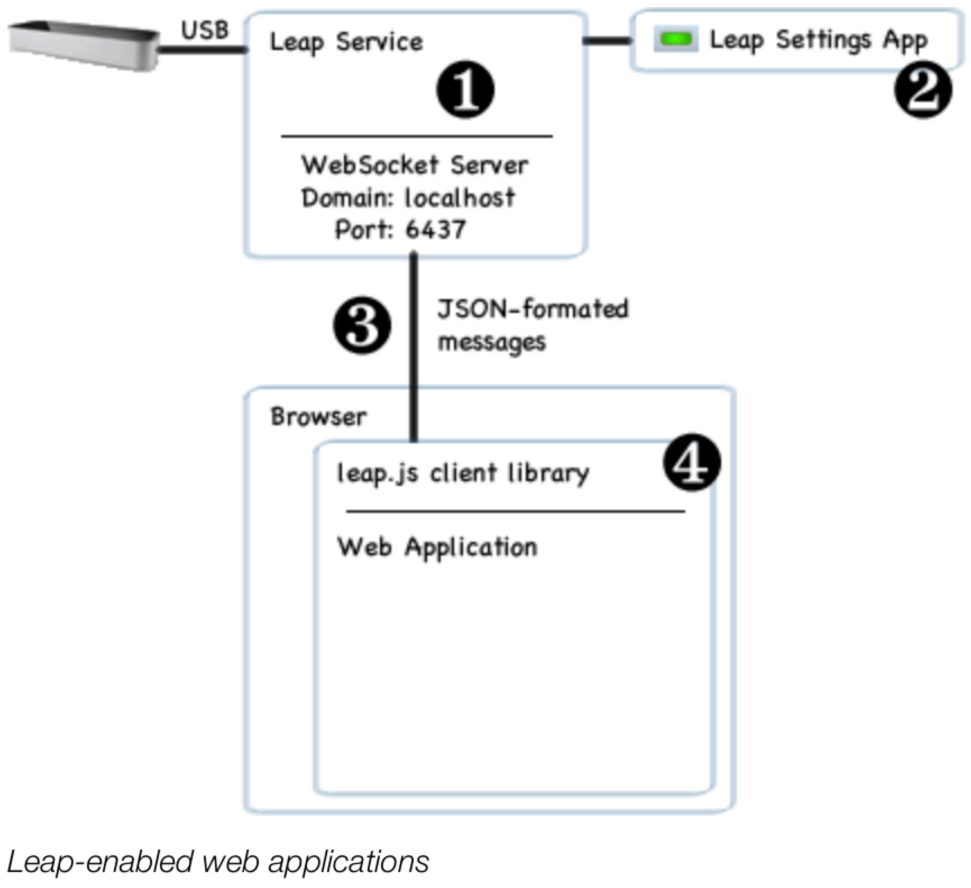


Figure 2: Leap Motion WebSocket Server[16]

As seen in Figure 2:

(1) The Leap Motion Service is the Leap Motion WebSocket server that provides a WebSocket server listening on its localhost domain, http://127.0.0.1:6437, on port 6437.

(2) Through the Leap settings, the user has the ability to enable and disable the WebSocket at any given time to enable and disable gestures.

(3) The Leap Server then sends tracking data in the form of JSON and the application configures messages to send back to the server about the gestures.

(4) The leap.js client library establishes a connection to the server and allows for the consumption of JSON messages for developers to display the data as he or she chooses[5].

### 2.1.2 Leap Motion Language Choice

The Leap Motion WebSocket server environment, using JavaScript, allowed us access to websites outside of Leap Motion applications. In addition, Google Maps API was written in JavaScript. Therefore, no extended libraries or access points were necessary to use the Google Maps API resource. Lastly, by using JavaScript we would be able to redirect users of our application easily to other travel websites to book their final hotel preference.

### 2.1.3 Leap Motion Web with Node.js

Node.js is a server side and networking runtime environment that maximizes throughput and efficiency of real time applications. It does this through asynchronous I/O library for files, sockets, and HTTP communication. It is made for data intensive real time applications that run across multiple devices[10]. Due to the fact that Node.js is able to provide quick data intensive transactions, this would enhance the speed of data being changed on the Google Maps API on our website.

### 2.1.4 Leap Motion Existing Applications and Gestures

Most of the applications currently on the Leap Motion App Store limit the number of gestures to one or two at most. The limited number of gestures prevents the sensor from confusing gestures that may be hard to differentiate between and allows for smoother navigation of the application[1]. Additionally, the number of gestures provided for Web development is lower than those provided for Desktop Applications.

## 2.2 Leap Motion as a Travel App

Leap Motion as an interaction mechanism for this travel application allows users to navigate with the computer in a unique way and find effortless ways to interact with a touch less screen. It provides a visual representation in which users may navigate between the hotels of the world, and which will help users understand where they are trying to go and the hotels in the surrounding area for his or her trip. It allows users to identify attractions based on relative distances to surrounding hotels.

### 2.2.1 Existing Travel Applications

Today, there is only one travel application in the Leap Motion App Store. This application identifies some of the strengths Leap Motion has to offer for travel planning. However, the applications do not integrate a way for users to search through hotels and switch to another website to purchase your ticket. The closest Web applications to this are Travel Seeker by Leap Motion developers, and RoadTripper, a website.

Travel seeker uses Leap Motion API to look at different vacation locations based on a certain price range. Once you are zoomed in enough to see a map of the United States, you can see price stickers for many major cities throughout the country. Prices on this application do not reference any given hotel or a breakdown of what is included in the price. In addition, this map does not allow you to zoom in and see locations up close.  Nevertheless, this application provides users an average cost of traveling to locations around the global and brings a new visual spin to planning your next vacation.

Aside from Travel Seeker, there are no other Leap Motion travel applications. The Leap Motion market is just starting to expand and it will take time before new applications become available specifically for Leap Motion. In the meantime, there are short-term solutions such as the Leap applications Touchless and Shortcuts, which have been developed to create gestures allowing users to navigate the Web using just Leap gestures instead of a mouse.

Touchless and the Shortcuts Leap Motion applications can be used in conjunction with travel websites such as RoadTripper, www.roadtripper.com, or other websites as an alternative to Travel Seeker. By using the Leap application and RoadTripper you are able to virtually travel the world utilizing Google Maps API and plan various trips using different routes. The application also allows you to save and share these routes. This application provides visual representation of different hotels, attractions, restaurants, and other areas of interest giving users the information needed to plan his or her trip accordingly. However, this is not a long-term solution because many travel maps can be rigid and do not promote the fluid movements of Leap Motion or quick interactions without typing on the keyboard or using the mouse.

# Chapter 3: Methodology

After first researching the existing travel applications using Leap Motion, we designed our own Leap Motion travel application: Map My Trip. We used Map My Trip as a prototype for user testing. We created a standard Web application that was similar to our Leap Motion application and then performed user studies with our final prototype to evaluate the strengths and weaknesses of using Map My Trip versus using other Web applications.

## 3.1 Requirements

In order for Map My Trip to be as efficient as possible, we determined a list of requirements for the application:

1. Implement gestures that are intuitive to users
2. Provide an intuitive flow of movements

Then, we shifted our focus to determine a list of requirements necessary to analyze the strengths and weaknesses of using our application:

1. Develop an application, similar to the Leap application Web page, using the mouse and keyboard
2. Perform user studies to compare and evaluate the different Web application technologies

## 3.2 Implementation of gestures

In order to select gestures that were familiar to the user, we researched pre-existing gestures for Android and iPhone applications. We also consulted Professor David Brown, a professor of Computer Science at WPI, who is an expert in the area of Human-Computer Interaction, for suggestions of common gestures. After this meeting, we chose a series of gestures that were intuitive and natural to the user, giving priority to those that were already implemented using the Leap Motion device. Once we accumulated our list of gestures, we implemented the Google Maps API. We researched extensively to find how Google Maps API interacted with Leap Motion. Once we understood that, we mapped each gesture to a specific action, thus creating our final application.

## 3.3 Gestures and Problems

The gestures used in our application are CircleGesture, SwipeGesture, KeyTapGesture, and a custom gesture. The CircleGesture was the first gesture to be implemented. This gesture can be seen below in Figure 3.

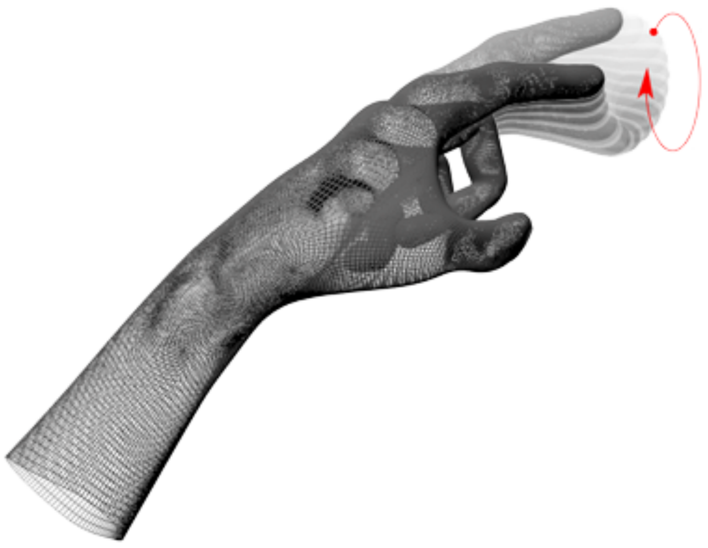


Figure 3 : CircleGesture using Leap Motion[3]

Originally, circling clockwise with one finger was used to zoom in, and circling counter-clockwise was used to zoom out. We decided to map the circle gesture to a different action because it would be more intuitive to be used for circling an area than for zooming in and out.

The circle gesture now causes hotels to appear in the approximate center of the on-screen area of the map.

In the beginning of our project, the circle gesture was interfering with our custom gesture. In order to compensate for the interference, we changed the circle gesture to only work within a given radius. We had programmed the gesture to read only one finger, but despite this, Leap was still translating an entire hand movement to be a circle gesture. In order to combat this, we used the stabilize hand position which allowed Leap to focus solely on the movement of one finger.

The SwipeGesture, as seen in Figure 4 below, was used to zoom.

We faced a number of challenges while implementing this gesture. First, the Leap Motion device would often read one swipe to be multiple swipes, thus zooming in or zooming out to a much higher degree than the user had intended.

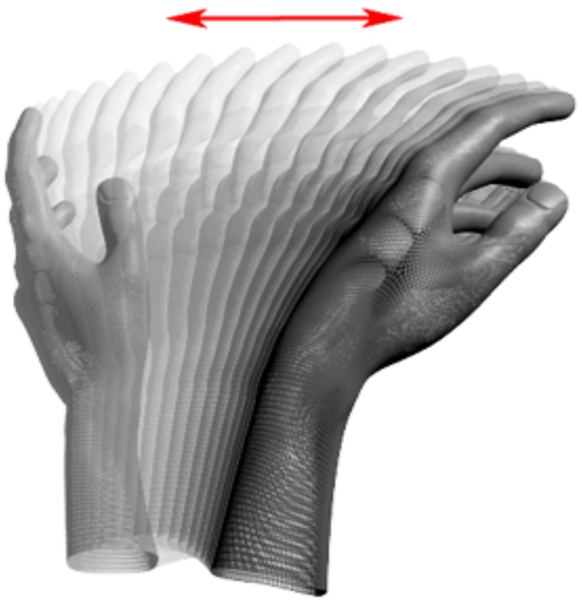


Figure 4: SwipeGesture using Leap Motion [14]

The gesture would also become unstable if more than one finger was being detected. In order to combat these issues and enhance the swipe gesture, we set a minimum distance and had the Leap Motion device detect the user's palm instead of his or her fingers. This allowed the gesture to stabilize and be more effective.

The KeyTapGesture, as seen in Figure 5 below, was first used as a 'click' function to allow users to select a hotel and view its information.

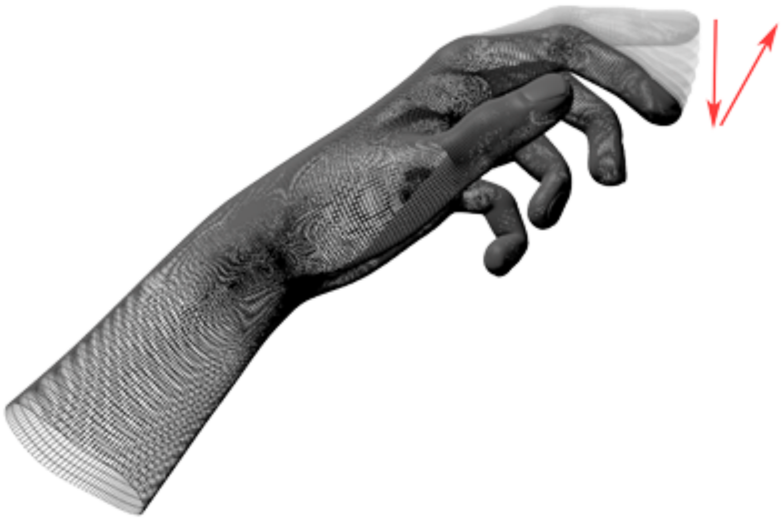


Figure 5: KeyTapGesture using Leap Motion 5

Unfortunately, translating the coordinates from latitude and longitude, which are Google Maps API coordinates, to x-axis, y-axis, and z-axis, which are Leap Motion coordinates, proved difficult. Thus, as an intermediate step, we changed the mapping of the KeyTapGesture from acting as a click gesture to a zoom gesture to see if this gesture could be utilized in a more efficient manner. In order to do this, the user would tap on the right side of the screen to zoom in and on the left side to zoom out. It was not very effective or consistent, and we decided that it would be difficult for a user to perform the gesture if they were not already accustomed to using

Leap Motion because this is not a common gesture on other mobile applications. We were ultimately unable to solve the issue with mapping the coordinate system, and so the KeyTapGesture returned to being used as a click gesture.

The ScreenTapGesture, as seen in Figure 6 below, was first used as a means to tap on a hotel. However, we decided that the KeyTapGesture would be more intuitive than using ScreenTapGesture.

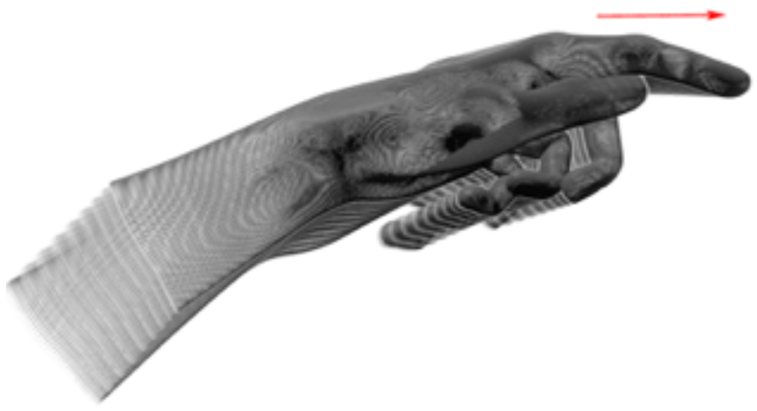


Figure 6: ScreenTapGesture using Leap Motion[13]

We then used ScreenTapGesture as a zoom gesture. In order to zoom in, users would tap on the right side of the screen, and to zoom out, users would tap on the left side of the screen. We ultimately decided that ScreenTapGesture would not be used because the gesture was not familiar and less responsive than the other gestures.

Our custom gesture was a fist, which we used to navigate the map. In order to drag the map to any location, the user could simply make a fist and drag the map in whichever direction he or she desired. This gesture is one of our most stable and consistent gestures throughout the development of the application.

## 3.4 Project Transition

As we created the Map My Trip application we encountered browser problems. The application was not able to handle two interaction boxes, which are box-shaped regions that are completely within the field of view of the Leap Motion controller, in different browsers. We tried a local server as well as Heroku, a cloud application platform that allows developers to build and deploy Web apps, to see if the problem persisted only on one version of the server. This was not the case. The next step was to create the same Web application as the Leap Motion Web page with mouse gestures and keyboard interaction. This application, that used mouse gestures, worked in all Web browsers. Therefore, we recognized that Leap technology was not supportive of all Web browsers and we had to transition our focus to understand the strengths and weaknesses of Leap Motion.

## 3.5 User Studies

In order to test our application and evaluate it in comparison with other travel applications, we needed to conduct user studies. We determined that for optimal testing, we needed three different testing groups. The first group tested our own application in which they used gestures to control the application. The second group tested our version of our application in which the users use mouse gestures, as opposed to CircleGesture, KeyTapGesture, fist gestures, etc., to control the application. The last group tested Booking.com, an online travel application that does not involve Leap Motion at all.

For each of the user studies, we provided the test subject with a brief introduction and some additional materials before starting the study. More specifically, the oral introduction that was given was an Institutional Review Board (IRB) approved method of obtaining consent as well as a quick way of informing the user about our project. The additional materials provided to the user included a list of tasks, which they must try to perform throughout the duration of the study. This list changed depending on the user study group due to the different capabilities of each technology, but the basis of the lists were similar in order to test the application at hand for the scope of the project.

In order to obtain participants for our survey, we reached out to the campus population. We focused on getting students for the user studies because this best aligned with the target audience for a Leap Motion application. The majority of our test subjects included members of social organizations on campus. We emailed these social groups asking for participants for our study, and we also asked the group members in person during popular meeting times. There were also other participants outside of these social organizations who were reached through face-to-face communication in the Campus Center.

For the group that used the Leap Motion application with gestures, instructions as well as images of the gestures were provided to inform the user how to perform various actions using the associated gestures. The user studies were also recorded using screen capture technology in order to be reviewed and analyzed at a later time. Lastly, when each user finished the tasks assigned for the study, he or she filled out a short survey so that we could gain insight on their experience and later analyze these experiences.

For the other two groups, the participants were simply given a list of tasks similar to those assigned to the Leap Motion group, and were also asked to complete a short survey afterward.

# Chapter 4: Results and Analysis

## 4.1 Limitations of Leap Motion

We shifted the focus from a usable product to a research project due to the number of issues we encountered with Leap Motion. Aside from facing problems with the gestures, we also realized that there were many issues with the Leap technology interacting with other types of technologies.

Leap Motion is an exceptional product for many desktop applications; however features built for the Leap Motion JavaScript Software Development Kit (SDK), in its current form, is quite limited for development and for user experience.  The JavaScript SDK does not have a large number of gestures, the precision of gestures in the application is limited. In addition, both the Wi-Fi connection and the Web browser choice can change the user experience.  There are instances where the Leap Motion SDK does not support all activity that a user may wish to use in which case the user may revert back to the mouse and keyboard. All of the limitations listed can be exemplified through our Travel Application website. These limitations are what ultimately caused us to change our project to be more research based than application based.

### 4.1.1 Gestures

The precision of the Leap Motion device using the Web API is not accurate enough to perform many tasks that require significant precision, such as to select a small point on the screen. For instance, trying to use the KeyTapGesture.  We had originally chosen the KeyTapGesture to click on a hotel icon because it seemed intuitive. However, doing so was arduous due to the small size of the markers as well as the proximity of markers to each other, requiring a high level of precision. Due to the nature of gestures in general, it was difficult to pinpoint the exact location where the gesture was made since a gesture itself is a movement. This meant that when a user would make the gesture to select a hotel, the point being selected would change while the gesture was being made. The result of this was either that no hotel or an undesired hotel was selected. This held true not only for selecting hotel markers, but also for any gestures that depended upon any amount of precision.

Another precision related issue concerned the number of fingers that were detected by the Leap device. Many gestures required only one finger to be detected by the Leap Motion device. This presented a serious usability issue. For example, at one point the swipe gesture was only to be recognized when a single finger was on the screen, as it could help differentiate between similar gestures. However, oftentimes the Leap device would inaccurately detect the number of fingers. This resulted in some cases where a correctly complete gesture occurred, but no gesture was registered. In addition, there were other cases where a gesture was supposed to occur a single time, but multiple instances of the gesture were registered. For example, when we had implemented the ScreenTapGesture to zoom in or out, we had created the function to zoom in or out when only one finger was making the gesture. When a ScreenTapGesture was performed with multiple fingers instead of just one, the action would be performed multiple times instead of just once or not at all.

There were also issues related to the limited number of available gestures and how these gestures interacted with each other. When originally brainstorming gestures and their correlating actions, we had thought that taking well-known gestures and translating them into Leap gestures would be the best option because most users would already be familiar with them. For example, when planning the zoom feature, an intuitive gesture would have been moving the user's fingers closer together or farther apart, similar to what is currently used for Android and iPhone zooming. In order to translate this motion to a Leap gesture, we thought of having users use two hands instead of two fingers. However, using opposite gestures to be representative of opposite actions, while familiar was almost impossible to implement using Leap. Using the zoom example from before, if a user wanted to zoom in, they would move their hands closer. However, if they wanted to zoom in multiple times, they would have to take their hands out of the field of view for Leap each time they wanted to zoom in to reset the gesture. In order to counteract this, specific constraints had to be made in order to have a usable product.  However, doing so reduced the ease with which the application was used.

Another option, which was more viable, was to have two gestures for opposite tasks that are not direct opposite representations of each other. For example, to zoom-in, the gesture was swiping horizontally, while the zoom-out gesture was swiping vertically. Ultimately, we had to weigh how intuitive the gesture was against its effectiveness. Although this reduced the usability of the application, we opted for gestures that were worked instead of ones that may have been more intuitive. We also faced issues with specific aspects of Leap Motion, such as creating a custom gesture and limitations with the interaction box, a box-shaped region that is completely within the field of view of the Leap Motion controller.

The hover gesture we had wanted to create would allow the user to hover over a hotel marker and view its information. If the user's hand was in place for a certain number of seconds, Leap would display the hotel information. After three weeks of attempted implementation, various forum posts, and posting on the Leap Developer site as well, we identified that this gesture was too time-consuming and that the KeyTapGesture would be capable of achieving the same action without investing nearly as much time. The reason creating a gesture was so difficult was because there was little to no documentation available on the Leap Developer website and, from our research, it seemed that no other developer had created a custom Leap gesture, although many had tried.

For our project we also had to use multiple interaction boxes - one for Google Maps and another for the other parts of the page. This was necessary because an interaction box of the whole view does not account for the interaction occurring within Google Maps. Additionally, we were forced to create two separate interaction boxes in order to enable Leap Motion gestures in two locations; one for the possibilities sidebar and one that exists within Google Maps

### 4.1.2 Internet/ Browsers

An issue with Leap Motion as a Web application is that a user's experience is entirely dependent on Internet connectivity, with a weaker Internet connection there could be an extended lag in the gesture motion. This poor Internet connection often resulted in the gesture becoming unresponsive or unpredictable and made it difficult for the user to move the icon because it could take several seconds to refresh an image each time any move was made. The severity of these Internet connection issues could be overcome depending on location.

Even if users have a good Internet connection, Leap Motion's Web Application is only accessible in Internet Explorer and Safari, without making any changes to the user's browser. Leap Motion Release Notes and Known Issues: V2.2.3.25971 notes the following errors with browsers including Mozilla Firefox and Google Chrome:

1 Secure WebSocket on Firefox requires Firefox version 32 or higher

2 The Close Firefox dialog window sometimes does not come to the foreground after installation (while Firefox was running)

3 In order for Secure WebSocket to work immediately on Firefox, please close Firefox prior to installing 2.2.0 or close when prompted

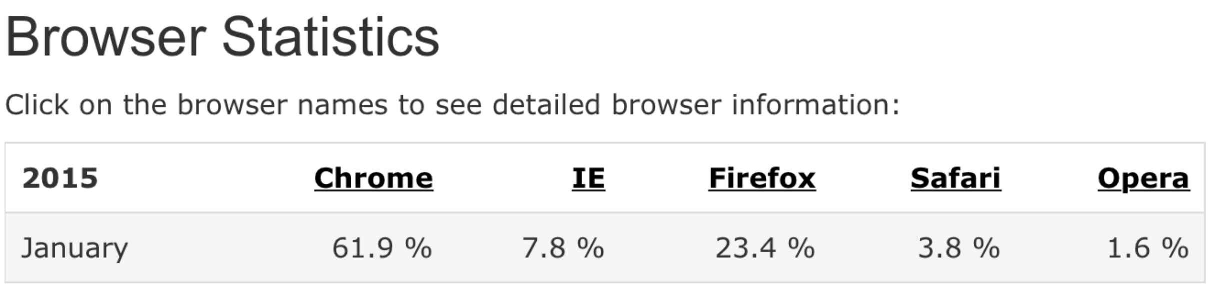
4 Secure WebSocket (TLS) on Linux Chrome requires running with --ignore-certificate-errors (caution: this also affects non-WebSocket services)

5 Chrome on Windows 7 sometimes fails to respond to emulated touch points – to fix this, click inside the program with a mouse

6 JavaScript client library and WebSocket protocol do not support setting gesture parameters

These errors mean that Google Chrome and Mozilla Firefox can be used with the Leap Motion device by enabling browser security settings. In addition, in the travel application project, it was identified that only one interaction window could be used per Leap Motion application. During implementation, this was a limiting feature because we needed two interaction windows – one for the Google Maps View and one for the hotel selection side bar. Google Chrome and Mozilla Firefox could not recognize when to switch from one interaction frame to another. In contrast, Safari and Internet Explorer were able to create this transition.

Although Safari and Internet Explorer can be used by all platforms, many users do not use these two Web browsers and they are not well supported for all operating systems.  In our case studies we have found that Internet Explorer creates a lag in gesture interaction and Safari is as supported for Windows platforms as it is for Apple platforms. In addition, as seen in Figure 7 over 80% of the Internet users use Mozilla Firefox or Google Chrome as their browser of choice.



## Browser Statistics

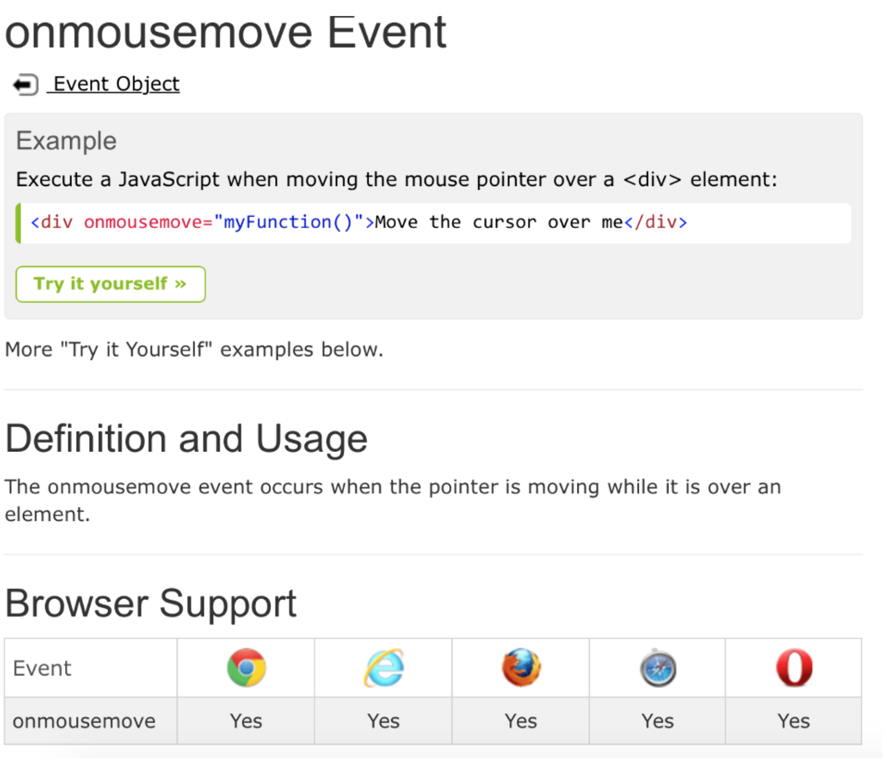
Click on the browser names to see detailed browser information:

| 2015 | Chrome | IE | Firefox | Safari | Opera |
|---|---|---|---|---|---|
| January | 61.9 % | 7.8 % | 23.4 % | 3.8 % | 1.6 % |

Figure 7: Browser Compatibility[2]

Thus, most users would often be forced to download additional Web browsers in order to use Leap Motion Web applications.

In contrast, this travel application could have been created using mouse gestures instead of Leap Motion gestures without sacrificing any of the features of the application. All browsers as seen in Figure 8 below support mouse gestures:



## onmousemove Event

Event Object

Example

Execute a JavaScript when moving the mouse pointer over a <div> element:

```
<div onmousemove="myFunction()">Move the cursor over me</div>
```

Try it yourself »

More "Try it Yourself" examples below.

## Definition and Usage

The onmousemove event occurs when the pointer is moving while it is over an element.

## Browser Support

| Event | Chrome | IE | Firefox | Safari | Opera |
|---|---|---|---|---|---|
| onmousemove | Yes | Yes | Yes | Yes | Yes |

Figure 8: Mouse Browser Information[9]

The mouse gestures can account for all of the same available Leap gestures for Web development and work on any browser. This has been proven in the travel project user studies. In addition, over time, a library could be built to support all of the gestures similar to Leap in order to create an easy build for future developers.

## 4.2 Benefits of Leap Motion

An example of a gesture that worked quite well with our application was the gripped hand gesture. This gesture occurred when the user would make a fist with their hand and 'drag' the map to their desired location. In comparison to the other gestures, this gesture worked almost flawlessly because the user had almost complete control.

Leap has a number of applications available in their app store. The applications that use only one or two gestures work very well because the Leap Motion device is able to easily distinguish the different gestures and therefore, shorten the response time. Another benefit of Leap Motion is that interaction boxes can be useful, provided only one interaction box is used in each application.

## 4.3 User Study Results

During our usability testing, we asked forty people to participate in our study. These forty participants were random members of the WPI Community. Participants were read a script (in Appendix A) prior to testing explaining why the study was necessary and what information would be used. We had three different groups for which the user could participate:

Group 1: Application using Leap Motion

Group 2: Application using mouse and keyboard

Group 3: Tradition hotel booking site

For groups 1 and 2, we used our travel application to test the usability of the gestures. For group 3, as a control group, we used booking.com.

### 4.3. 1 Application using Leap Motion

During our user studies, sixteen people agreed to participate in this study. The instructions that the users were provided with can be seen in Appendix D. Of the sixteen, eleven had not used or heard of Leap Motion. We asked users to rate how difficult the trial was, on a scale from 1-5 with 1 being the easiest.

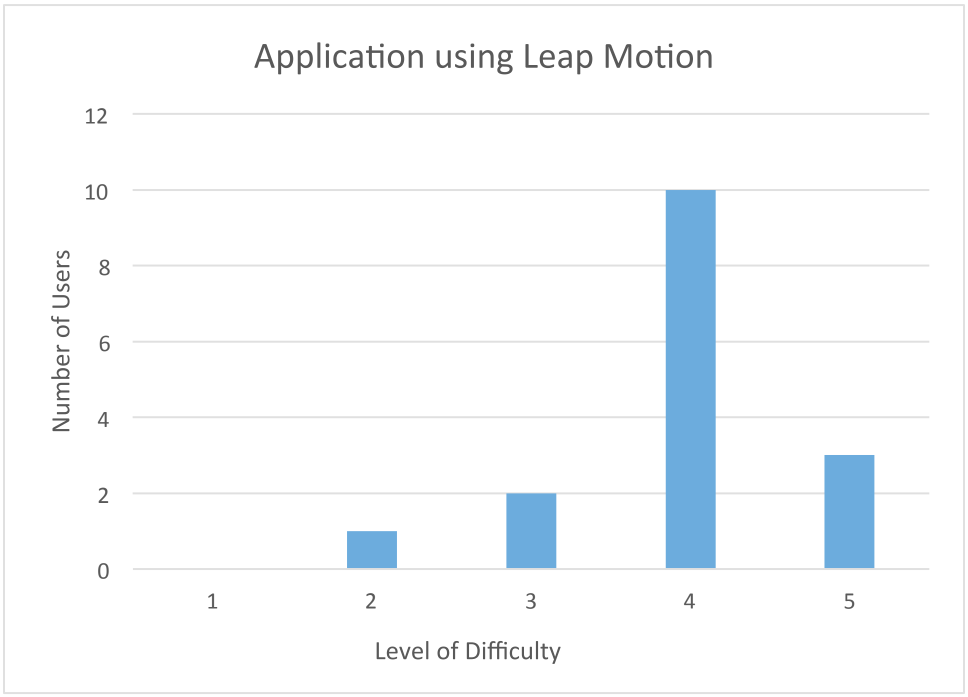


Figure 9: Map My Trip Leap Motion User Study

During the screen capture, we were also able to time each trial. The fastest trial was three minutes and thirty-nine seconds, while the longest trial was seventeen minutes and thirty-one seconds. The average for all sixteen trials was a little longer than ten minutes (10 minutes and 2 seconds).

In our survey, we asked users if the gestures were natural, and if not, which ones could be improved upon. The chart of the results of this question can be viewed in Figure 10 below.

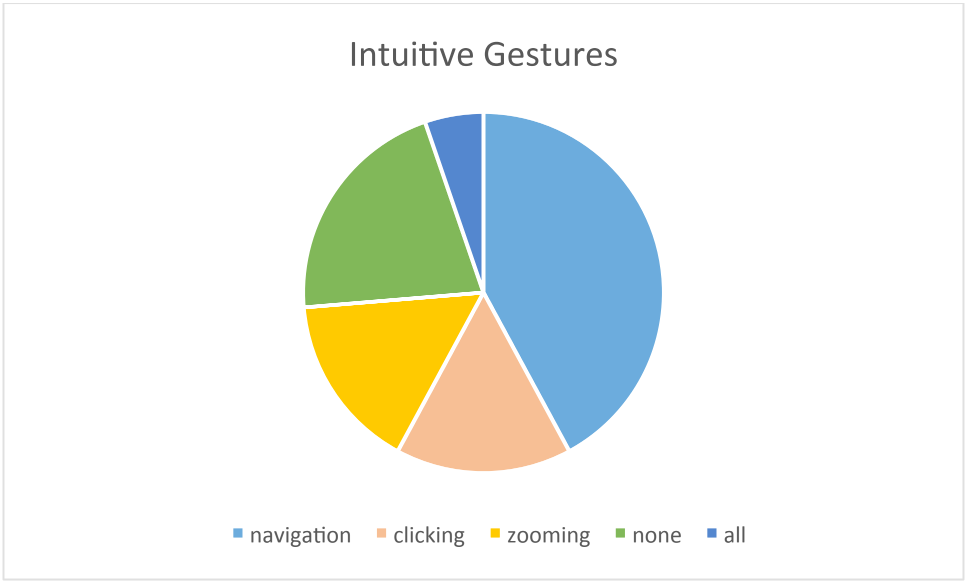


Figure 10: Chart of Intuitive Gestures

Of the six participants that made comments about the zoom gesture, 50% thought the gesture was intuitive, while the other 50% thought that it could be improved upon. There were also comments saying that selecting hotels and adding/taking away hotels in the side bar was especially difficult.

### 4.3.2 Application using mouse and keyboard

Twelve people agreed to participate in this study. All of the users were familiar with and had used Google Maps prior to this trial. In this user study we also provided instructions for users that can be seen in Appendix E. We asked the users to rate the difficulty of the trial, on a scale from 1-5 with 1 being the easiest.

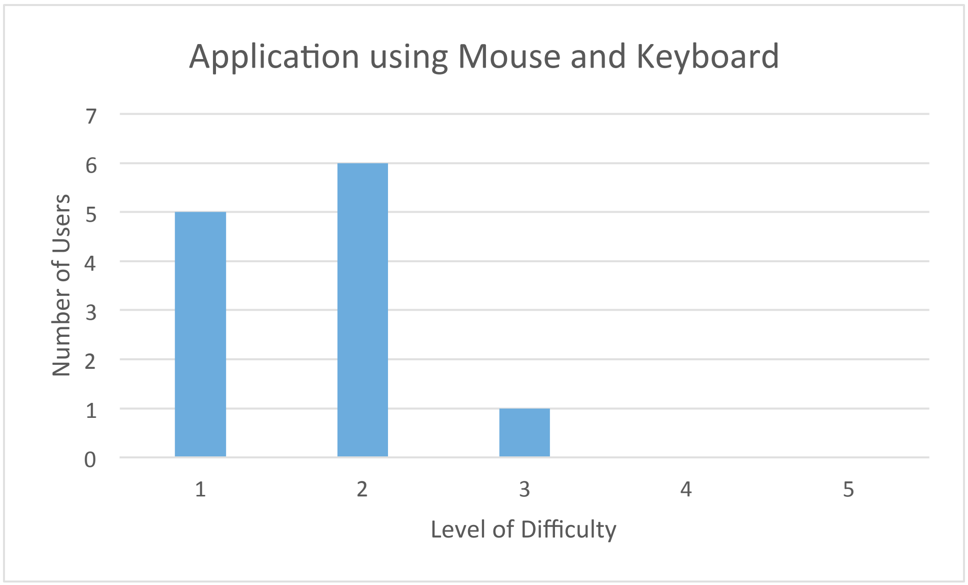


Figure 11: Leap Motion with Mouse User Study

Of the twelve, ten said that the gestures were intuitive. When asked if they would use the application in real life, two said they would not use this application to book hotels. Two suggested that the application "have a confirmation animation after you add a hotel to your list" and that "the appearance of selected hotels was awkward to find".

Aside from the survey feedback, we were also able to time each trial for each user. The fastest time was one minute and thirteen seconds, while the longest time was three minutes and fifty-eight seconds. The average time for all twelve trials was two minutes and six seconds.

### 4.3.3 Using a traditional hotel booking site

Twelve people agreed to participate in this survey.  The instructions for this survey can be seen in Appendix F. We asked the users how difficult the trial was on a scale of 1-5 with 1 being the easiest, and if they would want or need a different way of interacting with booking.com.

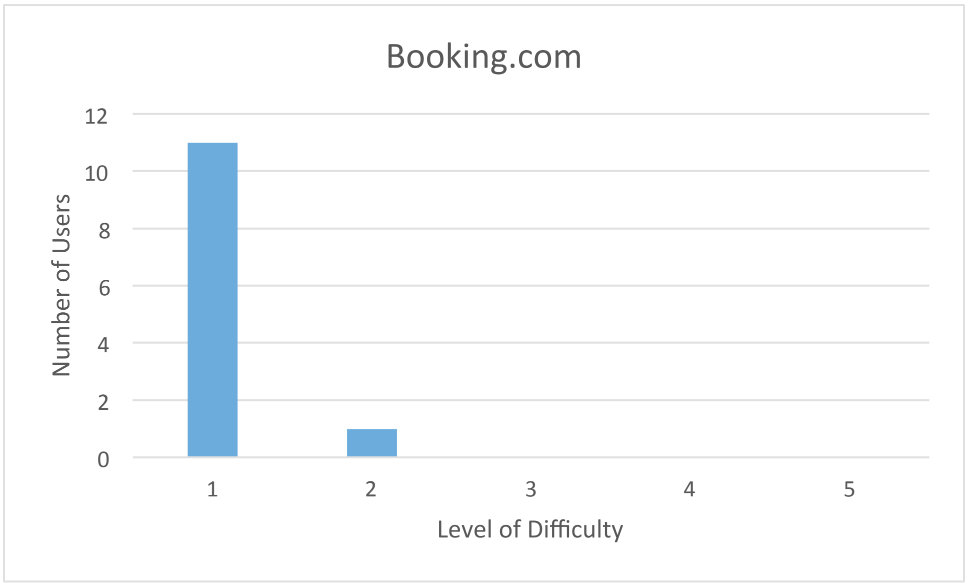


Figure 12: Booking.com User Study

All participants said they would not need or want any other way of interacting with the website. During the screen capture, we were also able to time each trial for each user. The fastest time was fifty-six seconds, while the longest time was one minute and forty-five seconds. The average time for all twelve trials was one minute and seventeen seconds.

## 4.4 Analysis

### 4.4.1 User Study Analysis

For the Leap Motion Web Application trials, in section 4.4.1, the majority of our users made comments that the application was incredibly frustrating due to Leap's inability to recognize gestures accurately as well as the browser issues that arose during every trial, regardless of group. In order to provide some level of consistency, we had users test the application using Mozilla Firefox because, of all the browsers, it had the minimal response time. However, when users were asked to move a hotel from the ‘Possibilities’ into the ‘Choices’ section, we had to switch into Internet Explorer because Mozilla Firefox did not allow users to interact with the sidebar. After switching to Internet Explorer, users were even more frustrated because it took Internet Explorer greater than five seconds to recognize gestures. The README seemed to aid users in completing the tasks for the app as users referred to the README multiple times during each trial.

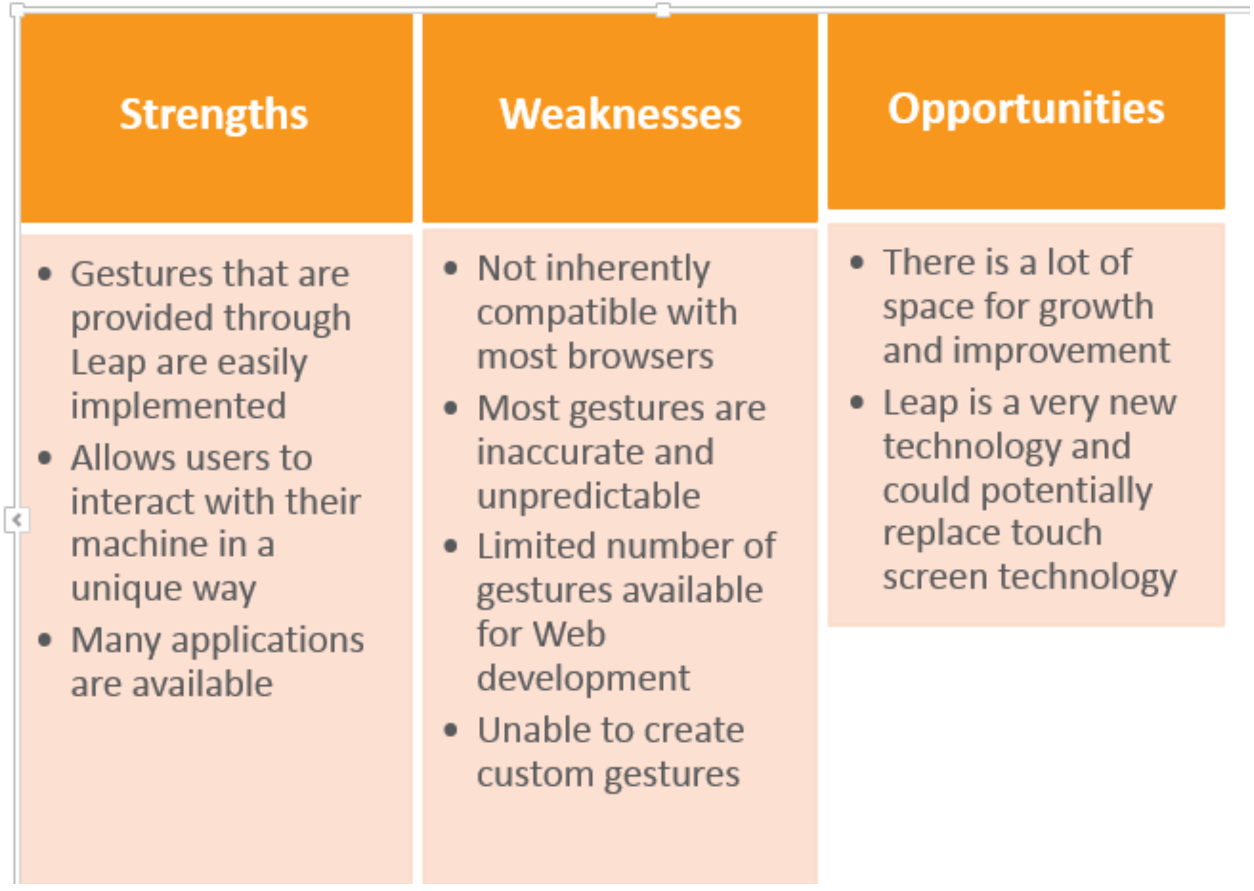
For the application using mouse and keyboard trials discussed in section 4.4.2, users mostly had an issue familiarizing themselves with the interface. Because of this, we concluded that users had an issue with the interface more than an issue with the technology. The users for these trials were not frustrated with the technology since Leap Motion was not used during this trial. Figure 11 confirms the fact that mouse gestures are familiar to our target group and that the gestures are not the problem. In addition, from the studies, when the user was given the option to navigate using gestures or type the name of the city in the search bar the majority of users chose to use the keyboard and search bar.

For the Booking.com trials discussed in section 4.4.3, users did not seem to have any issues. There were one or two trials where users struggled a bit to find specifically 5-star hotels, but all trials were still completed within two minutes.

In addition to all of the users in the Leap Motion Web Application trials becoming frustrated with the technology, these trials also took the longest time. The average time for the Leap Motion Web Application trials was ten minutes and two seconds. Compared to the mouse and keyboard application trials and the Booking.com trials, Leap took 7 minutes and 6 seconds longer and 8 minutes and 45 seconds longer, respectively. After comparing the three surveys, we concluded that Leap Motion, in its current state, would not be a useful or desired technology for our use.

### 4.4.2 Strengths, Weaknesses, and Opportunities of Leap Motion

After gathering all information from the user studies and analyzing this information we concluded our research noting the strengths, weaknesses, and opportunities of Leap Motion in it’s current state. This information can be seen in Figure 13 below.



| Strengths | Weaknesses | Opportunities |
|---|---|---|
| • Gestures that are provided through Leap are easily implemented<br>• Allows users to interact with their machine in a unique way<br>• Many applications are available | • Not inherently compatible with most browsers<br>• Most gestures are inaccurate and unpredictable<br>• Limited number of gestures available for Web development<br>• Unable to create custom gestures | • There is a lot of space for growth and improvement<br>• Leap is a very new technology and could potentially replace touch screen technology |

Figure 13: Strengths Weaknesses and Opportunities of Leap Motion

# Chapter 5: Conclusions and Recommendations

## 5.1 Benefits

A few of the benefits of our application were that many of the deals and discounts of other travel applications were removed and the application was simplified overall. In the end, the simplicity of this application allowed us to focus user tests around gesture navigation and compare this Leap motion technology and our Web application that uses Leap Motion to current travel Web applications and mouse gestures.

During the span of our project, we spent a lot of time attempting to find answers to the problems we encountered, and were unable to find sufficient solutions. Since Leap Motion is a relatively new technology, the value of documenting the benefits and downfalls of the device are invaluable to other developers looking to create a Web application using Leap Motion. This documentation may even be valuable to developers of Leap Motion so that they can address these issues in the future.

## 5.2 Suitability

Leap Motion is certainly useable with other applications, especially games, where precision is not such a crucial factor. For example, there is an application available in the Leap App Store called “Duck Hunt” where users can ‘shoot’ at ducks by moving their finger in an upward motion. Although this game only utilizes one gesture, it is fairly obvious that, even if the user performs a gesture correctly, the gesture will go unrecognized or not perform the action at the correct time. These unreliable responses are more acceptable for applications like Duck Hunt, where it is not as important that every single gesture be accurate and predictable.

With an application like ours, however, being unable to select a hotel or control the level of zoom presented severe usability issues. Not only did the application not work as intended, it also caused users to experience great frustration due to the difficulty of using the application as desired. Because users had to perform the same gesture multiple times slowly in order to receive some kind of response, the users did not find the application or the Leap Motion technology advantageous. The usability issues related to reliability and precision that are present in Leap Motion applications are currently too widespread and too complex to overcome for the device to be effectively used in applications requiring high levels of gesture accuracy.

## 5.3 Future Work

We believe that this document will be beneficial for developers looking to create Web applications using Leap Motion. Because the technology is still relatively new, documenting such problems would prevent developers from facing many of the same issues that we faced. One disadvantage of using Leap Motion is that there is no way to type or create words only using Leap Motion users must use a keyboard. In order to address this issue, Leap could implement some type of voice interaction. If given more time, we could research how Leap interacts with all applications instead of just Web applications and with other languages as well. Once these issues are addressed and a new version of the product is rolled out, we believe Leap Motion could be a very useful and widely used technology.

## References

1. "5 More Free Experiences Through the Web - Leap Motion Blog." *Leap Motion Blog*. N.p., 07 Aug. 2013. Web. 04 Mar. 2015. <http://blog.leapmotion.com/5-more-free-experiences-through-the-Web/>.
2. "Browser Statistics." *Browser Statistics*. W3schools.com, 1 Jan. 2015. Web. 08 Mar. 2015. <http://www.w3schools.com/browsers/browsers_stats.asp>.
3. "CircleGesture." *CircleGesture — Leap Motion JavaScript SDK V2.2 Documentation*. Leap Motion Inc., 1 Jan. 2015. Web. 14 Mar. 2015. <https://developer.leapmotion.com/documentation/javascript/api/Leap.CircleGesture.html#id44>.
4. Findlater, Leah. Froehlick, Jon. Fattal, Kays. Wobbrock, Jacob O. Dastyar, Tanya. *Age-related differences in performance with touchscreens compared to traditional mouse input.* ACM Digital Library. <http://delivery.acm.org/10.1145/2480000/2470703/p343-findlater.pdf?ip=130.215.9.226&id=2470703&acc=ACTIVE%20SERVICE&key=7777116298C9657D%2E71E5F5E88B9A3E17%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=484409395&CFTOKEN=28680060&__acm__=1425676655_17eb8b7f30c492ebdd93b89dc5f2892e>
5. "KeyTapGesture." *KeyTapGesture — Leap Motion JavaScript SDK V2.2 Documentation*. Leap Motion Inc., 1 Jan. 2015. Web. 14 Mar. 2015. <https://developer.leapmotion.com/documentation/javascript/api/Leap.KeyTapGesture.html>.
6. Kosner, Anthony W. "Leap Motion Controller Leaps Forward With Software, Sharpens Focus With Apps." *Forbes*. Forbes Magazine, 23 Nov. 2013. Web. 08 Oct. 2014. <http://www.forbes.com/sites/anthonykosner/2013/11/23/leap-motion-controller-leaps-forward-with-software-sharpens-focus-with-apps/>.
7. "Leap Motion Controller Review - Page 2 - CNET." *CNET*. CNET, 19 July 2013. Web. 08 Oct. 2014. <http://www.cnet.com/products/leap-motion-controller/2/>.
8. "Meadowhawk Module Featured in Mercedes CES Concept Car." *Leap Motion Blog*. N.p., 06 Jan. 2015. Web. 04 Mar. 2015. <http://blog.leapmotion.com/experimental-meadowhawk-module-featured-mercedes-benzs-ces-concept-car/>.

9. *Native Application Interface: Leap-enabled Applications*. Digital image. *System Architecture - Leap MotionJavaScript SDK V2.2 Documentation*. Leap Motion, 01 Jan. 2014. Web. 2 Mar. 2015. <https://developer.leapmotion.com/documentation/javascript/devguide/Leap_Architecture.html>.
10. "Node.js." *Node.js*. Joyent Inc, 1 Jan. 2015. Web. 14 Mar. 2015. <https://nodejs.org/>.
11. "Onmousemove Event." *Onmousemove Event*. W3schools.com, 1 Jan. 2015. Web. 09 Mar. 2015. <http://www.w3schools.com/jsref/event_onmousemove.asp>.
12. "Release Notes and Known Issues: V2.2.3.25971." *Skeletal Tracking*. Leap Motion, 1 Jan. 2015. Web. 12 Mar. 2015. <https://developer.leapmotion.com/features/faq>.
13. "ScreenTapGesture." *ScreenTapGesture — Leap Motion JavaScript SDK V2.2 Documentation*. Leap Motion Inc., 1 Jan. 2015. Web. 14 Mar. 2015. <https://developer.leapmotion.com/documentation/javascript/api/Leap.ScreenTapGesture.html>.
14. "SwipeGesture." *SwipeGesture — Leap Motion JavaScript SDK V2.2 Documentation*. Leap Motion Inc., 1 Jan. 2015. Web. 14 Mar. 2015. <https://developer.leapmotion.com/documentation/javascript/api/Leap.SwipeGesture.html>.
15. "System Architecture." *System Architecture — Leap MotionC++ SDK V2.1 Documentation*. Leap Motion, 1 Jan. 2014. Web. 08 Oct. 2014. <https://developer.leapmotion.com/documentation/skeletal/cpp/devguide/Leap_Architecture.html>.
16. *WebSocket Interface: Leap-enabled Applications*. Digital image. *System Architecture - Leap Motion JavaScript SDK V2.2 Documentation*. Leap Motion, 01 Jan. 2014. Web. 2 Mar. 2015. <https://developer.leapmotion.com/documentation/javascript/devguide/Leap_Architecture.html>.

## Appendix A: Glossary

CircleGesture : This is a circular finger movement. The Leap Motion sensor recognizes the finger motion when the finger draws a circle within the given field of view. This gesture can be seen in Figure 3.

Heroku: This is a cloud application platform that allows developers to build and deploy Web apps.

InteractionBox: This represents the box-shaped regions that are completely within the field of view of the Leap Motion controller.

JSON: JavaScript Object Notation.

KeyTapGesture : This is a tapping gesture by a finger. The Leap Motion sensors recognizes the finger moving down to the palm and then back up to the original position. This gesture can be seen in Figure 5

Leap Motion App Store: This is where all the applications that Leap Motion developers place applications to be bought and/or installed by the public.

ScreenTapGesture: Similar to the KeyTapGesture this is a tapping gesture. The Leap Motion sensor recognizes this gesture when the finger springing in toward the front of the Leap Motion field and then moving back. This gesture can be seen in Figure 6.

## Appendix B: Analysis of Travel Applications

We analyzed various travel applications using Leap Motion as well as those using Web based applications. We did this to find features that we could implement in our own application.

**Travel Seeker**: This is the only travel application that currently exists in the Leap Motion App Store. Some of its features allow users to:

1. Select an origin, budget, dates, and length of stay
2. Plan up to four different trips at the same time depending on factors such as climate, continent, and activity
3. View prices by zooming in on an area

Travel Seeker also implements the KeyTapGestures, SwipeGestures, and KeyTapGestures defined by Leap Motion.

**Roadtrippers:** This Web application uses Google Maps to allow users to enter start and end locations. Some of its features include:

1. Creating an account to save travel routes
2. View popular road trip routes
3. Select areas based on various factors (i.e. Shopping, attractions & culture, food & drink etc.)

The application also has zoom in and out features, markers showing each location or attribute, a search bar, a bar at the bottom of the screen which allows you to select trips, collections, and stories, as well as many other features.

**KAYAK**: a Web-based travel application that allows its users to compare hundreds of travel sites with one search. Some of its features include:

1. Provides links to other sites to compare prices based on search criteria

2. Allows users to narrow budget and to organize by different criteria (i.e. Rating, amenities, etc.)
3. provides multiple aspects of travel such as hotel booking, flights, cars, and other packages

## Appendix C: User Study Consent Script

Because our primary audience is students, we plan to recruit subjects by asking random members of the WPI Community to participate in various settings on and off campus. We will be reaching out to various student organizations as well as face-to-face interactions on campus. We will need about ten to twenty subjects for each usability group, which is roughly thirty to sixty subjects total. We intend only to use screen captures and keep the identity and any identifiable information anonymous for every subject.

### Consent script:

Team Member: Our MQP is testing the benefits of Leap Motion, which is a small device that plugs into any USB port and allows you to interact with your screen without a mouse or keypad. We are trying to assess the drawbacks and benefits of using Leap Motion compared to more traditional methods. We created a test application that will be used in two of the three surveys. You will be completely anonymous in this study; none of your personal or identifiable information will be available anywhere in our report. If at any point in the trial, you wish to stop, you can do so; this trial is completely voluntary. There are three surveys from which to choose, our application using Leap Motion, our application not using Leap Motion, and a traditional travel site. [choose which survey]

Would you like to participate in a user study to test our MQP?

### For Leap survey:

We will provide a README document, which details the gestures you can use and what they do and ask you to complete 5 tasks using these gestures. There will be a screen capture to see how easy or difficult it is to complete tasks, but you cannot be seen or heard with the recording. A clip of the screen capture may be used in our final presentation to show the use of Leap with this application. The trial will take about 10 minutes. At the end, I will ask you 4 questions about your prior familiarity with Leap, how difficult or easy the trial was, whether or not you would purchase a Leap Motion device after this trial, and which browsers you used.  Do you have any questions before we begin?

### For App without Leap:

In this trial, we are testing an application without using Leap Motion, in order to see how this trial compares with the other two trials. We will ask you to complete a set of tasks using your mouse and keyboard. The trial will take about 5 minutes. Your screen will be recorded, but you will not be seen or heard. A clip of the screen capture may be used in our final presentation to compare usability. At the end, I will ask you a few questions about the study. Do you have any questions before we begin?

### For booking.com:

In this trial, we are testing a normal hotel booking site in order to see how this trial compares with the other two trials. We will ask you to complete a set of tasks using your mouse and keyboard. The trial will take about 2 minutes. Your screen will be recorded, but you will not be seen or heard. A clip of the screen capture may be used in our final presentation to compare usability. At the end, I will ask you a few questions about the study. Do you have any questions before we begin?

# Appendix D: Leap Motion Survey

1. Did you know what Leap Motion was prior to this study?
2. On a scale from 1-5 (with 1 being easiest), how difficult was the trial overall?
3. Were the gestures natural/intuitive? If not, which ones could be improved?
4. After this trial, would you purchase a Leap Motion device?
5. Which browser(s) did you use?

## Tasks:

1. Zoom in and out
2. Navigate to Boston, MA
3. Find hotels in Boston, MA
4. Select one hotel
5. Move the selected hotel from ‘Possibilities’ into ‘Choices’ in the sidebar

## Results:

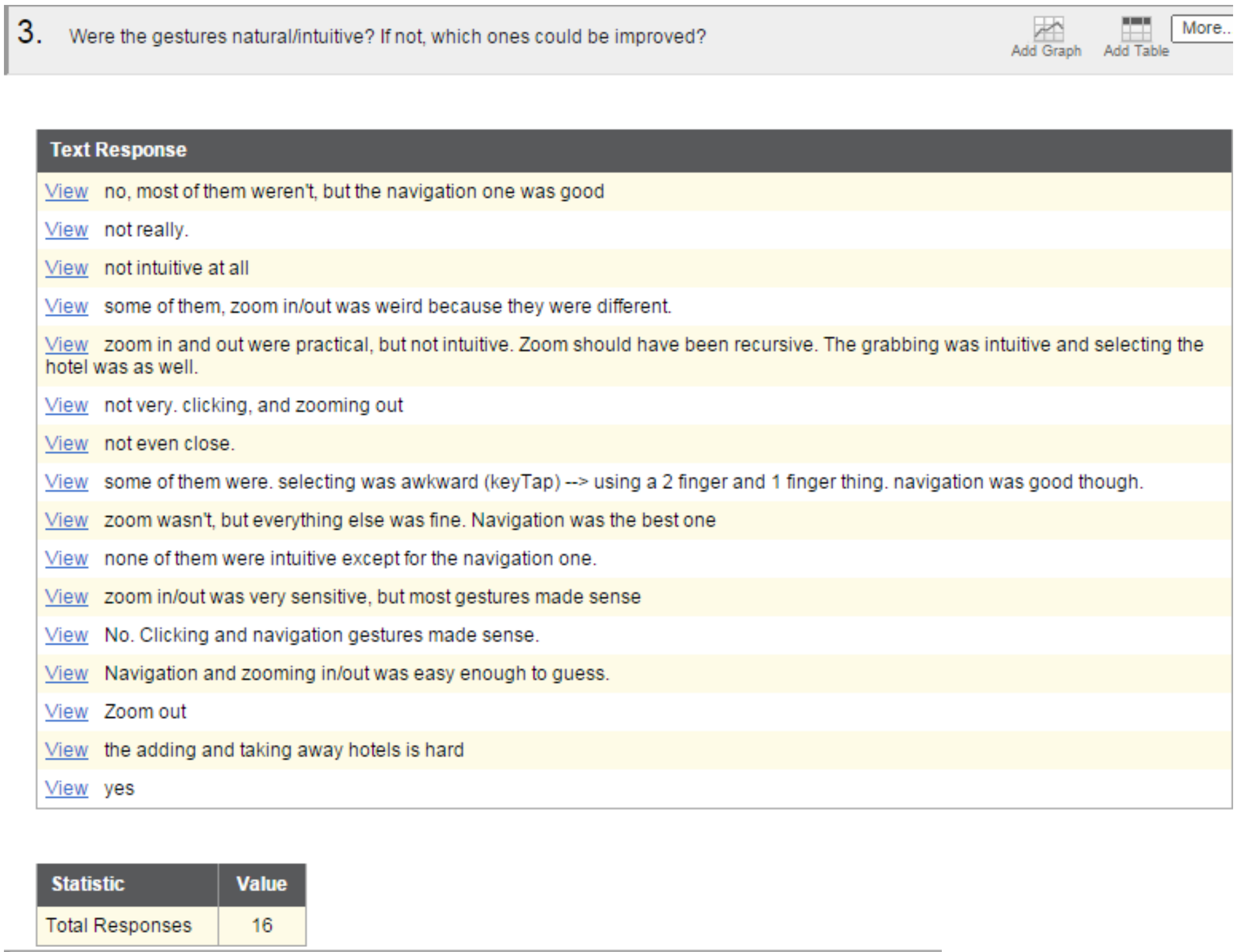
1. Did you know what LEAP Motion was prior to this study?

| Text Response | |
|---|---|
| View | User was not familiar with LEAP Motion prior to the study |
| View | motion sensing |
| View | yeah. |
| View | No |
| View | yes, but had not used it prior |
| View | no |
| View | no |
| View | yes, had played with the device prior to this study |
| View | no, I still don't know is entirely |
| View | no |
| View | not really |
| View | yes. |
| View | No |
| View | No |
| View | nope |
| View | No |

| Statistic | Value |
|---|---|
| Total Responses | 16 |

2. How difficult was the trial overall? (with 1 being the easiest)

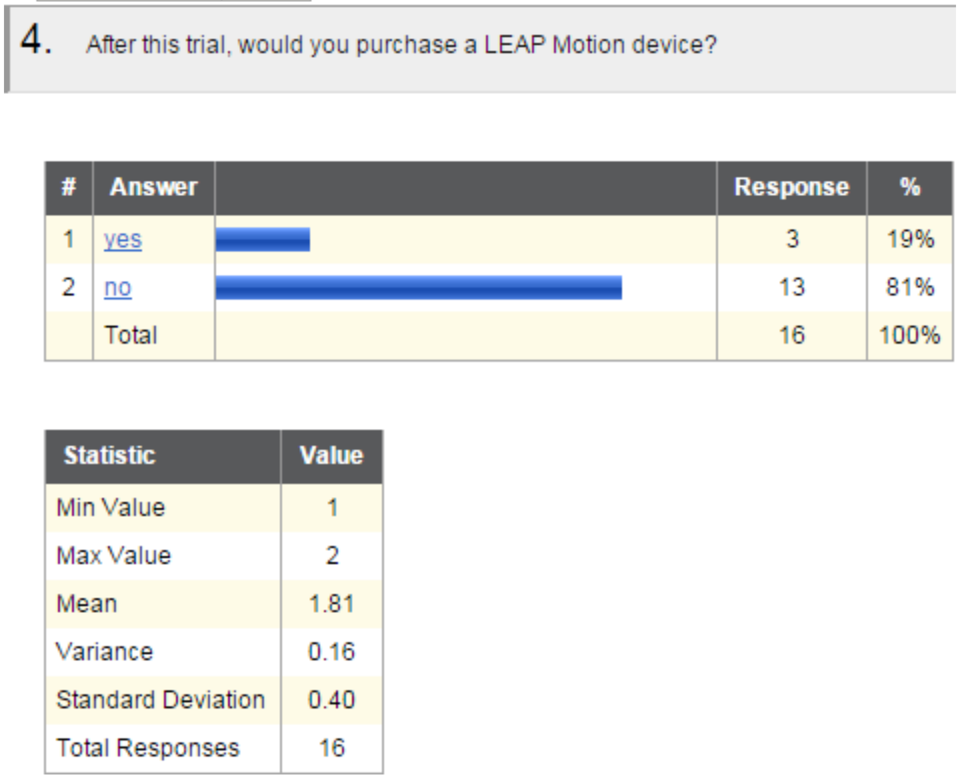
| # | Answer | Min Value | Max Value | Average Value | Standard Deviation | Responses |
|---|---|---|---|---|---|---|
| 1 | Difficulty | 2.00 | 5.00 | 3.92 | 0.86 | 13 |



3. Were the gestures natural/intuitive? If not, which ones could be improved?

| | Text Response |
|---|---|
| View | no, most of them weren't, but the navigation one was good |
| View | not really. |
| View | not intuitive at all |
| View | some of them, zoom in/out was weird because they were different. |
| View | zoom in and out were practical, but not intuitive. Zoom should have been recursive. The grabbing was intuitive and selecting the hotel was as well. |
| View | not very. clicking, and zooming out |
| View | not even close. |
| View | some of them were. selecting was awkward (keyTap) --> using a 2 finger and 1 finger thing. navigation was good though. |
| View | zoom wasn't, but everything else was fine. Navigation was the best one |
| View | none of them were intuitive except for the navigation one. |
| View | zoom in/out was very sensitive, but most gestures made sense |
| View | No. Clicking and navigation gestures made sense. |
| View | Navigation and zooming in/out was easy enough to guess. |
| View | Zoom out |
| View | the adding and taking away hotels is hard |
| View | yes |

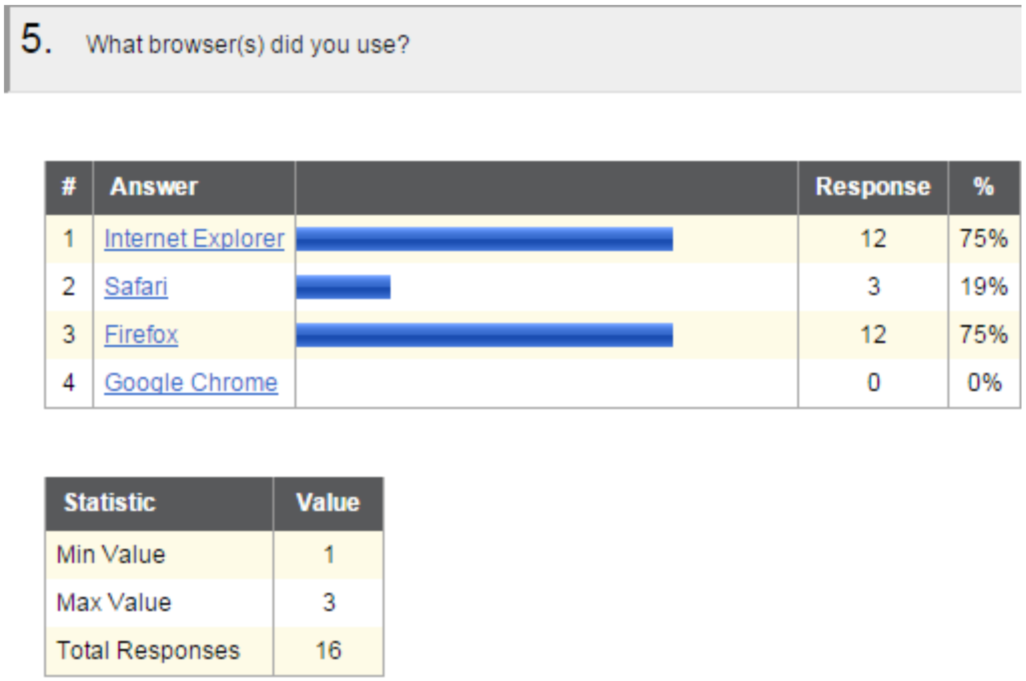
| Statistic | Value |
|---|---|
| Total Responses | 16 |



4. After this trial, would you purchase a LEAP Motion device?

| # | Answer | | Response | % |
|---|---|---|---|---|
| 1 | yes | | 3 | 19% |
| 2 | no | | 13 | 81% |
| | Total | | 16 | 100% |

| Statistic | Value |
|---|---|
| Min Value | 1 |
| Max Value | 2 |
| Mean | 1.81 |
| Variance | 0.16 |
| Standard Deviation | 0.40 |
| Total Responses | 16 |



5. What browser(s) did you use?

| # | Answer | | Response | % |
|---|---|---|---|---|
| 1 | Internet Explorer | | 12 | 75% |
| 2 | Safari | | 3 | 19% |
| 3 | Firefox | | 12 | 75% |
| 4 | Google Chrome | | 0 | 0% |

| Statistic | Value |
|---|---|
| Min Value | 1 |
| Max Value | 3 |
| Total Responses | 16 |

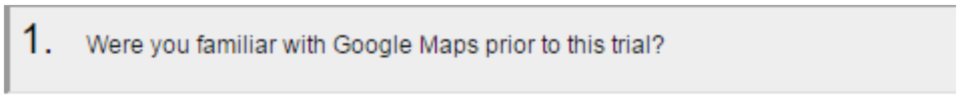
## Appendix E: Google Maps Web Application Survey

1. Were you familiar with Google Maps prior to this trial?
2. Were the gestures natural/intuitive? If not, which ones could be improved?
3. Would you use this app in real life?
4. On a scale from 1-5 (with 1 being easiest), how difficult was this trial overall?

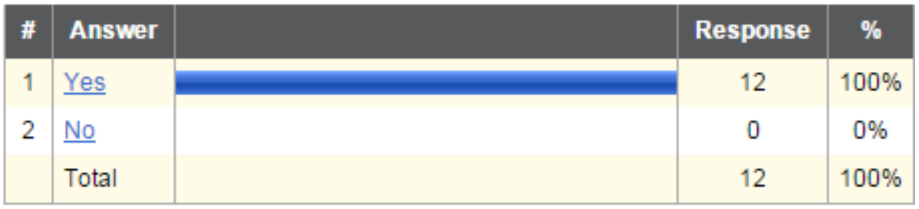
### Tasks:

1. Zoom in and out
2. Find hotels in Boston, MA
3. Select one hotel
4. Move hotel from 'Possibilities' to 'Choices' in the sidebar

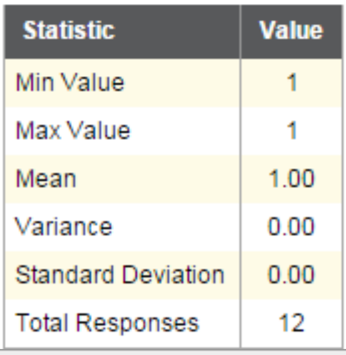
## Results:



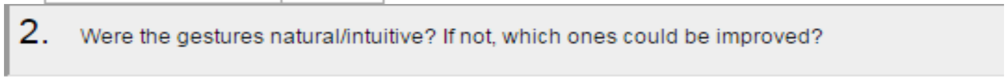
1. Were you familiar with Google Maps prior to this trial?



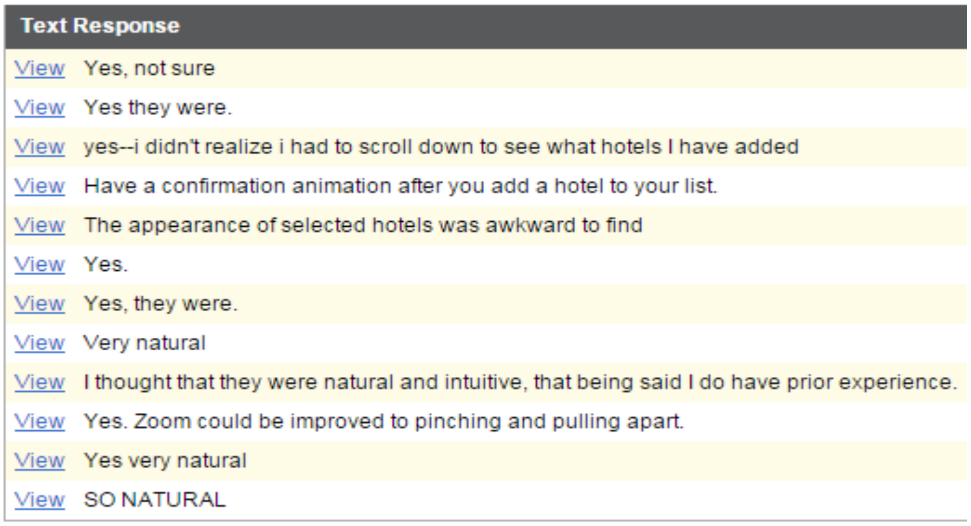
| # | Answer | | Response | % |
|---|---|---|---|---|
| 1 | Yes | ▬ | 12 | 100% |
| 2 | No | | 0 | 0% |
| | Total | | 12 | 100% |



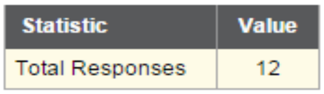
| Statistic | Value |
|---|---|
| Min Value | 1 |
| Max Value | 1 |
| Mean | 1.00 |
| Variance | 0.00 |
| Standard Deviation | 0.00 |
| Total Responses | 12 |



2. Were the gestures natural/intuitive? If not, which ones could be improved?



| | Text Response |
|---|---|
| View | Yes, not sure |
| View | Yes they were. |
| View | yes--i didn't realize i had to scroll down to see what hotels I have added |
| View | Have a confirmation animation after you add a hotel to your list. |
| View | The appearance of selected hotels was awkward to find |
| View | Yes. |
| View | Yes, they were. |
| View | Very natural |
| View | I thought that they were natural and intuitive, that being said I do have prior experience. |
| View | Yes. Zoom could be improved to pinching and pulling apart. |
| View | Yes very natural |
| View | SO NATURAL |



| Statistic | Value |
|---|---|
| Total Responses | 12 |

## 3. Would you use this app in real life?

| | Text Response |
|---|---|
| View | I would probably use google maps or trip advisor or something to find hotels |
| View | I would. |
| View | yes |
| View | Yes. |
| View | yes |
| View | Yes. |
| View | Yes |
| View | yes |
| View | I will use google maps in the future however I do not know if I will use it to book hotels. |
| View | Yes |
| View | yes |
| View | yes |

| Statistic | Value |
|---|---|
| Total Responses | 12 |

## 4. On a scale from 1-5 with 1 being the easiest, how difficult was this?

| # | Answer | | Response | % |
|---|---|---|---|---|
| 1 | Very Difficult | | 0 | 0% |
| 2 | Difficult | | 0 | 0% |
| 3 | Neutral | | 1 | 8% |
| 4 | Easy | | 6 | 50% |
| 5 | Very Easy | | 5 | 42% |
| | Total | | 12 | 100% |

| Statistic | Value |
|---|---|
| Min Value | 3 |
| Max Value | 5 |
| Mean | 4.33 |
| Variance | 0.42 |
| Standard Deviation | 0.65 |
| Total Responses | 12 |

## Appendix F: Booking.com Survey

1. On a scale from 1-5 (with 1 being easiest), how difficult was this trial overall?
2. Would you want/need a different way to interact with this website?

### Tasks:

1. Find 5-star hotels in Boston, MA
2. Find the hotel on the map
3. From the map, select a different hotel

## Results:

1. On a scale from 1-5 (with 5 being the hardest), how difficult was this?

| # | Answer | | Response | % |
|---|---|---|---|---|
| 1 | Very Difficult | | 0 | 0% |
| 2 | Difficult | | 0 | 0% |
| 3 | Neutral | | 0 | 0% |
| 4 | Easy | | 1 | 8% |
| 5 | Very Easy | | 11 | 92% |
| | Total | | 12 | 100% |

| Statistic | Value |
|---|---|
| Min Value | 4 |
| Max Value | 5 |
| Mean | 4.92 |
| Variance | 0.08 |
| Standard Deviation | 0.29 |
| Total Responses | 12 |

2. Would you want/need a different way of interacting with the website?

| # | Answer | | Response | % |
|---|---|---|---|---|
| 1 | yes | | 0 | 0% |
| 2 | no | | 12 | 100% |
| | Total | | 12 | 100% |

| Statistic | Value |
|---|---|
| Min Value | 2 |
| Max Value | 2 |
| Mean | 2.00 |
| Variance | 0.00 |
| Standard Deviation | 0.00 |
| Total Responses | 12 |