ENGR440 HCI - Final Project Report

Part One - Project Report

Iain Walker

300212708

Abstract

TODO

1. Introduction

2. Leap Earth System

The Leap Earth System created is intended to be a prototype for a more natural gestural interface for navigating a Google-Earth like application. Navigating current Google or Apple map systems with a mouse and keyboard or touchscreen is relatively straightforward. A more natural gestural interface is an interesting domain when considering applications that are easily grasped already with existing technologies.

The Leap Earth system simply consists of a map rendered on a globe that may be interacted with using gestures. As we zoom in and out, more or less detail is displayed on the globe. Actual features of mapping software (e.g. consider Google Maps direction function) has of course not been implemented. Rather the focus of implementation has been experimentation with various gestural control styles of the globe itself. One feature added to demonstrate additional layered interaction was the placing of markers on the map. This was added as part of consideration of the directions use-case for Google Maps; we must firstly place two markers that the mapping software will provide directions between.

Gestural features supported for globe interaction are primarily based on navigation around the globe. Moving the hand left moves the globe left; moving the hand right moves the globe right, and the same for up and down movements. Zooming is controlled by either moving the hand up or down, or via a finer-grained finger rotation gesture. Finally, perfoming a finger 'flick' action will place a marker at the center of the screen. Discussion of the supported gestural features is given in Section 3.

As well as the primary globe at center screen, a basic side-bar provides some extra functionality. There is a link to an 'about' page, that provides some description as to the purpose of the system, as well as the gestures supported. A stop rotation button allows for pausing of all animations, largely added for debugging purposes early in the project. A 'navigate to location' text box was also placed in the sidebar. This allows the user to type in some location - e.g. 'Wellington', which will cause the globe to rotate and zoom on that particular location. Finally, a Leap Hand is rendered in the sidebar, displaying precisely what the Leap Motion device sees in real time.

CONF 'yy, Month d–d, 20yy, City, ST, Country.
Copyright © 20yy ACM 978-1-nnnn-nnnn-n/yy/mm. . . $15.00.
http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

2.1 Implementation Tools

The Leap Earth system was implemented using the Leap Motion javascript API [], running on a basic Python HTTP server. The globe is rendered using the WebGL Earth API [], and interfaces to tools such as Google Maps API for lookup and coordinate systems. The interface has been tested to work with Google Chrome above version 9 (not currently installed on ECS workstations).

3. Interface Design Decisions

The core design decisions relate to the gestural input design, as the interface layout itself is very straightforward.

3.1 Navigation Gestures

The basis of the navigation gestures provided was built from imagining panning a virtual camera over the global map provided. This camera is held in the palm; thus moving the palm left moves the position shown left; right moves it right, and so on. Initially the approach I wanted to take revolved around holding a virtual soccer ball in two hands, and navigating around the globe as you might inspect or rotate such a virtual sphere in your hand. This proved too innaccurate and difficult to implement however; actual recognition of these quite detailed gestures necessary to rotate such an object is not something that the Leap Motion device is very good at to a high degree - immediate detection of gestures is not consistently forthcoming, as will be discussed in more detail in Section 4.

Having instead deciding on a camera-based approach, actual implementation strategies were considered. Leap Motion examples online actually demonstrate interaction with a Globe, using hand position to control camera position as desired. However these used an approach of directly mapping hand position in the plane detected by Leap Motion into a corresponding globe position and rotation. This is clearly not suitable when considering a mapping tool that must provide navigation capabilities up to very high levels of zoom, as very small movements would correspond to massive distances when implemented on the scale of the globe. Further, removing the hand from the leap motion space will reset the globe's position in these examples. Instead the relative magnitude of hand translation corresponds to movement of the globe, as shown in the formula below.

```
var newLng = lng +  Math.sin(rotateX * Math
    .PI/180) * getSensitivity(earth); //
    sensitivity changes based on current
    zoom level
```

The actual mapping from size of hand movement to amount of rotation induced relates to the current level of zoom, as demonstrated in the formula. This currently followed a simple 50/current-zoom squared formula, which was simply worked out through trial and error. This mapping from zoom to magnitude of translation is not perfect, as small movements at high levels of zoom still have quite large results in camera movement. With a better mathematical understanding of zoom levels to magnitude of movement required though, this could be improved.

3.2 Finer Zoom Control

The challenge of mapping small hand movements onto a surface as large as the globe affected implementation of zoom adjustment also. The Leap zone of detection was simply not large enough to only implement zoom related to how high the hand is held. As such, a circling finger gesture was added to facilitate finer grained zoom control. Circling the finger clockwise increases zoom, whilst anti-clockwise movement decreases zoom. This was simply based off intuition - 'lefty loosey righty tighty'. One of the challenges of course with adding further gestures to a system that implements globe rotation based off of hand position is that the Leap Motion regularly interprets the circling finger as the palm actually changing position. As a countermeasure, during detection of a current finger-circling gesture, any movement of the hand itself is ignored. This interpretation of gestures as hand movement was a difficulty with every single additional feature I attempted to add to the system - which we will review in Section 4.

A potential future solution to this issue of false-positive palm movement could be to implement a 'dead zone' of slight movements, which are just thrown away. In a basic sense I experimented with this - naiively completely ignoring any movements around 0.1 'Leap Units'. The problem with this approach was that it was still not enough to get rid of movement when circling the finger. Further, any larger dead zones made navigation around the world rather unintuitive, as small movements would result in no change, but when the dead zone boundary was breached a sudden large judder would result. This was detrimental enough that I considered the screen locking approach the best one for the prototype at least.

3.3 Placing of Markers

The final action possible on the main earth section is the placing of coordinate markers. This was considered as part of a possible wider future use case, in which users place markers and then for example request directions between the two markers. It was also a relevant way of investigating the very real problem of layering additional features onto a gestural system. The gesture to place coordinates is what Leap Motion calls a 'ScreenTapGesture' - a forward tapping movement by the finger. This seemed the most natural way of placing markers on a map as currently supported by the existing Javascript gesture API.

It is likely that at this stage the reader is thinking of the circling gestures, and how false palm movement detection resulted in undesired globe rotation. The output for a jerky finger tap is, as one would expect, even worse. Again, implementation of a dead zone could have helped with this, but at such a detriment to overall usability that this was considered infeasible. Other gestures were again considered. Certain hand positions (e.g. ball the fist) could have corresponded to placing the coordinate, or any other number of hand poses not requiring actual hand movement. However I considered part of the goal of the system to look at how natural and intuitive interaction styles would work - and really there was no sensible mapping from hand poses into marker placement. Further, what part of the pose constitutes actual placement - when one enters the pose, or exits the pose? These considerations made me prefer the flawed, but ultimately more natural finger tap gesture.

3.4 Sidebar Design Decisions

The Sidebar design was inspired by the overall layout of Google Maps, which has the simple UI of the map itself and a bar with relevant informative messages etc to the left of the page.

The only feature significant enough for discussion on the sidebar is the Leap Hand box, which renders a view of what the Leap Motion device sees in real time. This was originally implemented as a learning exercise and to debug Leap Motion gestures and hand positioning. Having the Leap Visualiser and my site open on a single-monitor workstation was proving impractical, and thus immediate feedback of what the device was detecting was useful for development purposes. However having shown the website to friends, I receieved comments that having the Hand shown helped with learning the mechanics of the Leap technology. Issues such as where the Leap detects, whether the Leap is detecting my hand(s) currently, and any potential occlusion are immediately highlighted by the Leap Hand box, which still manages to take up very little screen real estate.

3.5 Further Discarded Gestures

There were a few gestures I implemented that were ultimately discarded for a number of reasons. I intended to add a 'swiping' gesture, that would result in the earth rotating at a set speed in the direction of the swipe. Further swipes in the same direction would increase the speed of rotation, as with spinning a basketball on your finger for example. Implementing such a gesture unfortunately again clashed with the positional manipulation of the globe. Swiping was actually completed before the positional mapping was working correctly, but having added the positional control the side-effects of swiping were too greatly pronounced for the feature to remain.

We can see that further gestures clashing with existing ones, as well as gestures in general clashing with positional control as being a recurring theme in my project. As an inexperienced developer of gesture controlled interfaces, this was unfortunately something I did not anticipate. If I was to develop further interfaces with the Leap Motion device, I would certainly consider carefully what gestures could be implemented in tandum for the problem domain, as well as considering in advance whether the gestures would cause conflicts. With the technology at hand, simple interfaces with a small range of possible interactions may be the best for gesture based systems. Although the Leap Earth system is very basic, there was still room for gesture conflicts to occur, particularly with the positional control employed.

Having reviewed the Leap Earth system developed, and discussed core design decisions relating to the interface and gesture interaction, we will now look at the Leap Motion controller as an interaction device.

4. Leap Motion Interaction Critique

4.1 Consumer Focus

As a peripheral device for a computer, the Leap Motion is as easy to install and setup as one would expect it to be. Every system I have plugged the Leap Motion into detects the controller immediately and installs the relevant packages required to get the software running. The fact that a Leap Motion focused application store comes packaged with the basic software downloads increases the ease of learning how to control with the system. With a multitude of basic games at your fingertips, getting started is a breeze. The Leap device itself is also extremely unobtrusive. One does not notice it taking up any room on your desk. The technology behind the Leap Motion is also basic enough that the device is affordable to consumers. It uses infrared optics and cameras instead of the Kinects depth sensors, at much higher fidelity than the Kinect, at a refresh rate higher than computer monitors []. The software backing the device is what really sets it apart, though; algorithms for which are being (understandably) kept very secret. Thus from a consumer perspective the Leap Motion has taken the right approach.

4.2 The Good

I found the Leap Motion as a sensor incredibly accurate and fine-tuned with what it could detect.

4.3 The Bad

It was after getting started, and the exciting novelty of controlling applications with my hands wore off that real issues with the accuracy of the device became more noticeable. Whilst the fidelity of the device is extremely high,

5. Leap Motion Development Discussion

References