|  |
| --- |
| The University of Alabama |
| Sphero/LEAP Project |
| Motion and Robotic device connected by Web Development |

|  |
| --- |
| Author: Pereira, Kathleen  Mentor: Gray, Jeff  7-15-2015 |

Contents

[**Purpose** 2](#_Toc424764024)

[**Development** 2](#_Toc424764025)

[**The LEAP Device** 2](#_Toc424764026)

[**The Sphero** 3](#_Toc424764027)

[**Spherly** 4](#_Toc424764028)

[**Additions** 4](#_Toc424764029)

[**Integration Process** 4](#_Toc424764030)

[**Future work** 4](#_Toc424764031)

[**Conclusion** 5](#_Toc424764032)

[**References** 6](#_Toc424764033)

**Content of Figures**

[Figure 1: InteractionBox View [7] 3](#_Toc424763906)

## **Purpose**

Our purpose in this project was to integrate the LEAP Motion Controller with the Sphero Robotic Ball through a web application and Bluetooth technology. Thus, the Sphero would be controlled by the LEAP device through our application.

## **Development**

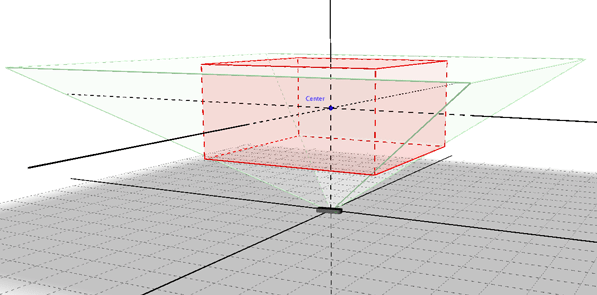
### **The LEAP Device**

The first step of this project was learning more about the LEAP device and how to work on it using JavaScript. The LEAP Motion Company has its own portal for developers, were some helpful material for this project was found [1]. Once that the LEAP architecture was understood, we started to develop our initial idea.

* **Tracking Movements**

The Controller class is used as our main interface to the LEAP device, through it we could gather valuable information to start tracking the hand’s movement [2]. Once that an instance of the Controller was created, we started to use the information from the frames that were collected by the LEAP. In the Frames class [3], we used another two classes:

* The InteractionBox class, which creates an imaginary box within the field of view of the LEAP [4].
* The Pointable class that reports the physical characteristics of a finger or tool [5].

Figure 1: InteractionBox View [7]

In the InteractionBox was possible to have values from 0.00 to 1.00, which made easier the analysis of the movement. In order to define the correct direction, the function defineDirections was created.

* **Defining Direction**

Once the information from the LEAP device was collected, the next action was the interpretation of this data, which is done In the function defineDirections; where we receive the coordinates of x, y and z. The algorithm in this function checks whether or not the value of the coordinates are valid; after this first verification, the new coordinates are compared with the old coordinates. If the difference between them is equal or greater than 0.05, then the code looks up on an array, called directions, that contains the directions in a string format.

Finally, with the correct direction we could send it as a command to our Sphero code, which will be discussed later on in Integration Process.

Note: The referred code can be found in *SpheroLeap/Spherly/bin/www/LEAPControl.js*

### **The Sphero**

In the Sphero’s side, we used an API that creates a server, which through Bluetooth connects to the Sphero and at the same time with the web application. This API is an open source code and it can be found at Github [6], this software was developed by Jake Trower and its name is *Spherly.*

### **Spherly**

This project used most of Spherly’s architecture to connect to the Sphero through Bluetooth. First, it was developed an HTML page in which we called some of Spherly functions (such as Connect, Sleep and Disconnect); also, the Spherly’s Dialog box was reused in this code. Until then, we did not change any of Spherly’s code.

### **Additions**

The first modification that was made in the code was on runblockly.js (which can be found at SpheroLeap/Spherly/bin/www/runblockly.js). We added a new function on line 392 that refers to a function in the *Sphero.js* (also found at SpheroLeap/Spherly/bin/www/Sphero.js). The final modification was on *Sphero.js* and it starts at line number 497; here is where we definitively send the commands to the Sphero.

### **Integration Process**

One of the challenges was how to integrate the LEAP side with the Sphero side without changing their core. The solution that was found has a place in the JavaScript of the HTML page that was created; once that we got the direction, we change the image that corresponds to the direction to another one (e.g. from *img/RIGHT.png* to *img/RIGHT\_Active.png*). When we change back, an attribute to the old image is added (view line 48 of *LEAPControl.js*); this new attribute is a trigger that calls the function *sendCommands* in *runblockly.js*, which later on calls the function of the same name in the object *Sphero* in *Sphero.js*.

## **Future work**

In a final analysis, we suggest some functionalities and modifications to a better performance of the software:

* The Sphero has a tail light to indicate which side is the Forward direction (which is always the opposite side of the tail light). A functionality to adjust the tail light would be added in order to change the position of the Sphero as is desired.
* One of the information that the LEAP collects is the current angle of the hand; this would be used to indicate to the Sphero to continue in the same direction. Differently, a flat hand would indicate a change of direction.
* This current version does not use the information of UP and DOWN direction; in fact, a new functionality would be added to use those information. The speed would be controlled based on the Y-axis, so UP would be faster and DOWN would be slower.
* Finally, the code that recognizes the movement should be improved because a more accurate reading will result in a high level of precision in the Sphero’s performance.

## **Conclusion**

This project used various tools offered by the JavaScript language; also, it promoted the integration of different platforms in a common goal. We hope that this project will become the first step for further research and work involving Web Development, Humam-Computer Interaction, Motion devices and Robotic devices.

## **References**

1. LEAP Motion Developer Portal: <https://developer.leapmotion.com/getting-started/javascript>
2. Controller Class, LEAP Motion: <https://developer.leapmotion.com/documentation/javascript/api/Leap.Controller.html>
3. Frames Class, LEAP Motion: <https://developer.leapmotion.com/documentation/javascript/devguide/Leap_Frames.html>
4. InteractionBox Class, LEAP Motion: <https://developer.leapmotion.com/documentation/javascript/api/Leap.InteractionBox.html>
5. Pointable Class, LEAP Motion: <https://developer.leapmotion.com/documentation/javascript/api/Leap.Pointable.html>
6. Spherly Code: <https://github.com/jakeonaut>
7. Figure 1: <https://di4564baj7skl.cloudfront.net/documentation/images/Leap_InteractionBox.png>