UNIVERSITY OF OXFORD

Hand Gesture Recognition via Leap Motion Sensor

Authors:
Jahangir IQBAL

Supervisor:
Dr. Irina VOICULESCU

A thesis submitted in fulfillment of the requirements for the degree of Masters in Computer Science

in the

Department of Computer Science

August 28, 2017

Declaration of Authorship

I, Jahangir IQBAL, declare that this thesis titled, "Hand Gesture Recognition via Leap Motion Sensor" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

University of Oxford

Abstract

Faculty Name
Department of Computer Science

Masters in Computer Science

Hand Gesture Recognition via Leap Motion Sensor

by Jahangir IQBAL

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

For/Dedicated to/To my…

Chapter 1

Rotation of the Hand UI Model

In this chapter the rotation of the Hand UI model will be discussed.

1.1 Description of Problem

The UI model of the hand is built from the Leap Motion sensor data as discussed in the chapter. This data represents the hand as it is displayed in reality above the Leap Motion controller device. Originally the representation of the hand was built from this hand without any further modifications. However, to demonstrate why this is sometimes not the ideal situation, see Figure 1.1. It shows the hand as it is displayed in reality; it is in parallel plane above the device.

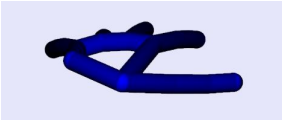

FIGURE 1.1: This shows the user's hand in a flat orientation; as determined by the Leap Motion sensor.

It should noted how difficult it is to see the fingers and thumb. They are only barely visible. Therefore the user would be forced to force his or her hand into a vertical postion by straining their wrist just so they can see the gesture they are performing on the screen. Figures 1.2 and 1.3 show other orientation the hand can take; these figures show the hand after a yaw rotation and after a roll rotation respectively. Again, note the slight difficulty in figuring out the thumb and fingers positions and orientations when the hand is rolled around the z-axis. The hand that is shown in the yaw rotation in Figure 1.2 is able to seen a little clearly because the wrist was strained upwards.

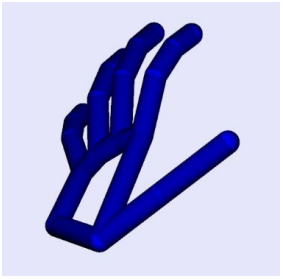

FIGURE 1.2: The user's hand after the certain yaw rotation around the y-axis.

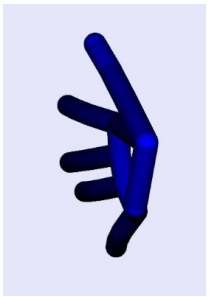

FIGURE 1.3: The user's hand after the certain roll rotation around the z-axis.

Finally all of these different orientation can of course overlap with each other to create a complex orientation of the hand as shown in Figure 1.5, which shows the user's left hand in a weird handshake sort of position while being rotated to the right and tilted upwards.

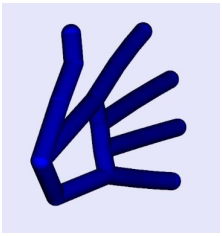

FIGURE 1.4: This shows the user's hand in a flat orientation; as determined by the Leap Motion sensor.

This chapter will explain how these different possible orientations the user's hand might be in while they are performing gestures shown on the screen are accounted for and "undone". Doing this results in the user's hand to be displayed in a set vertical orientation on the screen despite the different ways the user may have oriented their hand above the device in reality. The final resulting hand for all of the figures seen previously will be as is shown in Figure **??** after the composite rotation have transformed it to a vertical position.

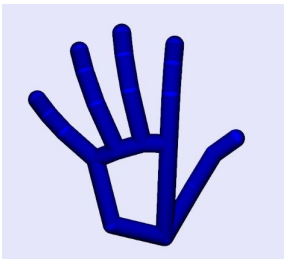

FIGURE 1.5: This shows what the user's hand will look like after all of the possible rotational transforms have been undone and the hand has been fixed to a vertical orientation.

This feature of the application makes it easier for the user to see what his or her hand is doing without requiring them to always contain their hand in a specific orientation. The goal of this application is to measure the accuracy with which a user is able to complete certain gestures, not the orientation of their hand. In fact, the algorithms that are used to grade the correctness of the user's attempted gesture do not take the user's hand orientation into account. They focus on the fingers bones and their relative orientations to each other. At the beginning of this project, one of the ideas discussed was to build some sort of rig which would be used to place the user's hand in a set orientation. However because of what will be explained in this chapter such a rig is no longer necessary.

1.2 JavaFx Coordinate System vs Leap Motion Coordinate System

The Leap Motion controller has a different coordinate system from JavaFX. This distinction is very important to get out of the way as the rest of the chapter will rely on

such an understanding. The coordinate system for the Leap Motion device is shown in Figure 1.6. Note the placement of the green LED light in the Leap Motion device as shows the orientation of the otherwise symetrical picture.

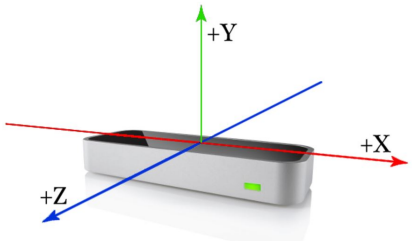

FIGURE 1.6: The coordinate system that is used by the Leap Motion sensor. Data collected will based upon these axes.

This coordinate system is referred to as a right-handed coordinate system because of the easy way in which it can be represented by the first three fingers of the right hand as shown in Figure 1.7.

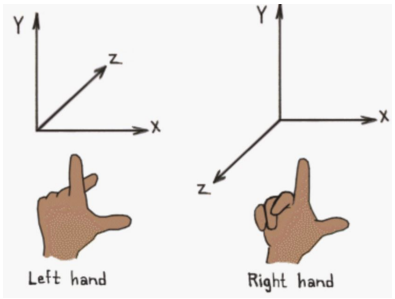

FIGURE 1.7: Left and right handed coordinate systems and how they can be shown via fingers.

In contrast, the traditional coordinate system which is used in computer science and which is what JavaFX also follows is shown in Figure 1.8. The JavaFx coordinate system is not left handed or right handed as far as we can tell from the Figures shown.

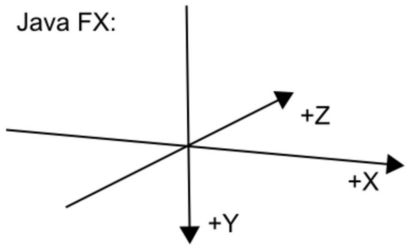

FIGURE 1.8: The coordinate system used in JavaFX. Setting transforms on objects will be based on this coordinate system.

Understanding these somewhat subtle difference and being able to switch from one context to the other was very important when considering the various rotational transforms that needed to be applied to the Hand UI model in order to straighten it to a vertical position. For example, one of the things that caused me some headaches sometimes when I was fixing and debugging my code was the fact that 3D rotations are measured in a very specific way. When it is said that some object is rotated

around the y-axis by 90 degrees, this means that the if we imagine ourselves to be an observer standing on the y-axis and looking down the negative direction of this y-axis, then we would observe the object to rotate counter-clockwise about the y-axis. Therefore, it becomes very clear why it is significant to have clear understanding of the coordinate system that is being used for a particular section of code. Leap Motion data is returned with the default settings of the Leap Motion coordinate system. This data must be appropriately modified when the code using it is based upon the JavaFX coordinate system. Likewise composite rotations around certain axis might become jumbled up if one is not careful.

1.3 Ineffective Leap Motion Data

1.3.1 Pitch Roll Yaw

1.3.2 Negative Zeros

1.4 Simplified Hand Model

1.5 Composite Linear Transformations

1.6 Rotational Matrix