GESTURE BASED UI DOCUMENTATION

Donal Burke | Kevin Delassus

Purpose of the Application

The purpose of this application was to control a mechanical arm with a leap motion controller through Java remote method invocation (RMI). As we quickly found out that control through RMI was not going to be possible we changed our focus on controlling the arm with the Leap Motion Controller on from a single computer. This project was more of a proof of concept that we could move a mechanical arm with a Leap Motion Controller. In theory if this could be done there is nothing to say that this could not be done over a network. A surgeon could be controlling a mechanical arm which is operating on someone from his office across the world.

The setup involved in running the application are as follows

- Plugging the Leap Motion Controller and Lego Mindstorm made mechanical arm into a single PC
- Start the application on the Java Lego brick computer
- Start the Java application on the PC

Once the setup is complete you can now run the gestures. More information on the gestures can be found in the below section. A more detailed explanation on the setup and running of the of the application can be found in the repository Readme.md file.

To test how the hardware could be combined with gestures we first had to research the Leap Controller. We needed to determine how the gestures were made and how we could convert the gesture into the motor running on the mechanical arm. One of the tests we preformed was getting the motors moving by keyboard inputs. Each Keyboard input was mapped to a certain motor movement. This gave us a overview of what and how many gestures were going to be needed.

Gestures Identified as Appropriate for this Application

For this application, we decided to use gestures that would be natural for the user. We took inspiration from control's that are used in heavy machinery such as diggers, excavators and cranes. Since the Lego arm works in a similar way to an excavator with regards to the directions it can move in, such as up, down, rotate left and rotate right, we tried to implement gestures that felt similar to using these hard controls but with gestures instead.

The gestures we used in the application are listed below with an explanation of why we decided these gestures were suited for controlling different aspects of how the Lego arm operates.

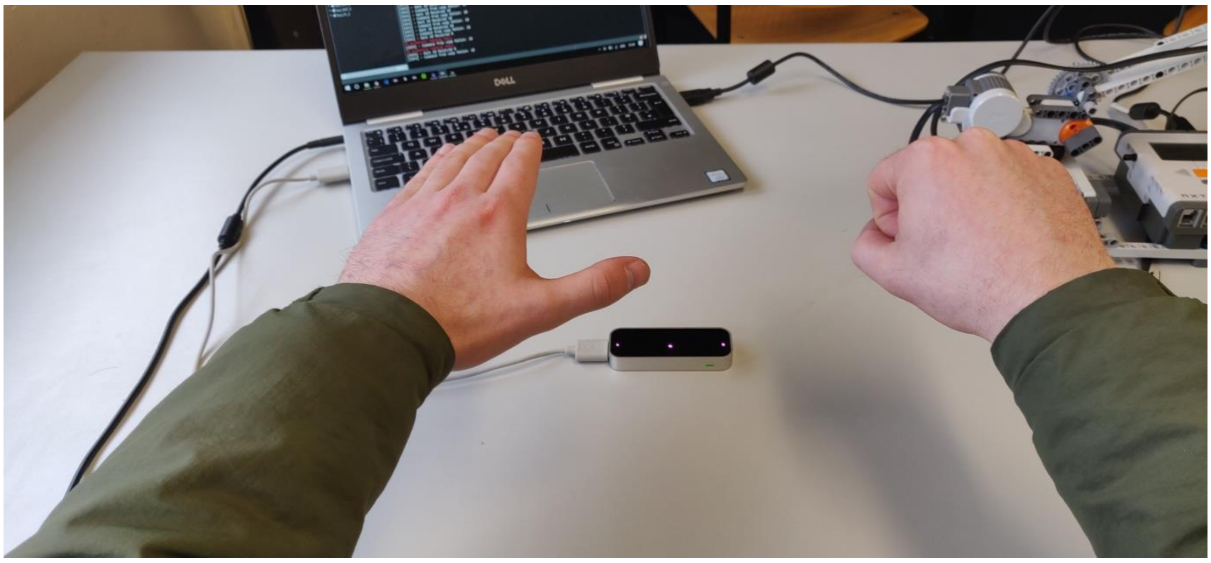
The gestures for the Lego arm can be separated into two different areas. Gestures for the right hand and gestures for the left hand.

Right Hand Gestures

We decided that for this project it would be best if the claw movements were controlled by the right hand. The right-hand gestures that we used are listed below:

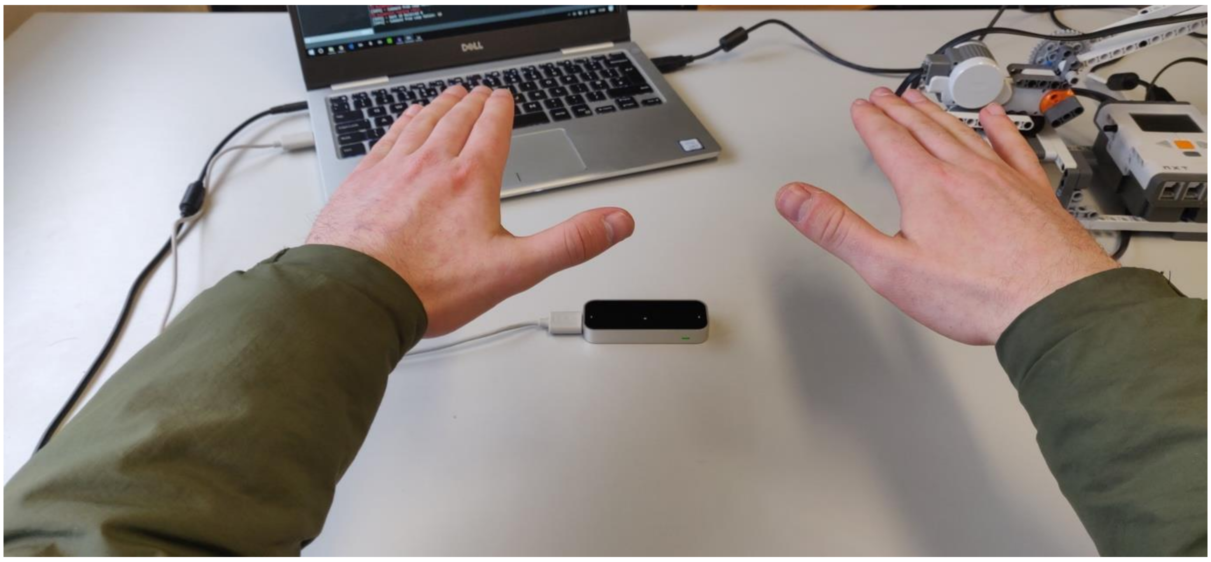
1. Closed Hand
2. Open Palm Facing Down
3. Open Palm Facing Up

Closed Hand Gesture



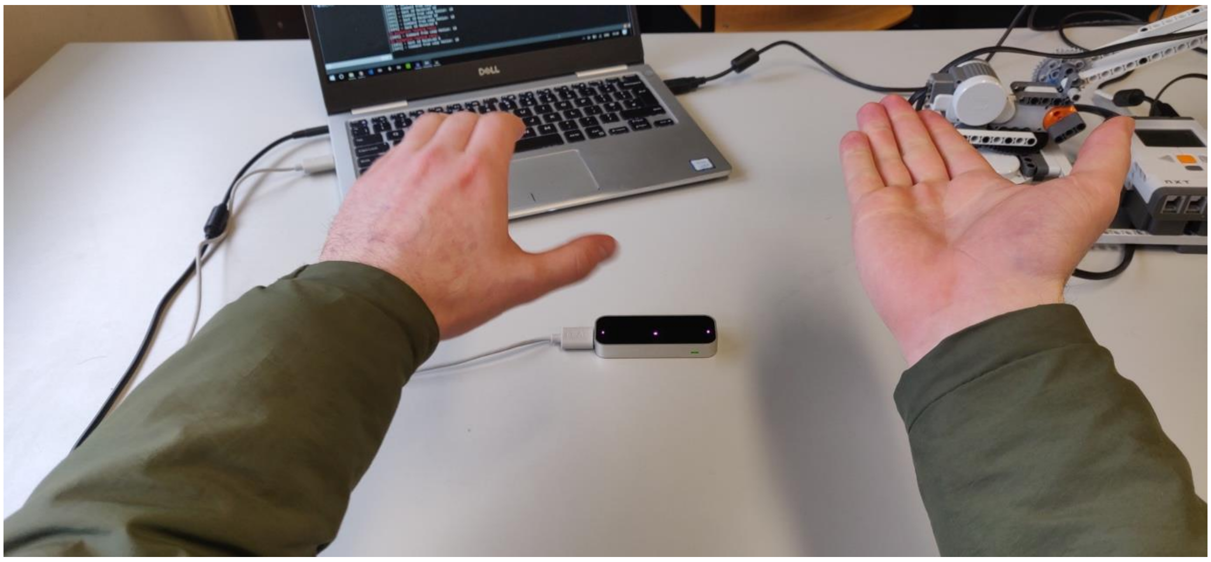
When the user clench's their right hand into a fist the leap motion registers this movement as a closed hand. This gesture controls the Lego arm by moving the motors on the claw to the forward rotation causing it to close. We felt that this gesture was appropriate because movements performed by the claw and the users hand are similar, meaning that if the user closes their hand then the claw closes as well.

Open Palm Facing Down Gesture



This gesture will stop all claw movement by sending commands to the Lego arm to stop all motor movement. We decided to use this gesture as it felt like the most natural hand resting position while using the Leap Motion Controller.

Open Palm Facing Up Gesture



This gesture is used to open the claw on the Lego arm. We felt that this was the most natural open position for a user's hand. We also felt that the accuracy of using the gestures was higher than others because of how the Leap Motion Controller software operates. We were able to calculate the user's wrist position by using check the roll along the z axis. This allowed us to accurately detect whether the user is using an open palm gesture or not.

Left Hand Gestures

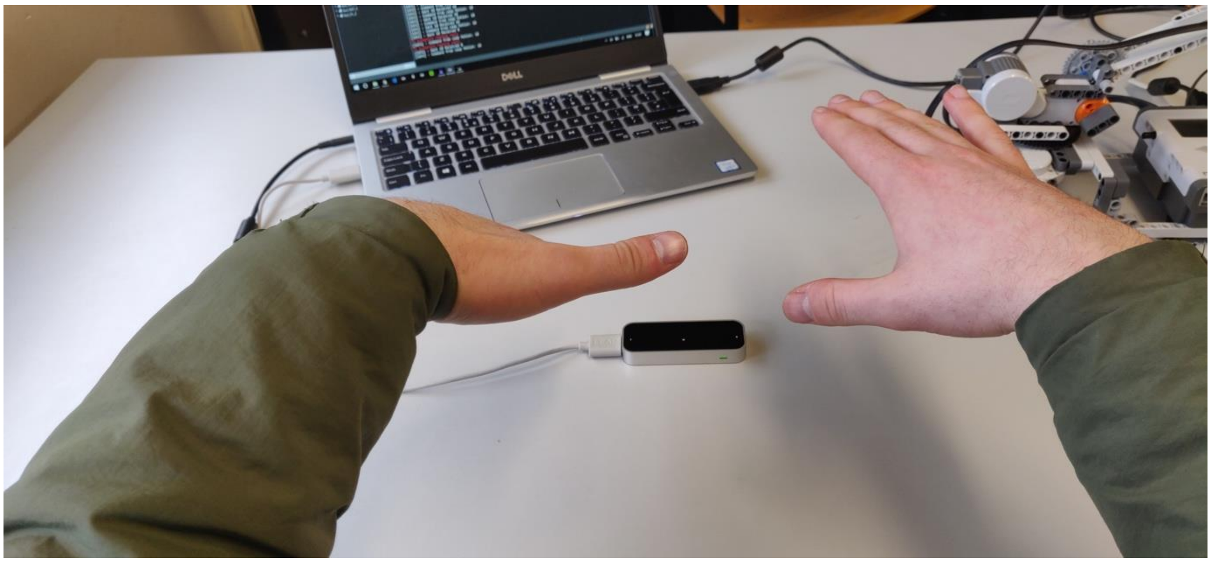
We decided that the gestures on the left hand would be in control over the movement of the Lego arm. The directions the Lego arm can move in are Up, Down, Left and Right. We tried to use a range of gestures that would feel natural for a user when trying to control the arm. The gestures we used for the left hand are listed below:

1. Circle Gesture (Clockwise/Anti-clockwise)
2. Swipe Gesture (Up/Down)
3. Closed Fist

Circle Gesture

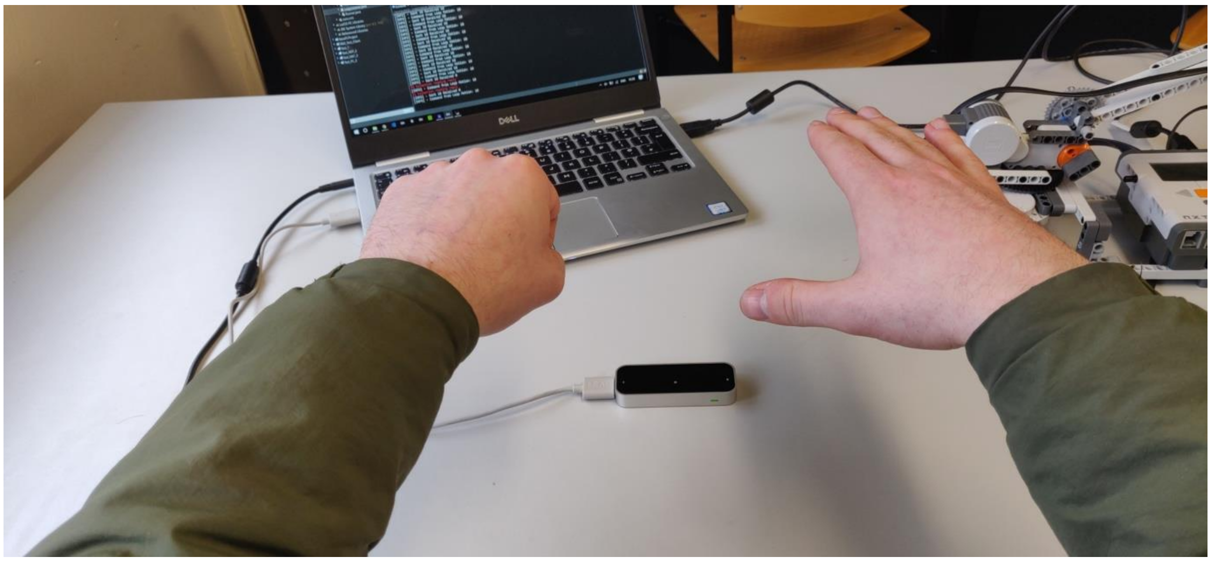
To control the left and right movement of the Lego arm we decided to use the circle gesture. This was mainly because we tried using the Swipe Gesture for all directional movement (Up/Down and Left/Right) but we found that with the swipe gesture the leap motion control had difficulty differentiating what axis the hand movement was performed on. This ended leading to weird movement’s that were happening with the Lego arm during testing. We then decided it would be best to trying and use a different gesture that was supported by the Leap Motion SDK. We then came across the Circle Gesture and decided that it suited our needs. The circle gesture works by drawing a circle motion with the user’s left hand, but what’s important to note is what direction the circle is drawn. By drawing the circle in a clockwise rotation, the Lego arm will move right. By drawing the circle in an anti-clockwise direction, the circle will move left.

Swipe Gesture



For Up and Down movement, we felt that the swipe gesture was perfect for our needs, as it required no extra development from our part as we did not need to code custom gestures using the Leap Motion Controller as the swipe gesture is one of the default gestures that is provided with the SDK. By doing a swipe motion Up or Down the Lego arm will move in those directions. We felt that this gesture felt the most natural as we are trying to emulate similar movement between the Lego arm and the movement a user would perform with their arm.

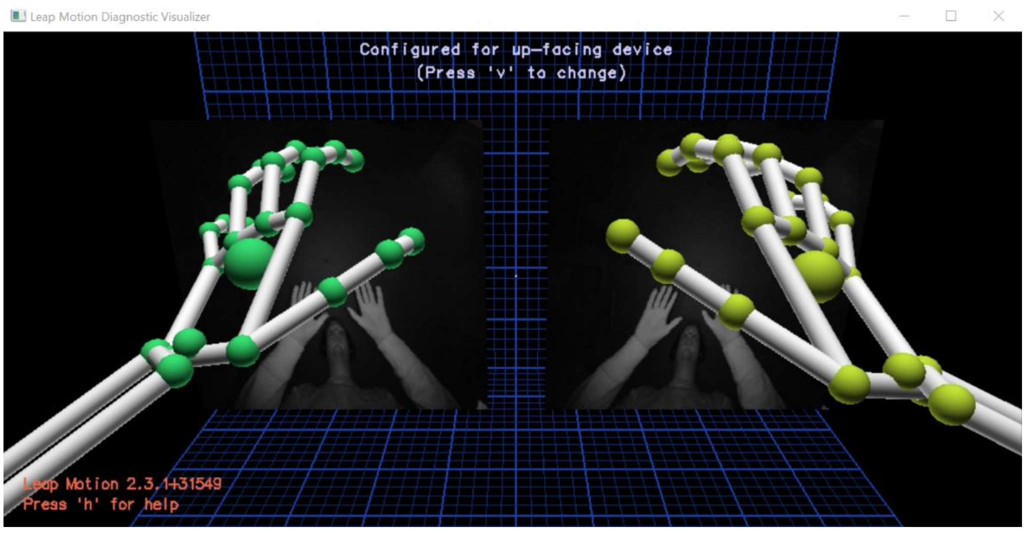
Closed Fist (Left Hand)



The closed fist gesture for the left hand is used to stop the arms motors from moving. We felt that this gesture felt seamless with all of the other gestures we implemented previous.

Hardware Used in Creating the Application

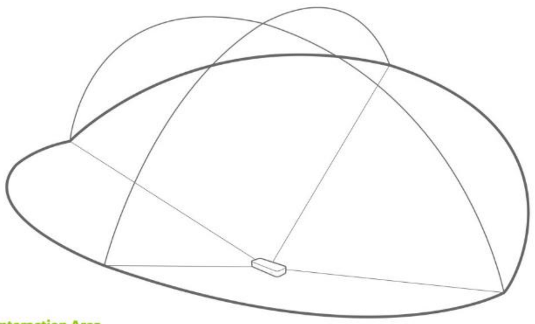
The hardware that we choose to use are the Leap Motion Controller and the Lego Mindstorm NXT.


Leap Motion Diagnostic Visualizer
Configured for up-facing device
(Press 'v' to change)
Leap Motion 2.3.1+31549
Press 'h' for help

The Leap Motion Controller is a device that scans an area of roughly 8-cubic feet above the device. It tracks both hands and all 10 fingers as they move though the open space between you and your device. The tracking is done by two cameras and three infrared LEDs. These track infrared light with a wavelength of 850 nanometers, which is outside the visible light spectrum. An example of the interaction area can be seen below.

The software installed to run the Leap Motion Controller detects your hands and fingers and translates the data into information that your computer can read. An example of how the both hands are picked up by the Leap Motion Controller can be seen below.

The Leap Motion was primarily designed for Virtual Reality such as the Oculus Rift or HTC Vive but can also be used for desktop applications without the use of a virtual reality headset. The main reason why we decided to use the Leap Motion Controller for our project is due to the wide range of gestures available. Leap offers a set of predefined gestures but also gives you the option to create your own gestures. As our project involved controlling multiple motors having the option of multiple gestures was a big advantage. Alternatives to using the Leap Motion Controller were to either use the Myo Armband or to control the mechanical arm by voice control. The reason why we chose not to use the Myo Armband was due to its available gestures. The gestures did not fit with what the mechanical arm was intended.



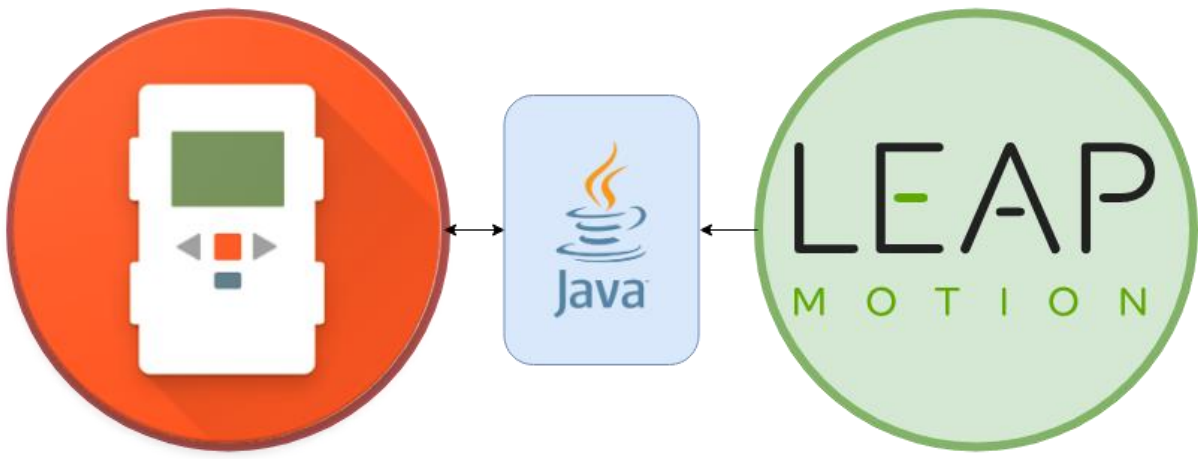
Interaction Area
2 feet above the controller, by 2 feet wide on each side (150° angle), by 2 feet deep on each side (120° angle)

The Leap motion gestures were a lot more fluid. The reasons why we chose not to use voice control was due to its unreliability and again not fitting with the project outcome.

The Lego Mindstorm is a hardware software platform produced by Lego for the development of programmable robots. Each version of the system includes an intelligent brick computer that controls the system. The brick computer can control a set of modular sensors and motors. The NXT version of the Lego Mindstorm series of products was the second release. It was released in 2006. It allows you to control up to 3 motors and 4 sensors. You also had the option to connect two brick computers which would double the amount of motors and sensors you could use. The main reason why we decided to use the Lego Mindstorm set was the fact we could create what we wanted to control. We wanted to control a mechanical arm and by using the Lego Mindstorm kit we were able to easily create an arm instead of buying a premade made mechanical arm. Another reason why we used Lego Mindstorm was due to being able to flash your own operating system onto the brick computer. In our case we wanted to control the arm using the Leap Motion Controller and to connect both we needed to add a different operating system. Multiple operating systems can be flashed onto the brick computer. Some of these include BricxCC (C Language), leJOS NXT (Java), pyNXC (PYTHON) and Ruby-nxt (Ruby). In our case we chose leJOS NXT.

Architecture for the Solution

The diagram below describes the architecture we used in the creation of this application. The language we used was mainly Java. We used Java because it's a language that we are very familiar with. Both the Lego NXT application and the Leap Motion software were implemented in Java.



Conclusions and Recommendations

In conclusion, we felt that the overall end result for this project met the overall objectives and aims that we had set from the start of this project. We felt that we met the expectations set out by the project outline and we were very happy with the feedback we received from our lecturer and peers in the presentation deliverable of the project as-well. Overall, we felt that we learnt a considerable amount about gesture based application development and we tried our best to implement these features in our application.

For recommendations, we recommend using a newer version of the leap motion tracking software as we found some complications with how accurate the leap motion hardware was for us.