Design Document of Gesture Based UI Project

Laura Flaherty G00348586

Github : https://github.com/LauraFlaherty/GestureBasedURI

Video of app in use: https://youtu.be/y_JfN7yPqgU

Purpose of the application:

The purpose of this application is to experiment with the functions of the leap motion controller engine. By the end of this project I hope that I will be able to successfully pick up a cube and be able to "swipe" a ball. All these actions are to be contained within a "gate", so objects do not misfire away from the user. If the hands "hover" over the shapes than the shape will change colour. If I had a VR headset it would be a fully immersive experience. Leap Motion consists of **two cameras** hidden under a protective sheet of dark plexiglass. When the device is plugged in, the cameras are not visible - but the three infrared LEDs, which are aimed at the space above it, light up. (*******)

Gestures identified as appropriate for this application and Gesture Implementation

The gestures I have used is the

- Pinch
- Hover
- Swipe

The **Pinch** gesture is when the middle finger, index finger and thumb are held together. The reason I have used this gesture in particular is because it is simple to some one picking up something. I have implemented the use of only three fingers because it is similar to how a person would actually pick something up. It is also very simple and only requires one hand so it allows for someone who may not be fully able bodied to use.

The **Hover** gesture is when the hand is placed over the object. In this case, when the hand is placed over the shape it changes colour. Again, this gesture is very simple and very self-explanatory. The advantage of using this technique is that it allows the user to be spatially aware. Some users may find it difficult to be fully aware of the position of certain shapes in relation to themselves. This feature is an easy way for the user to get used to the leap motion controller environment.

The Swipe gesture is not to be confused with a phone swipe. This is when the user uses their hand to bat away an object. The use uses the back of their hand to "swipe" the object away. This gesture is widely used all across the world. For example, if an insect was to fly around you, you might "shoe" them away using this gesture. Most people would have used this gesture in their daily lives, so for that reason I have used it in my app.

The tests are at the end of this document.

To implement any gestures the developer must use attach the prefabs Interaction Hand Left and Interaction Hand right to the handModels game object. Then they must attach the Interaction Manager to each hand. This allows for the configuration of the movements and fingers used.

Each object must be have the rigid body component and Interaction Behaviour script. They must also attach the Interaction Manager to the Interaction Behaviour script.

The pinch is mildly successful. For the average user it takes about four times to pinch the object and pick it up.

The hover is mostly successful. It depends how the leap motion controller hardware is operating. It can be unreliable at times. The event types must be added to the object such as the PrimaryHoverStay and the PrimaryHoverEnd. These events dictate what colour is used.

The swipe gesture takes a few attempts also. It is extremely effective when used.

Hardware used in creating the application:

For the purpose of this project I an using the Leap Motion Controller. I have decided to use this hardware because of its compatibility with Unity. The **Leap Motion controller** is a small USB peripheral device which is designed to be placed on a physical desktop, facing upward. It can also be mounted onto a virtual reality headset. (*) In my opinion this hardware seems like an exciting opportunity to experiment with both the Unity game engine and the Leap Motion Interaction Engine. The **Interaction Engine** provides physics representations of hands and VR controllers fine-tuned with interaction heuristics to provide a fully-featured interaction API: grasping, throwing, stable 'soft' collision feedback, and proximity. (**)

There are of course cons to using the Leap Motion Controller. The price of the leap motion controller is approximately 100 euro.(***) If I did not have access to GMIT's hardware I would not have chosen this hardware.

There is also the matter of relevancy to consider. The Leap Motion Controller is an older technology. At the peak of its hype in 2013, Leap Motion (which created the Leap Motion Controller) was valued at approximately $306 million, however in in May 2019 Leap Motion had agreed to sell for around $30 million. The device never caught on as a PC peripheral, but it arrived just in time to ride a wave of interest in virtual and augmented reality, offering an early interface system before the launch of ubiquitous handheld motion trackers (******).

I still believe that the Leap Motion Controller allows more freedom for creativity.

I considered using the Microsoft Speech Engine for Unity. The **Speech Application Programming Interface** or **SAPI** is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications (****). I am aware that there are pros to using the Microsoft Speech Engine. Most electronic devices have an inbuilt microphone, allowing for an easier time for development because there is no added hardware to plug in or set up.

Speech recognition is already a part of our daily lives, with digital assistants like Amazon's Alexa, Samsung's Bixby and Google Home. Speech recognition in general is a growing technology but for now is still limited to relatively simple commands. (*****).

There are, of course, disadvantages of speech recognition technology. The development and implementation of speech recognition is very dependent on the microphone's capabilities. For example, the voice input from a laptop's microphone may be heavily distorted rendering the speech recognition function of an application useless. I also believe that there is less allowance for

creativity. Grammar must be very precise and difficult to create if the developer does not have an above average grasp of the spoken language.

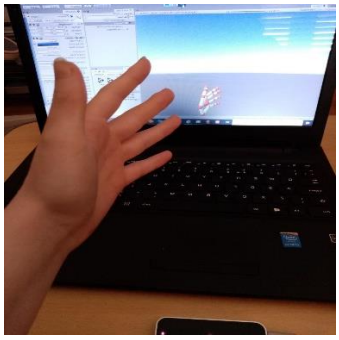
Architecture for the solution:.

Unity 2018.3.7
Unity core assets 4.3.4
Leap Motion Interaction Engine 1.1.1
Leap Motion Controller Orion Set Up 3.2.

1) Open Unity
2) Create Project
3) Import core assets.
4) Within Leap Motion Folder Navigate to Core -> Examples -> Leap Hands Demo Desktop
   This scene is the example scene where the game is built from.
5) Import custom Package: Leap Motion Interaction Engine 1.1.1

Test 1:

Play the game in from the example scene. If virtual hands are moving with your hands: Success



Test 2:

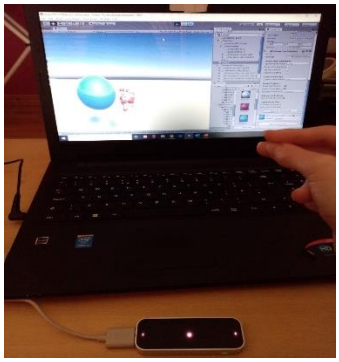
Place sphere and plane on scene. If hands can reach sphere: Success

Test 3:

If hands can collide with sphere: Success

Test 4:

Use thumb and two fore fingers to pinch and hold collide sphere: Success



Test 5:

When hand hovers over sphere, sphere turns red: Success

Test 6:
When hand is finished hovering over sphere, sphere turns blue: Success

Test 7:
Repeat tests 1 – 6 for a different shape.

Test 8:
Use hand to bat away the sphere.

Conclusions & Recommendations

I have learned that in order to use the Leap Motion Controller the device in which it is plugged into must be well capable. The LMC uses a lot of power as it extremely sensitive and must be positioned on a flat surface The LMC is not a "plug in and play" type of hardware. It requires the correct downloads the be used properly. There are multiple downloads to select from as this hardware has been around for over seven years. However as of May 2019 the Leap Motion Controller software updates have been put on hold, since the sail of it's parent company.

I have also learned about the importance of speech recognition. If speech recognition were to disappear tomorrow, there would be a lot of people effected. People use speech recognition software for texting, for using their computer, and sometimes even for controlling their own homes. Speech recognition is an essential part of the growth of smart living. It is constantly being used to make our lives easier and reduce time spent typing.

If I were to undertake this project again, I would implement more game elements. Add a scoreboard, sound and a game menu to make this app more fun and immersive.

References:

(*) https://en.wikipedia.org/wiki/Leap_Motion

(**) https://leapmotion.github.io/UnityModules/

(***) https://www.robotshop.com/eu/en/leap-motion-controller.html

(****) https://en.wikipedia.org/wiki/Microsoft_Speech_API

(*****) https://signalprocessingsociety.org/publications-resources/blog/what-are-benefits-speech-recognition-technology

(******) https://www.theverge.com/2019/5/30/18645604/leap-motion-vr-hand-tracking-ultrahaptics-acquisition-rumor

(*******) https://www.notebookcheck.net/Review-Leap-Motion-Motion-Control-Technology.98821.0.html