Mobile Interface in Augmented Reality

Divyansh Upreti (SBU ID:112026646)
Prabuddha Kumar (SBU ID: 112076553)

***Abstract*— With the growing involvement and advancement in the field of computer vision and virtual reality, one can envision a day where there would be no need to hold the mobile devices in your hand and rather with gestures or portable controllers, you will be able to fully control your device and view it on large screen. Imagine, that you are walking to a new place and you don't have to look down on your phone as it is already in a large field of view. You can perform any work and all the instructions will be visible right in front of your eyes. This is all possible with Augmented reality, which is fascinating and definitely has a great potential.**

Thinking, in this direction, we decided to design a mobile interface in Augmented reality in which we used magic leap as the AR device to project the 3D phone screens on user's field of view to create an immersive AR experience. We developed an application that can give us a very easy access to one's smartphone's apps, notifications and can provide him with a much needed and alternative way of viewing his phone.

I. Introduction

Owing the ever rising popularity of Augmented Reality devices, the future of interaction is destined to change drastically. Additionally, smartphones these days are deeply entangled in everyones life. It is our gateway, our personal space, as well as the world outside. At some point, it is inevitable that the two types of devices will have an overlap of purpose. If wearables become ubiquitous in the coming few years, it will be ideal to have ones mobile phone screen in Augmented Reality, always available in front of a users eyes. In our project, we create a mobile phone interface in Augmented Reality, to allow a user to interact without holding a mobile phone at all times.

As a means to demonstrate the proof of concept, we have used Magic Leap One, an augmented reality device that overlays virtual objects on the real world, thus fitting our use case perfectly, A 6DoF controller provides a great means for interaction with the virtual objects on the screen in a precise manner.

We aim to extend the work that we have begun, by extending the project with further improvements that we have mentioned in the Future Work Section.

II. Functionality:

The application enables users to interact with their mobile phone application via a Magic Leap Application.
Components Used:

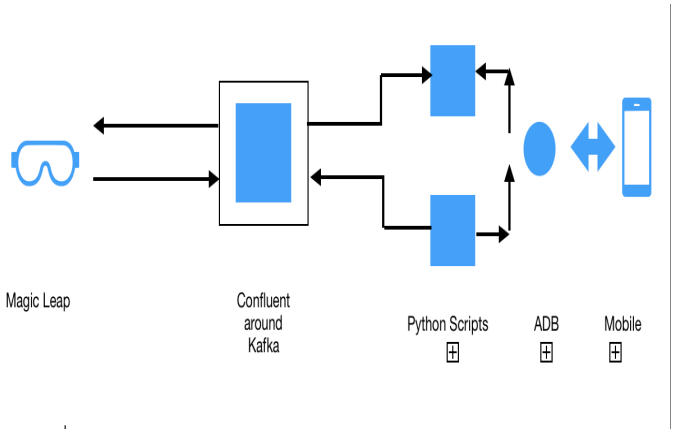

Fig. 1. Architecture Diagram

1.) Android Mobile Device: An android mobile device is the target device that will be projected onto the Augmented Reality interface. Android has been chosen as a platform as it is slightly easier to set up with an Android Debugging Bridge.

2.) Magic Leap: Magic Leap One superimposes 3D computer-generated imagery over real world objects, by projecting a digital light field into the user's eye, involving technologies potentially suited to applications in augmented reality and computer vision. It is attempting to construct a light-field chip using silicon photonics.

3.) Android Debugging Bridge: Android Debugging Bridge is a tool provided by the Android SDK that allows for a user to connect with an Android device via command line. [1] For us, it provides distinct value as it allows us to implement clicks, extract application data, and trigger applications on the phone based on a users input on the Magic Leap.

4.) Kafka: Apache Kafka is a distributed streaming platform. [1] It allows users to publish and subscribe streams of data, store streams and process them. It is extremely useful for building data and streaming pipelines that are extremely scalable and reliable.

We utilize Kafka for streaming data related to user events, relaying from the Magic Leap and back. The user events that have been mapped include screen touch events, icon click events etc. These events are then used downstream

Fig. 2. App running in Unity

to trigger actions on the users mobile phone.

The following concepts are pertinent to our project and have been used extensively as the data channel for the project:

Topics: A topic is a particular data topic that is written by multiple consumers. It can be subscribed to a consumer and written by a producer. In our case, we have a topic to stream the clicks that a user makes on the controller while wearing the magic leap, and another for sharing the users app information.

Consumer: A consumer reads data from a particular stream, and takes action based on it. In the project, for example, a python intermediary script consumes downstream to relay click on the Android Mobile Device.

Producer: A producer writes data to a particular stream, and thus acts as a publisher in a pub/sub model. For example, in the project, the App on the Magic Leap acts as a producer, and sends click locations to the mobile device,

5.) Confluent:

Confluent provides tools for a user to deploy Kafka streams and exposes a REST API for reading from and writing to streams. This is an advantage over simply using a client library for Kafka, as that creates a reliance on the underlying platform. We have used the REST API for consuming and sending data in the streams to power our application.

6.) Python Scripts:

We use python scripts to get device information from the mobile devices, as well as to implement events on the devices. These scripts read from, and write to Kafka streams to channel the commands across the application.

III. Device Architecture

1.) Magic Leap Application: The Magic Leap Application allows the user to interact with the Android Mobile Device in Augmented Reality. This application implements a variety of features and allows the user a number of options to interact with the mobile device.

Functionalities: The app displays the mobile phone screen, top application icons, and notification for the user to interact with. The user can communicate with the application using the Magic Leap Control, that ships with Magic Leap One. This allows the user to interact with the application as if wielding a laser pointer and uses ray casting under the hood. The user can then press the bumper button to invoke a click operation. Additionally, the user can click the floating icons to launch the corresponding applications. The user can also click on a notification button to launch the application corresponding to the notification. button.

To get the application information and relay the click information, the Magic Leap Application receives data from, and sends data to Kafka streams that relay data into and out of the device.

2.) Android Application:

A screen recording android application * streams the devices screen in real time, and allows us to relay the screen in mjpeg format. Our Magic Leap Application connects to this stream to get a live feed of the users screen via Http requests.

3.) Kafka Data Streams:

There are two primary tasks to be handled by the Kafka Topics:

1.) Send application data from Android Device to the Magic As the Magic Leap application starts, some initializing information is relayed from the phone to the Magic Leap. A python script is running in the background and is responsible for sending this data into a Kafka stream. This includes: The application information (App Name, Package Name and Icon), which are used to render the icons (app name, package name), and to invoke the right application on the mobile device on clicking. Information about the current notifications on a users mobile phone, so that they can be rendered on the Magic Leap Application.

2.) Send user input information from Magic Leap to the Android Device: When a user clicks on an interactive item, the information needs to be relayed back to the Android device. Upon a user click, the information is sent into a Kafka stream and is consumed by a background python script that forwards the request to the Android device. The events that are shared to the Android Device are: Click on the screen: If the user clicks the screen, the click event is sent to the stream and results in a click on the same spot. Click on an app icon: If the user clicks on a floating app icon, then the package name of the corresponding application is sent in the stream, resulting in the app being opened. Click on the notification panel: Upon clicking the notification panel, the app corresponding to the notification is opened.

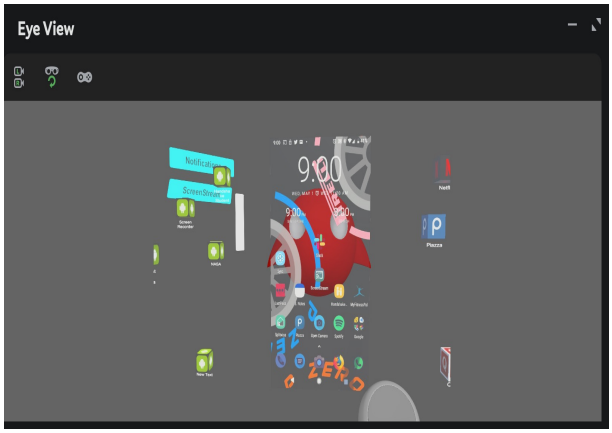

Fig. 3. App running in MLRemote

Fig. 4. App running in Magic Leap

4.) Python Scripts:

Two python scripts running in the background power the communication with the Android Device. To facilitate this communication, it leverages an Android tool, ADB (Android Debugging Bridge). This allows us to execute screen clicks and app invocations on the mobile device. There are two python scripts:

1.) AppInfoExtractor: This script is responsible for sending initialization information to the Magic Leap Application. This includes information about the apps on the users device and the notifications that the phone is currently displaying.

2.) RunCommands: This is a more inward facing script and is responsible for executing the interaction events that the user performs with the Magic Leap. This includes the clicks and the app invocations that the user performs on the Magic Leap Application.

5.) Unity app features:

Inside Unity, we created a plane over which the stream from our mobile will be projected. The streaming was simulated by scripting and plane acted as a projecting device. On its side, the notification menu is composed of a panel, which has constraints on its vertical layout. This property helps to consolidate the notifications if they arrive in a very large number. The menu is at an angle to the main screen. Encircling these is a large sphere where we have pre decided the locations of the icons where they will be float upon instantiation. The icons float in the air because of an animation script that we have given to them to make the app more interesting. There are consumer and producer functions that start respective coroutines when they run and thus consume and produce data respectively. We have also included a ray to point at the world objects that utilizes the position and orientation of the controller of magic leap.

IV. Future Work

Given the scope of the project, we believe that there are interesting things to follow up this work with. Some of the things that we want to pursue in the near future are:
1.) UI: Having an app drawer, as well as a desktop, with the favorites. Make the icons more interactive, increase in size when the ray hits. Swipe to make icons appear/disappear.
2.) Ability to place the screen on the walls in order to have custom placement. 3.) Enable multi-screens spread in the workspace

References

[1] Android Development Bridge Information
https://developer.android.com/studio/command-line/adb
[2] Screen Stream Application
https://github.com/dkrivoruchko/ScreenStream