# Eye-Based Shopping: Gaze-Assisted Interaction and Communication in Offline Shopping

**Mengxiang LIN**
**44201050-0**

Master of Engineering

Supervisor: Prof. Jiro TANAKA
*Graduate School of Information, Production and Systems*
*Waseda University*

January 2022

# Abstract

Along with the growth of digital technology, online shopping has gradually ushered in a period of rapid development. However, traditional entity stores are not going to be replaced by e-commerce anytime soon. While there are many benefits to online shopping, entity stores also have their own unique advantages, especially for consumer goods with high liquidity. Also, many people think offline shopping is an enjoyable outing with family or friends. Even so, in terms of technological development, e-commerce based on digital technology performs better than entity stores. Therefore, it is important for retailers to combine information technology with their own sales methods to bring consumers an easy, convenient and personalized shopping experience. Traditional offline shopping leaves some problems. It lacks the personalized interactions and the real-time communication, which reduces the shopping efficiency. Over the years, with mixed reality technique developing, a growing number of potential possibilities has been provided for tangible interactions in commerce offering.

In this paper, we propose an eye-based system that combines augmented reality with eye-tracking to enhance shopping interaction and communication. Our system mainly has two part: eye-tracking interaction and real-time communication. By tracking the user's eye movement instead of hand ray, the system is able to capture the interests of the user efficiently to support natural interactions. Additional information in AR is only revealed after the system has recognized that the user's visual attention is focusing on the item. Also, the system aims to promote communication through sharing items or suggestions using eye-tracking technology. Users can share the 3D model for others who are in the same net room. And by detecting the users' eye gaze points and indicating them on the 3D model, the user can let other user know which context he is focusing on. By combining AR and eye-tracking technologies to visualize the user's focuses and share them with other users, the efficiency of the exchange of ideas and suggestion for offline shopping can be powerfully enhanced.

**Keywords:** offline shopping, augmented reality, eye-tracking, communication

# Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof. Jiro Tanaka, for his advice and guidance throughout my research work. Working under his supervision and guidance has been very enjoyable and I have learned and grown a lot in the field of human-computer interaction. This is undoubtedly a precious treasure that is rare for my future work and life.

In addition, I would like to thank all my dear partner in the IPLAB who have provided me with help and guidance. With their opinions and corrections, my research can be completed on schedule. Meanwhile, their enthusiasm and friendliness have infected me, allowing me to overcome my academic difficulties and move on to the next challenge step by step.

Finally, I would like to thank my family for supporting and encouraging me throughout my entire learning process. In the process of completing the thesis, if there is no support that they give me in life, I will not have the opportunity and time to complete the learning tasks and rectification work.

# Contents

# List of figures

# List of tables

# Chapter 1

# Introduction

## 1.1 Background

Augmented Reality (AR) is a technology that cleverly combine the real world with virtual information. In visualized AR technology, the user is supposed to see a new reality by superimposed virtual objects on the real world by head mounted displays, enriching the real environment to achieve a sensory experience beyond reality. AR technology are being widely used in many activities such as marketing, architecture, media, education, medicine, interior design, advertisement, shopping, etc [1]. AR has attracted a great deal of attention in daily life of human. How this leading-edge technology can be used to enhance the experience of human life has been a sustainability discussed topics. [2].

Although online shopping is gradually entering people's daily lives nowadays, offline shopping also has its own unique advantages. In the *Nielsen Global E-Commerce and New Retail Survey* [3], it was revealed that 69% of respondents think offline shopping was an enjoyable outing with family or friend, and 67% enjoyed the pleasure and satisfaction of shopping in entity shops in China. Therefore, entity shops are not going to be replaced by e-commerce at short notice. But the report also points out that for retailers of entity shops, they must change with the times through advanced digital technology on the traditional offline sales methods. And using more flexible means to engage offline consumers, many key factors are critical, such as: enhancing the interaction between items and consumer, sharing the shopping experience with others, etc. AR as augmented physical environment technology excellently meets this need pretty well.

However, it seems that wearable AR devices are not becoming a mainstream shopping tool in shopping nowadays. To investigate whether providing customers with the shopping tool in AR leads to an increase in purchase rates over traditional shopping applications, a study was done by Hamraz Javaheri et al. [4]. They tend to assess the effect of wearable AR on customers' shopping behavior and attitudes by comparing experiments with HoloLens and tablets. From the analysis of their results, it is observed that using the wearable AR shopping app observably increased participants' final purchase amount compared to traditional shopping apps on tablets.

In order to solve the problem of lack of effective guidance and incentives for offline shopping customers, our research focuses on AR technology and eye-tracking technology. We propose a method to combines AR with eye-tracking to enhance interaction and communication in offline shopping experience. Our system differs from the current AR shopping systems that relies on hand rays, transforming hand rays into eye gazes for the interaction between virtual and reality. It reduces the intrusiveness of the AR system to the customer's shopping flow and provides more attractive shopping experience in a physical environment. The purpose of the system is to improve the offline shopping experience in the following ways:

- Provide the interaction between the real world and the virtual world in eye-tracking form. Most of the interactions are designed to be completed smoothly by eye gaze alone. And some of the interactions that cannot be performed independently by eye gaze will use input streams that do not interrupt the user's shopping flow, as well as voice commands.

- Provide the method that dynamically reveals information on demand. Virtual Additional information is only revealed if the user shows interest to this item through eye gaze.

- Provide the platform for users to share their shopping experience with others. In such an environment, it allows users to communicate information and ideas more efficiently with their peers.

## 1.2 Organization of the Thesis

The rest of the thesis is organized as follows: In chapter 2, we will introduce some related works. In chapter 3 we will describe the goal and approach of our research and also some use cases of our system will be told briefly. In chapter 4, we will present the concept of the system design. System hardware devices, software environment and our innovative design ideas will be introduced in detail. In chapter 5, we will show the implementation details for the system development. In chapter 6, we will make a conclusion for our implemented content and talk about some potential possibilities for our future work.

# Chapter 2

# Related Work

## 2.1 AR Shopping System

Using augmented reality technology can powerfully support adding additional information to the objects in real world. Especially, offline shopping malls are an excellent place to introduce augmented information. Because when shopping offline, there are often so many items in the shopping mall that it is difficult for users to choose. In this case, additional information about the items is required, such as its composition, purchase history, other users' recommendation, comparison with other items of the same type in the shopping mall, etc. The use of augmented information is a good way to attach unlimited additional supporting information to a limited realistic environment. There are several researches and applications that use augmented reality technology for shopping.

Augmented Reality, as a technology that can transfer information from 2D to a more immersive 3D, can effectively enhance everyday users' life experience, especially the shopping experience. The features that online shopping can offer are rapidly increasing due to the development of digital technology, while offline shopping is undergoing major technological changes. Rashid et al. [5] provide some Augmented Reality interfaces that developed for cell phone devices that connected to physical environment to narrow the gap between online and offline retail store. Their preliminary evaluation indicates that users are willing to use such a smart system to help them interact with the items in the physical environments.

With the advent of the era of business interaction, the sales industry can no longer use traditional sales methods. They have to combine the sales industry with digital information technology to create more innovative methods to enrich the customer's shopping experience. Jayananda et al. [6] are interested in smartphone augmented reality applications. They did a research to assist shoppers in navigating to the target products. The main purpose of their research is to design a feature-rich mobile application. It has an innovative shopping navigation system with Augmented Reality technology.The subsequent part of this research is to optimize the efficient handling of customers' shopping lists, use object detection to provide personalized recommendations in augmented content and remote shopping.

What's more, Junho et al. [7] has explored the effectiveness of augmented reality technology in the context of grocery shopping in mobile device. They develop an assistance system in aid of customers to assist them in making decisions about which products to purchase. They constructed an AR-assisted grocery shopping application published on mobile devices. It is intended to make personalized, real-time recommendations about targeted products to users and highlights products to give the additional information. Their preliminary evaluation of a typical grocery retailer strengthens that adding AR tags to products can greatly reduce the search time to seek out targeted items. And with the aid of attaching the meaningful colors to AR tags, the user's ability to distinguish recommended product can be powerfully improved.

The primary concern of current commerce services is to satisfy customers by means of a series of business and technology solutions. Therefore companies need to explore different approaches in not only desired products navigation, but also to enable users to enjoyed the pleasure and satisfaction of shopping in order to revolutionize the in-store shopping experience. Meegahapola et al. [8] has presented Smart Phone based Mixed Reality Application (SPMRA). The system is based on a smartphone-based mixed display platform that gamifies the entire in-store shopping experience with an easy-to-use and low-cost solution. This highly interactive, intelligent platform makes it easy for retail owners without technology experience to create mixed reality-based experiences, and also enable customers to experience unique shopping pleasures in-store.

In summary, the use of augmented reality technology to enhance the entire shopping process can be a great method to increase the purchase intentions of consumers and enrich the offline shopping, which will give the consumers an interactive real experience in the entity stores.

However, these studies also remain some shortcomings.

1. **Intrusive User Interfaces:** Whether in Ahn's or Jayananda's research, their system prototypes were built on mobile phones. This means that customers need to use at least one hand to hold the phone at all times during shopping to display augmented information. Obviously, it is very intrusive as customers has to be occupied with some hand resources and the shopping experience would be interrupted compared to the usual way.

2. **Lack of Communication:** In all of the above studies, the systems were essentially supported only in single-user mode. Usually, it is quite hard for a person to pick out one of items from several types. This is the reason why people often want to hear the suggestions from others (friends, family) and go shopping together. Therefore, in real offline shopping, AR shopping system should be developed for multiply users. So, allowing the interaction of information communication between multiple users is also a major point of AR shopping system.

## 2.2 Eye-Tracking

The eye, or vision, is one of the most critical senses for humans. Nearly 80% of all sensory impressions are transmitted to the human brain through visual channels. [9]. Many studies show that eye movements are closely linked to cognitive process. Eye movements can provide information about our thoughts and desires to others, which is the meaning of "eyes are a window to the mind" [10]. So by capturing the movement of the human eye, the user's intention can be well grasped.

The major advancement in the history of eye-tracking technology was the invention of a head-mounted eye gaze tracker, this technique is still widely used [11]. From previous researches to the present, there are various methods to record the position of the eye relative to the head. Here are some typical eye-tracking systems using head mount have been described.

Hartridge et al. [12] use an original apparatus cto capture motion through a cinema or other camera with proper magnification. When their method is used, head movement is subject to various restrictions. They recorded the actual head movements of the experimenter through an artificial cornea. This cornea was attached to a suitable fitting which was clamped to an incisor. Meanwhile, in order to record head and eye movements, they also analyzed data on the light from the UV lamp falling on two photocells after reflecting on the corneal surface.

From the 1950s to 1960s, Yarbus [13] did some studies in eye-tracking. He used a original device to do the research on eye movement: a rubber suction cup with a mirror. Since it is directly attached to the sclera of the human eye, the light reflected from the mirror on the photographic paper can indicate the eye movement. During the experiment, he had to bite down on a special plastic all the time. No one else wanted to repeat this experiment due to it's extremely uncomfortable experience. Furthermore, the efficiency of this research is also very low. It could last only a few minutes, whole it took weeks or even months to process the data obtained in those minutes.

Mackworth et al. [14] used a simple head-mounted optical eye-tracking device to understand where people is currently looking in a given real situation. Their eye-tracking device was composed of a cine-camera and a periscope. When the subject moves his eyes with head, the periscope would emit a light spot reflected from the subject's eyes, while the cine-camera takes images of the changing scene. To indicate the position of the subject's sight-line, the image of the physical environment is momentarily marked as a spot in the cine-camera.

Additionally, previous considerations of how to record the position of the eye must prevent or allow for the effects of head and body movements. This is because this factor shifts the head datum line relative to the field of view. However, B. Shackel [15] described a method that allows for head and body movement, thus providing a more comfortable and convenient recording of the eye's point of view in motion. The immediate visual field is first recorded by the head-mounted motion camera, while the eye's point of view is marked by the superimposed white spot of the eye position recording device. This was a breakthrough for eye-tracking research at the time.

With the development of eye-gaze sensing technology, these methods now can track the human eye more accurately while being less intrusive to the human eye. In consequence, there is an increasing number of studies using eye gaze as an input stream. It has led to the creation of a number of devices that can track the direction of human eye. But for an ideal tracking device, there are some requirements that are still not fully met by existing technology:

- Provide an unobstructed field of view.
- Allow comfort and natural interaction.
- Compatibility with much more real-world context.
- Non-contact with users.
- Provide good temporal dynamics and speed of response.
- Enhanced accuracy and precision.

From the beginning of the study till today, with the continuous development of eye-tracking technology, its application has expanded to almost every field of human daily life. Eye-tracking research provides insights into human behavior specifically for business and scientific purposes [16]. Some spacial examples that use eye-tracking are as follows:

1. **Disabled People:** As some people with disabilities who use their eyes almost exclusively as a means of communication with the outside world, eye-tracking provides a convenient and natural way for them to interact. Eye-tracking technology is becoming a mainstream assistance for them. [17] [18]. Santis et al. [19] present an effective eye-tracking device that can track the user's eyes without additional infrared illumination.

2. **Ophthalmology:** Eye-tracking methods that record eye movements to measure the process of human eye position and gaze point movements fit well with the needs of the field in ophthalmology. It can help ophthalmologist better understand eye movements and is practical for developing means of prevention, diagnosis and treatment. These researches and the methods that they utilize eye imaging in medical field are explored in depth by Harezlak et al. [20].

3. **Neuropsychology:** For decades, researchers have extensively studied eye movements in physiological and psychological research by using various types of eye-tracking devices [21]. They have defined and detected a variety of different eye movement events and tried to find the relation between these events and cognitive processes as well as perception [22] such as attention, emotional expression and learning ability. For low-age children, Xu et al. [23] have used eye-tracking technology to assess cognition on intelligence-related scales.

4. **Marketing:** Motivated from the growing importance of visual marketing in practice, using eye-tracking can investigate the cognitive consumer purchase behaviors which commonly known today as Neuromarketing [24]. Evaluating eye-tracking studies can help understand the human behavior related to consumer cognition in marketing product. Previous research by Pieters [25] provides a case study of the application of eye-tracking to ad pretesting. Moreover, Zamani et al. [26] reveals that products of larger sizes tend to be more likely to be preferred.

After the above mentioned achievements of eye-tracking in a wide range of fields, we focus on the application of the shopping field. Users' implicit shopping intention, i.e. whether I should buy this thing or not, is hard to identify in physical reality. But if some special measures are used to recognize it, it will be significant and valid information.Eye-gaze as a most direct human subconscious can be recognized to help shopping systems to individual customers' shopping motives.

Lee et al. [27] proposed an experimental paradigm to identify the implicit intentions of customers. From the result of experiment, fixation count, fixation duration and multiplication of first fixation duration, as well as visit count, indicates a variety of tendencies about whether customers have intention to buy it or not. By comparing above tendencies, the implicit shopping intention of customers can be well analyzed.

In addition, tracking eye gaze We further can not only be used to determine the subconscious of consumer behavior in decision making and their purchasing process, but also be used as an input stream for shopping interactions.

Phelippe et al. [28] have discussed an AR solution based on eye-tracking for customers who want to enhance their overall shopping experience through augmented information content. When customers shop offline, it is often difficult to find fundamental information (e.g. expiration date) about products because of the limited display space in offline shopping mall. The application they published on the HoloLens equipped eye-tracking technology to set target and support the information display. It indicated that the convenience of gaze-based interaction and having information displayed around products will make shopping more enjoyable and pleasant.

# Chapter 3

# Research Goal and Approach

## 3.1 Goal

The goal of our research is to provide a gaze-assisted system that combines AR technology with eye-tracking to enhance shopping interaction and communication. Not only can users interact with virtual objects and real environments with each other, but users also establish a connection with each other. By tracking the user's eyes, it can recognize users' implicit shopping intention well, so as to provide the most comfortable interaction experience to users. Our goals can be divided into the following points:

1. By tracking the user's eye movement instead of hand ray, the system can capture the interests of the user to support natural interactions. Through the "object detection" function, our system can detect items in shopping mall to reveal corresponding augmented content and accept eye gaze as an event trigger to start the corresponding interaction;

2. Apply the Real-time communication feature to allow our shopping system to support multi-user use. The system aims to build a platform to promote communication between users who are going shopping together.

## 3.2 Research Approach

### 3.2.1 Approach

In order to enhance the user's offline shopping experience when going outing, we equipped two approaches to make offline shopping more attractive and enjoyable:

- **Approach 1:** Implement eye-tracking interaction between users and objects instead of hand-based interaction.
- **Approach 2:** Allows users to communicate with each other in real time while shopping offline.

### 3.2.2 Novelty

Compared with other existing offline shopping systems, the novelty of our research is mainly reflected in the following aspects:

1. We propose a gaze-assisted interaction instead of the conventional hand-based interaction. Our design tends to use eye gaze as the mainly input stream for shopping interactions in our system and explore the subconscious decision in users' shopping experience.
2. We provide an eye gaze method that allows users to get information about items on demand. Additional information of the target object is only revealed if the user shows interest to this item through eye gaze.
3. Based on networking structure provided by Unity, we apply the multi-user connection feature to support user communication promotion. It allows users to sharing items or suggestions through eye-tracking technology .

## 3.3 Use Case

When users enter an offline physical shopping mall, they can put on an AR headset and activate the gaze-assisted shopping system if they want to have a richer and more convenient shopping experience. Since the user is already in the shopping mall, a complex physical environment, the augmented virtual objects will not be visible to the user's view at first. (Our system is designed to assist without disturbing the user's normal shopping process.)

When the user finds a physical items of interest that he can personally hold in his hands, he select the real interaction mode of the system to use. After the user gets the items in his hand, he wants to see the details of the items, such as the manufacturer or the composition. So he gazes at the items with his eyes for 1 second and then moves his eye gaze point to the right, and this augmented additional information appears on the right side of the items. After reading the details, he also wants to know if there are any items similar to this one in the current shopping mall, which can help he make a decision whether he should buy the current items in his hand. Therefore, he moves his eye gaze to the left and the list of similar items is revealed on the left side of the target items. Through this comparison, the user quickly picks out the most favorite items in that category.

The user finds out another item in the next shopping store, but that item is a valuable item and the staff can not bring out the physical item. At this point, the user switches the interaction mode to virtual mode. After the user gazes at the target item he likes, a 1:1 size virtual model of the item appears in his AR field of view. The user gets a series of additional information about the item by moving the eye point. In addition, the user also manipulates the item at close range by combining his own hand gestures.

When the user still can not make a decision, he meets his friend in the store who was also using the same gaze-assisted shopping system. Suddenly, he invites his friend to shop with him together. The friend then joins the network room created by the user. After allowing synchronized access to each other, the user and his friend can see where each other's eyes were gazing at. The user shared the virtual model of the item he had been hesitant to buy with his friend for exchanging their ideas. Because they can visually see where the other person is focusing on, they were able to communicate their ideas about the item pretty smoothly.

# Chapter 4

# System Design

## 4.1 System Overview

Figure 4.1 shows the overall structure of our system. Based on eye-tracking technology, the system mainly focuses on 2 parts: Eye-Tracking Interaction and Real-Time Communication.

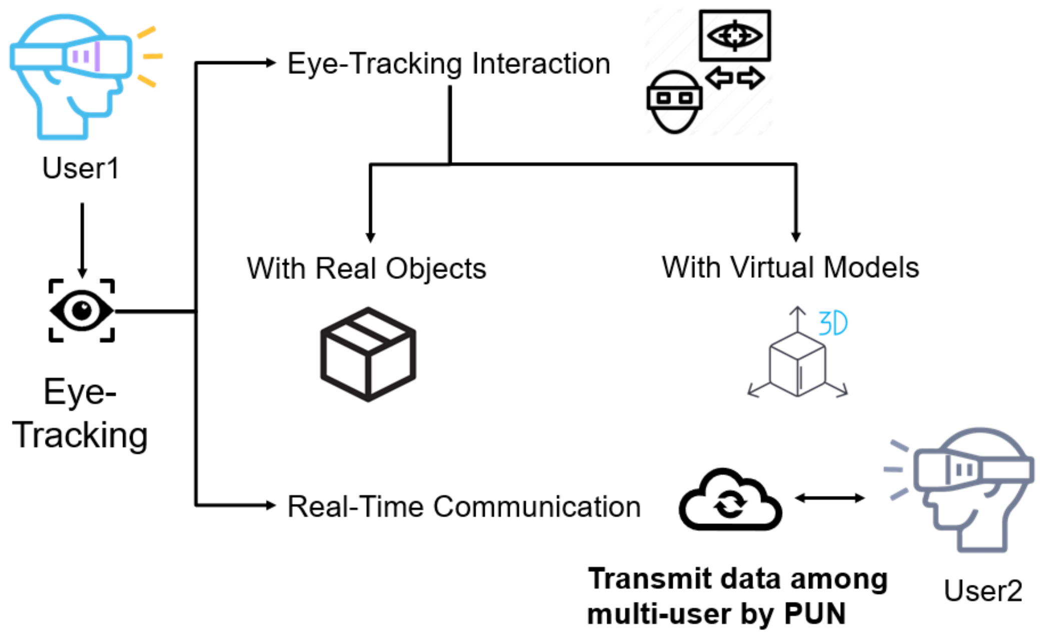


Fig. 4.1 System overview

Eye-tracking interaction is the core part of the system. The mainly function is that it can detect the users' eye gaze points to filter out which items the user is interested in. We propose using eye-tracking that allows the system to reveal the information on users' demands. That means, additional information is only revealed if the user shows interest to this item through eye gaze. It is useful for use cases of striking a balance between informative AR view and information overload.

In real-time communication section, the user can share the virtual models for others. When another user accepts the request, they can manipulate the virtual object together and share the movements. All participants in a shared experience can view each other's interactions and communicate with others. It can help the exchange of opinions between users. Moreover, by detecting the users' eye gaze points and indicating the points on the shared model,the user can let another user know which context he is focusing on to promote communication.

For summary, our system provides an eye-based system that combines augmented reality with eye-tracking technology to enhance shopping interaction and communication. By tracking the user's eye movement instead of hand ray, the system is able to capture the interests of the user efficiently to support natural interactions. Also, the system aims to promote communication through sharing virtual objects and visualization of concerns using eye-tracking technology.

Following subsections contain information about key parts of this research. We will describe the details of our system design in each of five parts:

1. Hardware support.
2. Software environment including software and technical support.
3. Object detection to find the interactive target in real world.
4. Details of eye-tracking interaction design.
5. Details of real-time communication design.

## 4.2 System Hardware

### 4.2.1 Mobile Display Device

Our augmented reality eye-based shopping system is designed to enhance the user's shopping experience, so the system needs to be developed on a mobile platform. If a mobile phone is chosen as the platform for users to see virtual 3D objects in the real environment, it will take up the user's hand resources when user is interacting with commodities. This would severely disrupt the user's normal shopping experience and bring a sense of fragmentation to the user. In addition, a camera is needed for our system. It is not only used to recognise objects in the real environment, but also needs to be able to be used to track the user's eyes to achieve eye-tracking interaction that is the core part of our system. Therefore, we choose Microsoft HoloLens (2nd gen) instead of a mobile phone as the mixed reality HMD (head-mounted display) device to achieve this goal. As Microsoft HoloLens 2 fulfils above requirements perfectly, it can bring an attractive and non-intrusive shopping experience to users.

Microsoft HoloLens 2 as the mixed reality HMD device is extremely good in terms of Field Of View (FOV), size, weight and wearing comfort. Its lightweight and integrated design is well suited to offline shopping. HoloLens 2 is equipped with eye-tracking and hand tracking technology, which means it is capable of detecting the hands or gaze point position without using any additional device. This contributes to the user feeling relaxed and fully engaged in shopping because they don't have to hold any controllers to interrupting interaction in shopping. The depth camera on HoloLens 2 performs an iris analysis to support the eye-tracking interaction of our system. Figure 4.2 is the HoloLens 2 device we use in the research.

### 4.2.2 Development Device

As for the development device, we used a PC with the windows 10 operating system to develop our system. It provides a platform for us to process programs and release projects to the HoloLens. In addition, as required by the system, it is desirable to have a hard device

Fig. 4.2 Microsoft HoloLens 2

other than HoloLens to act as a server to transfer the data. The PC can also serve as this server to connect the 2 HoloLens to the same local area network for data transmission. The configuration of the PC we use is shown in Figure 4.3 and Table 4.1.

Fig. 4.3 PC

| | |
|---|---|
| Operation System | Microsoft Windows 10 |
| CPU | Intel® Core™ i7-7700HQ CPU @2.80GHz 2.80GHz |
| Graphics Card | Intel® HD Graphics 630 |
| Ram | 8.00 GB |

Table 4.1 The PC setup information

## 4.3 Software Environment

### 4.3.1 Development Tools

We used the Unity, a cross-platform game engine, to develop our system. Unity is one of the leading real-time development platforms that all development scripting is done in C Sharp. It provides a development platform for Universal Windows Platform (UWP) applications that support running on HoloLens. Also, Unity's built-in XR framework can upgrade the system from a video-based experience to an AR experience. The Unity version we used is 2019.4.28f1.

### 4.3.2 Technical Support

The software environment support is shown below:

- Mixed Reality Toolkit (MRTK) V2.7.2 [29]: MRTK is an open source project for shared base components, a building block for common interaction and user interface controls. Unity-based MRTK provides some features and components that is used to promote efficiency of cross-platform mix reality app development in Unity. Moreover, it supports a wide range of platforms, including Head-Mounted Displays (HMD) such as HoloLens. When developing apps for HoloLens or other UWP, developers are required to set up the right input module, camera, cursor or motion controller for the scene so that the user can interact with the content. MRTK makes all this works much easier.

- Vuforia Engine SDK V10.1.4: Vuforia [30] is an AR development platform that transforms real-world objects into interactive experiences. It aims to help developers create a new level of interaction between real-world objects and virtual objects. Vuforia SDK encapsulates the computer vision modules used to recognise and track images in real time. With its features, developers can connect the real world with the digital world. We used Vuforia Engine and apply the object detection function to locate real-world objects and augment the interactive content.

- Photon Unity Networking (PUN) [31]: PUN is a Unity package for multi-user application. It is ideal for developers to develop and launch multiplayer applications in a networked environment. PUN is the standard cross-platform multiplayer service such as mobile, desktop, web or consoles. We used it to connect multiple HoloLens and transfer virtual objects‘ data.

## 4.4 Object Detection

Our system is based on the object detection technology. Only by being able to detect the individual items in the shop can the rich interactions be achieved. We use Vuforia Engine [30] to implement this function and built the system with unity, which Vuforia Engine can be easily imported into.

Firstly, we get the physical object data with its significant vantage points through The Vuforia Object Scanner. The Vuforia Object Scanner [32] is an Android application used to scan a physical 3D object. Then, we have assembled these model data to build a model database so that our system can recognise the different objects in the shopping mall with the support of this feature points data. Next, when an object whose features have been recorded enters the hololens camera acquisition range, corresponding virtual objects are generated at the object's location. Such virtual objects are managed by the scripts that we have written, and the state changes between visible and invisible to striking a balance between informative AR view and information overload.

## 4.5 Eye-Tracking Interaction

The vast majority of today's AR shopping systems use the hand as a medium to build interactions between the real world and the virtual world. But generally speaking, people's hands are busy when they are shopping, such as picking out objects, holding a shopping bag or a backpack, etc. If an AR system requires the user to reach for their phone or use gestures to control the augmented content in the AR view while shopping, it can always be overwhelming. Furthermore, as these are non-shopping related actions, they will make the user feel that their shopping experience is being interrupted and complicate the user's direct and natural interaction with the objects. Therefore, instant of hand ray interactions, our system needs to provide some other method to support more fluid interaction designs that do not take up hand resources.

In our system, we have designed a number of eye-based interactions based on eye tracking technology to replace hand rays. This means that people do not need to use hands to interact with virtual objects while shopping, but only need to move their eyeballs to complete most of the information interaction.

The main interaction of our system is that proposing using eye-tracking that allows the system to reveal the information on users' demands. It can detect the users' eye gaze points to filter out which items the user is interested in. That means, additional information is only revealed if the user shows interest to this item through eye gaze. With such a core interaction design, we provide two object interaction modes for users to choose from. Users can interact with real objects or virtual models. The two mode both contain the core interactions described above, and also are equipped other interactions in their respective modes that fit their own modes.

Summarizing the above opinions, our system provides an eye gaze method that allows users to get information about items on demand. It also provides 2 mainstream scene interaction modes to respond to the needs of different situations. In this section, we will discuss the construction of the eye-tracking interaction.

### 4.5.1 Core Interaction

HoloLens 2 provide a powerful method of eye-tracking technology to enable users to quickly and effortlessly engage with holograms across their view and can make the system smarter by better identifying a user's intention. For eye gaze, in order to give users confidence in what they're about to interact with and to not override the interaction, we use a faint eye cursor with some transparency to indicate the current position of the user's gaze. Furthermore, an arrow is equipped to mark the visual targets to provide confidence about what the user is about to interact with.

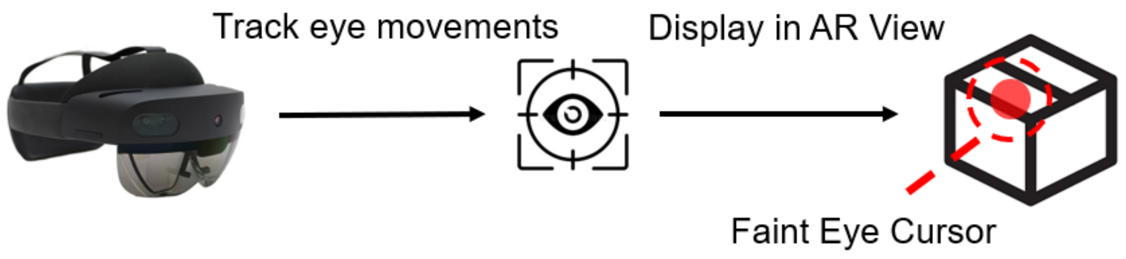


Fig. 4.4 Eye Gaze Point

Figure 4.5 shows the eye-tracking interaction design for information revelation on demand. The interaction process consists of two parts: the targets identification and the information revelation. The user first needs to identify the target that he want to interact with from plenty of objects in his view. Further information interaction is controlled through the input of eye gaze movement.

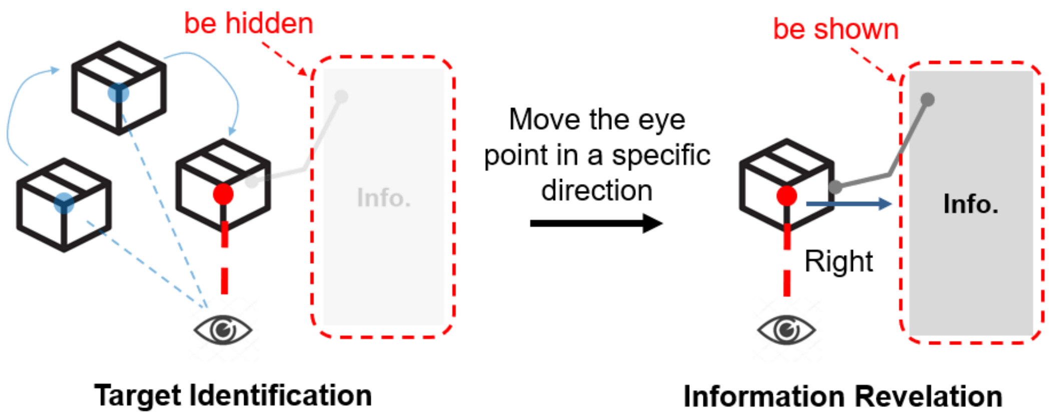
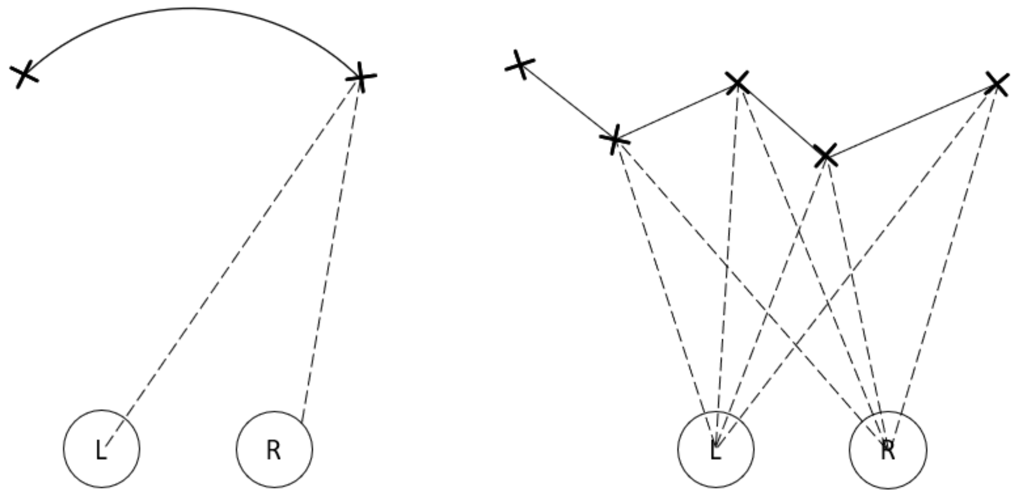


Fig. 4.5 Core Eye-Tracking Interaction Design

- **Target Identification:** As the eyes can be regarded as an indicator for the human brain's performance ("eyes are a window to the mind"), the saccade lengths and fixation duration [33] [34] can be an intuitive representation of how people process information. Fixation is the type of eye movement that keeping the visual eye-gaze on a single or same location for a duration. And saccade means a person rapidly moves his gaze point from one point to another. Figure 4.6 shows two basic elements of eye movement. Therefore, short saccade length with long fixation duration present a satisfied and deep processing of a particular gazing target, which indicating that the person is interested in this object. In our system, when the user's gaze point keeps jumping between multiple objects, the system will not set the interaction target. Instead, when the user stops moving the gaze point and keeps looking at an object for a pre-defined amount of time, it will be identified as the interactive target.



(a) Fixation (b) Saccade

Fig. 4.6 Types of Eye Movement

- **Information Revelation:** We propose using eye-tracking that allows the system to reveal the information on users' demands. Information is not visible on the left or right of the item at first. It will become fully visible when the user's visual attention explicitly falls upon the respective area. That means, additional information is only revealed if the user shows interest to this item through eye gaze. For example, moving the eye gaze point from the center to the left will only indicate some information about similar items for users to make a comparison. Figure 4.7 shows the overview of information revelation design.

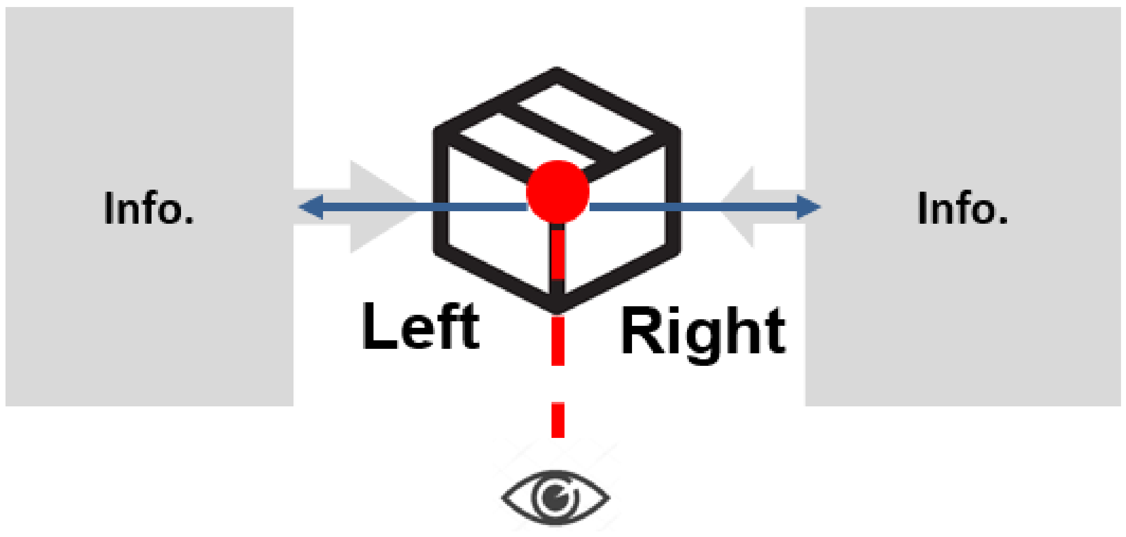


Fig. 4.7 Information Revelation

When users are shopping in a mall, the point task of the user is to select and purchase their favorite item, and the revelation of additional information about the items is only performed as a support to help the user select the item. Therefore, as illustrated in Figure 4.8, we assign three different states to the revelation of item additional information to accommodate the various behaviors of the user:

- **Idle:** When the user do not show his interest in the item, e.g., he does not drop his eye gaze point on the item, the additional information windows are in a hidden state and not visible.

- **Standby:** The additional information window is already in a ready-to-interact standby state. The item states will be changed from idle to standby after the user has been looking at the corresponding item for a duration. If the state have been changed to standby successfully, the current item will become interactable and a virtual arrow is generated above the real object to indicate this object is being interacted with at that time. Of course, since the user's focus is still on the item itself at that moment, the additional information window remains hidden.

- **Active:** When the user's eye gaze point moves from the item towards the additional information window, for example in the left or right direction of the item (or up if in real mode), the additional information window will change to active state and becomes completely visible.

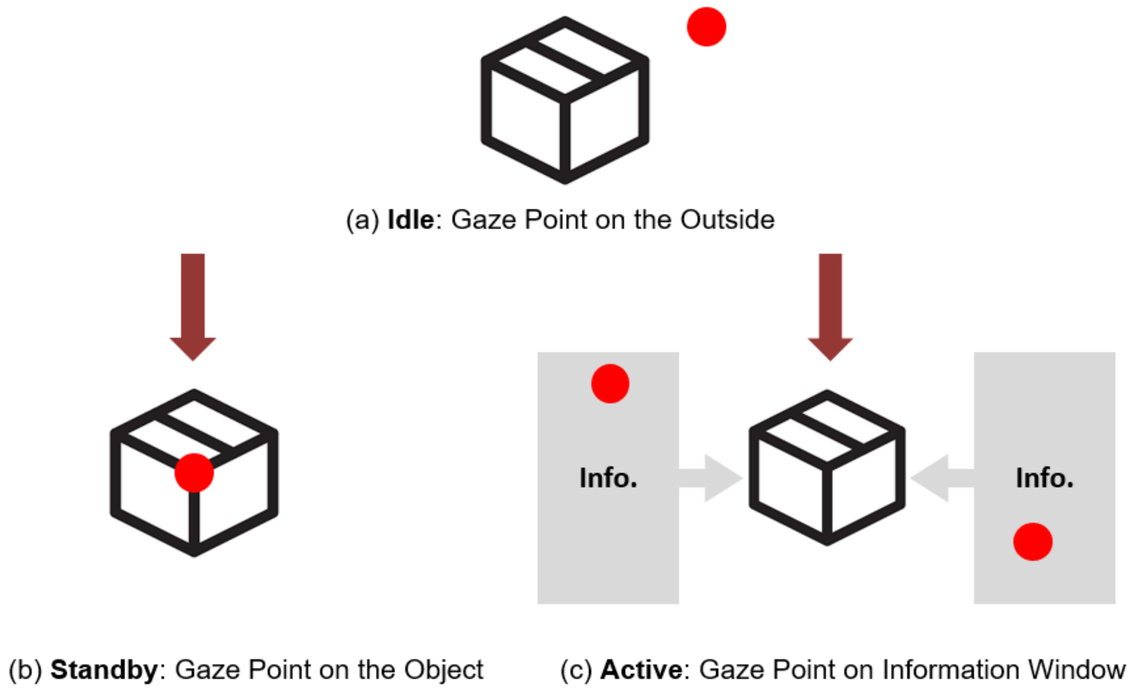

(a) **Idle**: Gaze Point on the Outside
Info.
Info.
(b) **Standby**: Gaze Point on the Object
(c) **Active**: Gaze Point on Information Window

Fig. 4.8 Gaze Point States

In addition, some items will have a lot of additional information. If we set the size of the virtual window too large, it will interfere with the user's real-world interaction, while setting it too small will make it difficult to include such a large amount of additional information. Usually, it is necessary to use sliding windows to reveal information and gestures to turn pages. Here our system uses text scrolling based on eye tracking. Through eye gaze, user can scroll texts without lifting a finger.

Figure 4.9 shows that there are two active regions at the top and bottom part of the virtual window. When the user reach the end of the displayed text and his eye point falls

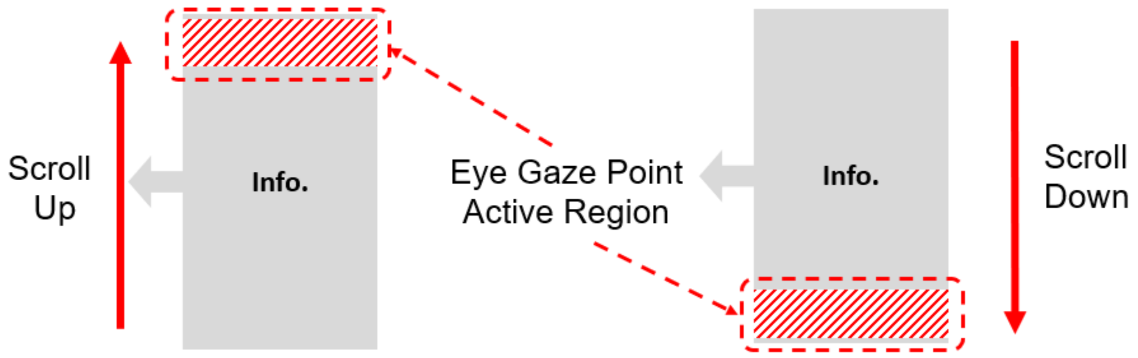


Fig. 4.9 Automatically Scroll

within the active region, the text will automatically scroll up to reveal more content. With the text scrolling automatically, the user can simply continue reading below or review above by changing where he is gazing.

As stated at the beginning of this section, there are two mode to respond to the needs of different situations: interaction with real objects or virtual models. We make a relation between real and virtual modes that the user can switch between the two modes by voice command "Change". It allows the user choose the preferred mode to interact with items. We will discuss the individual specific designs for each mode in detail in the following subsections.

### 4.5.2 Interaction With Real Objects

In interaction with real objects mode, various augmented information will be directly bound to the real object. This kind of interactive mode provides users a more natural and attractive offline shopping environment. The problem of limited space in offline shopping mall is well solved by directly attaching additional information as the augmented content to the real object.

Firstly, to get the interactive target from lots of objects in real mode, the system does not set the target as soon as the gaze point falls upon it. Interaction will start only after the user has been looking at it for a duration. If the target is interactable, a virtual arrow (Figure

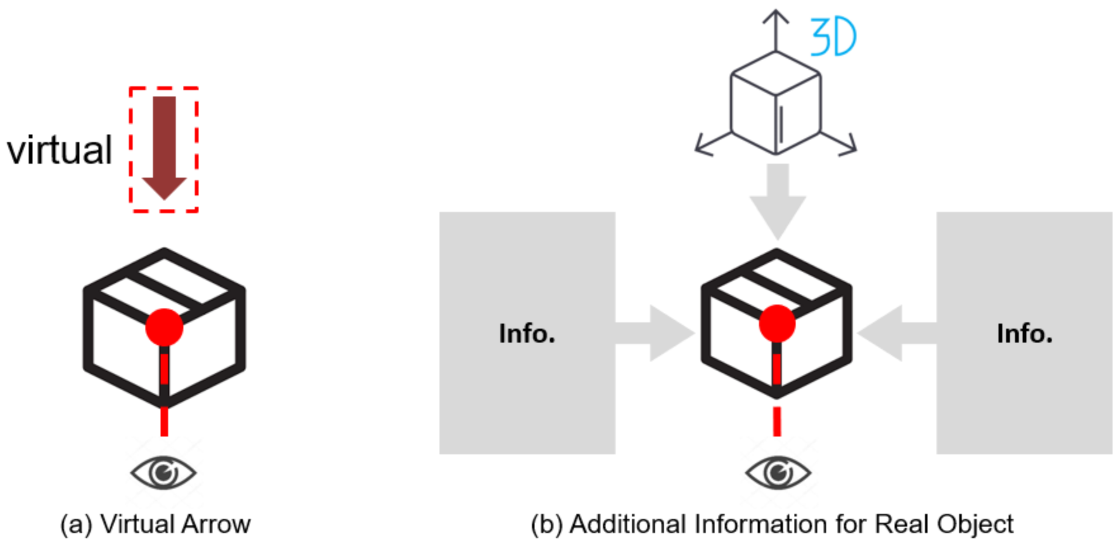


Fig. 4.10 Real Object Interaction

4.10 (a)) will be generated above the real object. The arrow indicate the object that the user is currently interacting with.

Figure 4.10 (b) describes some relationships between the real object and additional information in AR view. According to the user's intention, the user can move his eye gaze point in the following three directions (up, left or right) to reveal respective specific information:

- **Up:** When the user's gaze point is moved from the center to the top of the real object, the system will generate a virtual model equal to the real object. The virtual model can be a realistic representation of the item inside the packaging box. It can also be a transparent representation of the item so that the user can better understand the internal structure of the item.

- **Left:** The area on the left shows a list of similar items to the current interactive target for comparison and reference. Normally, offline shopping is unlike online mall where users' data can be used to personalize and recommend the most suitable items for each user. The inability to quickly compare with similar items is the biggest problem of offline shopping.

- **Right:** The user is able to move the eye gaze point to the right will indicated detailed information related to this item, such as manufacturer, brand, composition, component performance, shelf life, etc. And with the help of the auto-scroll function, the additional information of the item can be presented in this virtual window in as much detail as possible.

### 4.5.3 Interaction With Virtual Models

Different from the real mode, virtual mode is not constrained to constantly interact with real objects, which tends to be more convenient and flexible than the real mode. In this mode, even if the user cannot physically touch the item, he can also get a better understanding of the item in favor with the virtual model.

In Figure 4.11, the basic interaction of virtual mode is briefly summarized. It is divided into two parts:

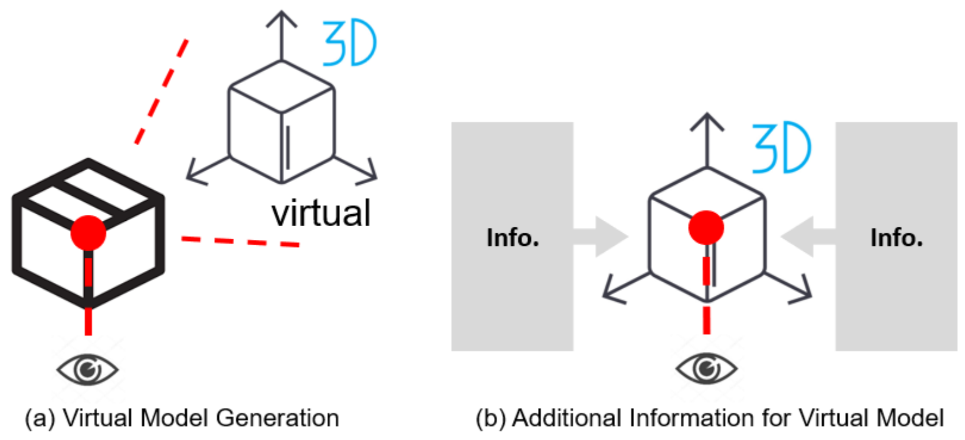

3D
virtual
(a) Virtual Model Generation
Info.
3D
Info.
(b) Additional Information for Virtual Model

Fig. 4.11 Virtual Model Interaction

- **Virtual Model Generation:** When the system determines the real object that the user intends to interact with through the same approach as above, a virtual 3d model is generated directly without binding to the target. Its state will be kept in the user's AR view. Alternatively, the user can tend to put it away and call it back up when needed. It is just similar to that the user temporarily adds an item to his shopping cart and carries it around with him during his shopping. Later, when the user finds an item that needs to be compared, he can easily and conveniently call out a virtual model of the same item for discussion.

- **Additional Information for Virtual Model:** The interaction design of additional information in the virtual mode is much the same as that of the real mode. Since the user already interacts directly with the virtual model, the virtual 3d model reference in the upper area is omitted here. Only the recommendation of similar items on the left and the detailed information of the current item on the right are left.

Since eye gaze can provide limited input, in addition to using eye tracking technology in this mode, we will incorporate certain gestural movements to make the interaction smoother. Primarily, after the user generates the virtual model in his AR view by gazing, he can continue using eye gaze instead of hand rays as the target selection input of the interaction with virtual models. As shown in Figure 4.12, just like the user sets an real object as the interactive target with gaze in real mode, a floating arrow will be generated above the current virtual model since the user has gazed at it for a duration. After that, user can use gestures to manipulate the virtual model.

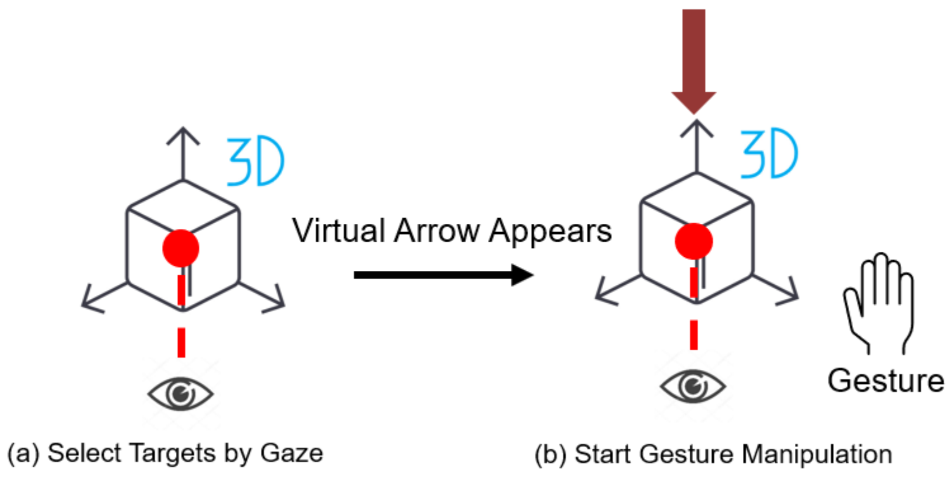
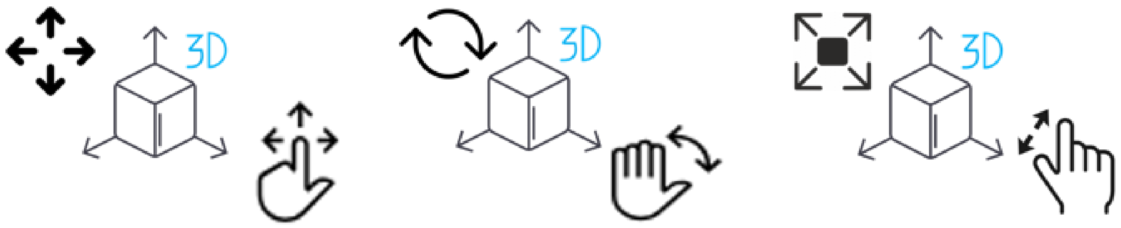


Fig. 4.12 Using Gestures to Manipulate Virtual Model

Figure 4.13 shows the gesture interactions with virtual models. The interaction with virtual models includes movement, rotation and zooming. We have assigned different gestures to each interaction. Since it is a virtual object generated by the AR system, the user does not have to worry about breaking the items in the process of manipulation to avoid causing unnecessary damage.



(a) Move (b) Rotate (c) Zoom

Fig. 4.13 Gesture Interaction with Virtual Models

## 4.6 Real-Time Communication

Usually, it is difficult to make up one's own mind, so people spend a lot of time picking out their favorite items from a wide range of items. This is one of the reasons why people often want to hear the suggestions from others (friends, family members) and go shopping with them together. In addition, due to offline shopping, people communicate in a more abstract way, often relying only on speaking or body language to express their opinions. Unlike online shopping where users can visually express ideas through pictures or videos. Therefore, we propose to use augmented reality technology to visualize the abstract content for smooth unhindered communication.

In our system, we have implemented a real-time communication module to promote communication between people who are shopping with each other. Users can visualize their abstract ideas or intentions in real time with augmented content. Users can quickly gain a focus about the person they are shopping together through AR content, with such a system being useful for exchanging ideas, analyzing items of interest, or other relevant shopping situations.

In this section, we will discuss the interaction design of real-time communication module. We propose two functions in this part to promote users' communication. The first one is ability for users to share the movements of virtual models. When a user shares a virtual model with another user, both two users can manipulate the model and the system synchronizes transformation of model movement to promote information sharing. For the second one, users can share their gaze point in real-time. By detecting the users' eye gaze points and displaying them on the surface of the virtual model or augmented in virtual environment, the user can let another user know which context he is focusing on to help the communication.

## 4.6.1 Sharing Object Movements

Sometimes, in order to find the most suitable items for ourselves more efficiently, we will tend to go to different stores of the same type separately from our peers. In this case, since the we are in different spatial environments, we can only use our own abstract words to describe the characteristics of the items in our own store. Even if we send the picture of the item to our peers, it is still not pretty convenient. Due to improving their communication experience, we support the ability of users to share the virtual models they have obtained, based on the virtual model we talked about before. Though HoloLens 2, the user can share the virtual model of the items he is interested in for the other user. Users can compare and discuss similar items in different shops in one mall. Figure 4.14 shows the relationship between users with virtual model.

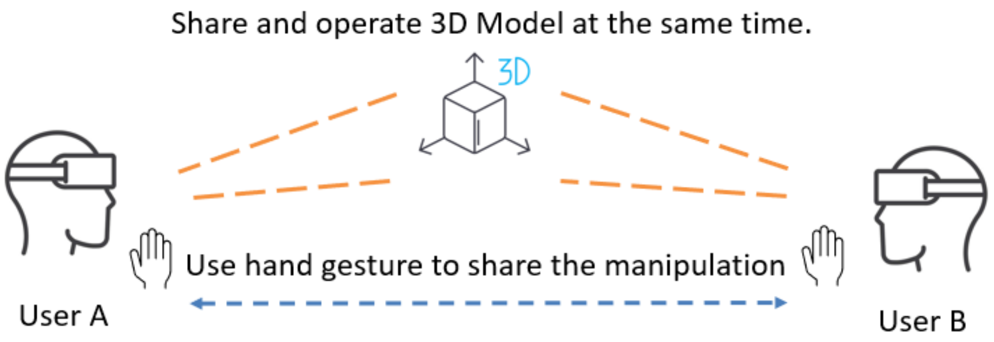

Share and operate 3D Model at the same time.
3D
Use hand gesture to share the manipulation
User A
User B

Fig. 4.14 Sharing Virtual Model Movement

Once a user obtains a virtual model of a item through the virtual model introduced in the previous sub-section, he can share this model with his peers through a combination interaction of eye gaze and hand gestures. And when the peer accepts the request, the system generates the same virtual model of the item in the same location in the peer's field of view. Although they are different users, they can share a copy of the virtual model data. Then a communication platform is built for users through the function of sharing voice. When a user manipulates the shared virtual model with his own gestures, the same manipulation will be synchronized to the virtual model in the shared user's view.

### 4.6.2 Sharing Gaze Points

People have some argument about the items with their peers. They argue for a long time before noticing that they are not discussing the same item. Therefore, when exchanging opinions on shopping, it is very important to make the discussed context of each other clear to both sides. When a user shares an virtual model with another user, they can share not only the movement of the shared model, but also their eye gaze points. By detecting the users' eye gaze points and indicating them on the surface of the virtual model, the user can let another user know which context he is focusing on. That means, our system provide a method to help the current user understand the context of each user in communication by focusing on their concerns when going offline shopping.

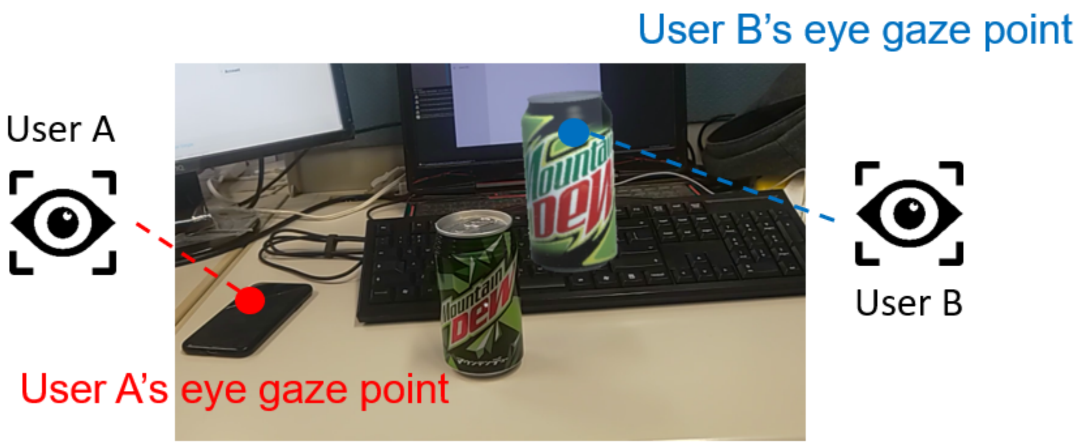


Fig. 4.15 Sharing Gaze Point

As Figure 4.15 is shown, once users agree to share their eye gaze points for other users, they can both see where the other is currently gazing at in their respective AR views. In order to distinguish whom each eye gaze point belongs to, the other user's eye cursor is attached to a different color than the current user. When both users are in different spatial locations, only the gazing situation of the shared virtual model are shared. Meanwhile, when both users are in the same spatial location, the spatial awareness system of HoloLens allows the users to share the gazing situation of all the current spatial information.

Summarizing the system design chapter, we have discussed the overall design structure of the entire system. First, we define the platform for which our system is oriented, and describe the hardware devices support of our system and the software environment for development. Next, we describe the object detection that the foundation in our system. The object detection technique is used as a bridge to bind virtual objects to real objects for our following interaction design. The main interaction of the system we divided it into 2 modules. The first one is eye-tracking interaction. The core interaction is that our system tend to reveal the information on users' demands. Based on this core interaction, we designed 2 interaction modes to adapt to the user's needs in different scenarios. The second one is to provide a real-time communication platform for users in augmented content. Users can share the movements of virtual objects or the location of the content that their are interested in. So far, we have completed the system design part, and in the next chapter we will discuss the implementation of our system.

# Chapter 5

# System Implementation

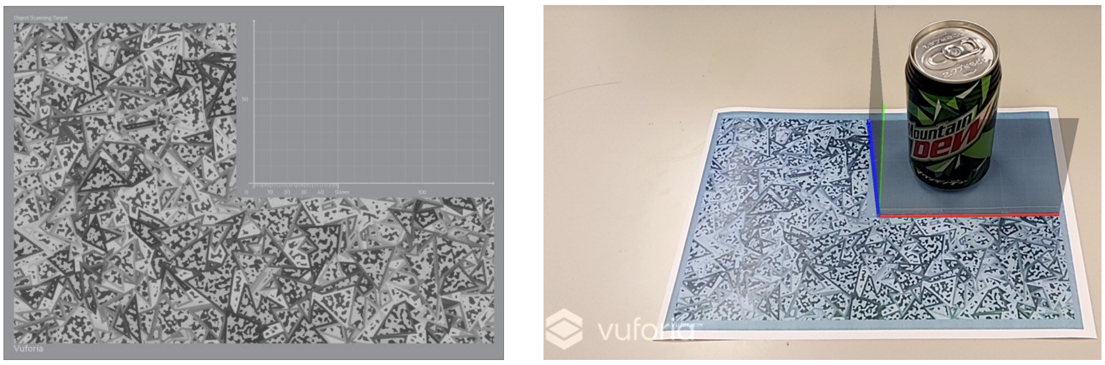
In this chapter, we will introduce the implementation of our eye-based shopping system. At first, the object detection technology that is used as the root of interaction in this system will be introduced. Then, the implementation method and effects of various interaction designs in this system will be explained.

## 5.1 Real Object Detection

Firstly, we need to implement the root function of our system. Since our system needs to recognize individual items in order to provide corresponding virtual augmented content, the first step in system development is to implement object detection. Object recognition allows our system to detect and continuously track the intricate 3D objects within the field of view. The main design of our system is to unlock new in-app content when an object is recognized.

### 5.1.1 Object Targets Generation

We use the application called "Object Scanner" [32] with object target scanning image (Figure 5.1a) to easily scan the detailed object and generate the object data.



(a) Object Scanning Target Image (b) Coordinate System in Target Image

Fig. 5.1 Scan Preparation

- **Object Scanner [32]:** The Vuforia Object Scanner, an Android application, is used to scan the physical object. It is easy to generate a marker from an image or 3D object. Through combining this application with the target image, we can easily get an Object Data file including the source data that can define the physical Object Target.
- **Object Scanning Target Image [35]:** When we scanning an object to produce object data, the paper (as shown in Figure 5.1a) as the target image is needed. It is able to maintain the dimensional parameters between the 3D object and the real object, while it also defines the position and orientation of the object target with respect to its local coordinate space origin. The feature area of the target of the target image allows the user to accurately identify the pose of the physical target within the grid area. In addition, it also defines the rejection region of the scan space (which will not be incorporated into the object data). Figure 5.1b shows the coordinate system established through this paper. The center of the target image (the intersection of the red, green and blue lines) is the local origin of the scan, which is also the local (0,0,0) origin of the obtained 3d model data after the scan.

Once we have installed the application with the downloaded apk and printed the target image paper, we can start scanning the object to get the feature point data.

Firstly, we need to place the object that is about to be scanned on the grid area of the target image. Secondly, press "Record" button on the right to start scanning and capture the

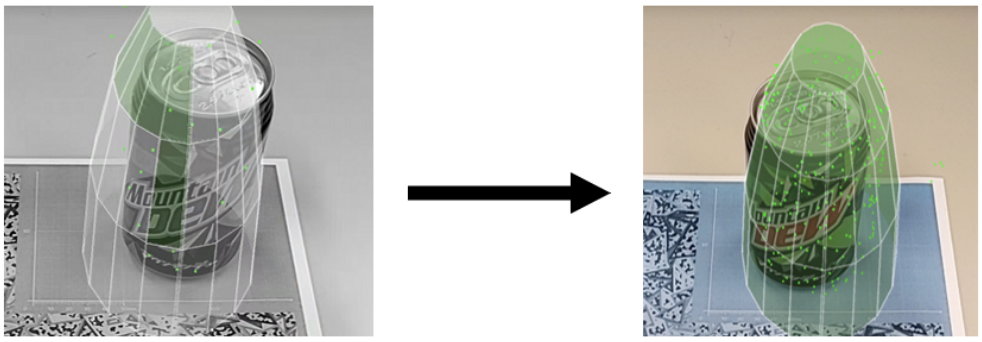
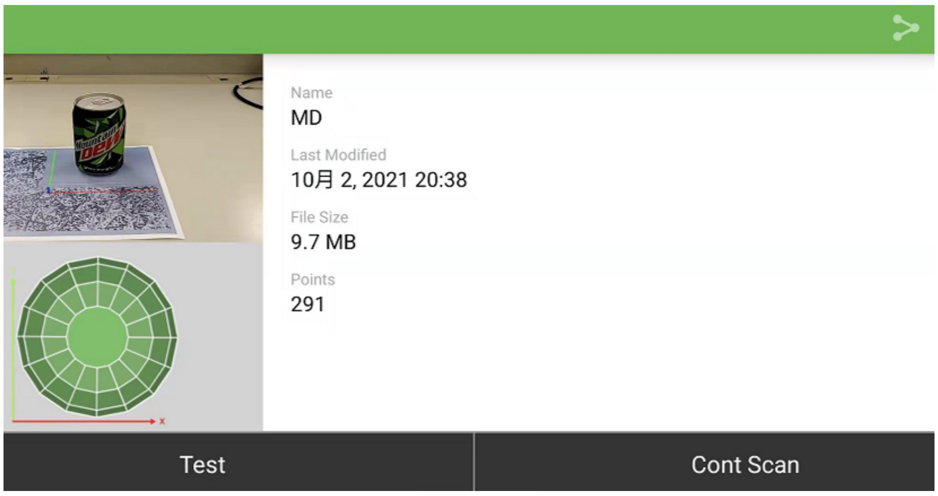


Fig. 5.2 Capture Surface Region

significant vantage points with the phone camera. When all the characteristic points of a surface area is completely captured (The left Figure in Figure 5.2), its corresponding facet will turn green (The right Figure in Figure 5.2). If we need to scan points about objects in other directions, it is better for us to move the scanner instead of the scanned target while recording a scan. Changing the position of the target frequently will produce recognition impurities. Rotate the phone around the object so that the more points shown in the upper left corner, the more efficient the recognition will be afterwards. Once we have captured enough of the surface area required to identify the object, we can press the "Stop" button to stop scanning.



Name
MD
Last Modified
10月 2, 2021 20:38
File Size
9.7 MB
Points
291

Fig. 5.3 Summary Screen

After the scan, we need to test whether the object is actually recognized. After finishing scanning an object, we will get a summary screen window showing the detailed results of the scan. We can press the "Test" button (as shown in the lower left corner of Figure 5.3) to test whether this recognition data is accurate enough for vuforia to recognize the scanned object. The augmentation shows that our system has been able to recognize the scanned object target successfully. In order to ensure that the object can be recognized in all environments, it is desirable to test it in different backgrounds. Figure 5.4 represents a good result of the object scan in the test. Also, we can improve or add to the scan through selecting "Cont Scan" button in the summary screen, which tend to reload the Object Data file and refine the scan.

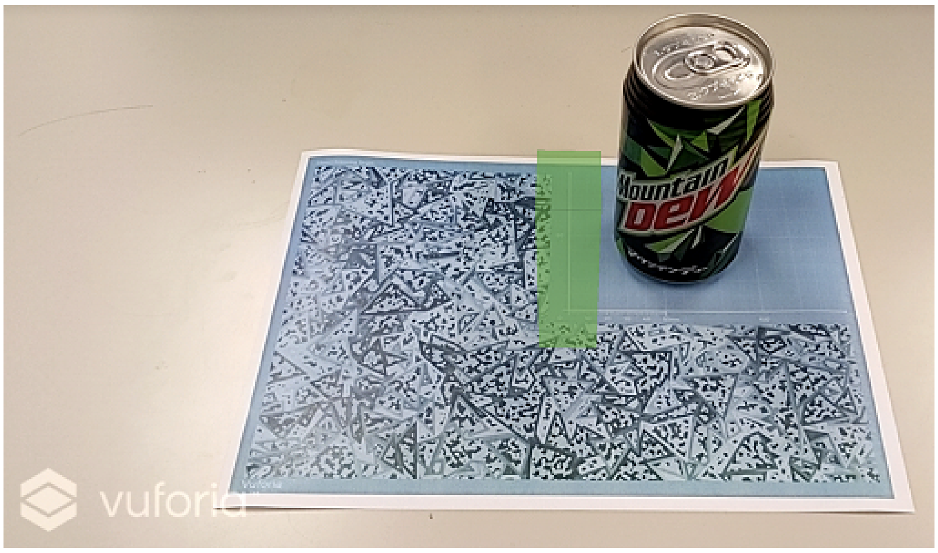
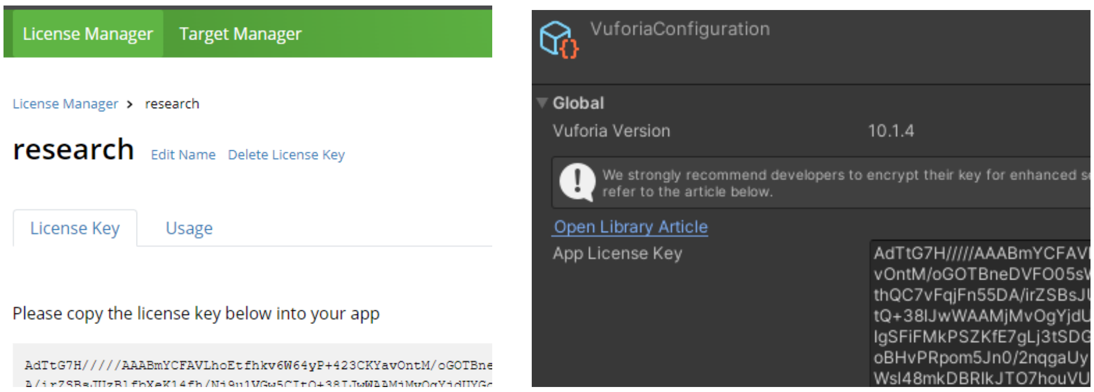


Fig. 5.4 Test the Results of the Object Scan

### 5.1.2 Object Target Management

In this sub-section, we intend to illustrate how to manage and upload object data and eventually add it to our unity project. In order to use the Vuforia Engine to either develop or deploy an application, a license key is required. It uniquely identifies the apps and enable them to access features of the Vuforia Engine. We can get the free license from the License Manager page and license keys are created in the Vuforia Developer Portal's License Manager (Figure 5.5a). Then we just need to add this license key to our own unity project to get features of the Vuforia Engine. Figure 5.5b presents the result of adding the license to Unity project.


License Manager Target Manager

License Manager > research

research Edit Name Delete License Key

License Key Usage

Please copy the license key below into your app

AdTtG7H/////AAABmYCFAVLhoEtfhkv6W64yP+423CKYavOntM/oGOTBne

VuforiaConfiguration

Global

Vuforia Version 10.1.4

We strongly recommend developers to encrypt their key for enhanced s
refer to the article below.

Open Library Article

App License Key AdTtG7H/////AAABmYCFAV
vOntM/oGOTBneDVFO05s
thQC7vFqjFn55DA/irZSBsJ
tQ+38lJwWAAMjMvOgYjdU
IgSFiFMkPSZKfE7gLj3tSDG
oBHvPRpom5Jn0/2nqgaUy
Wsl48mkDBRlkJTO7houVU

(a) Vuforia License Manager (b) Vuforia License in Unity

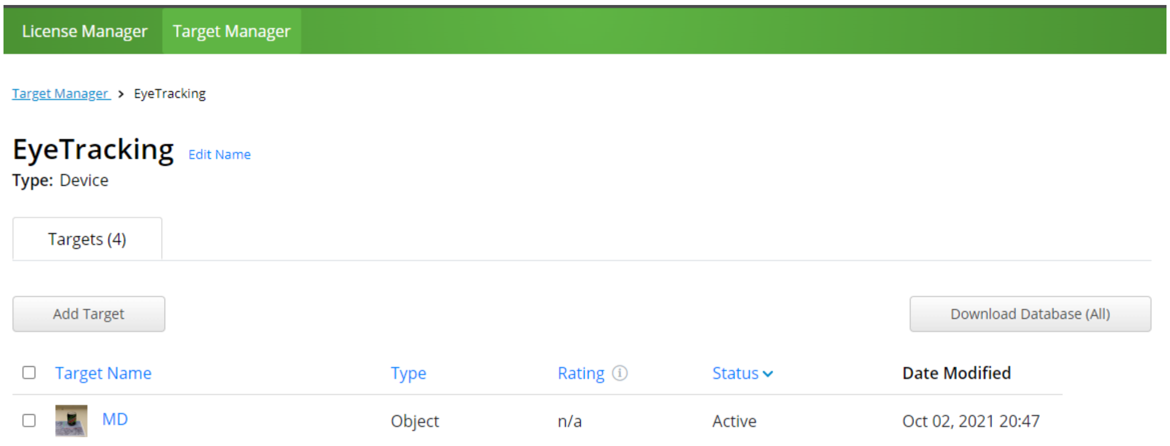
Fig. 5.5 Vuforia License

So far, we can start managing the scan data of the target objects. Once we have scanned a target object through the method that described in the previous subsection, we can upload its scan data to vuforia's cloud database for use in our unity project.

1. **Share Data File:** Press the "Share" icon (the upper right corner of Figure 5.3) on the scan summary screen. Select the storage method of the displayed list to store the scan data.

2. **Upload Data File:** Open the Target Manager in the Vuforia Engine Developer Portal page. Add our own new target to the selected database. Since we want to track physics objects, the type chosen here is "3d objects". Then, after uploading the object file saved in the previous step, we have complete uploading the scanned object data. The satisfied result is shown in Figure 5.6.

3. **Use in Unity:** Finally, download the scanned target database for Unity implementation. Since we have already built our hololens development environment via MRTK tools, we only need to add the script component named Vuforia Behaviour from vuforia SDK to the MainCamera. Next, import the downloaded database with an Object Target as a .unitypackage. After that, we create an ObjectTarget (3D Scanned) GameObject and



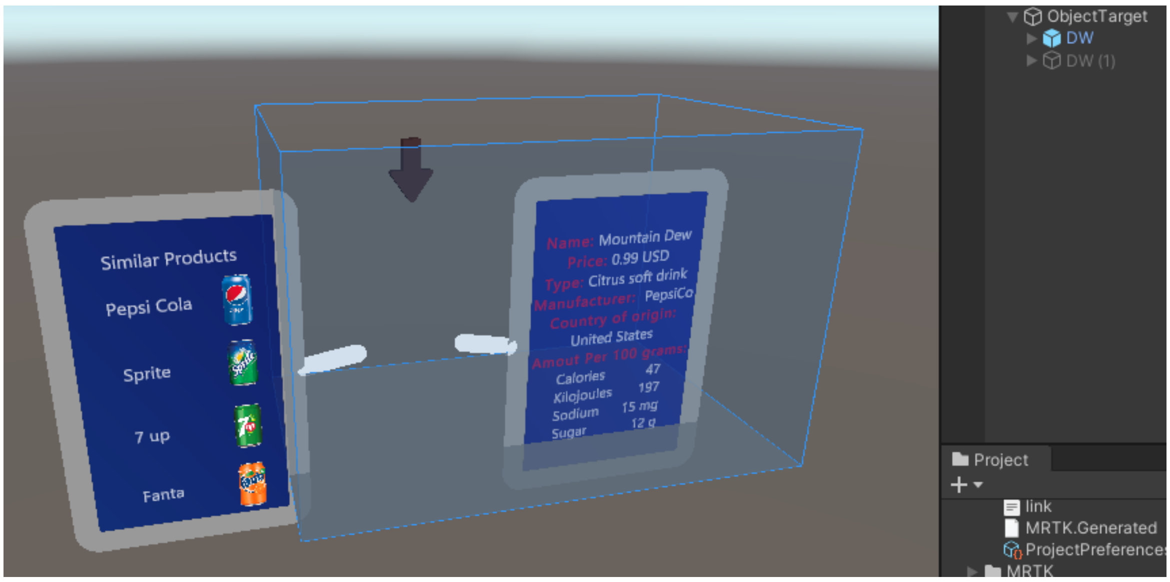
EyeTracking

Type: Device

| | Target Name | Type | Rating | Status | Date Modified |
|---|---|---|---|---|---|
| ☐ | MD | Object | n/a | Active | Oct 02, 2021 20:47 |

Fig. 5.6 Target Database

configure its Database and Object Target that we want to associate with the Object Target instance in Object Target Behavior component. Finally, add the content as a child to the above Object Target GameObject. The result that implemented in unity is shown in Figure 5.7.



Name: Mountain Dew
Price: 0.99 USD
Type: Citrus soft drink
Manufacturer: PepsiCo
Country of origin: United States
Amout Per 100 grams:

| | |
|---|---|
| Calories | 47 |
| Kilojoules | 197 |
| Sodium | 15 mg |
| Sugar | 12 g |

Fig. 5.7 Unity Implementation

In this way, once the camera of our device recognizes the features of the scanned object, it will reveal our augmented content added as a child at the location we have defined.

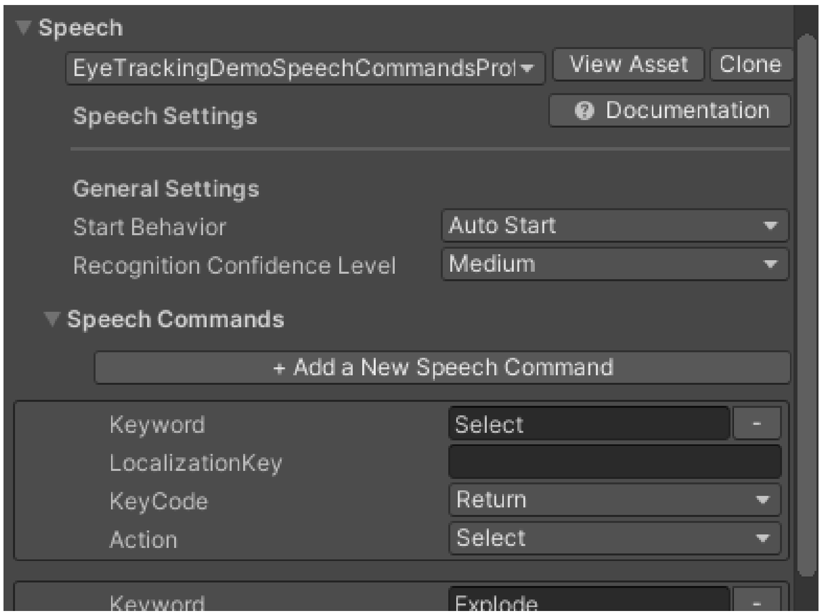
Afterwards, we will use some scripts to develop interactive functions based on the function as detecting real objects and generating virtual objects.

## 5.2 Eye-Tracking Implementation

The main idea in our system is that it proposes a gaze-assisted system instead of the conventional hand-based interaction in offline shopping. In this section, we will discuss the detailed implementation of eye-tracking interaction in our project.

### 5.2.1 Mode Relation

As we described above, our whole eye interaction is divided into 2 modes: interaction with real objects mode and interaction with virtual model mode. Considering that using eye tracking as the input for this function would put a lot of burden on the user's eye, we need to choose other inputs to switch between the 2 modes in order to switch more smoothly. Here, voice is a excellent input method since speech commands is very invasive and convenient for shopping interaction.



Start Behavior: Auto Start
Recognition Confidence Level: Medium

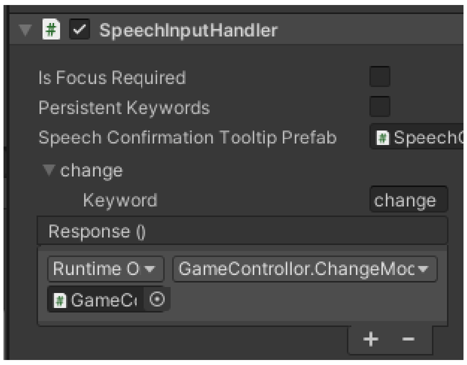
Keyword: Select
LocalizationKey:
KeyCode: Return
Action: Select

Keyword: Explode

Fig. 5.8 Speech Commands Profile in MRTK

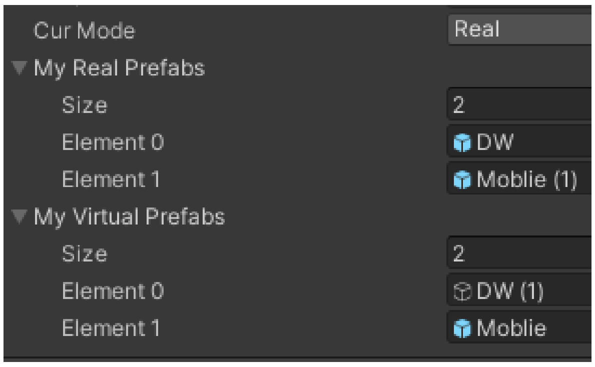
As for the speech commands, the system use the microphone of the HoloLens and the function provided by MRTK API to realize the mode switching for the eye-tracking interaction. Figure 5.8 shows where we can configure the keywords for the system to recognize. Simply add the voice command words we want to use as input into the "keyword" to register voice keywords. Also, we can specify a key code that will produce the same speech event when pressed, which can be used for local testing in unity.

The Speech Input Handler script provided by MRTK can be added to a GameObject to handle speech commands using UnityEvents. To use the speech commands to switch the mode, we add the Speech Input Handler script to a GameObject named SceneController, which integrates the global properties of the control scene. The Figure 5.9a shows the detail of this GameObject. When we open the "SpeechInputHandler" script, it automatically shows the list of the defined keywords from the Speech Commands Profile. Also, we create a script on the SceneController component that implemented a function of controlling whether the objects is activated or not. As Figure 5.9b shows, the *My Real Prefabs* list is the GameObject used in real mode, and virtual mode use the GameObject in the *My Virtual Prefabs* list. When the system recognize the user's voice command, the bound event will be triggered. The system will deactivate the objects in the currently active list and then activate another list of objects in the inactive list. Through controlling whether the object is activated in this way, it is possible to switch between the two modes.



```
SpeechInputHandler
Is Focus Required
Persistent Keywords
Speech Confirmation Tooltip Prefab    SpeechC
change
    Keyword                           change
Response ()
Runtime O    GameControllor.ChangeMoc
GameC
```

(a) Speech Input Handler script



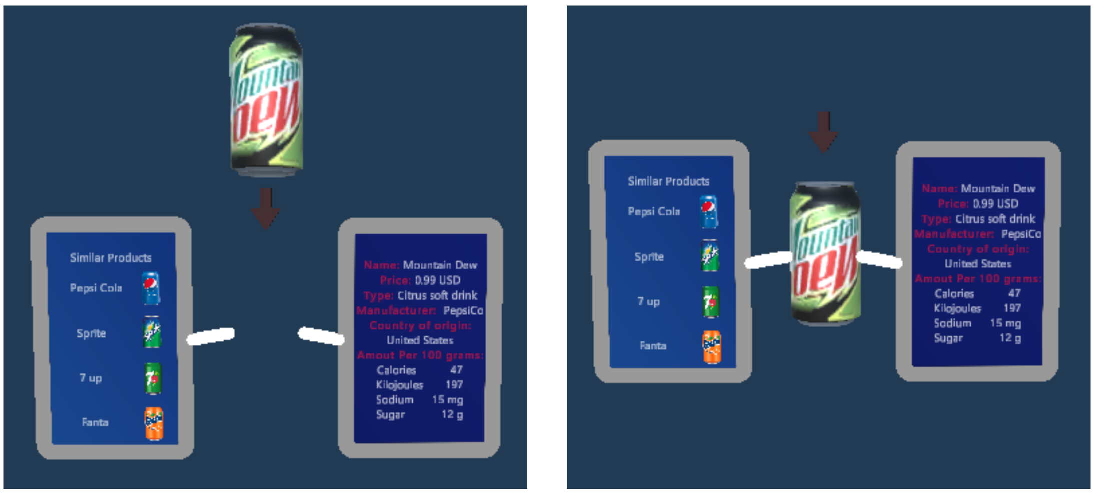
```
Cur Mode              Real
My Real Prefabs
    Size              2
    Element 0         DW
    Element 1         Moblie (1)
My Virtual Prefabs
    Size              2
    Element 0         DW (1)
    Element 1         Moblie
```

(b) Change Mode Script

Fig. 5.9 Speech Command Implementation

### 5.2.2 Information Revelation on Demand

As shown in Figure 5.10, two information windows containing different information are created on the left and right of the item. The left window displays a list of items that are similar to the current items, while the detailed information of the current item are displayed in the right window. In addition, an virtual 3d model of the current item is also created on its top in real mode. Since the virtual mode is already allowing the user to interact with the virtual model, this interaction can be removed. We have proposed to hide the additional information of the shopping items until they are explicitly gazed by the user. It can avoid confusing the user by always presenting all additional AR information of every item and do not create too much burden for them.

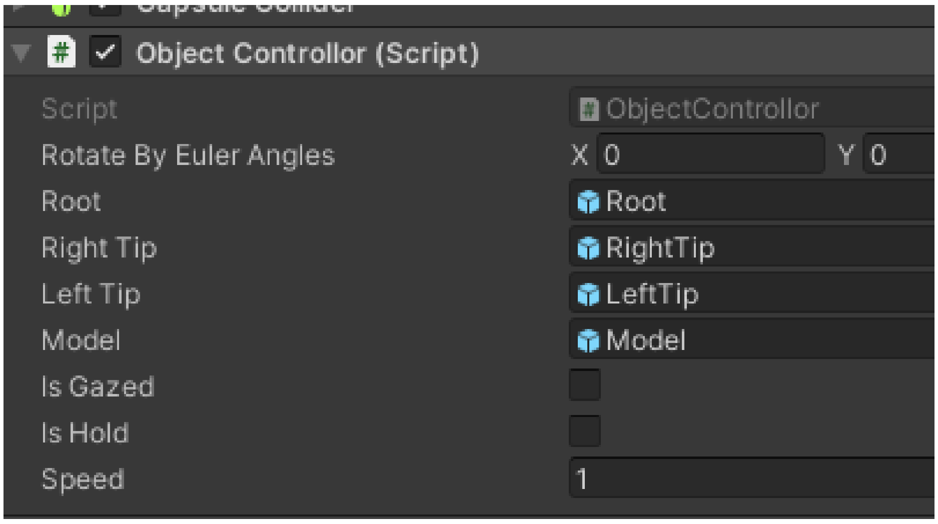


(a) Real Mode (b) Virtual Mode

Fig. 5.10 Dynamic Information Overlays

As we assumed in the System Design Chapter, in order not to overwhelm the user, we have assumed 3 states for the additional information window: Idle, Standby and Active. For the purpose of controlling the state of additional information window, we have decided to use a state controlling variable named *isGazed* (as shown in Figure 5.11) to manage the transformation of the state.

In order to implement eye-tracking interaction, we also need "EyeTrackingTarget" script (shown in Figure 5.12) that provided by MRTK. It includes some eye-tracking function and


Object Controllor (Script)

Script ObjectControllor
Rotate By Euler Angles X 0 Y 0
Root Root
Right Tip RightTip
Left Tip LeftTip
Model Model
Is Gazed
Is Hold
Speed 1

Fig. 5.11 The State Controlling Variable

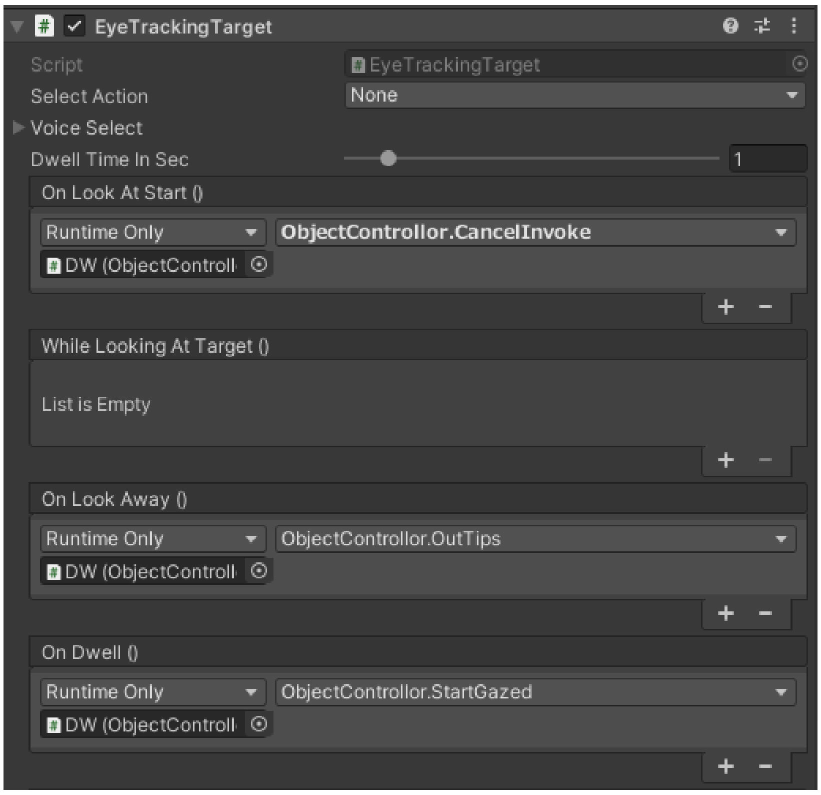
can be added to a GameObject to enables objects to become interactable targets for eye tracking.

Through "EyeTrackingTarget" script, we can easily develop the eye-tracking interactions. For the basic content, we use the three properties of this script: OnDwell(), OnLookAway() and OnLookAtStart().

- **OnDwell():** Since we designed the interaction to be triggered after the user has been looking at the target for a duration, we chose to use OnDwell() instead of OnLookAtStart(). In OnDwell() function, event is triggered when the target has been looked at for a given predefined duration (dwellTimeInSec). When the system detects the user's eye data falling on the object for a duration, it will trigger the OnDwell() function, which will trigger ObjectControllor.StartGazed() function we bound in. At that point, the current object's isGazed variable will turn to true, indicating that the object has entered the standby state and being ready for interaction.

- **OnLookAway():** Similar to OnDwell() properties, OnLookAway() will trigger the bound event when the user is looking away from the target. Therefore, ObjectControllor.OutTips() function that we have written will be triggered once the user's eyes are removed from the current object. But at this point isGazed does not immediately turn to false. In the ObjectControllor.OnLookAway() function, the code that turns isGazed to false is set to a delayed call. Only after this delayed function is called successfully, the object's state will switch back to Idle exactly.

- **OnLookAtStart():** When the user starts to look at the target, the event will be triggered in OnLookAtStart(). We can use this feature to cancel the delayed function call in OnLookAway() above. That is, when the user accidentally moves the eye out of the object's interactable range, as long as the eye is immediately returned to the current object's valid range within a certain amount of time, then isGazed remains true and the object will not be switched to Idle. Such a fault-tolerant design for the user's behavior can lead to a smoother interaction experience.


EyeTrackingTarget

Script: EyeTrackingTarget
Select Action: None
Voice Select
Dwell Time In Sec: 1

On Look At Start ()
Runtime Only — ObjectControllor.CancelInvoke
DW (ObjectControll

While Looking At Target ()
List is Empty

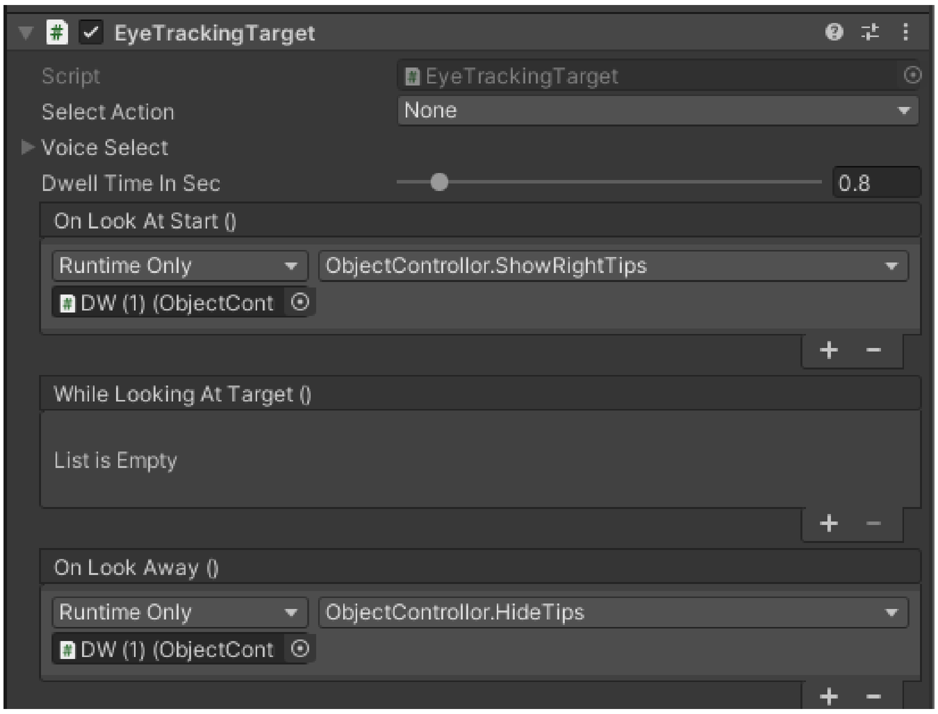
On Look Away ()
Runtime Only — ObjectControllor.OutTips
DW (ObjectControll

On Dwell ()
Runtime Only — ObjectControllor.StartGazed
DW (ObjectControll

Fig. 5.12 Eye Tracking Interaction Components

After completing the implementation of the Idle and Standby states of the interactive content, the next step is the event that switch state to Active. Obviously, the event that activates the additional information window is separate from the event that recognize the interactive object, so we also added the "EyeTrackingTarget" script for the information window.

The configure setting of this script is indicated in Figure 5.13. The logic here is relatively simple, so only two properties of "EyeTrackingTarget" script are used: OnLookAtStart() and OnLookAway().


EyeTrackingTarget

Script: EyeTrackingTarget

Select Action: None

Voice Select

Dwell Time In Sec: 0.8

On Look At Start ()

Runtime Only | ObjectControllor.ShowRightTips

DW (1) (ObjectCont

While Looking At Target ()

List is Empty

On Look Away ()

Runtime Only | ObjectControllor.HideTips

DW (1) (ObjectCont

Fig. 5.13 Information Window Components

- **OnLookAtStart():** We tend to achieve the interaction that the additional information window reveals directly when the user's eye point moves in corresponding direction. So here we do not need to require the user to continue to look at a duration before starting to interact. Instant of using OnDwell() function, we decided to use the OnLookAtStart() function, which is triggered when the user starts to look at the target. In this function, as long as the user moves the eye gaze point to the preset trigger range of the information window, the system will directly activate the information window for the user to read the information of the item. And at the same time, since the user's eye gaze point moves from the response range of the object body, it also triggers the delay function for the object state switch. We need to run the appropriate code to cancel this state switch function to ensure that the isGazed variable is still true.

- **OnLookAway():** Similarly, the system inactivate the information window immediately after the user's eye point leaves the information window's point of view. But at that time, this does not sunddenly change the value of isGazed to false and switch the state to Idle. The same delay function is used to allow the user to return to the object for a short duration to continue the eye-tracking interaction. This also prevents the user from having to start over again to determine the target of interaction because of an inadvertent fixation that moves the eye gaze point elsewhere.

One more thing, the above implementation is also applicable to virtual model that only used in the real mode. This can be achieved rapidly by replacing the revealed content with the virtual model.

In addition, since the Active state can only be entered from the Standby state, the system only reacts to the corresponding function after the isGazed variable of the current object is changed to true. Therefore, the function written in either of the above two properties will initially determine whether the isGazed variable is true or not. if isGazed is false, no action will be performed.

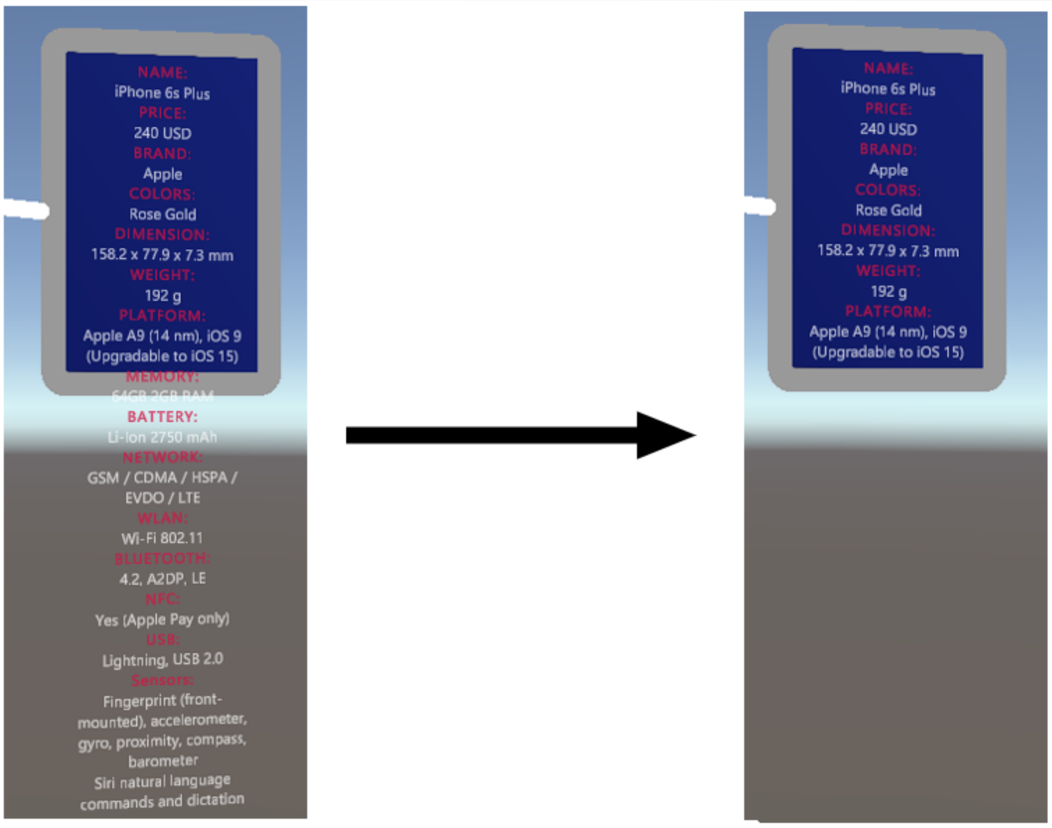
### 5.2.3 Automatically Scroll

There are times when the item's additional information is too much to be revealed completely within the limited range of the virtual information window, so we designed the auto-scroll interaction. When the user is reading information on an information revelation window and he reaches the bottom part of the text in the display window box, the text of additional information will be automatically scrolled up to show more content. Also, if the user wants to review information that has been slid away, he simply gazes the top of the displayed text, and the text will automatically scroll down to show the previous content.

The principle of auto-scroll is that only revealing part of the text in the area by masking, and then moving the position of text up or down by changing the Transform components of the text GameObject to achieve the effect of auto-scroll.

**1. Text Mask**

Firstly, we have to implement the function of mask. What we mainly need in this function is the two components called "Canvas" and "Rect Mask 2D". Since the text is the 2d object, it is better to use “Canvas” to restrict its position. "Canvas" belongs to Unity’s UGUI, which provides powerful visual editing. It can powerfully improves the development efficiency of GUI. Then a "Rect Mask 2D" is a masking control that can restrict the display of a child element to the rectangular extent of the current object. RectMask2D is usually used to display a small part of a larger area, which is perfectly met the requirement in our system. “Rect Mask 2D” does not depend on any Image component, and its cropping area is the rect size of its RectTransform.



NAME:
iPhone 6s Plus
PRICE:
240 USD
BRAND:
Apple
COLORS:
Rose Gold
DIMENSION:
158.2 x 77.9 x 7.3 mm
WEIGHT:
192 g
PLATFORM:
Apple A9 (14 nm), iOS 9
(Upgradable to iOS 15)
MEMORY:
64GB 2GB RAM
BATTERY:
Li-Ion 2750 mAh
NETWORK:
GSM / CDMA / HSPA / EVDO / LTE
WLAN:
Wi-Fi 802.11
BLUETOOTH:
4.2, A2DP, LE
NFC:
Yes (Apple Pay only)
USB:
Lightning, USB 2.0
Sensors:
Fingerprint (front-mounted), accelerometer, gyro, proximity, compass, barometer
Siri natural language commands and dictation

NAME:
iPhone 6s Plus
PRICE:
240 USD
BRAND:
Apple
COLORS:
Rose Gold
DIMENSION:
158.2 x 77.9 x 7.3 mm
WEIGHT:
192 g
PLATFORM:
Apple A9 (14 nm), iOS 9
(Upgradable to iOS 15)

(a) Without Mask (b) With Mask

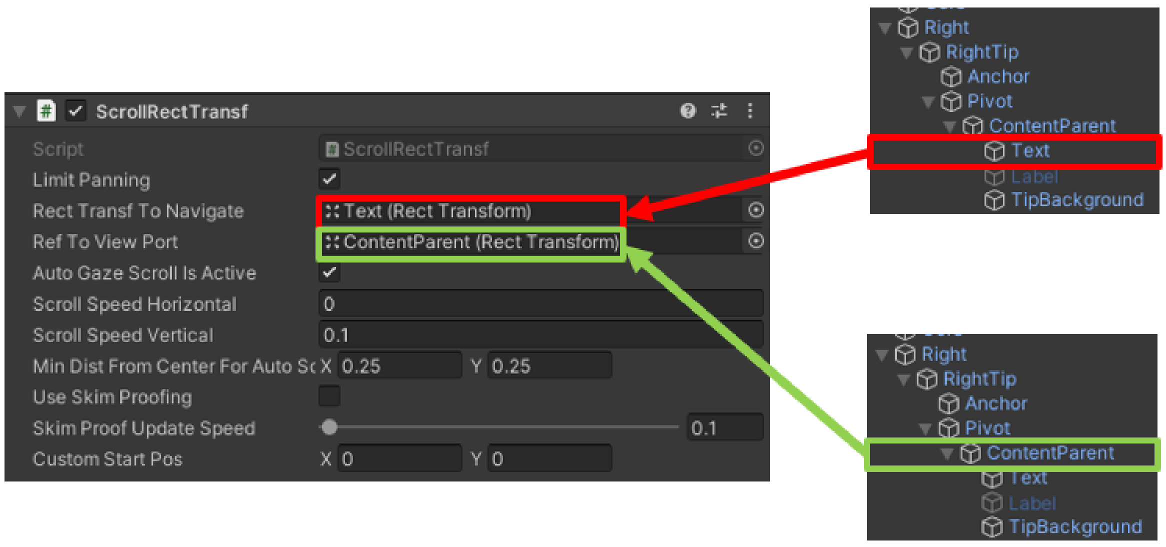
Fig. 5.14 Text Mask

Figure 5.14 shows the result that using the mask to hide the content. Figure 5.14a shows the GameObject state of the complete text without the mask. When a mask is used, the extra text is hidden and the remaining content within the set range is revealed (as shown in Figure 5.14b, only the text in the information window is displayed.)

**2. Scroll with Eye Gaze**

After completing the implementation of the mask, the next step is to scroll the text according to the position of the user's eye gaze point.

Here we use "ScrollRectTransf" script component to implement this function. After adding the script component to the root GameObject, we need to bind 2 other GameObjects to this component. As shown in Figure 5.15, "RectTransfToNavigate" is the reference to the RectTransform of the target text to scroll in as the reading navigation for the user. The other is "RefToViewport", which is the reference to the scrollable content's parent RectTransform to determine the proper bounds and offset.



| ScrollRectTransf | |
|---|---|
| Script | ScrollRectTransf |
| Limit Panning | ✓ |
| Rect Transf To Navigate | Text (Rect Transform) |
| Ref To View Port | ContentParent (Rect Transform) |
| Auto Gaze Scroll Is Active | ✓ |
| Scroll Speed Horizontal | 0 |
| Scroll Speed Vertical | 0.1 |
| Min Dist From Center For Auto Sc | X 0.25 Y 0.25 |
| Use Skim Proofing | |
| Skim Proof Update Speed | 0.1 |
| Custom Start Pos | X 0 Y 0 |

Fig. 5.15 Scroll Component

Next we need to enable "AutoGazeScrollIsActive" to support that the navigated text will be automatically scrolled up or down depending on the relative active region where the user is gazing. "MinDistFromCenterForAutoScroll" then specifies the minimum distance from the active region to the center of the strike box to the scroll. We can continuously test and debug to get an optimal activation position to respond to the user's scroll request.

The content shown in Figure 5.16 is the final result of the automatically scrolling interaction in our system. As it describes, the huge amount of text content is not all displayed in the information window. Instead, the information window is used as a mask to reveal only a section of the text content within its range. Through pre-debugging, the eye gaze hit boxes are set up to suit the user's smooth interaction. When the system tracks that the user's eye point falls within the range shown by the red box (which is not actually assisted in the system), it performs the corresponding text scrolling up or down.

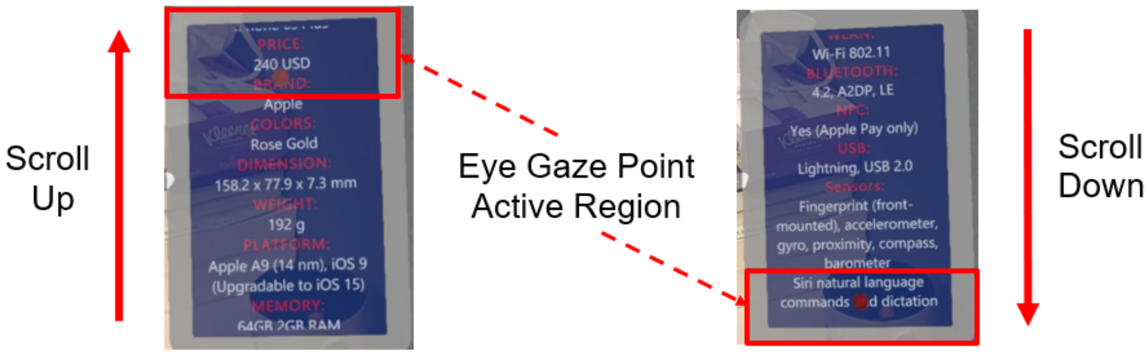


Fig. 5.16 Automatically Scroll Implementation

## 5.3 PUN-based Multi-User Environment

The second part of our system is that it aims to promote the real-time communication between different users. Interaction in our system is not only between humans and objects, but also allows human-to-human interaction. In this section, we will describe the detailed implementation of multi-user real-time communication in our project. In this part, we mainly combine Photon as a multi-device synchronization technology and azure as a perceptual space technology to achieve our desired goal.

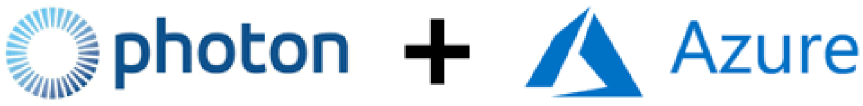


Fig. 5.17 Photon and Azure

### 5.3.1 Multi-Device Connection

In order to achieve real-time interaction, the first thing we need is to build an environment that connects different users together. Here we decided to use Photon Unity Networking (PUN) to build our server and make network connections.

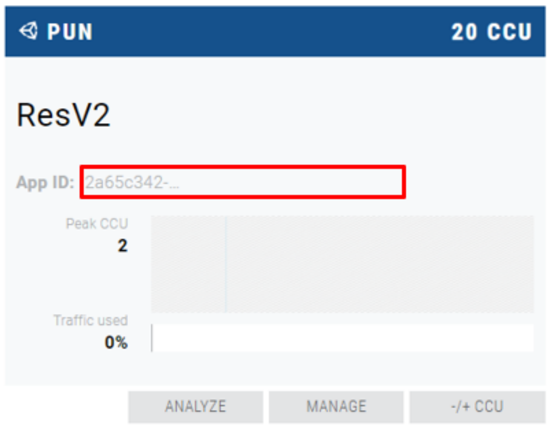
Photon Unity Networking (PUN) is a Unity package for multi-user application. It allows flexible matching to let plenty of players enter rooms where objects can be synchronized over the network [31]. We just let a user connect to Photon's loadbalanced server as a server, and then other users as the client can simply and quickly connect by joining the room. In this case, different users can exchange GameObject data in the scene to get a remote interactive experience.

We divide the multi-device connection implementation into two parts:

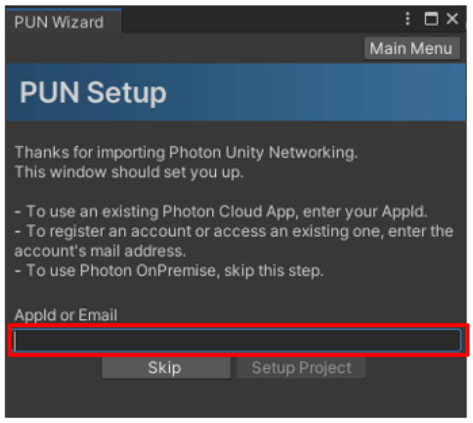
**1. Setting up Photon Unity Networking**

As the preparation for connection, we should enabling some additional capabilities for Unity project. In "Configuring the Unity project", the following additional capabilities need to be enabled: InternetClientServer capability and PrivateNetworkClientServer capability. The other preparation we should do is that downloading and importing the three packages in the following order: AzurespatialAnchors SDK, GettingStarted.unitypackage, AzureSpatialAnchors.unitypackage and MultiUserCapabilities.unitypackage.

Then we need to download and import "PUN 2 - FREE" from the "Asset Store" window into our project. After that, a Photon account is needed to manage the PUN application for our Unity project. We navigate to the Photon dashboard website to create the account and sign in in order to get a new PUN app. Once the creation process is completed, the created PUN Cloud App will be shown on the dashboard, on which we can copy the App ID (as shown in the red box in Figure 5.18a) it to the PUN Setup menu (as shown in Figure 5.18b) of our Unity project.



(a) PUN App ID



Thanks for importing Photon Unity Networking.
This window should set you up.

- To use an existing Photon Cloud App, enter your AppId.
- To register an account or access an existing one, enter the account's mail address.
- To use Photon OnPremise, skip this step.

(b) PUN Setup Menu

Fig. 5.18 PUN Application Connecting

Once we have successfully created a PUN app and connected it to our own Unity project. Next, we tend to allow connections with other users so that multiple users can see each other.

**2. Connecting Multiple Users**

In this section, we are going to connect the AR devices to the multi-users environment that we have created in the previous section. "NetworkLobby" GameObject is used in our game scene to manage the user's connections. The lobby for our project exists on the Master Server to list rooms for our scene. It does not enable players to communicate with one another unless the users are in the same room.

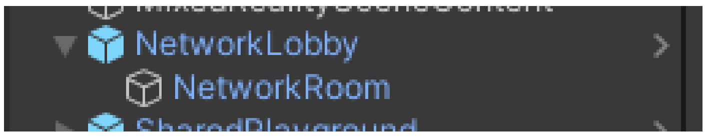
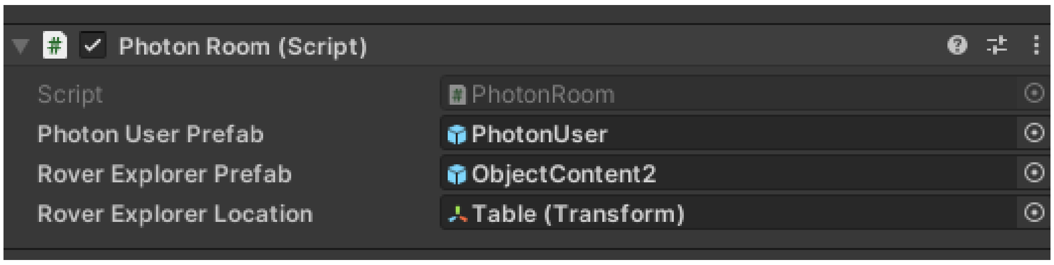


Fig. 5.19 NetworkLobby GameObject

To manage the users' profiles, there is a GameObject named "NetworkRoom" as a child of "NetworkLobby" GameObject. "NetworkRoom" includes "PhotonRoom" script component that can create a GameObject using the prefab to represent the users at the start of the game, and shares it when the users enters the room. As shown in Figure 5.20, we assign a user prefab (a sphere 3D object) to the “Photon User Prefab” field. In addition, each time a corresponding prefab is generated for a user entering the room, a different color is added to it to distinguish each user.


Photon Room (Script)

| | |
|---|---|
| Script | PhotonRoom |
| Photon User Prefab | PhotonUser |
| Rover Explorer Prefab | ObjectContent2 |
| Rover Explorer Location | Table (Transform) |

Fig. 5.20 PhotonRoom GameObject

So far, we have successfully configured the project to allow multiple users to connect to the same experience and see each other’s movements. Figure 5.21 gives a satisfactory result of the multi-device connection.

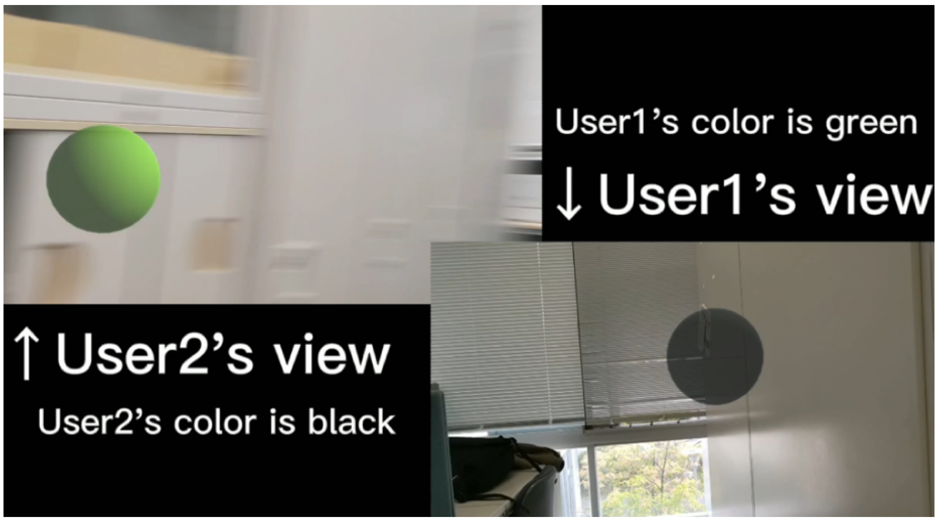


Fig. 5.21 Multi-User Connection

### 5.3.2 Object Target Management

At last, we need to implement the sharing of multi-player manipulation for virtual 3D objects on the multi-user environment that established in the previous section. In addition to "NetworkLobby" GameObject described above, another GameObject named "SharedPlayground" is also needed to manage the shared objects.

We add another GameObject named "TableAnchor" to our scene as a child of the "SharedPlayground" object. It will manage the various spatial anchor. With the "TableAnchor", object movements sharing across devices can be implemented. The spatial anchor as the shared information will be processed and updated in real time by continuously analyzing the anchor of the AR camera. We just need to drag the "Main Camera" component into the Camera field of the "TableAnchor"'s "AR Session Origin" component. After that, it collects the behaviors of each local user and updates the shared objects data through the behaviors of all users in the current room server. Figure 5.22 shows all of the components that the "TableAnchor" GameObject has.

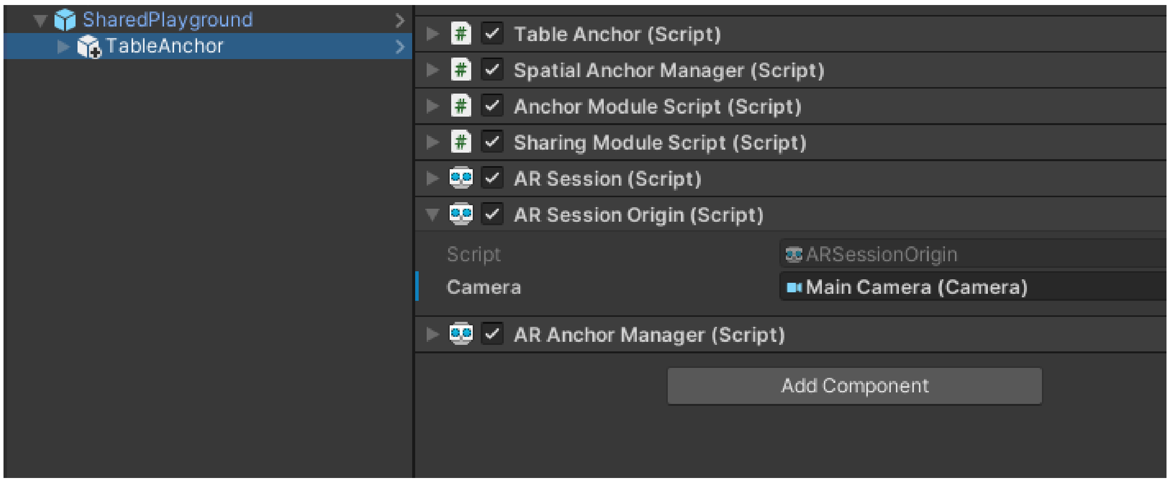


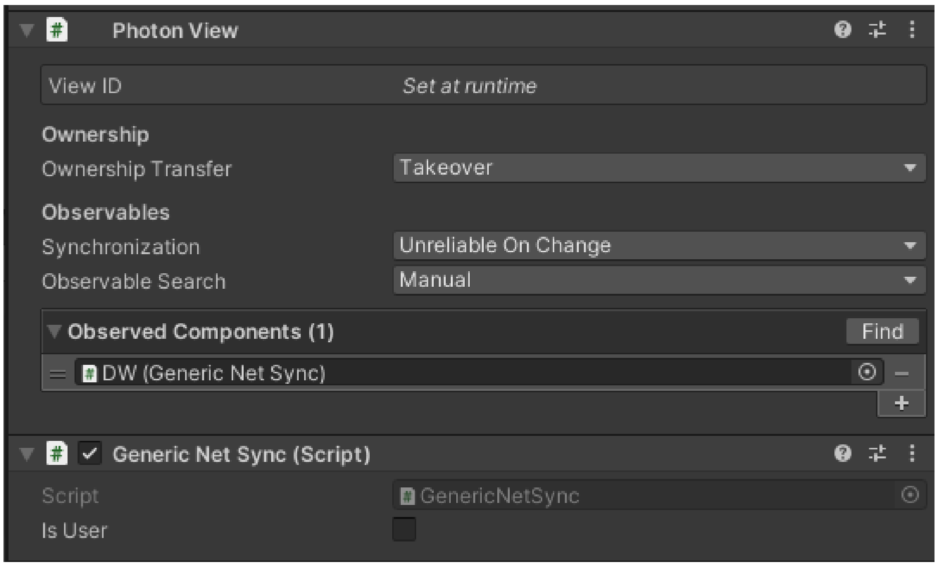
Table Anchor (Script)
Spatial Anchor Manager (Script)
Anchor Module Script (Script)
Sharing Module Script (Script)
AR Session (Script)
AR Session Origin (Script)
Script ARSessionOrigin
Camera Main Camera (Camera)
AR Anchor Manager (Script)

Fig. 5.22 SharedPlayGround GameObject

After we have prepared the interaction scene, the next step is to configure the PUN to instantiate the object. We will configure the project to use the GameObjects that created before and define where it will be instantiated. To start with, we need to package all our already created AR interactive GameObjects into an empty GameObject and make it into a prefab called ObjectContent. As shown in Figure 5.20, to the Rover Explorer Prefab field, we assign this prefab from the materials storage folder. The GameObject added to the Rover Explorer Prefab will be shared across multiple devices.

**1. Shared Object Movements**

To synchronize the target content, a PhotonView component needs to be added for shared objects. PhotonView is a component that connects the various object instances on each device together. In PUN, each GameObject in Unity can be controlled by only one client. When a GameObject is instantiated, then it is only owned by the client that generated it. If we want to give control permissions of a GameObject to other clients, we need to configure the setting in PhotonView (shown in Figure 5.23). Here, we use "Takeover" so that any client in the current room get permission to take over the GameObject from the current owner. Also, PhotonView defines which or how components are to be observed. That means, the scripts that need to be continuously updated during synchronization should be added into the Observed Components field of the PhotonView. Since in our requirements it is sufficient to make changes only to the Transform component of the shared objects. The GenericNetSync script component implements the control of the Transform component. Moreover, we can set the IsUser variable to indicate whether the currently bound object is a user or not. If it is a user, the data from the MainCamera of each device will be used to update the current object's transform.


Photon View

| | |
|---|---|
| View ID | Set at runtime |
| **Ownership** | |
| Ownership Transfer | Takeover |
| **Observables** | |
| Synchronization | Unreliable On Change |
| Observable Search | Manual |

**Observed Components (1)** Find

DW (Generic Net Sync)

Generic Net Sync (Script)

| | |
|---|---|
| Script | GenericNetSync |
| Is User | ☐ |

Fig. 5.23 Synchronization Basics

Next, to determine the relative position where the shared GameObjects will be instantiated, we need assign the Transform component of target position from the Hierarchy window to the Rover Explorer Location field. In this way, whenever a user connects to the current room, the object in Rover Explorer Prefab will be instantiated on the Transform bound in Rover Explorer Location as his initial position.

So far, we have successfully configured our project to synchronize object movements so users can see the objects move when other users move them.

**2. Shared Gaze Points**

Subsequently, we can use the methods in the previous sub-section to implement a new application. Each user can only control the eye-gaze points that belong to him or her, updating them constantly in the shared space so that the users in this room know what others are focusing on.

Therefore, in order to implement the eye gaze points sharing feature, we need to create a GazeManager GameObject (GameObject details are shown in Figure 5.24) to manage each eye-gaze point. As before, in order for the content it contains to be synchronized in real time, a script component for synchronization such as PhotonView needs to be added to GazeManager.

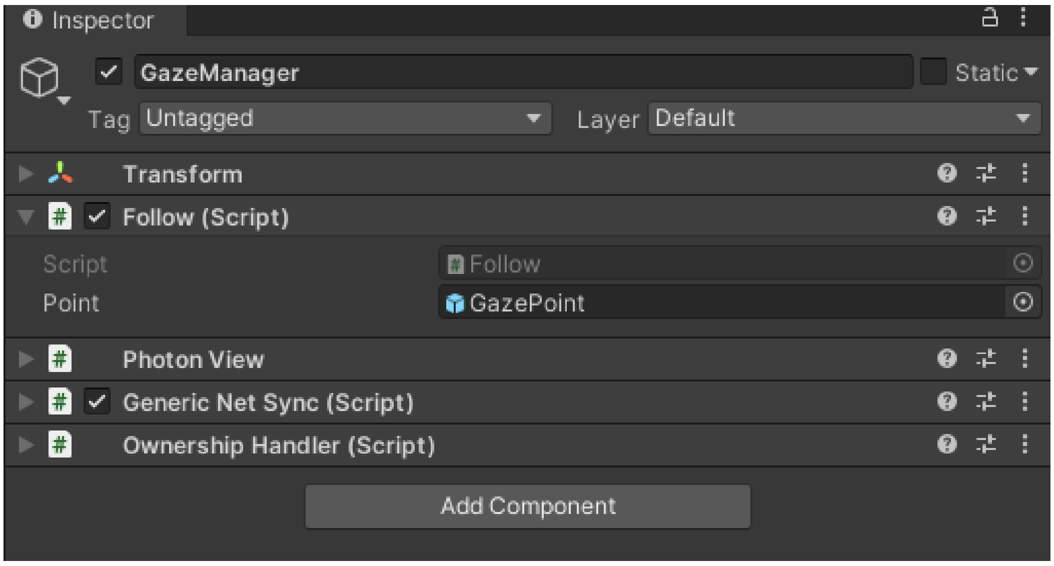
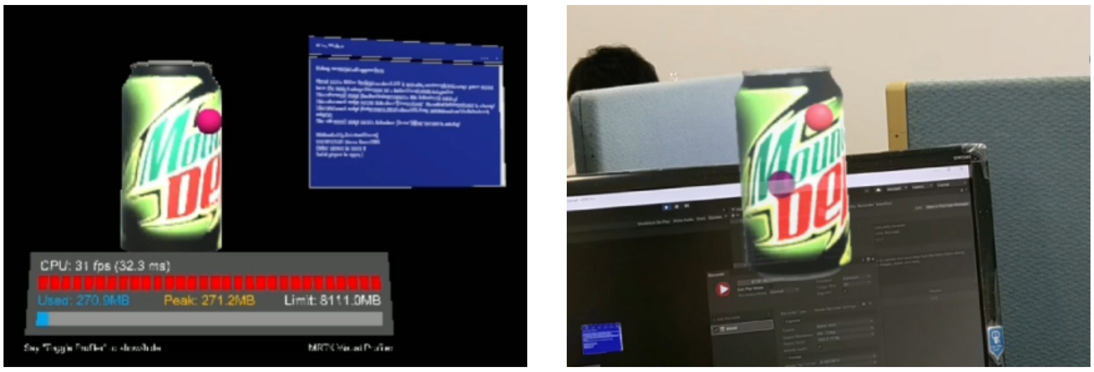

Inspector
GazeManager
Static
Tag Untagged
Layer Default
Transform
Follow (Script)
Script Follow
Point GazePoint
Photon View
Generic Net Sync (Script)
Ownership Handler (Script)
Add Component

Fig. 5.24 GazeManager GameObject

After enabling GazeManager to synchronize across multiple devices, we need to write the appropriate script to control the individual user's eye gaze points. First we add pre-created eye point prefabs to the Points field of the script (as shown in Figure 5.24). This prefab will be used to create an exclusive eye gaze point for each user when they enter the current room. Since the system has already add a random color material to the user, we can read this material and add the same color material to the corresponding eye gaze point. In this way, other users will be able to efficiently match the user's prefab with the eye gaze point prefab that in the same color.

Figure 5.25 shows the actual result of this feature. Each eye gaze point has its own color, and when moving a eye gaze point of one device, the eye gaze points on the other devices will move together with it.



(a) PC View (b) HoloLens2 View

Fig. 5.25 Gaze Points Sharing

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In this thesis, we present an AR shopping system based on the interaction design in eye-tracking for shopping assistance. Through augmented reality and eye-tracking technology, we improve the offline shopping experience and increases the efficiency of offline shopping in terms of both user interaction with the physical environment and communication with other users.

First, the addition information of the items in each entity store in the shopping mall can be displayed in virtual 3D and superimposed on the real environment. The system is designed to improve the efficiency of offline shopping and reduce the intrusion of the AR system into the user's shopping flow by not basing the input stream on the user's hand rays but on eye movements. Users do not need to use their hands to trigger the interaction with virtual content, and most of the interaction can be completed by the movement of the eye gaze point independently. Compared to other existing hand ray based AR shopping systems, our system greatly improves the offline shopping fluency.

In addition, we designed and implemented a communication platform that allows multiple people to share information. Our system can powerfully promote communication through sharing items or suggestions using eye-tracking technology. Though HoloLens, the user can share the 3D model with others and synchronize various manipulation of 3d models.

By detecting users' eye gaze points and indicating them on the 3D model, users can let another user know which context he is focusing on.

## 6.2 Future Work

In the future, we plan to improve our AR gaze-assisted shopping system. The interaction design in our system is not rich enough now, so we will design and implement more eye tracking based interactions between users and items. We will also optimize the approach that users communicate with each other in our multi-user part to enrich the sharing experience of offline shopping.

In addition, we will continue to keep doing evaluations. Since our system requires the user to gaze at the target object for a duration before triggering a subsequent interaction, we need to evaluate the optimal duration among multiple groups of volunteers. We plan to implement an interface for users to adjust the gazing duration by themselves. Furthermore, we will also do the evaluations to compare the user experience of our system with similar other augmented reality shopping systems.

# References

[1] Rick van van Krevelen and Ronald. Poelman. A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality*, Vol.9, No.2: pages 1–20, Jan. 2010.

[2] C Cavanaugh. Virtual reality and augmented reality will change brand experiences, 2017.

[3] The Nielsen Company. *The Future of Grocery: E-commerce, Digital Technology and Changing Shopping Preferences Around The World.* Apr. 2015.

[4] Hamraz Javaheri, Maryam Mirzaei, and Paul Lukowicz. How far can wearable augmented reality influence customer shopping behavior. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '20, page 464–469. Association for Computing Machinery, 2020.

[5] Zulqarnain Rashid, Enric Peig, and Rafael Pous. Bringing online shopping experience to offline retail through augmented reality and rfid. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 45–51, 2015.

[6] Kasun Jayananda, D.H.D. Seneviratne, Pradeep Abeygunawardhana, L.N Dodampege, and A.M.B Lakshani. Augmented reality based smart supermarket system with indoor navigation using beacon technology (easy shopping android mobile app). In *2018 IEEE International Conference on Information and Automation for Sustainability (ICIAfS)*, pages 1–6, 2018.

[7] Junho Ahn, James Williamson, Mike Gartrell, Richard Han, Qin Lv, and Shivakant Mishra. Supporting healthy grocery shopping via mobile augmented reality. *ACM Trans. Multimedia Comput. Commun. Appl.*, pages 1–24, Oct. 2015.

[8] Lakmal Meegahapola, Lakmal and Indika Perera. Enhanced in-store shopping experience through smart phone based mixed reality application. In *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 1–8, 2017.

[9] Monika Płużyczka. The first hundred years: a history of eye tracking as a research method. *Applied Linguistics Papers*, pages 101–116, Dec. 2018.

[10] Saara Khalid, Jason Deska, and Kurt Hugenberg. The eyes are the windows to the mind: Direct eye gaze triggers the ascription of others' minds. *Personality and Social Psychology Bulletin*, Vol.42, No.12: pages 1666–1677, Aug. 2016.

[11] Todd E. Hogue, Charlotte Wesson, and Derek E. Perkins. Eye-tracking and assessing sexual interest in forensic contexts. 2016.

[12] H Hartridge and L C Thomson. Methods of investigating eye movements. *The British journal of ophthalmology*, Vol.32, No.9: pages 581–591, Sept. 1948.

[13] Benjamin W Tatler, Nicholas J Wade, Hoi Kwan, John M Findlay, and Boris M Velichkovsky. Yarbus, eye movements, and vision. *i-Perception*, Vol.1, No.1: pages 7–27, 2010.

[14] Norman H. Mackworth and Edward Llewellyn Thomas. Head-mounted eye-marker camera. *J. Opt. Soc. Am.*, Vol.52, No.6: pages 713–716, Jun. 1962.

[15] B. Shackel. Note on mobile eye viewpoint recording. *J. Opt. Soc. Am.*, Vol.50, No.8: pages 763–768, Aug. 1960.

[16] EyeSee. Eye tracking through history, 2014.

[17] Vijay Rajanna and Tracy Anne Hammond. A gaze-assisted multimodal approach to rich and accessible human-computer interaction. *ArXiv*, pages 1–4, 2018.

[18] Maria Laura Mele and Stefano Federici. A psychotechnological review on eye-tracking systems: Towards user experience. *Disability and rehabilitation. Assistive technology*, Vol.7: pages 261–81, Nov. 2011.

[19] Alberto De Santis and Daniela Iacoviello. Robust real time eye tracking for computer interface for disabled people. *Computer Methods and Programs in Biomedicine*, Vol.96, No.1: pages 1–11, 2009.

[20] Katarzyna Harezlak and Pawel Kasprowski. Application of eye tracking in medicine: A survey, research issues and challenges. *Computerized Medical Imaging and Graphics*, Vol.65: pages 176–190, 2018. Advances in Biomedical Image Processing.

[21] Nicholas Wade. Pioneers of eye movement research. *i-Perception*, Vol.1: pages 33–68, Nov. 2010.

[22] Christopher Was, Frank Sansosti, and Bradley Morris. *Eye-Tracking Technology Applications in Educational Research.* Sept. 2016.

[23] Hong Xu, Xiaoyan Xuan, Li Zhang, Wenxin Zhang, Min Zhu, and Xiaoke Zhao. New approach to intelligence screening for children with global development delay using eye-tracking technology: A pilot study. *Frontiers in Neurology*, Vol.12: pages 1–6, Nov. 2021.

[24] Rene Santos, Jorge Oliveira, Jessica Rocha, and Janaina Giraldi. Eye tracking in neuromarketing: A research agenda for marketing studies. *International Journal of Psychological Studies*, Vol.7: pages 32–42, Feb. 2015.

[25] Rik Pieters. A review of eye-tracking research in marketing. *Review of Marketing Research*, Vol.4: pages 123–147, Jan. 2008.

[26] H. Zamani, A. Abas, and M.K. MAmin. Eye tracking application on emotion analysis for marketing strategy. Vol.8: pages 87–91, Jan. 2016.

[27] Dong-Gun Lee, Kyeong-Ho Lee, and Soo-Young Lee. Implicit shopping intention recognition with eye tracking data and response time. In *Proceedings of the 3rd International Conference on Human-Agent Interaction*, HAI '15, pages 295–298, New York, NY, USA, 2015. Association for Computing Machinery.

[28] Phelippe Souza-Herod and Abdelwahab Hamam. Augmented reality shopping framework. In *SoutheastCon 2021*, pages 1–6, 2021.

[29] Mixed reality toolkit. https://github.com/microsoft/MixedRealityToolkit-Unity, Accessed 2021.

[30] Vuforia sdk. https://developer.vuforia.com/, Accessed Sept. 2021.

[31] Photon unity networking. https://dashboard.photonengine.com/, Accessed Oct, 2021.

[32] Vuforia object scanner. https://developer.vuforia.com/downloads/tool, Accessed Sept. 2021.

[33] Kenneth Holmqvist and Richard Andersson. *Eye-tracking: A comprehensive guide to methods, paradigms and measures*. Nov. 2017.

[34] Andrew Duchowski. *Eye Tracking Methodology: Theory and Practice*. Jan. 2007.

[35] Vuforia object scanning target image. https://developer.vuforia.com/downloads/tool, Accessed Sept. 2021.