

# Using Multiple Kinds of Markers and Hand Gesture to Enhance User Interactions in Marker-Based AR System



**Xitong Sun**

**44181573-5**

Master of Engineering

Supervisor: Prof. Jiro Tanaka

*Graduate School of Information, Production and Systems Waseda  
University*

July 2020



# Abstract

Nowadays, AR (augmented reality) technology is increasingly popular among people.

Many AR applications are used in people's daily life.

Markers are visual cues can be used in AR system. Some markers can be used to trigger some AR content, and we call them AR marker. Some markers can just provide some information. We call a system that uses AR marker to display AR content and other markers to provide control information as marker-based AR system.

Hand gesture is a popular way for human interactions. It is widely used in AR system.

Users can use his hand gesture to interact with virtual content.

In current marker-based AR system, different markers are usually independent of each other and users lack intuitive interactions with virtual content or the system.

In this research, we propose a marker-based AR system which combines multiple kinds and numbers of markers and hand gesture together for some interactions. And we propose virtual markers which can be combined with real markers for visual programming and some other interactions. Based on the framework of our system, we make some applications.

**Keywords:** Augmented Reality, Multiple Markers, Hand Gesture, Virtual Marker, Visual Programming

# Acknowledgement

I would like to express my sincere gratitude to my supervisor, Professor Tanaka. In the past two years, I have learnt a lot from Professor. Thanks to his guidance and advice, I have made some progress. In the process of master thesis, I encountered a lot of difficulties, and even stopped for a while. Professor gave patient guidance and provided many useful comments, which helped me a lot for my work. I am really honored to have the opportunity to spend these two years as a student of Professor Tanaka.

I also want to express my gratitude to all members of IPLAB. I learned a lot from them. In the process of master thesis, I get some advice and comments from them, which are really helpful.

Special thanks to my parents. They always supported me when I was in trouble, which really encourage me a lot.

Thanks to WASEDA University and IPS for providing this excellent working conditions for these two years.



# Table of Contents

List of figures .....	VII
Chapter 1 .....	1
Introduction .....	1
1.1 Introduction .....	1
1.2 Organization of the thesis.....	1
Chapter2 .....	2
Background .....	2
2.1 Augmented Reality .....	2
2.2 Visual Markers .....	2
2.3 Hand Gesture Interactions.....	2
Chapter3 .....	4
Research Goal and Approach .....	4
3.1 Problem .....	4
3.2 Goal .....	4
3.3 Approach .....	4
Chapter4 .....	5
Related Work.....	5
4.1 Combining Multiple Markers.....	5
4.2 Hand Gesture Recognition .....	5
4.3 Visual Programming.....	6
Chapter5 .....	7
System Design.....	7
5.1 System Overview .....	7
5.2 Multiple Kinds of Markers.....	8
5.2.1 NFT (Nature Feature Tracking) Marker.....	8
5.2.2 3D Object Marker .....	9

5.2.3 ATK (ARToolkit) Marker .....	10
5.2.4 ArUco Marker .....	10
5.2.5 QR (Quick Response) Code .....	11
5.2.6 Hide Marker .....	11
5.2.7 Combining Multiple Markers Information.....	12
5.3 Hand Gesture.....	13
5.3.1 Hand Information Acquisition.....	13
5.3.2 Hand Gesture Recognition .....	14
5.3.3 Hand Gesture Interaction .....	15
5.4 Interactions with Multiple Markers and Hand Gesture .....	15
5.4.1 Combining Multiple Markers and Hand Gesture .....	16
5.4.2 Interactions with Multiple Markers and Hand Gesture .....	17
5.5 Visual Programming with Multiple Markers and Hand Gesture.....	18
5.5.1 Markers Sequence .....	18
5.5.2 Grammar Structure and Markers Design .....	19
5.6 Virtual Marker .....	25
5.6.1 Generating Virtual Marker .....	26
5.6.2 Manipulating Virtual Marker .....	26
5.6.3 Combining Virtual Markers and Real Markers .....	28
5.7 Example of Applications .....	28
5.7.1 Virtual Timer .....	28
5.7.2 Hide Marker Controller.....	31
5.7.3 Controlling AR Content with Hand Gesture .....	35
5.7.4 Visual Programming – Fibonacci Sequence.....	36
5.7.5 Visual Programming – Controlling AR Content .....	39
Chapter6 .....	42
System Implementation.....	42
6.1 System Hardware .....	42

6.1.1 Web Camera and Tripod.....	42
6.1.2 PC .....	43
6.1.3 Leap Motion .....	43
6.2 Development Environment .....	44
6.3 Multiple Markers Recognition .....	44
6.3.1 NFT (Nature Feature Tracking) Marker Recognition .....	44
6.3.2 3D Object Marker Recognition.....	46
6.3.3 ATK (ARToolkit) Marker Recognition .....	47
6.3.4 ArUco Marker Recognition.....	48
6.3.5 QR Code Marker Recognition .....	48
6.3.6 Hide Marker Configuration.....	49
6.3.7 Multiple Markers Combination.....	50
6.4 Hand Gesture Recognition .....	50
6.4.1 Environment Configurations .....	51
6.4.2 Hand Shape Recognition.....	51
6.4.3 Hand Motion Detection.....	55
6.4.4 Hand Gesture Recognition .....	56
6.5 Visual Programming with Multiple Markers and Hand Gesture.....	58
6.5.1 Markers Sequence Recognition.....	58
6.5.2 Execution of Visualization Function .....	59
6.6 Virtual Marker .....	59
6.6.1 Virtual Marker Modeling .....	59
6.6.2 Generating Virtual Marker in Scenery .....	60
6.6.3 Manipulations of Virtual Marker.....	60
6.6.4 Visual Programming with Virtual Markers and Real Markers.....	61
Chapter7 .....	62
Preliminary Evaluation.....	62
7.1 Questionnaire .....	62

7.2 Result and Analysis .....	63
7.3 Summary .....	64
Chapter8 .....	65
Conclusion and Future Work.....	65
8.1 Conclusion.....	65
8.2 Future Work.....	66
Reference.....	67

# List of figures

Figure 1 System Overview .....	7
Figure 2 An NFT marker which contains enough feature points .....	9
Figure 3 A 3D object marker which contains enough feature points .....	9
Figure 4 An example of ATK marker with a black frame and central image.....	10
Figure 5 An example of ArUco marker with a sole id number 2 .....	10
Figure 6 An example of QR code which contains an URL in marker.....	11
Figure 7 An example of hide marker definition .....	12
Figure 8 Leap Motion is connected to PC when working.....	13
Figure 9 Framework of hand gesture recognition .....	14
Figure 10 The static fist hand shape.....	14
Figure 11 Virtual cars and virtual hand Motion in one scenery .....	16
Figure 12 Framework of interactions with markers and hand gesture.....	17
Figure 13 Marker 1 to Marker 8 are arranged in 2 lines .....	18
Figure 14 Variable X marker and variable Y marker .....	19
Figure 15 Constant marker.....	20
Figure 16 Hide marker for numbers input.....	20
Figure 17 AR number on variable marker.....	20
Figure 18 Numbers input with pairing .....	21
Figure 19 Markers to define loop structure for visual programming.....	22
Figure 20 Left hand defining gesture .....	22
Figure 21 If-Condition definition.....	23
Figure 22 Then definition and else definition .....	23
Figure 23 Define if-then-else structure for visual programming .....	24
Figure 24 Function structure definition markers.....	24
Figure 25 Right hand defining gesture.....	25
Figure 26 Action markers to control AR content .....	25
Figure 27 Touching real marker to generate its virtual marker .....	26
Figure 28 Hand gesture to control virtual markers .....	27
Figure 29 Using finger ray to select virtual makers in AR scenery .....	27
Figure 30 Four number markers to show AR numbers of the time.....	29
Figure 31 Semicolon markers to show AR semicolon .....	29
Figure 32 Virtual timer using multiple markers .....	30
Figure 33 Hand gesture to control the timer .....	30
Figure 34 AR markers to generate virtual cars.....	32

Figure 35 Hide marker controller to control virtual cars .....	32
Figure 36 QR code to change material of the virtual car .....	33
Figure 37 Hand gesture for pairing input and output .....	33
Figure 38 Some visual clues for pairing input and output .....	34
Figure 39 Hand gesture to control AR content.....	35
Figure 40 If marker definition.....	36
Figure 41 Then marker definition .....	37
Figure 42 Else marker definition.....	37
Figure 43 Fibonacci Function marker definition.....	38
Figure 44 Makers for visual programming with Fibonacci function marker.....	38
Figure 45 Executing gesture for markers sequence .....	39
Figure 46 Executed outcome of markers sequence .....	39
Figure 47 Combining action marker and grammar structure .....	40
Figure 48 Next step gesture.....	41
Figure 49 Logicoool HD Pro Webcam c920r.....	42
Figure 50 Velbon EX-Mini S .....	42
Figure 51 TOSHIBA PC.....	43
Figure 52 Leap Motion.....	43
Figure 53 Vuforia Target Manager .....	44
Figure 54 Importing Vuforia database to Unity .....	45
Figure 55 NFT markers configurations in Unity.....	45
Figure 56 AR content on an NFT marker.....	45
Figure 57 3D object scanning information in a smartphone .....	46
Figure 58 AR content around an 3D object marker .....	46
Figure 59 Register an ATK marker in the database.....	47
Figure 60 AR number on an ATK marker .....	47
Figure 61 Red AR cube on an ArUco marker .....	48
Figure 62 Recognized information from QR code shown on UI .....	48
Figure 63 ZXing API used in the recognition C# script.....	49
Figure 64 Virtual Button Area on NFT marker .....	49
Figure 65 Virtual Button Behavior script.....	49
Figure 66 C# Script to combine multiple markers information together .....	50
Figure 67 Virtual hands in Unity 3D Scenery .....	51
Figure 68 Hand shapes in the system.....	52
Figure 69 Fingertips, palm and wrist position on hand.....	54
Figure 70 Hand shape prediction outcome shown on Console window .....	55

Figure 71 Current palm velocity data shown on Console window .....	56
Figure 72 Finger ray from virtual hand.....	57
Figure 73 ATK markers position in Unity scene view .....	58
Figure 74 Executed code saved in .txt file .....	59
Figure 75 Real Marker and its corresponding virtual marker .....	60
Figure 76 Questionnaire for participants.....	62
Figure 77 Results from participants .....	63

# Chapter 1

## Introduction

### 1.1 Introduction

Augmented reality (AR) is a live real-world environment whose elements are augmented by computer-generated sensory input such as sound, video, graphics or GPS data [1].

AR markers are images that can be detected by a camera and used with software as the location for virtual content placed in a real scene [2].

There are many kinds of visual markers. Some of them can be used as AR marker, for example, ARToolkit marker, and some of them can be used only to provide some information, for example, QR code.

We call a system that uses AR marker to display AR content and other markers to provide control information as marker-based AR system.

Hand gesture is a popular way for human interactions [3]. It is widely used in AR and VR system. Users can use his hand gesture to interact with virtual content.

In this research, we will combine multiple markers and hand gesture in marker-based AR system for interactions. We can use multiple markers and hand gesture for visual programming and some other interactions. Based on the framework, we can make some applications of our system.

### 1.2 Organization of the thesis

The rest of the thesis is organized as follows: Chapter 2 will talk about the background of the thesis. Chapter 3 will tell research goal and approach. Chapter 4 will be about some related works. Chapter 5 will introduce the design of our system. Chapter 6 will be about the implementation of the system. In Chapter 7, we will make some preliminary evaluation of the system. And in Chapter 8, we will make some conclusions of our research.



# Chapter2

## Background

### 2.1 Augmented Reality

Augmented Reality is a technology that supplements the real world with virtual information that appears to coexist in the same space as the real world [4].

In AR scenery, we can combine virtual object and real environment. And we can make some real-time interactions with virtual content [5].

Now augmented reality technology is gradually becoming more and more popular. AR is also used in many fields, for example, education [6], commerce [7], collaboration [8] and social use [9].

There are many kinds of AR. Some are based on GPS location [10], some are based on plane recognition [11] and some are based on visual markers. In our research, we focus on marker-based AR system.

### 2.2 Visual Markers

Visual markers are widely used in marker-based AR system. Markers can be tracked and recognized with Web camera. If the visual marker serves as AR marker, it can be used to display some virtual content [12]. And visual markers can also provide some information to the system for some interactions, for example, QR code.

There are many kinds of visual markers. In this research, we will use nature feature tracking marker, ARToolkit marker, ArUco marker, 3D object marker and QR code marker for interactions and further application.

### 2.3 Hand Gesture Interactions

With the development of human-computer interaction technology, how to use natural and efficient interaction methods in the virtual environment has become a hot topic of research. Hand gesture is one of the most important interaction methods of human, which can

effectively express users' intentions [13].

Hand gesture is a mode of non-verbal interaction medium and can provide the most intuitive, originaive and natural way to interact with computers. It can make the interaction between human and computer as natural as the interaction between humans [14].

Hand gesture is popular in AR field. HoloLens uses some hand gesture for interactions with the machine. Leap Motion are also widely used in some AR applications for human interactions with AR world.

Using hand gesture in AR system can make the interactions natural and intuitive. With hand gesture, users can get started with the AR system more conveniently.

# Chapter3

## Research Goal and Approach

### 3.1 Problem

There are some problems of current marker-based AR system.

1. Different markers are usually independent of each other and their information lacks interaction with each other.
2. There are little interactions between hand gesture and markers.
3. Users lack intuitive interactions with AR content or the system.

With these problems, users have limited interactions with marker-based AR system. However, interactions are important for users experience for an AR system.

### 3.2 Goal

To solve current problems, our research goal is to combine multiple kinds and numbers of markers and hand gesture together for more intuitive interactions in marker-based AR system.

### 3.3 Approach

Firstly, we need to recognize and combine multiple kinds and numbers markers information together.

Secondly, we need to introduce hand gesture in marker-based AR system.

After the first step and second step, we need to combine multiple markers and hand gesture together and make some interactions with virtual content and the system.

Finally, we can use them for some applications.

# Chapter4

## Related Work

### 4.1 Combining Multiple Markers

We need to combine multiple markers and make them work together.

In Fang's work [15], he provides some ways to combine multiple markers to control some AR content. However, in his work, control method is simple and markers function are limited.

In Tada's work [16], he combines multiple ARToolkit markers for control as visual programming. However, in his work, markers can only be put in one-dimension, which will be limitation for relatively complex functions.

Fang's work and Tada's work provides some possibilities to combine multiple markers information. In this research, we will use multiple markers for control or some other functions. Markers will be put with some sequence in two-dimension and will work as visual programming with relatively complex functions.

### 4.2 Hand Gesture Recognition

We need to introduce hand gesture in marker-based AR system. Therefore, we need to recognize user's hand gesture in real time.

Leap Motion is a popular device which can be used to detect user's hand information and return some data for processing and analysis.

In Lu's work [17], she classifies hand gesture with 3 levels, and provides some good ways to recognize user's hand gesture in real time for some interactions in VR.

Zeng's work [18] provides some ways to use Leap Motion for Hand Gesture Recognition. Michahial's work [19] provides some possibilities to use SVM for hand gesture recognition, but for dynamic hand gesture, there will still be some limitations. And Filho's work [20] talks about hand gesture recognition with Unity Engine.

In this research, we will use Leap Motion and SVM for hand gesture recognition in AR

scenery, and there will be some static and dynamic hand gesture for interactions with virtual content or the system.

### 4.3 Visual Programming

We will use multiple markers and hand gesture for visual programming.

Tada's work [16] provides some ways to use some visual markers for visual programming, which can be used for control. His framework includes loop structure and branch structure. However, in his work, there is no function structure for visual programming. And Numbers expression is not enough.

Rose's work [21] provides some simple visual programming examples. However, in the examples, grammar structure for visual programming is still simple, which is not enough for relatively complex operations.

In this research, we will use multiple visual markers and hand gesture together for visual programming. The framework will include function structure, which means we can make some relatively complex operation for control. With the framework, we can make some applications.

# Chapter5

## System Design

In this chapter, we will introduce design of our system. Firstly, we will introduce the overview of the system. Our system mainly contains 2 parts, multiple kinds of markers interactions and hand gesture interactions. And then, we will introduce about these 2 parts separately. After that, we will introduce some interactions with multiple markers and hand gesture, and some applications of our system.

### 5.1 System Overview

In our system, we use multiple kinds and numbers of markers and hand gesture together to make some interactions with virtual content in AR world [Figure 1].

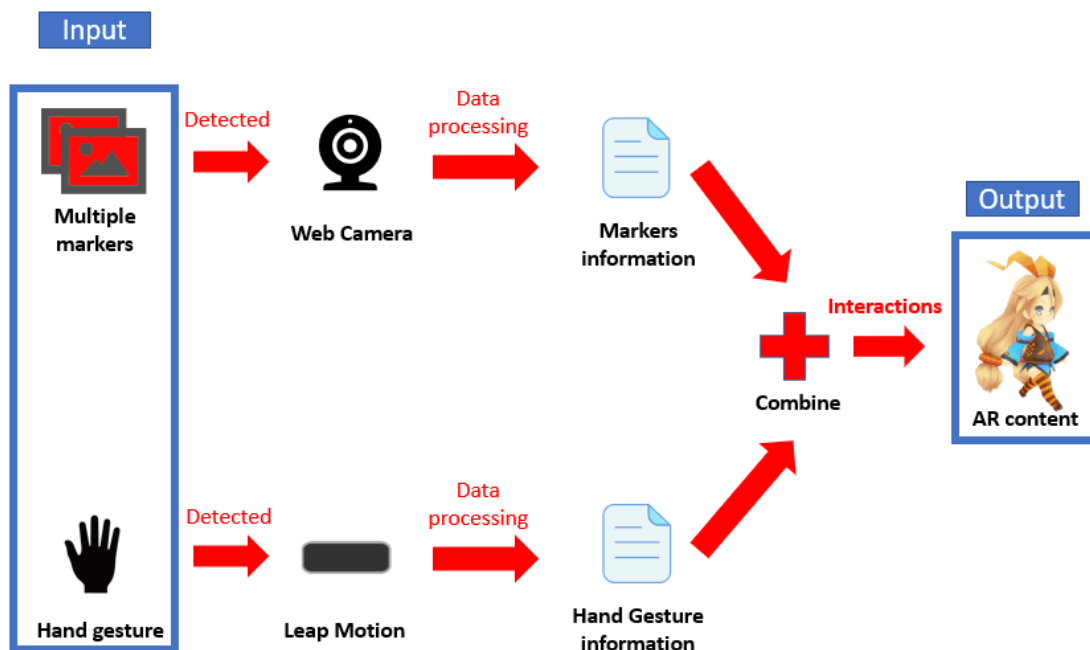


Figure 1 System Overview

For multiple kinds of markers, in our system, we can combine some of NFT (Nature Feature Tracking) marker, 3D object marker, ARToolkit marker, ArUco marker and QR code together to make some interactions. As a first step to use these markers together, we need to track and recognize them with Web camera. And after data processing, we can get

markers information from them. We need to combine information of these markers for some interactions.

For hand gesture, we need to define and recognize some hand gestures. To detect and recognize hand gesture, we need to use Leap Motion as a deep sensor to track user's hand. The recognized hand gesture can be used for some interactions with AR content or markers. The hand gesture definition should be explicit and easy to make by users.

As a next step, we can combine multiple markers and hand gesture together, and use them to make some interactions with AR content.

Our system can be used for some applications, for example, visual programming, virtual timer and AR content controlling. We will introduce them in the following sections.

## 5.2 Multiple Kinds of Markers

In our system, we will use multiple kinds and numbers of markers for interactions. They are NFT marker, 3D object marker, ARToolkit marker, ArUco marker and QR code. We need to recognize them and combine their information together for some interactions with AR content.

### 5.2.1 NFT (Nature Feature Tracking) Marker

NFT marker is a kind of 2D image marker. It uses nature feature points of the image as basis of tracking and recognition. Every 2D image can serve as NFT marker only if it has enough nature feature points for tracking [Figure 2]. In our system, we mainly use NFT markers as AR marker, which means we mainly use it to show some AR content we want to interact or control on the marker.

There are many ways to track NFT marker. In our system, we mainly choose to use Vuforia Engine to help us track and recognize NFT marker.



Figure 2 An NFT marker which contains enough feature points

### 5.2.2 3D Object Marker

3D object marker is a kind of 3D marker. 3D object marker has a similar recognition method to NFT marker. It uses nature feature points of an object as the basis of tracking and recognizing. We also mainly use 3D object marker as AR marker to show some AR content.

A good 3D object marker for tracking also need enough nature feature points. So, we need to choose some objects containing many points to serve as marker [Figure 3].

In our system, we use Vuforia Engine to track 3D object marker.



Figure 3 A 3D object marker which contains enough feature points



### 5.2.3 ATK (ARToolkit) Marker

ATK marker is a kind of 2D marker. It uses its black frame for tracking and its central image for recognition [Figure 4]. ATK marker can show its content explicitly through the central image designed by user. In our system, we mainly use ATK marker to provide some control information for AR content.

We use ARToolkit SDK to help us track and recognize ATK marker.



Figure 4 An example of ATK marker with a black frame and central image

### 5.2.4 ArUco Marker

ArUco marker is a kind of 2D marker. It is tracked with a sole id number [Figure 5]. Its recognition way is similar to ATK marker. We can use ArUco marker to serve as AR marker, and we can also use it to provide some control information in our system.

We use ArUco SDK to help us track and recognize ArUco marker in our system.

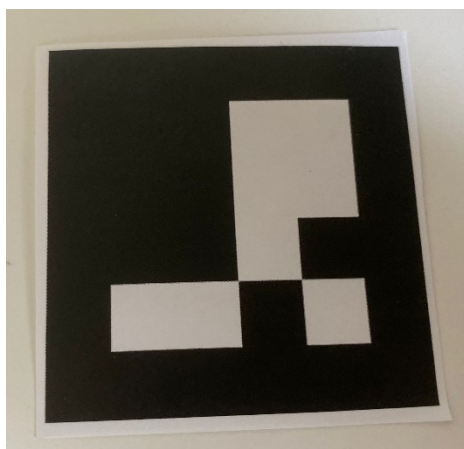


Figure 5 An example of ArUco marker with a sole id number 2

### 5.2.5 QR (Quick Response) Code

QR code is a kind of 2D marker. It can contain some information such as an URL in marker [Figure 6]. In our system, we use it to contain some control information which can be used to interact with AR content.



Figure 6 An example of QR code which contains an URL in marker

We use ZXing API to help us read and recognize QR code.

### 5.2.6 Hide Marker

Hide marker is a kind of NFT marker. It also uses its nature feature points for tracking and recognizing. However, it has some special features comparing to traditional NFT markers.

If some part of a hide marker is blocked, the system can return some response. The user can define the hide part by himself. And a hide marker also can contain many hide parts for multiple response. [Figure 7] shows an example of hide marker definition. The image is the NFT marker, and the blue square parts on it are user defined hide parts. After definition, the marker can serve as hide marker. If the user blocks the hide parts in the AR camera view, the system will give some response.

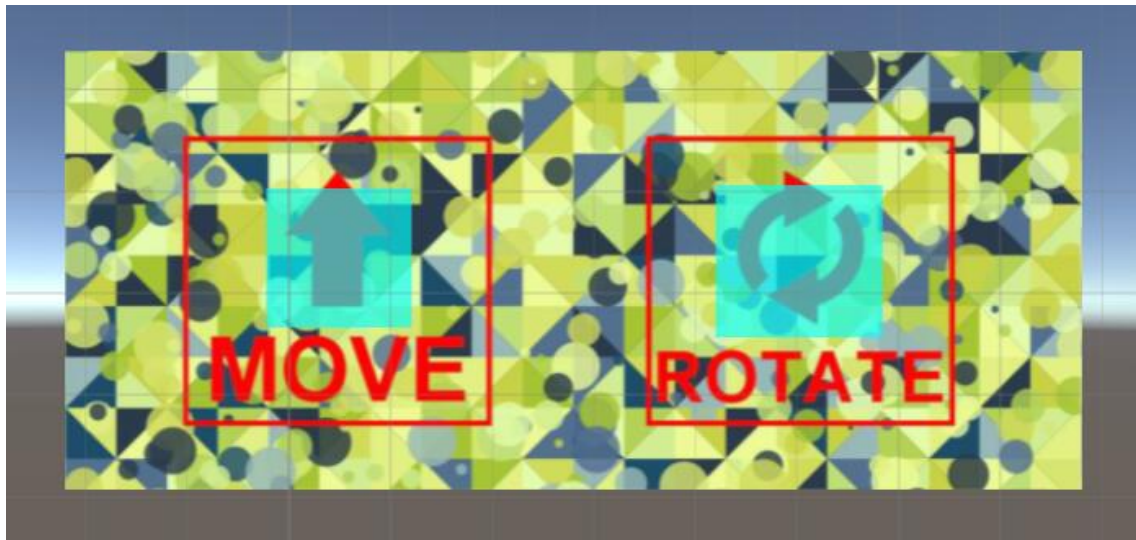


Figure 7 An example of hide marker definition

We can use the feature of hide marker to make some input in our system. We can touch different part of a hide marker for different input and output, and we can also touch multiple parts of a hide marker for a multiple input to the system. With hide marker, we can make more input to the system with just one marker. It will save markers space that using one marker to replace a series of markers as the input.

With hide marker, we can make more interactions in our system.

We will use Vuforia Engine to help us to define and recognize hide markers. With some scripts, we can define many functions on a hide marker.

### 5.2.7 Combining Multiple Markers Information

Combining multiple kinds and numbers of markers together, we can make more interactions with AR content.

Using different kinds of SDK and API, we can recognize different kinds of markers respectively. As a next step, we need to combine these SDK or API in one Unity scenery for use. To make them work together, we need some scripts to manage them and combine recognized information from them. And after that, we can get the combined information from multiple markers and we can use the information for interactions with AR content. For combined multiple markers, some markers will serve as input markers and some will serve as output markers. In our system, output marker will be AR marker. It can show

some AR content which is used for interactions. And other markers will be used as input markers. We can use them to provide some information as input to control or make some interactions with AR content shown on AR marker.

## 5.3 Hand Gesture

In our system, we will use hand gesture for some interactions. To detect and recognize user's hand gesture, we need to collect user's hand information and process them for some analysis. After that, we can use the recognized hand gesture to make some interactions with markers or AR content.

### 5.3.1 Hand Information Acquisition

As the first step to use hand gesture, we need to collect user's hand information for recognition and interactions.

We will use Leap Motion as the sensor to detect and track user's hand information in real time. Leap Motion can detect hand shape and motion information for every frame, and the information can be used for hand gesture recognition and interactions after data processing.

In our system, Leap Motion will be linked to PC through USB cable [Figure 8]. The hand information detected by Leap Motion will be transported to PC for further use.



Figure 8 Leap Motion is connected to PC when working

### 5.3.2 Hand Gesture Recognition

After we access hand information from Leap Motion, we need to process and analyze the information for hand gesture recognition.

To recognize user's hand gesture, we separate hand gesture to 2 parts, static part and dynamic part [Figure 9]. The static part is the shape of hand, which also means how does the hand look like, for example, a fist hand [Figure 10] or an open hand. The dynamic part is the state of hand movement, for example, the hand palm movement velocity or the direction of the fingers. We can analyze these 2 parts separately. After that, we can combine the analysis outcome of these 2 parts to decide the gesture of user's hand.

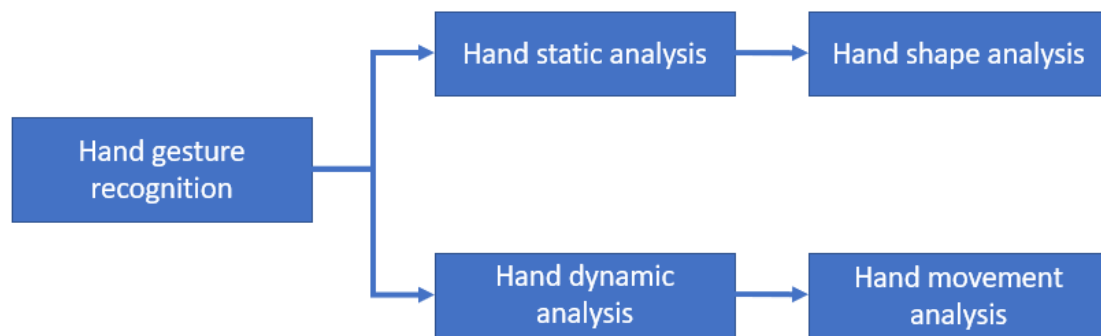


Figure 9 Framework of hand gesture recognition



Figure 10 The static fist hand shape

For hand shape analysis, we can get the position information of fingers, palm and wrist of user's hand from Leap Motion. Based on the position information, we can use Multiple Class SVM as the classifier to recognize and classify different kinds of hand shapes.

For hand movement analysis, we can also access the hand motion information from Leap Motion, for example, the palm velocity, the palm direction and the finger direction. After some analysis, we can judge the hand movement statement from the motion information. After static and dynamic analysis, we can combine these 2 parts to analyze the current hand gesture state of user's hand. With the recognized hand gesture, we can make some interactions with markers or AR content.

### **5.3.3 Hand Gesture Interaction**

After hand gesture recognition, we can design some interactions with the hand gesture. In our system, there are mainly 2 kinds of hand gesture interactions. They are:

- (1) Using hand gesture to interact with AR content.
- (2) Using hand gesture to interact with markers.

We can use hand gesture to interact with AR content. For example, we can use hand gesture to control AR content's size and position, or we can choose some AR content with hand gesture for some effects. To interact with AR content, we need to match hand gesture to its effect on AR content.

We can also use hand gesture to interact with markers. In our system, there are multiple kinds and numbers of markers. We can use hand gesture to interact with them to combine multiple markers and hand gesture together, which will bring more functions and interactions in our system.

With hand gesture, the interactions in the system will be natural and easy to use.

## **5.4 Interactions with Multiple Markers and Hand Gesture**

In our system, we will combine multiple markers and hand gesture together for some interactions between user and AR world.

After multiple markers recognition and hand gesture recognition, we can use multiple markers and hand gesture in our system. With some configurations, we can combine

multiple markers and hand gesture in a scenery and make some interactions. Based on this, we can design some applications of our system.

#### **5.4.1 Combining Multiple Markers and Hand Gesture**

With multiple markers recognition, we can combine multiple markers information in the system and use it for some interactions.

With hand gesture recognition, we can recognize user's hand gesture in the system for some interactions.

We can combine multiple markers recognition and hand gesture recognition in one system with their SDKs and some configurations. After that, we can use both markers and hand gesture for interactions in one scenery. [Figure 11] shows an example of combining multiple markers and Leap Motion. In the AR camera view, we can see virtual cars generating from markers and virtual hand generating from Leap Motion, which are combined in one scenery. The virtual hand can make some gestures, which can be recognize with hand gesture recognition in the system. In this case, we can combine markers and hand gesture together.

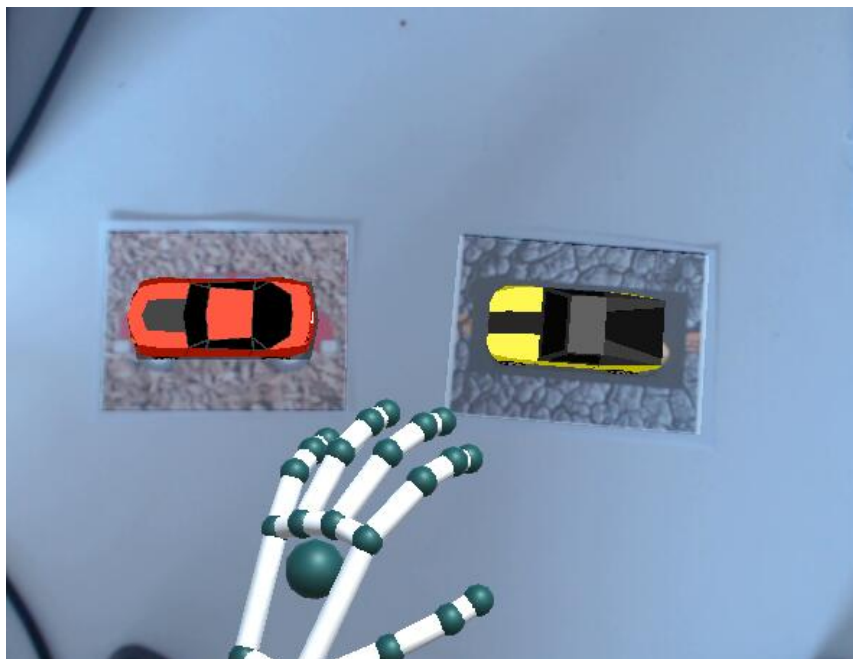


Figure 11 Virtual cars and virtual hand Motion in one scenery



### 5.4.2 Interactions with Multiple Markers and Hand Gesture

To make some interactions in our system, we need to correlate the control information of markers and hand gestures with some responses of AR content or the system. Based on different applications or user case, we can design different responses and interactions.

The overall design idea of this part is shown in [Figure 12]. We can get markers information with multiple markers recognition. And we can get current hand gesture information of user with hand gesture recognition. After that, we can get the interaction information in the system based on the combined markers information and hand gesture information. With the interaction information, we can control or interact with AR content in the scenery or some output of the system.

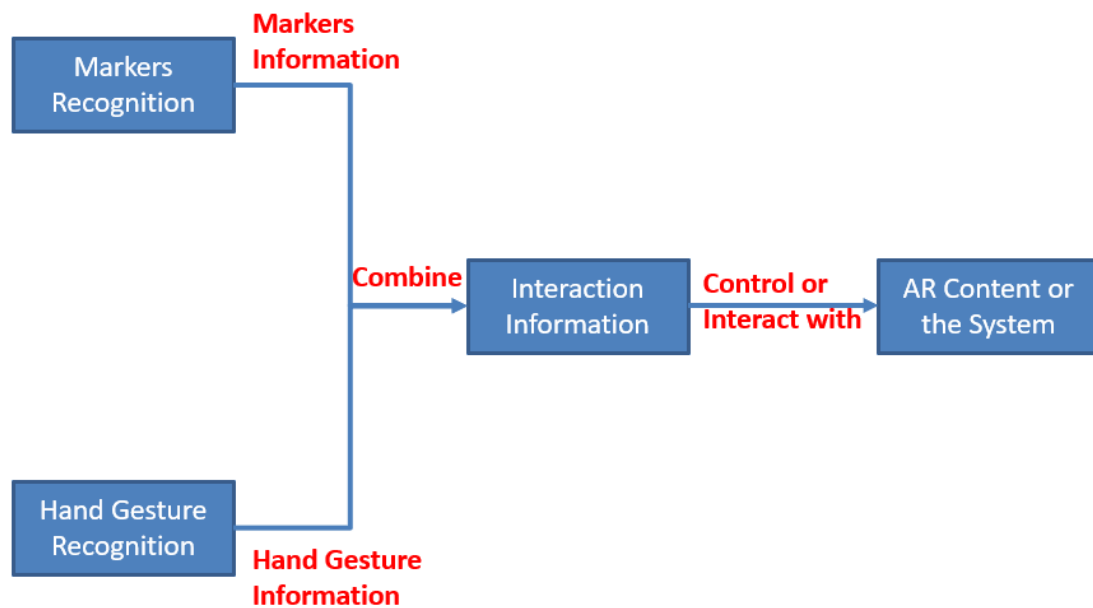


Figure 12 Framework of interactions with markers and hand gesture

For different application or user case, we can design different interactions with AR content or the system, and we can also use different markers and hand gesture. In the following sections, we will introduce some applications of our system and the design of them.



## 5.5 Visual Programming with Multiple Markers and Hand Gesture

We will use multiple markers and hand gesture for visual programming. Each marker will contain some information, and multiple markers will be put with some sequence in the scenery. A series of markers will serve as a visualization program, and we can execute the program for some output in the system.

We have some language grammar structure for visual programming, for example, loop structure or function structure. With these structures, we can make some relatively complicated operations.

And the program defined by multiple markers can be executed for some output.

### 5.5.1 Markers Sequence

A series of markers will serve as a visualization program.

We will put multiple markers in 2-dimension with some lines. For each line, markers will be put from left to right. And markers lines will be put from up to down. Markers sequence for visualization program will be from left to right and line by line. In [Figure 13], there are 8 markers put in 2 lines. And for each line, there are 4 markers put from left to right. The markers sequence for them will be marker 1-2-3-4-5-6-7-8.

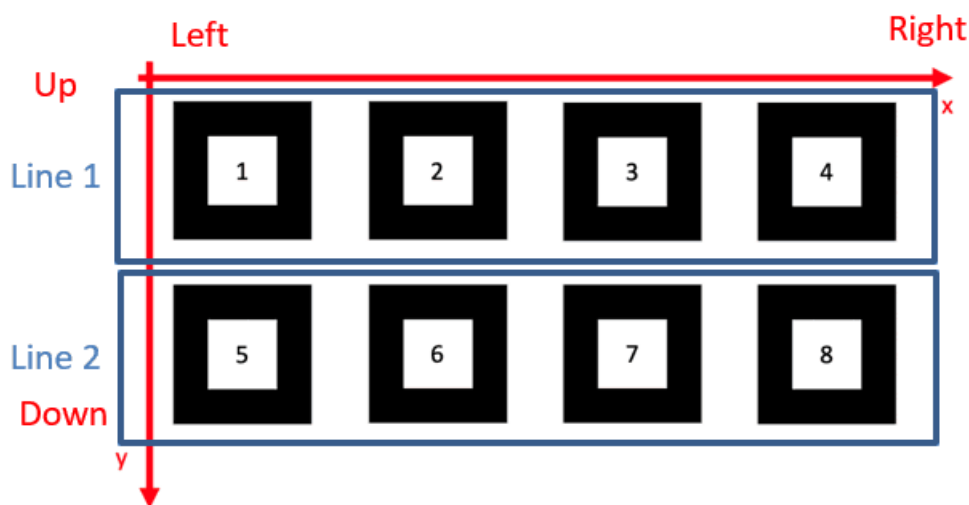


Figure 13 Marker 1 to Marker 8 are arranged in 2 lines

Markers sequence represents the logic of visualization program. Therefore, markers recognition sequence and execution sequence will be same as markers arranging sequence.

### 5.5.2 Grammar Structure and Markers Design

For visual programming, we have some language structure. And we will design some markers for these grammar structure.

#### *Sequence Structure*

For sequence structure, markers will be recognized and executed one by one as markers sequence we have defined. And there will not be some special markers for grammar structure.

#### *Variable and Number*

For visual programming, we may need some variable for use. Therefore, we designed variable markers [Figure 14], which can serve as a variable in the system and can contains some number for use.



Figure 14 Variable X marker and variable Y marker

We will also use numbers for visual programming. In this application, we use constant marker [Figure 15] to save some number, and we can assign numbers to variable marker or constant marker with hide marker [Figure 16]. By touching hide marker, we can input some numbers to the system, and the system will assign the input numbers to the variable marker or constant marker.

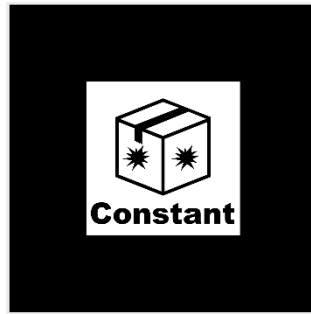


Figure 15 Constant marker

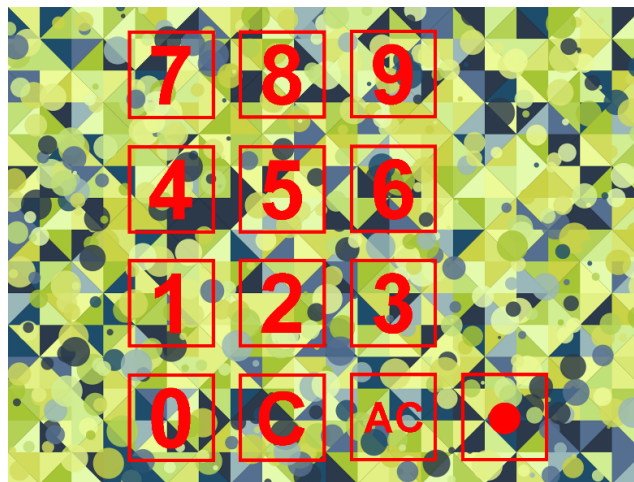


Figure 16 Hide marker for numbers input

Variable marker and constant marker will show their value with an AR number on the marker [Figure 17]. We can check their value through Web camera.



Figure 17 AR number on variable marker

For numbers input, if there are more than one variable markers in the scenery, we can use hand gesture to pair the hide marker and variable marker that we want to assign the value [Figure 18]. The selected marker will be covered with red highlight and the paired markers will be linked with virtual line. With the pairing, we can make numbers input explicitly.

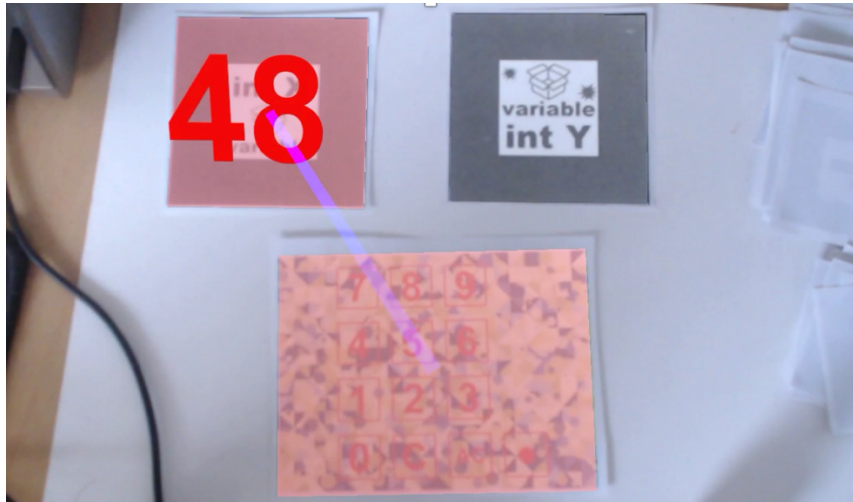


Figure 18 Numbers input with pairing

Variable and numbers are the basis of functions structure in this application. With variable markers and numbers input, we can use variable and numbers for visual programming.

### ***Loop Structure***

Loop structure is a basic grammar structure for programming language. We will also include loop structure in visual programming.

We will use loop marker and some other markers to define the loop structure for visual programming [Figure 19]. The loop marker is used in the beginning of the loop structure definition. Variable X marker is used to define the loop time with its value. We can put a series of markers after variable X marker to define the loop body.



```

Loop (x)
{
    Loop Body;
}

```

Figure 19 Markers to define loop structure for visual programming

After arranging markers for loop structure, we can make left hand defining gesture [Figure 20] to define the loop structure. After definition, the loop structure information will be saved in loop marker. And we don't need variable marker and loop body markers anymore. We can use loop marker to replace the whole loop structure for use.



Figure 20 Left hand defining gesture

### ***Branch Structure***

We will include if-then-else branch structure for visual programming.

We will use if marker, then marker and else marker to define the branch structure. To simplify if-then-else structure definition, we will define if-condition struct, then struct and else struct separately and combine them as the branch structure.

We will use a series of markers to define if-condition struct [Figure 21]. We will use a

series of markers to define the condition for the branch structure, and with left hand defining gesture, we can define the condition information in if marker. After definition, we can use only if marker to replace these markers.

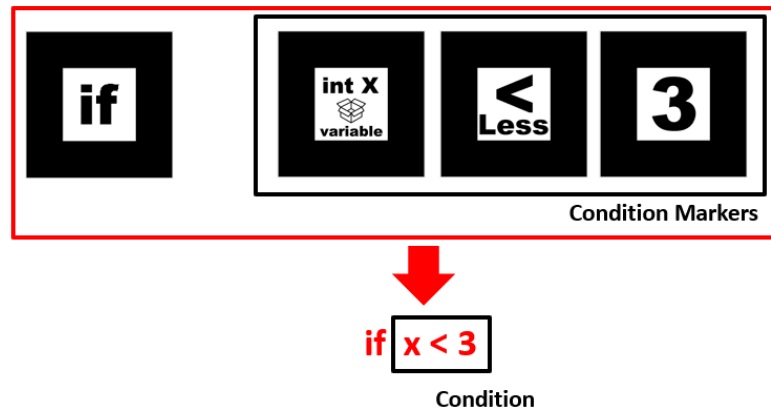


Figure 21 If-Condition definition

With same ways of definition, we can define then struct and else struct with a series of statement markers [Figure 22]. With left hand defining gesture, we can use then marker and else marker to replace a series of markers for definition.

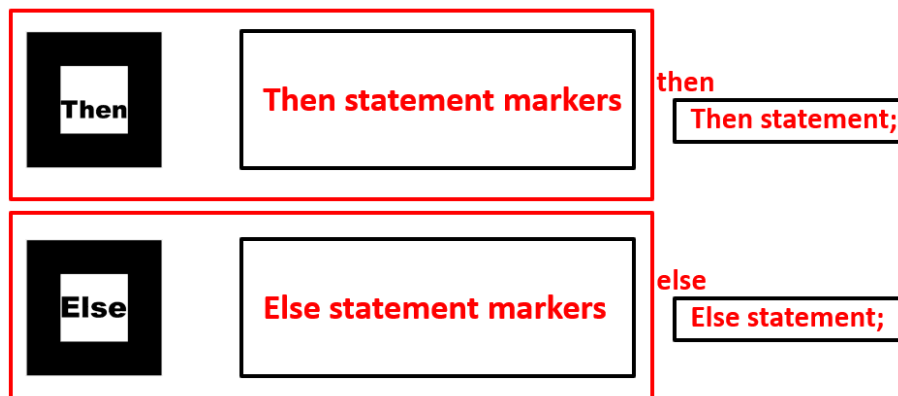


Figure 22 Then definition and else definition

After definition of if marker, then marker and else marker, we can combine them to form the if-then-else structure for visual programming [Figure 23].



**if** condition  
**then** Then statement;  
**else** Else Statement;

Figure 23 Define if-then-else structure for visual programming

### *Function Structure*

We will include function structure for visual programming.

We will use a function marker as the function name and use a series of markers to define the function body [Figure 24].



**Function ()**  
**{**  
**Function Body;**  
**}**

Figure 24 Function structure definition markers

With right hand defining gesture [Figure 25], we can save the function information in function marker. After that, we can use only one function marker to replace a series of

markers for function definition.



Figure 25 Right hand defining gesture

### ***Action Markers***

We can use some markers to control the AR content to do some actions. We call them action markers. In this application, we have some action markers [Figure 26]. They will make the human-like AR content do running action, jumping action and walking action separately.



Figure 26 Action markers to control AR content

These markers can be combined with grammar structure markers to make AR content do a series of actions as the program.

## **5.6 Virtual Marker**

In this application, we propose virtual markers.

Virtual markers are markers which only can be seen through AR camera, but can work with real markers together. we can put multiple real markers and virtual markers in an AR scenery. When the system is running, information from real markers and virtual markers will be combined for interactions or some control.



Virtual markers will also have the same appearances to real markers. Therefore, they can be used with real markers naturally.

Virtual markers will connect real world and virtual world in our system. Virtual markers are some virtual objects, but they can work with real markers, which means they can interact with real world. They can also interact with virtual world, for example, finger ray from virtual hand. Therefore, in our system, virtual markers are key points to connect real world and virtual world.

We can combine multiple virtual markers and real markers for visual programming or some other interactions.

### 5.6.1 Generating Virtual Marker

We can generate virtual markers in the scenery by touching its corresponding real markers through AR camera [Figure 27].



Figure 27 Touching real marker to generate its virtual marker

If a real marker is touched, the system will log the information of real marker, and generate a virtual marker which have a same appearance on the position that the real marker is put in the scenery. After that, we can remove the real marker from the scenery, and the corresponding virtual marker will still exist in the scenery. And we can use the generated virtual marker for some interactions.

### 5.6.2 Manipulating Virtual Marker

Virtual markers are some virtual objects. Therefore, they can be manipulated with virtual hand and hand gesture [Figure 28].

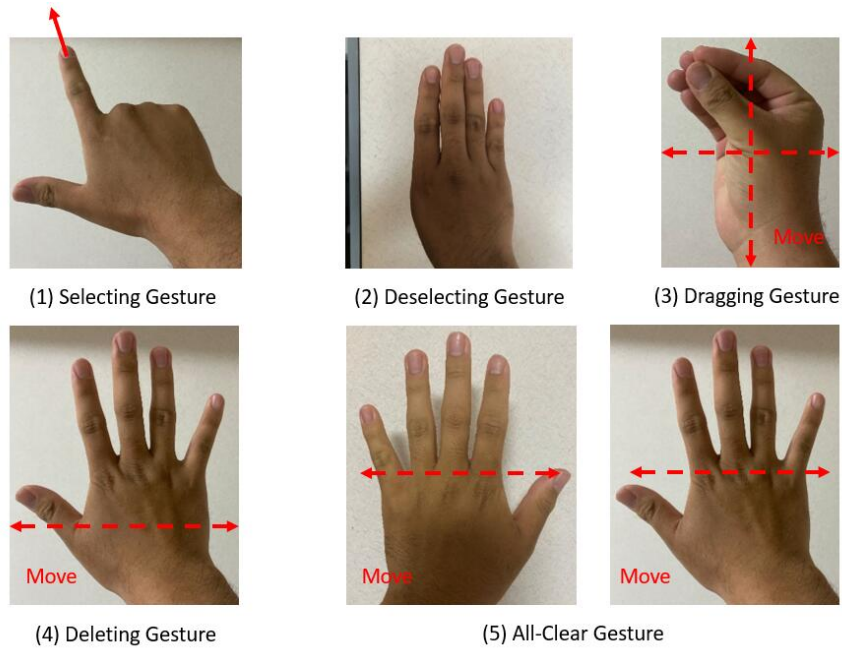


Figure 28 Hand gesture to control virtual markers

There are some descriptions of these hand gesture:

- (1) Selecting Gesture: To make this gesture, the user needs to open his thumb finger and index finger. There will be a finger ray from the index finger for selecting use in the AR scenery. The user can use the finger ray to select virtual markers for manipulation. To express the selected state of virtual markers, we will cover the selected virtual markers with red highlight [Figure 29].

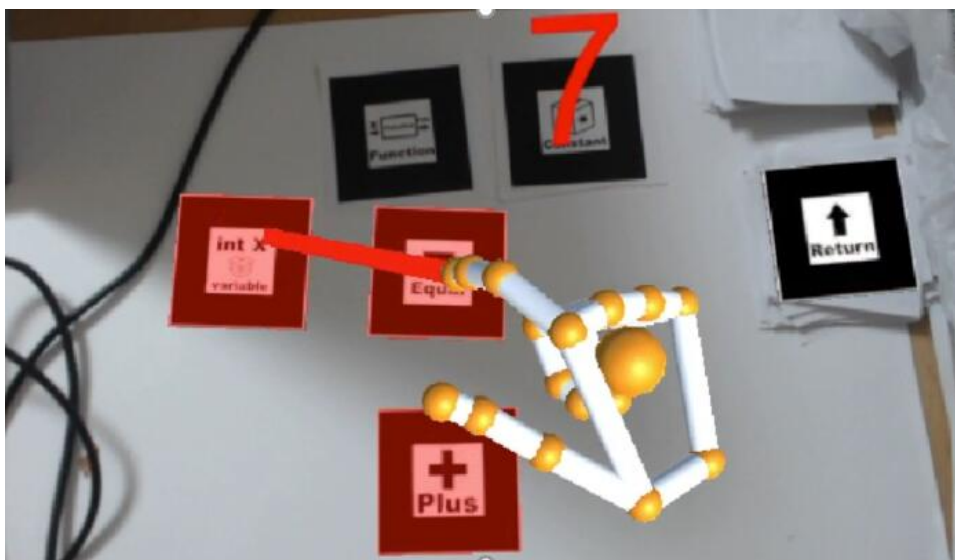


Figure 29 Using finger ray to select virtual makers in AR scenery

- (2) Deselecting Gesture: To make this gesture, the user needs to open all his fingers except the thumb finger. Using this gesture, the user can cancel the selected state of virtual markers.
  - (3) Dragging Gesture: This gesture is same to dragging gesture in application 5.4.4. Using this gesture, we can drag and move the locations of virtual markers in selected state. We can use this gesture to arrange virtual markers.
  - (4) Deleting Gesture: To make this gesture, the user needs to open all his fingers and hand palm and move his hand left and right with some speed. With this gesture, we can delete the selected virtual markers in the scenery.
  - (5) All-Clear Gesture: To make this gesture, the user needs to open all his fingers and hand palm for both hands and move his both hands left and right with some speed simultaneously. With this gesture, the user can delete all virtual markers in the scenery.
- With these hand gesture, we can manipulate virtual markers in the scenery.

### **5.6.3 Combining Virtual Markers and Real Markers**

We can put some virtual markers with some real markers together for visual programming. In the scenery, we will see some virtual markers and real markers existing together. These markers will locate in same position coordinate system in the scenery. Therefore, we can get markers sequence composed of multiple virtual markers and real markers. And as a next step, we can get the combined information from markers sequence. With the combined information, we can make our visualization program for some interactions.

## **5.7 Example of Applications**

### **5.7.1 Virtual Timer**

An example of our system is the virtual timer based on multiple markers and hand gesture. We will show AR time numbers on multiple ARToolkit markers. And we will use hand gesture to control the running of the virtual timer.

In this application, we only consider the minutes and seconds of the timer, not the hours. Therefore, we will use up to 4 number markers to show numbers of the timer [Figure 30].

And we also use a semicolon marker to show the AR semicolon to separate minutes number and seconds number [Figure 31].

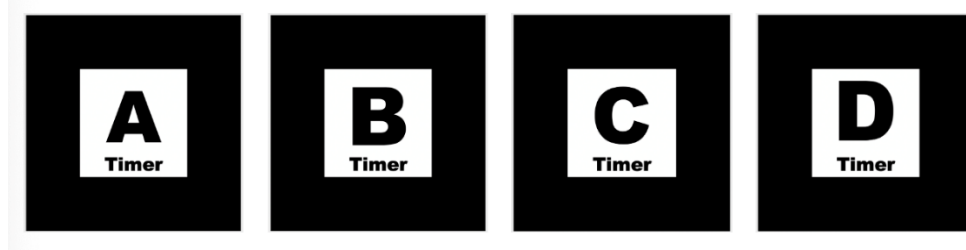


Figure 30 Four number markers to show AR numbers of the time

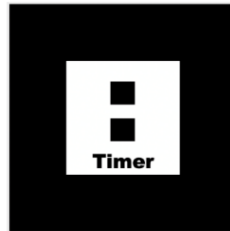


Figure 31 Semicolon markers to show AR semicolon

If we just put one number marker in the scenery, the number marker will display an AR number on it, and the number will increase 1 per second.

If there are four number markers, they will work as a timer [Figure 32]. The system will recognize number markers from left to right. The 2 rightmost markers will serve for seconds, and the 2 leftmost markers will serve for minutes. The time of the virtual timer will increase like real timer, and each number marker will display only one digit of the time one it. And we can put the semicolon marker between minutes markers and seconds markers to show an AR semicolon to separate them.



Figure 32 Virtual timer using multiple markers

We can set the start time of the timer. The default start time will be 00:00.

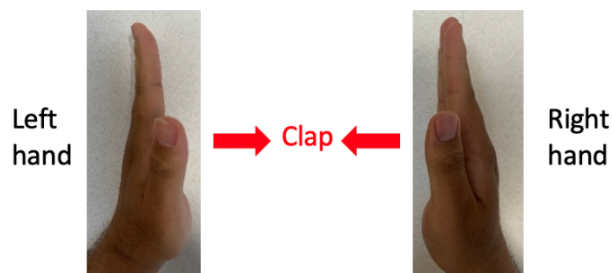
We can control the time of timer with hand gesture. In this application, we define 3 hand gesture to control the time [Figure 33].



(1) Pause Gesture



(2) Continue Gesture



(3) Restart Gesture

Figure 33 Hand gesture to control the timer

There are some descriptions of these hand gesture:

- (1) Pause gesture: To make this gesture, the user needs to clench his fist. This gesture will make the timer pause.

- (2) Continue gesture: To make this gesture, the user needs to open his palm and straighten his fingers. This gesture will make the timer continue working from pause state.
- (3) Restart gesture: To make this gesture, the user needs to use his two hands to clap. This gesture will make the timer reset as 00:00.

With these hand gesture, we can control the time on markers.

Combining these markers and hand gesture, we can make a virtual timer and make some interactions with it.

### **5.7.2 Hide Marker Controller**

Hide marker can be used as the input in our system. We can use hide marker as the controller to input some information to control the action of AR content.

In this application, we will use AR marker to generate virtual cars as the controlled AR content, and we will use hide marker to control its movement. Furthermore, we will use QR code to change the material of the controlled virtual car.

We will also use hand gesture for interactions. In this application, output will be actions of AR content, and input will be information from hide marker and QR code. We will pair the input marker and output marker with hand gesture for accurate controlling.

#### ***AR Marker Design***

As a first step, we need to design AR markers which will be used to generate AR content for interactions. AR marker should have enough nature feature points for tracking and recognizing. The picture of AR marker needs to intuitively reflect its corresponding AR content. In this application, we design 2 AR markers [Figure 34] to generate virtual cars for interactions.



Figure 34 AR markers to generate virtual cars

### ***Hide Marker Design***

Hide marker will be used to control virtual cars actions. By touching some parts of hide marker, we can input some control information to the system. In this application, we will control virtual cars to move forward or rotate with hide marker controller.

We design a hide marker for controlling [Figure 35]. By touching MOVE part, we can control the virtual car move forward. By touching ROTATE part, we can control the virtual car rotate. If we touch both MOVE part and ROTATE at same time, the controlled virtual car will move and rotate simultaneously.

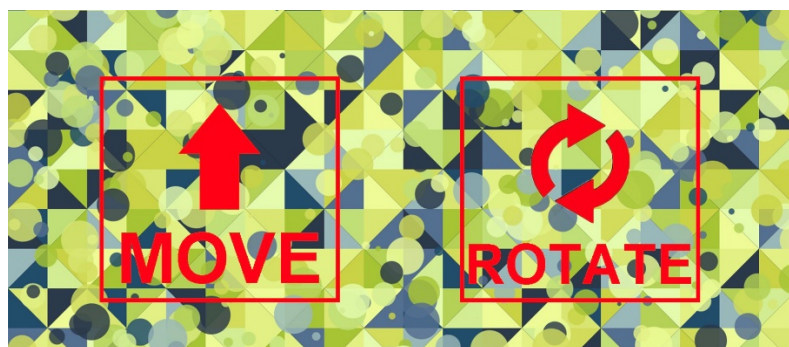


Figure 35 Hide marker controller to control virtual cars

With hide marker controller and some configurations in the system, we can control the actions of virtual cars in real time by touching the hide marker.

### ***QR Code Design***

We will use QR code to contain some information. After QR code recognition, the system



will recognize the information from QR code. And based on the information, the system will give some response on the AR content, which will be the output.

In this application, we will use QR code to change the material of controlled virtual cars. We design 3 QR code [Figure 36] for interactions with virtual cars. They will make the virtual car change to yellow color, blue color and transparent material respectively.



Figure 36 QR code to change material of the virtual car

### ***Pairing Input and Output***

In this application, there will be multiple input markers and multiple output markers. AR markers will be output markers. Hide marker and QR codes will be input markers. Therefore, we need to pair input markers and output marker for an accurate control.

We defined 2 kinds of hand gesture [Figure 37] to pair input marker and output marker.

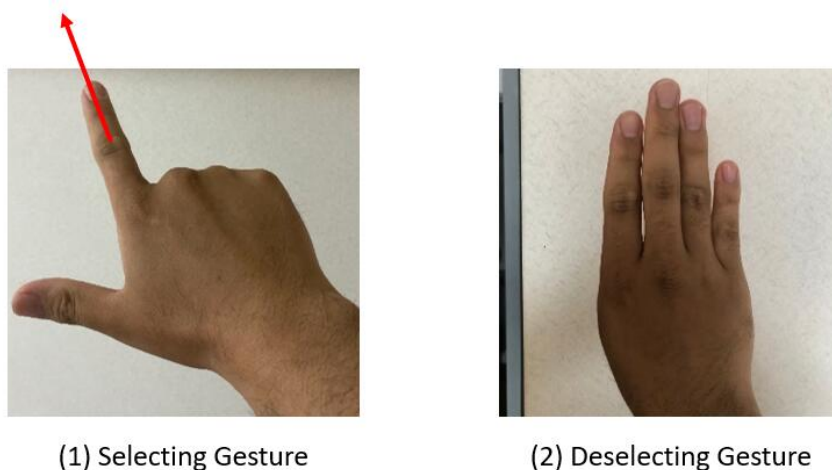


Figure 37 Hand gesture for pairing input and output



We can use selecting gesture to select some markers. The selected input markers and output marker will be paired, which means only the selected output marker will be controlled by selected input markers.

We can use deselecting gesture to cancel the selected state of markers.

We will use some visual clues to make the control explicit [Figure 38].

To express the selected state of markers, we will cover each marker some highlight. If the marker is in selected state, it will be covered with red highlight. And if the marker is not selected, it will be covered with white highlight.

To express the relationship between input markers and output marker, we will use virtual line to link each input marker to the output marker. And if there are more than one input markers, the linked lines will have different colors.

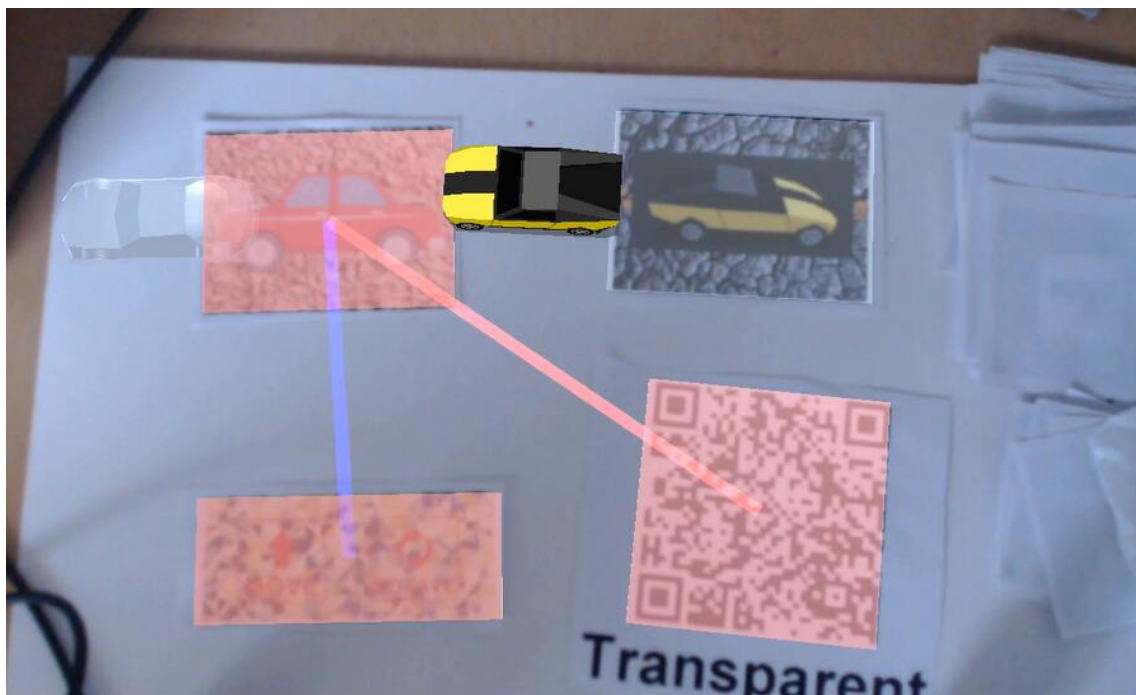


Figure 38 Some visual clues for pairing input and output

With these designed markers hand defined hand gesture, we can make some interactions with virtual cars in the application, and we can control the virtual car in real time with some explicit visual clues.

### 5.7.3 Controlling AR Content with Hand Gesture

We can use hand gesture to control the AR content directly. In this application, we will use AR marker to generate AR content. And we will use hand gesture to control the size of AR content, move the AR content and rotate the AR content.

#### *AR Content Generation*

We will use AR marker to generate AR content for control.

#### *Hand Gesture Definition*

In this application, we will use hand gesture to control AR content directly.

We define 3 kinds of hand gesture to control AR content [Figure 39].

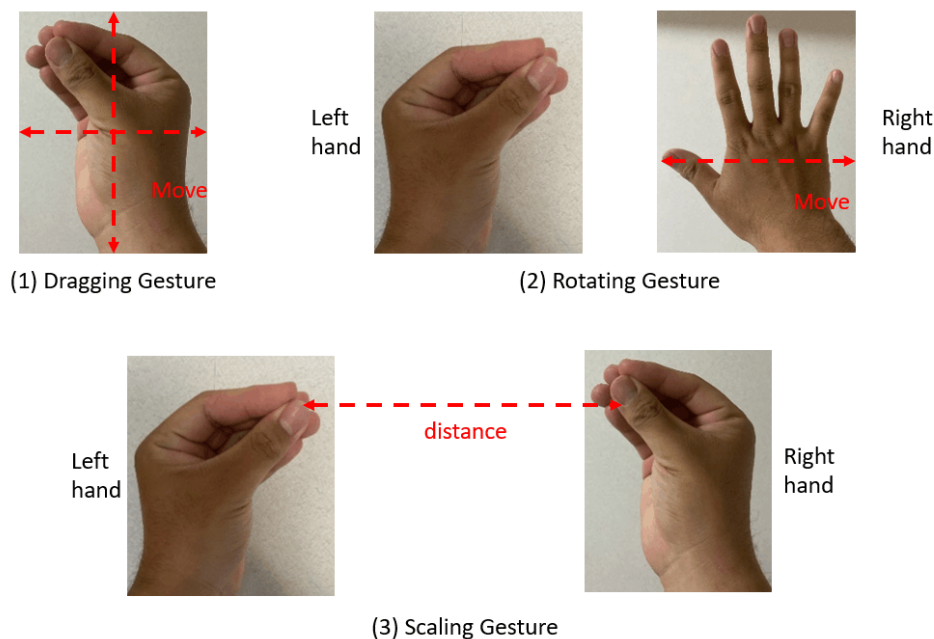


Figure 39 Hand gesture to control AR content

There are some descriptions of these hand gesture:

- (1) Dragging Gesture: To make this gesture, the user needs to concentrate all his left-hand fingers on one point and move his hand. The controlled AR content will have a same movement direction with the user's hand. And the movement speed of controlled AR content will be proportional to the movement speed of the palm. With this hand gesture, the user can drag and move the AR content to anywhere with has hand.
- (2) Rotating Gesture: To make this gesture, the user needs to use his left hand make the

dragging gesture hand shape and open his right-hand fingers and hand palm. The controlled AR content will rotate if the right-hand moves left and right. The rotation speed of the controlled AR content will be proportional to the movement speed of the right-hand palm. With this hand gesture, the user can rotate the AR content with his hands.

- (3) Scaling Gesture: To make this gesture, the user needs to concentrate his fingers to one point for his both hands and move his two hands. The controlled AR content will zoom in and out with the movement of both hands. The AR content will scale proportionally with the distance between the thumbs of both hands. With this hand gesture, the user can control the scale of the AR content with his both hands.

With these hand gesture, we can control AR content generating from AR marker in our system directly.

#### **5.7.4 Visual Programming – Fibonacci Sequence**

In this application, we will use visual programming in our system to define Fibonacci function and execute it.

##### ***Fibonacci Function Definition***

We will use a series of markers to define if-condition structure of Fibonacci function [Figure 40]. The defined condition, “if  $x < 3$ ” will show on if marker through Web Camera.

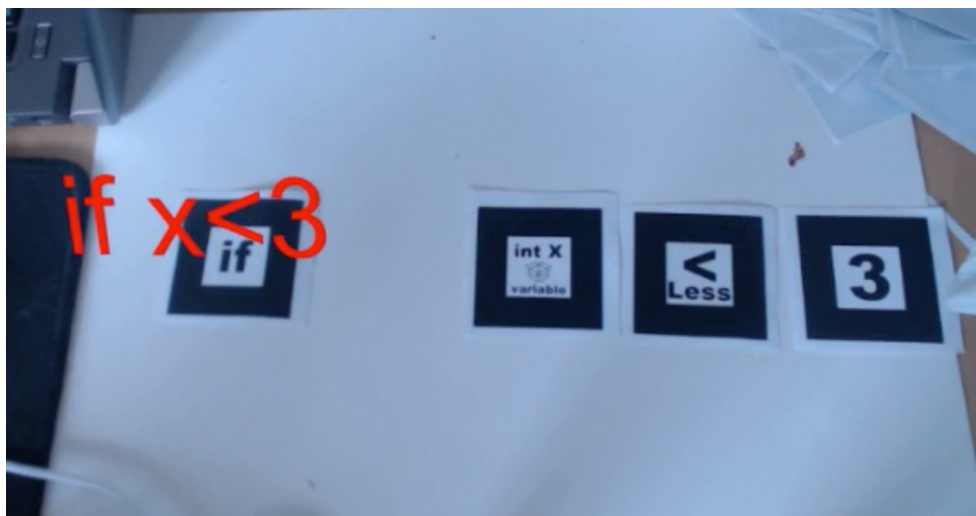


Figure 40 If marker definition

We will use a series of markers to define then statement of Fibonacci function [Figure 41]. The defined then statement, “then 1” will show on then marker through Web Camera.

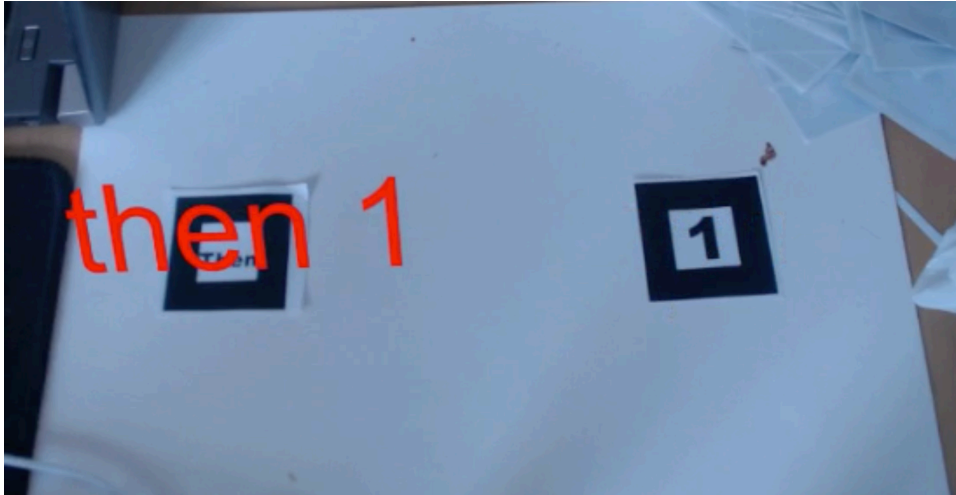


Figure 41 Then marker definition

We will use a series of markers to define else statement of Fibonacci function [Figure 42]. The defined else statement, “else  $F(x-1) + F(x-2)$ ” will show on then marker through Web Camera.

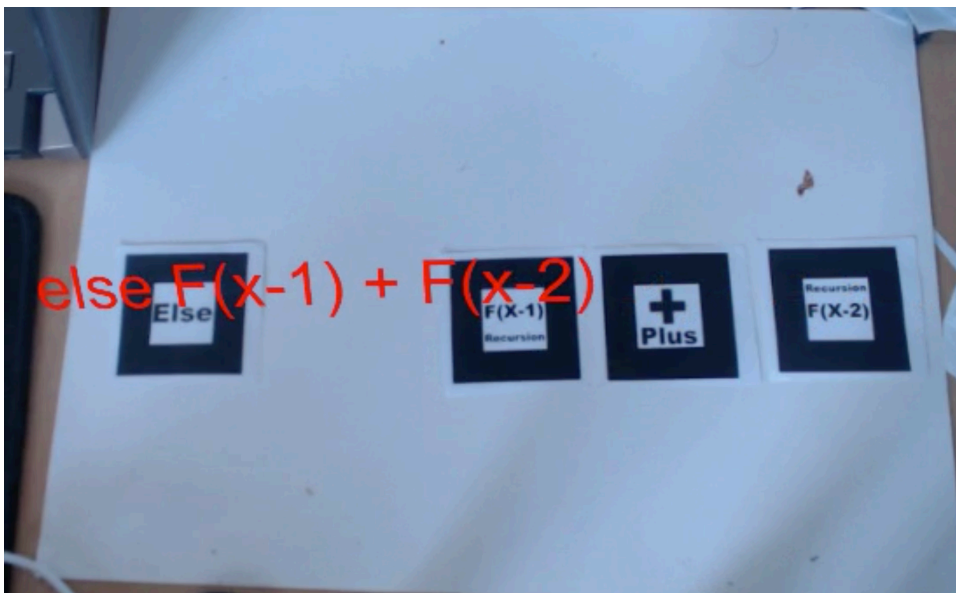


Figure 42 Else marker definition

After definition of these markers, we can combine if marker, then marker and else marker to define Fibonacci in function marker. The defined function will show on the defined

function marker [Figure 43].

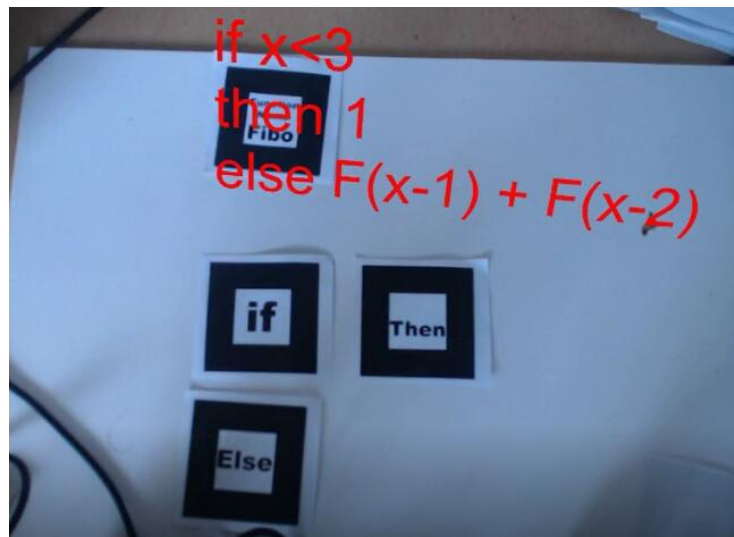


Figure 43 Fibonacci Function marker definition

Now we define the Fibonacci function on function marker.

### ***Fibonacci Function Execution***

After definition, we have the defined Fibonacci function marker, and we can execute it for calculation of the Fibonacci sequence.

We can execute the defined function marker with a series of markers [Figure 44]. We will use a constant marker to assign value for variable X. And value of variable Y will be defined as Fibonacci(X).

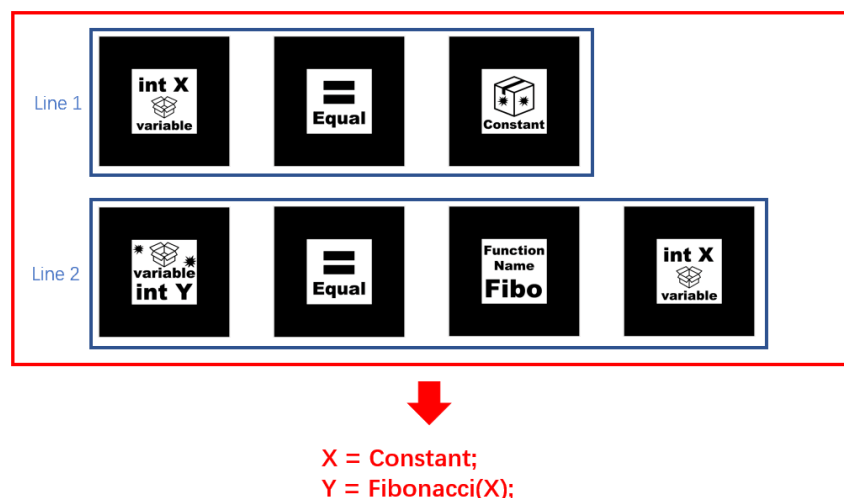


Figure 44 Markers for visual programming with Fibonacci function marker

Executing the markers sequence with executing gesture [Figure 45], we can get the executed outcome. The value of variable X and variable Y will show on their markers [Figure 46]. We set constant value as 10. Therefore, after execution, the value of variable X will be 10, and value of variable Y will be Fibonacci (10), which means 55.

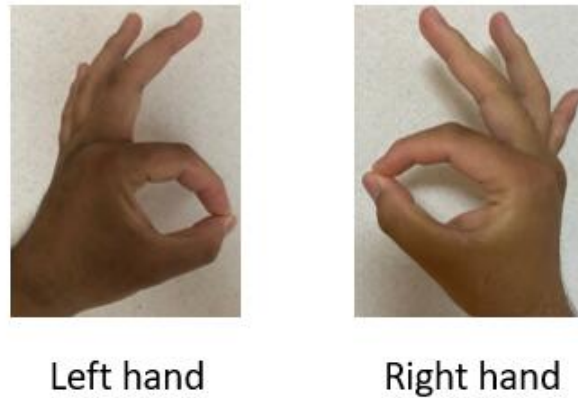


Figure 45 Executing gesture for markers sequence



Figure 46 Executed outcome of markers sequence

### 5.7.5 Visual Programming – Controlling AR Content

We can combine grammar structure and action markers to make the visualization program, which can be executed to control AR content to do some actions.

#### *Combining Grammar Markers and Action Markers*

We can combine grammar markers and action markers as a marker sequence to control

AR content. For example, we can combine loop structure and running marker [Figure 47]. Combining them, we can make the controlled AR content do running action for x times as a loop.

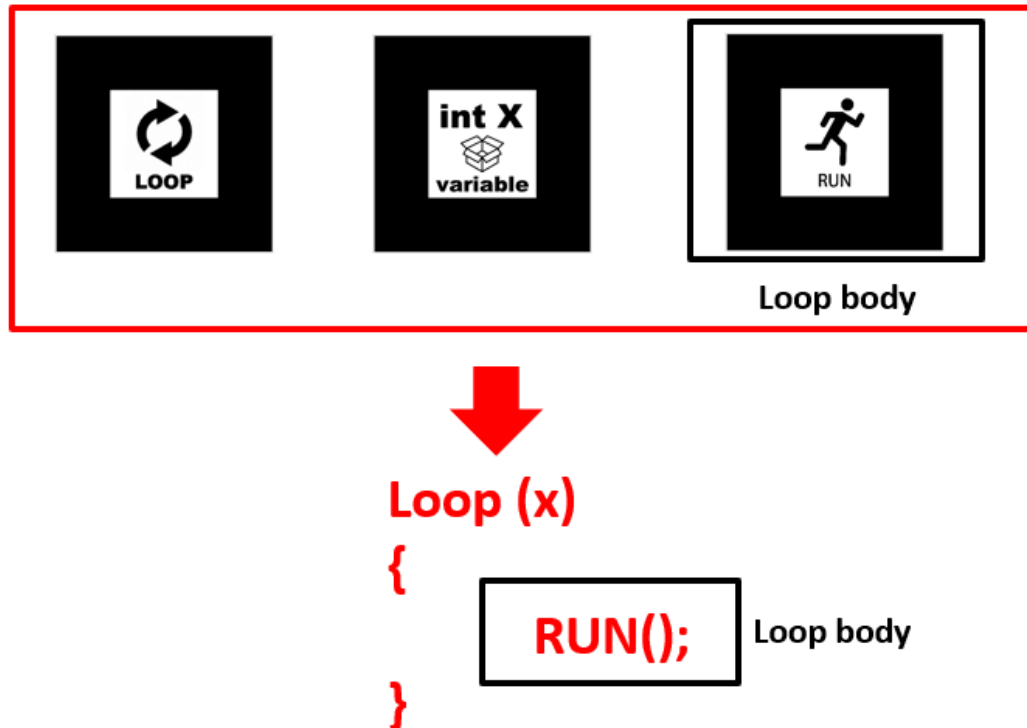


Figure 47 Combining action marker and grammar structure

We can use the loop marker to replace these markers after definition.

We can also combine other grammar structure in our visual programming with action markers.

### ***Executing Markers to Control AR Content***

After combination and definition, we can combine multiple defined grammar markers to define a function marker. Executing the function marker with AR marker which will generate an AR content for controlling, we can control AR content do a series of actions one by one as we defined in the visualization program.

### ***Working State of Execution***

We have 2 working modes of execution.



When the system works in normal mode, the AR content will do actions as the user defines. AR content will do each action for one second until he finishes all actions.

When the system works in debug state, the AR content will do actions one by one as definition. After finishing an action, AR content will stop actions and wait for next step gesture [Figure 48]. When the system detects that the user makes the next step gesture, AR content will continue next action in actions flow we have defined.

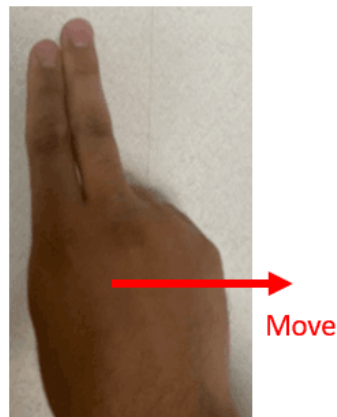


Figure 48 Next step gesture



# Chapter6

## System Implementation

### 6.1 System Hardware

#### 6.1.1 Web Camera and Tripod

We used Logicoool HD Pro Webcam c920r [Figure 49] as the Web Camera to help detect and recognize markers in our system. And We used Velbon EX-Mini S [Figure 50] as the tripod to fix the web camera for a stable camera view.



Figure 49 Logicoool HD Pro Webcam c920r



Figure 50 Velbon EX-Mini S

### 6.1.2 PC

We used PC as the development platform. In our system, we used TOSHIBA Dynabook [Figure 51] as PC.



Figure 51 TOSHIBA PC

### 6.1.3 Leap Motion

We used Leap Motion [Figure 52] as the depth sensor to detect user's hands data. Leap Motion can track user's hands and fingers position and transfer these data to our system which will help us to recognize user's hand gesture.



Figure 52 Leap Motion

## 6.2 Development Environment

The development operation system is Windows 10 Pro.

The software we mainly used is Unity 3D Engine, which will provide a support for AR development.

We mainly used C# Programming Language as the script language to realize the system. For markers recognition, we used Vuforia Engine, ARToolkit 5 SDK, ArUco SDK and ZXing API.

To use Leap Motion in our system, we need Leap Motion Orion 4.0.0 and Leap Motion Core Asset Package as the software support [22].

For hand gesture recognition, we used SVM for classification. In our system, we used Accord.NET Framework API [23] to help us recognize user's hand gesture.

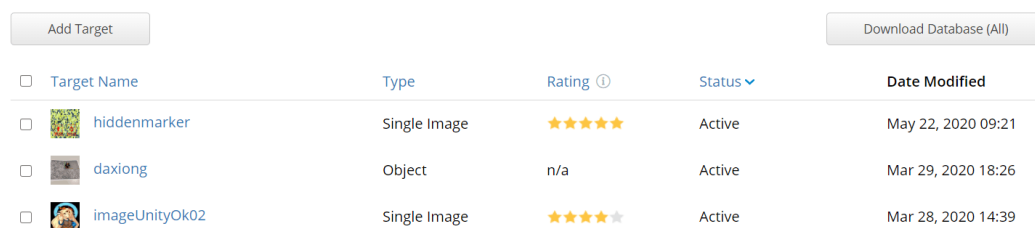
## 6.3 Multiple Markers Recognition

In our system, we will use multiple kinds and numbers of markers. We need to recognize them and combine their information together for further interactions.

### 6.3.1 NFT (Nature Feature Tracking) Marker Recognition

We use Vuforia SDK to help recognize and track NFT markers.

To recognize NFT marker, we firstly need to register markers information in the Vuforia Target Manager [Figure 53].



The screenshot shows the Vuforia Target Manager interface. At the top left is an 'Add Target' button, and at the top right is a 'Download Database (All)' button. Below these is a table with the following columns: Target Name, Type, Rating (with a help icon), Status (with a dropdown arrow), and Date Modified. The table contains three entries:




<input type="checkbox"/> Target Name	Type	Rating ⓘ	Status ▾	Date Modified
<input type="checkbox"/>  hiddenmarker	Single Image	★★★★★	Active	May 22, 2020 09:21
<input type="checkbox"/>  daxiong	Object	n/a	Active	Mar 29, 2020 18:26
<input type="checkbox"/>  imageUnityOk02	Single Image	★★★★☆	Active	Mar 28, 2020 14:39

Figure 53 Vuforia Target Manager

Vuforia Target Manager will rate for the uploaded marker images. If the image has enough nature feature points for tracking, the system will give it 4 or 5 stars rating, which means the marker will be detect and tracked well when the system is running.

After registering marker's information, we can download the markers information

database and import it to Unity 3D Engine [Figure 54].

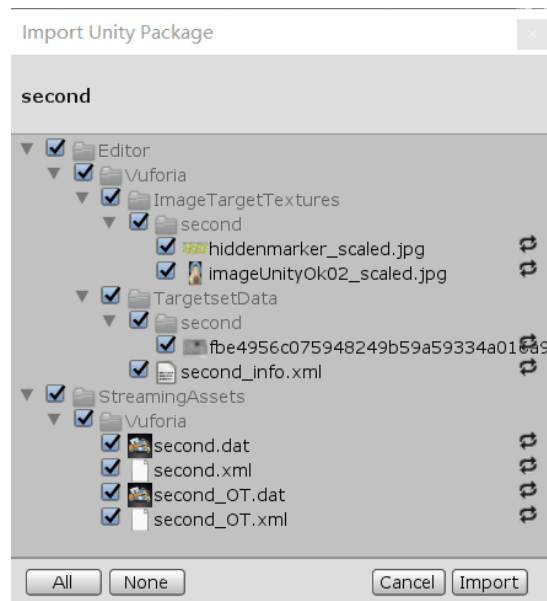


Figure 54 Importing Vuforia database to Unity

As a next step, we need make some configurations in Unity [Figure 55]. After that, we can recognize NFT markers and attach some AR content on it [Figure 56].



Figure 55 NFT markers configurations in Unity

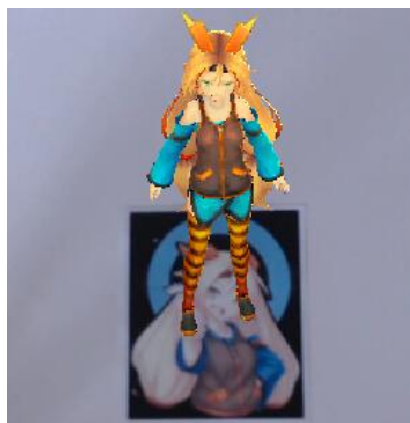


Figure 56 AR content on an NFT marker

### 6.3.2 3D Object Marker Recognition

In our system, we also use Vuforia SDK to recognize 3D object marker.

As a first step, we need an android smartphone as the scanner to scan and log the nature feature points of a 3D object [Figure 57], which we will use as the marker.

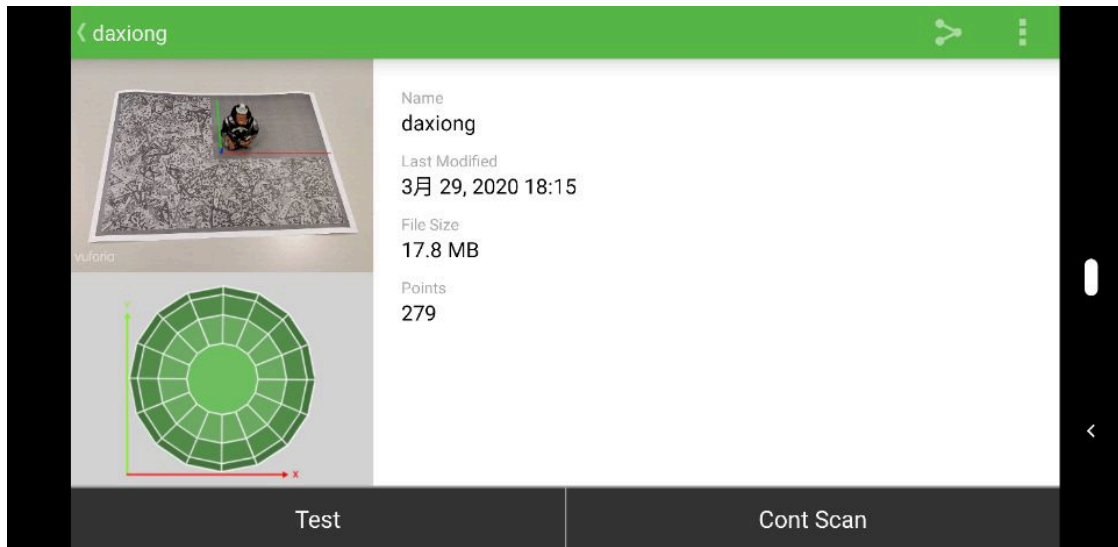


Figure 57 3D object scanning information in a smartphone

After scanning the feature points of the object, we need to upload the scanning data to Vuforia Target Manager. This step is same to NFT markers recognition.

After uploading and processing, we can download the Vuforia database and import it to Unity 3D Engine, and after some configurations, we can recognize a 3D object marker and show some AR content around it [Figure 58].

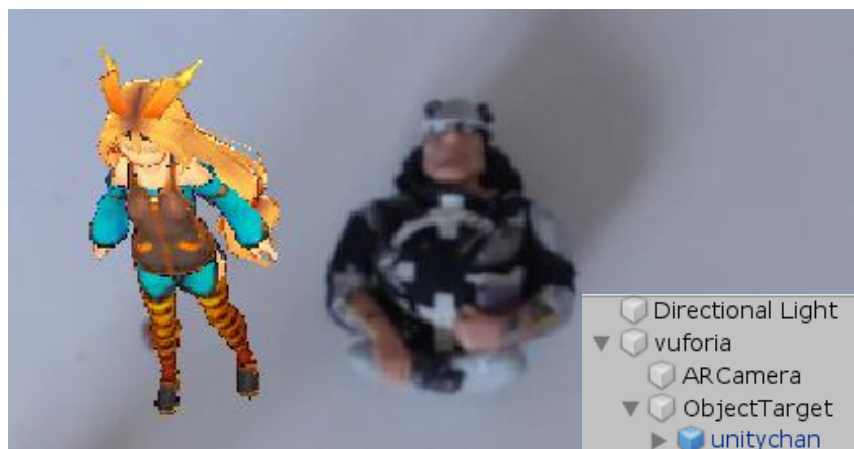


Figure 58 AR content around an 3D object marker

### 6.3.3 ATK (ARToolkit) Marker Recognition

In our system, we use ARToolkit SDK to help recognize and track ATK marker.

As a first step, we need to register an ATK marker in the ATK markers database [Figure 59]. The ATK markers recognition information will be saved as a .txt file.

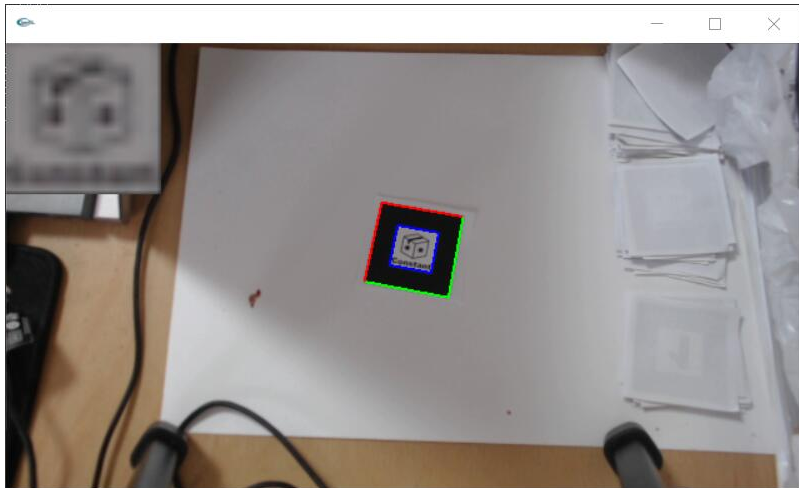


Figure 59 Register an ATK marker in the database

After some configurations in Unity, we can recognize and track ATK marker. We can just use ATK marker to save some information, and we can also use it to show some AR content [Figure 60] for interactions.



Figure 60 AR number on an ATK marker



### 6.3.4 ArUco Marker Recognition

In our system, we use ArUco SDK to help recognize and track it.

After importing ArUco SDK to Unity 3D Engine, we can recognize and track ArUco markers. We can use ArUco marker to save some information, and we can also show some AR content on it [Figure 61].

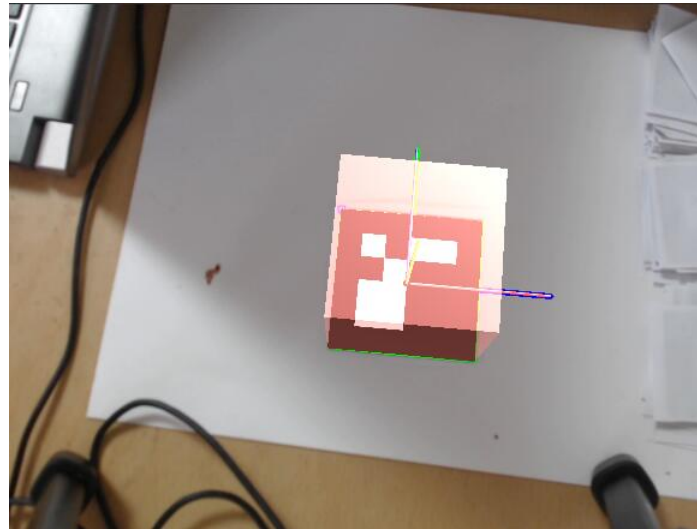


Figure 61 Red AR cube on an ArUco marker

### 6.3.5 QR Code Marker Recognition

In our system, we can recognize QR code [Figure 62] through Web camera with the help of ZXing API [Figure 63].



Figure 62 Recognized information from QR code shown on UI

```

using UnityEngine;
using System.Collections;
using System;
using Vuforia;
using System.Threading;
using ZXing;
using ZXing.QrCode;
using ZXing.Common;
using UnityEngine.SceneManagement;
using static Vuforia.Image;
using UnityEngine.UI;

```

Figure 63 ZXing API used in the recognition C# script

### 6.3.6 Hide Marker Configuration

We will use hide marker for some input in our system. With the help of Vuforia SDK, we can recognize hide marker and make some interactions.

Firstly, we need to add some virtual button area on the NFT marker [Figure 64] with the Virtual Button Behavior script [Figure 65].

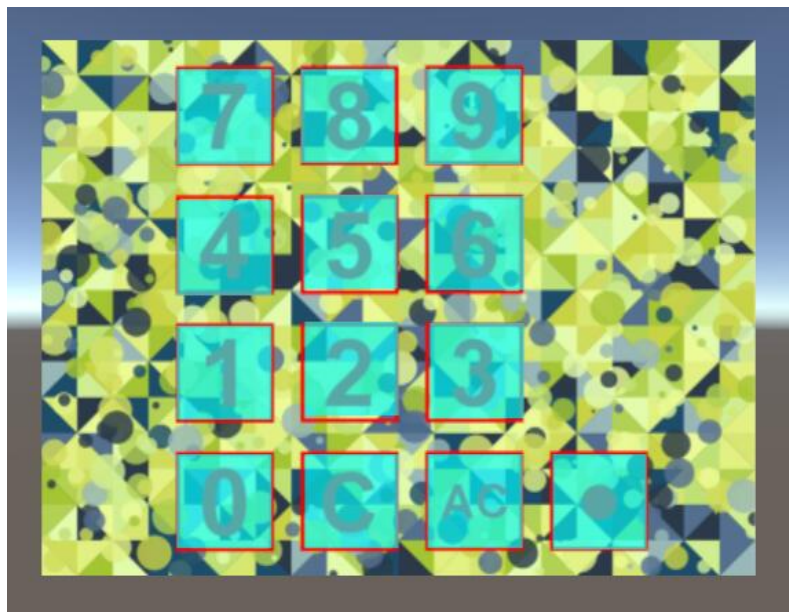


Figure 64 Virtual Button Area on NFT marker

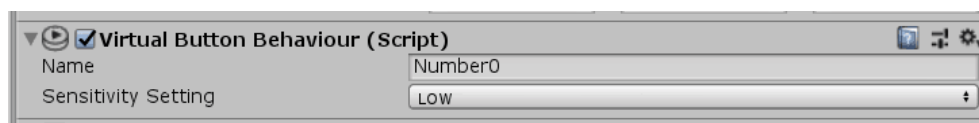


Figure 65 Virtual Button Behavior script



Then, we need to assign trigger event for each virtual button area. For numbers input, we set the trigger event as inputting number to the system.

After trigger event configuration, we can use hide marker as the input marker in our system. Through Web camera, we can touch its virtual button area to trig the event for input in the system.

Hide marker also can be combined with other markers for input and some interactions.

### 6.3.7 Multiple Markers Combination

Using different kinds of SDK or API, we can recognize multiple kinds of markers respectively. And as a next step, we can combine multiple kinds and numbers of markers information together in one scenery for some further use and interactions.

Firstly, we need to import different kinds of SDK in Unity 3D Engine. And after some configurations [Figure 66], we can combine multiple markers information together. Based on the combined information, we can make some interactions or some further applications.

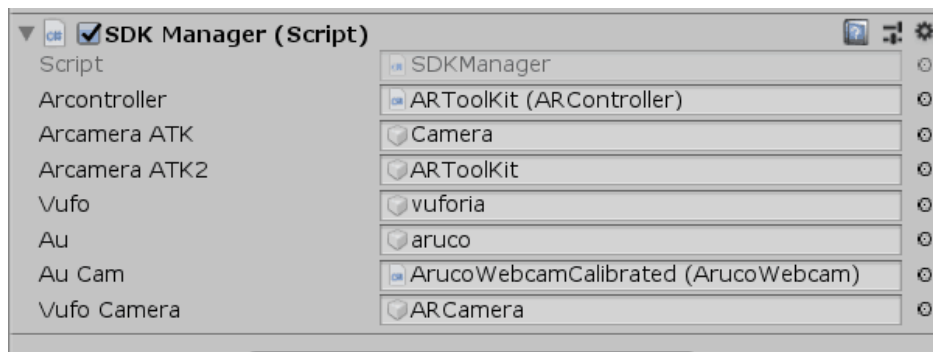


Figure 66 C# Script to combine multiple markers information together

## 6.4 Hand Gesture Recognition

In our system, we use Leap Motion as the depth sensor to track user's hand data. Leap Motion can track fingers, palm and wrist position data of hands in every frame. We can access these data and use them for hand gesture recognition.

To recognize user's hand gesture, firstly we need to classify user's hand shape. And as a next step, we need to access some hand motion data. Combining hand shape and hand motion data, we can decide user's hand gesture.

### 6.4.1 Environment Configurations

To use Leap Motion in Unity 3D Engine, we need some configurations.

As a first step, we need to download and install Leap Motion Orion 4.0.0, which can drive Leap Motion to work with PC.

And after that, we need to import Leap Motion Core Asset Package to Unity 3D Engine, which will make it possible to get hand data in real-time from Leap Motion in Unity.

After configurations, we can see our virtual hands in Unity Scenery [Figure 67]. The virtual hands have same hand shapes and motions to our real hands.

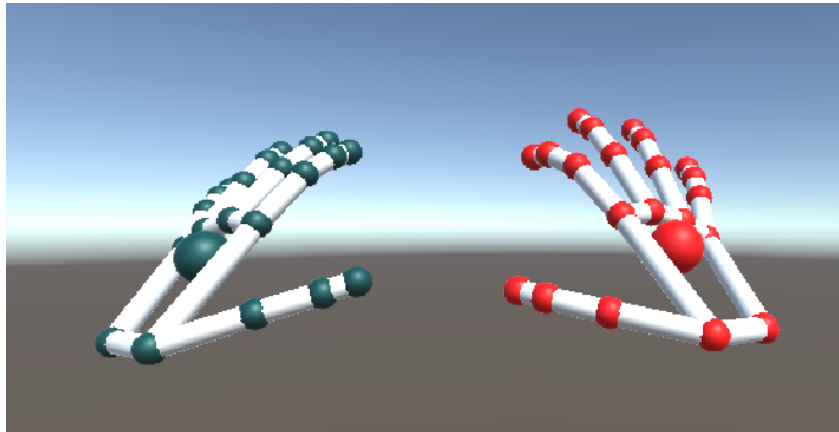


Figure 67 Virtual hands in Unity 3D Scenery

### 6.4.2 Hand Shape Recognition

In our system, we have static hand gesture and dynamic hand gesture, and we also have single-hand gesture and double-hand gesture. To recognize different kinds of hand gesture, as the first step, we need to recognize and classify user's hand shape, which represents the static characteristics of user's hands. In our system, we have mainly 7 kinds of hand shapes [Figure 68]. They are the basis of hand gestures we will use in our system.

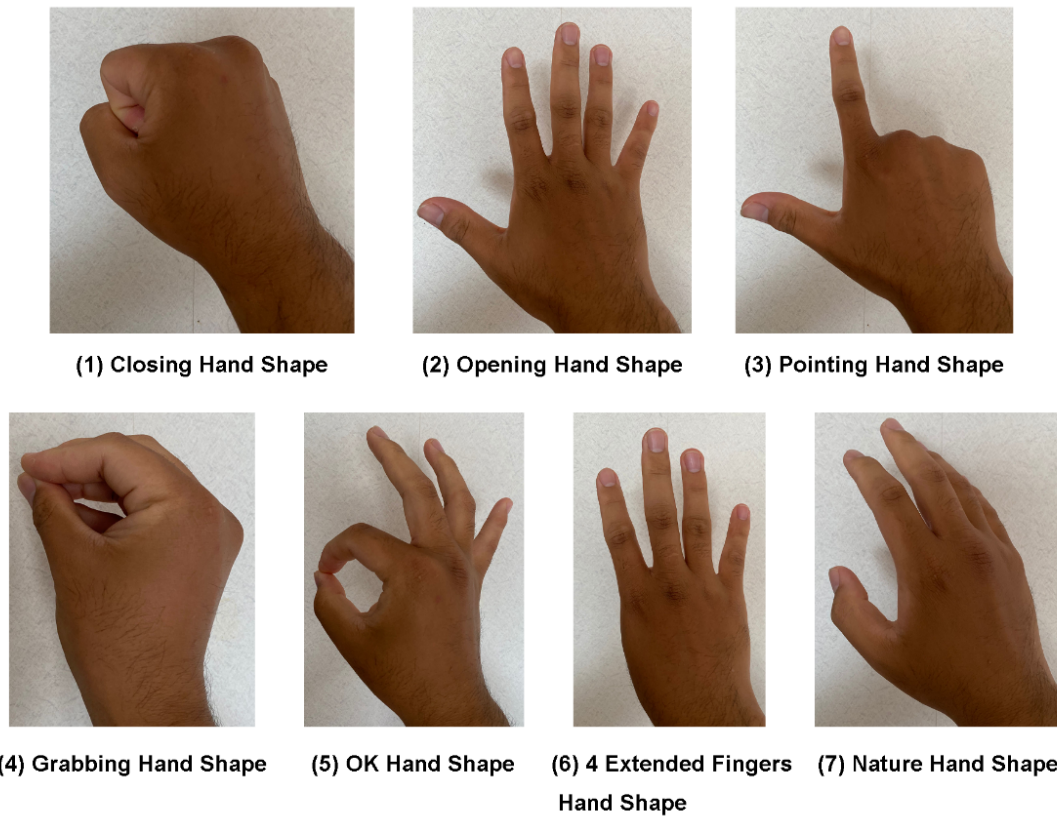


Figure 68 Hand shapes in the system

Here are some descriptions of these hand shapes features, which will serve for the hand shape recognition in the following work.

**(1) Closing Hand Shape**

In this state, the user's five fingers are tightly closed.

**(2) Opening Hand Shape**

In this state, the user's five fingers are spread out.

**(3) Pointing Hand Shape**

In this state, the user's thumb and index finger will stretch out.

**(4) Grabbing Hand Shape**

In this state, the five fingertips of the user will gather at the same point and make the shape of grabbing an object.

**(5) OK Hand Shape**

In this state, the user's thumb and index finger will gather at one point and make an

"OK" action.

(6) 4 Extended Fingers Hand Shape

In this state, the user's four fingers except the thumb will stretch out.

(7) Nature Hand Shape

In this state, the user's hand is naturally relaxed. The five fingers are neither fully extended nor tightly closed.

In our system, we adapt Multi-Class SVM to classify and recognize these different kinds of hand shapes. In order to get better compatibility with Unity 3D, we apply Accord.NET Framework to provide some API for SVM learning and using in our system.

There are some steps for multi-class classification:

(1) Data Collection

(2) Data Processing and Normalization

(3) SVM Model Training

(4) Hand Shape Prediction

***Data Collection***

Leap Motion can provide some position data of user's hand. In this step, we need to confirm what kinds of hand data we will need for hand shape recognition and classification.

After analyzing different kinds of hand shapes that we need to classify, we choose five fingertips position, hand palm position and wrist position on hand for use [Figure 69]. By combining these points position information, we can distinguish these different hand shapes.

We can access these positions information directly from Leap Motion API.

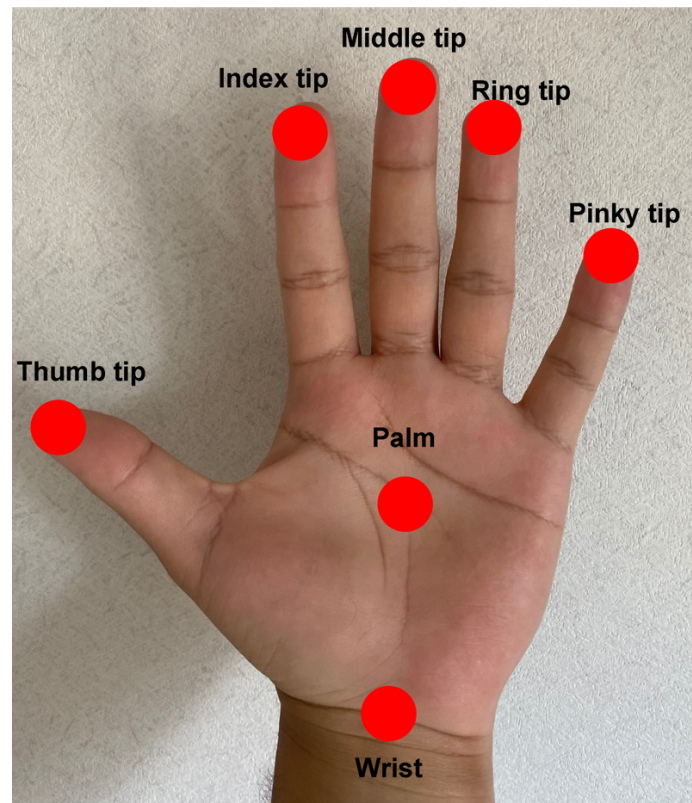


Figure 69 Fingertips, palm and wrist position on hand

### ***Data Processing and Normalization***

In this step, we need to process and normalize hand information data we have got.

Firstly, we need to process position information of fingertips, palm and wrist. To describe features from these data, we will calculate the distance between the five fingertips to the palm of the hand, the distance between the five fingertips to the wrist, and the distance between the other four fingertips to the thumb. These distance data will be used for SVM Modeling and Predicting.

After processing, we need to scale all distance data to  $[0, 1]$  for normalization.

### ***SVM Model Training***

In this step, we need to train a Multi-Class SVM model using the processed and normalized data. The trained model can be used to predict which hand shape the newly entered hand data belongs to.

The system will read the hand shape data for training and perform SVM learning. After training, we can get a trained model that can be used for hand shape classification.

In our system, we prepare 100 groups hand data information for each hand shape. These data will be used for SVM model training.

### ***Hand Shape Prediction***

After getting the trained SVM model, we can use it for hand shape recognition and classification.

We can access hand data in real time from leap motion, and then the system will process and normalize these hand data. The processed data will serve as the input for hand shape classification. And the system will classify which hand shape the input hand data belongs to based on the trained model.

Now, we can recognize and classify the user's hand shape [Figure 70].

For each hand shape, we tested 100 times based on the trained model. And the recognition accuracy rate is higher than 93%, which shows that our hand shape recognition method is effective.

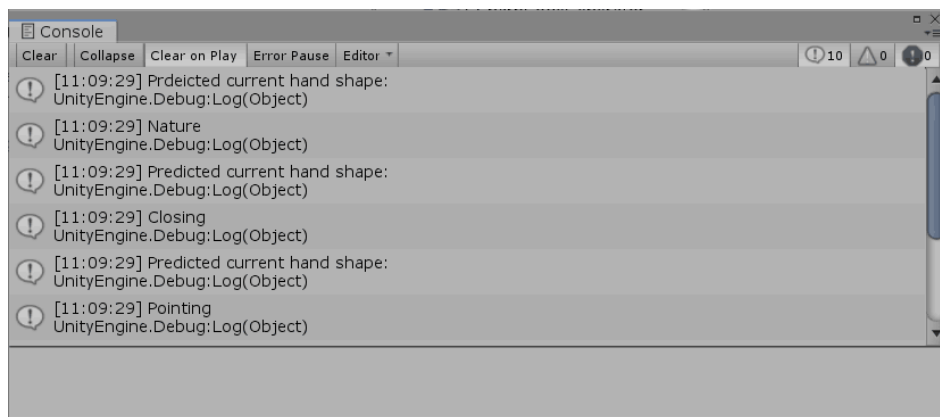


Figure 70 Hand shape prediction outcome shown on Console window

### **6.4.3 Hand Motion Detection**

With SVM, we can classify and recognize user's static hand shape. As a next step for hand gesture recognition, we need some motion data of user's hand to analyze the dynamic features.

Leap Motion SDK can provide some interfaces for us to access user's hand motion data every frame when Leap Motion Controller is connected to the PC and works normally [Figure 71].

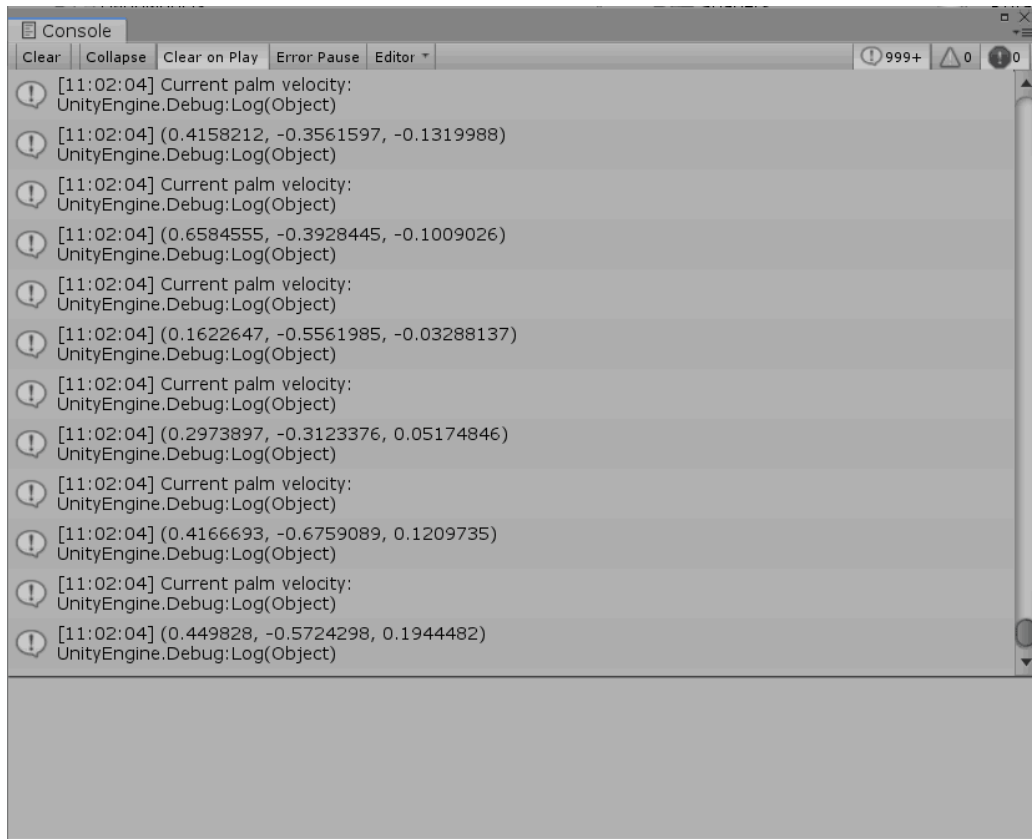


Figure 71 Current palm velocity data shown on Console window

From Leap Motion, we can access hand's palm velocity, palm direction, finger velocity and finger direction, which will help us define and classify different kinds of hand gestures. These motion data will also support some interactions between AR content and hand gesture.

The hand motion data provided by Leap Motion is based on real hand motion. If we want to use these data for interactions with AR content in Unity scenery, we need to process and adjust these data for different interactions.

After accessing and processing hand motion data, we can combine them with hand shape classification for hand gesture recognition.

#### 6.4.4 Hand Gesture Recognition

We define a hand gesture with a hand shape and some hand motion data. Therefore, we can combine recognized hand shape and real-time hand motion data, we can recognize user's hand gesture in real time.

For different kinds of hand gesture, there will be different decision condition. If the current recognized hand shape and hand motion data satisfy its decision condition, it will be recognized as the corresponding hand gesture.

For example, we will define swiping left hand gesture with open hand shape and the palm velocity larger than 0.7m/s from right to left. If user's hand satisfies this condition, the system will recognize user's hand as swiping left hand gesture.

A special hand gesture is selecting hand gesture.

When user make the selecting gesture, the finger ray will generate from user's index fingertip of virtual hand [Figure 72]. Use can use finger ray to make some interactions with AR content. And in our system, it is mainly used to select some AR content for further interactions.

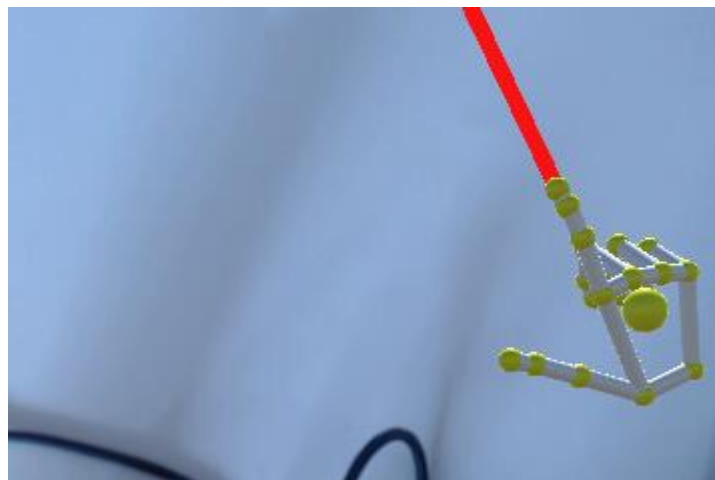


Figure 72 Finger ray from virtual hand

To realize this hand gesture, we need to recognize it at first.

If the system recognizes that user makes the pointing hand shape, user's hand gesture will be recognized as selecting hand gesture.

As a next step, we will access user's index fingertip direction from Leap Motion.

After that, we will use user's index fingertip position as the start point of finger ray and use user's index fingertip direction as the direction of finger ray. With these hand data, we can draw the finger ray.

If the finger ray collides with some AR content in the scenery, the system will give some response. With the response, we can decide which AR content is selected by user.



With hand shape recognition and hand motion detection, we can realize hand gesture recognition. With hand gesture recognition, we can make some interactions with AR content or the system.

## 6.5 Visual Programming with Multiple Markers and Hand Gesture

### 6.5.1 Markers Sequence Recognition

In our system, we will use a series of markers for visual programming. And markers sequence represents logics of visualization program. Therefore, we need to recognize markers with defined sequence.

In our system, we define the markers sequence from left to right, and then, line by line from up to down.

As the first step, we need to recognize all markers in the scenery and access their coordinate positions in 2-dimension from Unity Scene [Figure 73].

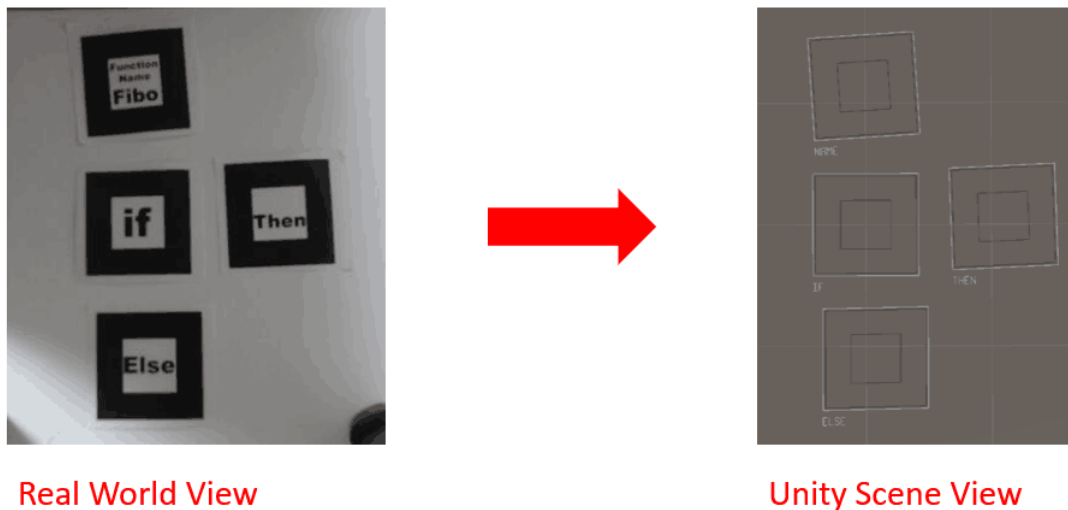


Figure 73 ATK markers position in Unity scene view

Next, we will use their 2-dimension coordinates to get their sequence as we define.

In this step, because we can't accurately place the two markers on the same x coordinate or y coordinate, we need to set some threshold for coordinate position to decide whether

2 markers are in same line or same list.

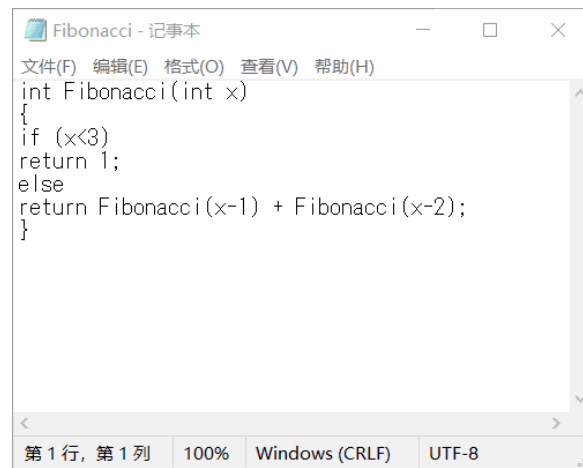
After that, we will get the 2-dimension markers sequence for processing.

### 6.5.2 Execution of Visualization Function

With markers sequence, we can define the visualization function program. As the next step, we need to execute the defined function.

We use the execution of Fibonacci function as the example.

Firstly, we need to transform markers sequence to meaningful and executable code. We will save the code in a local .txt file [Figure 74].



```
int Fibonacci(int x)
{
    if (x<3)
        return 1;
    else
        return Fibonacci(x-1) + Fibonacci(x-2);
}
```

Figure 74 Executed code saved in .txt file

When we want to use the saved codes, we can read the defined function codes in .txt file. And using C# reflection, we can create the dynamic class with the defined function and use it when the project scenery is running. With this, we can use the defined visualization functions when executing it.

## 6.6 Virtual Marker

We will use some virtual markers working with real markers for visual programming. Virtual markers will have same outlook with real markers, and they will form markers sequence together.

### 6.6.1 Virtual Marker Modeling

We need to model virtual marker and use them in our system.

To make virtual marker in our system, we need to attach real markers picture on virtual card [Figure 75].

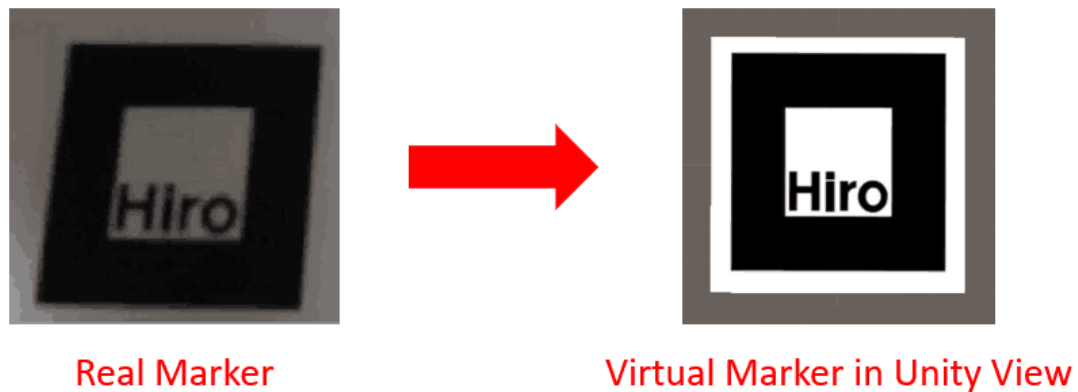


Figure 75 Real Marker and its corresponding virtual marker

After that, we need to save prepared virtual markers models in the prefab file in the project document for use.

### 6.6.2 Generating Virtual Marker in Scenery

We want to use virtual markers together with real markers. Therefore, we need to generate virtual markers in AR scenery.

After virtual markers modeling, we have virtual markers models in the prefab file. We will check the state of existing real markers in the scenery. If the system detects that a real marker is touched, it will log the position of the touched marker and generate the corresponding virtual marker on the same position.

### 6.6.3 Manipulations of Virtual Marker

We can manipulate virtual markers in our system.

With hand gesture recognition, we can realize hand gesture we have defined about virtual markers.

Combining the selected virtual marker with the user's real-time gesture information, we can realize the manipulation of virtual markers.

#### **6.6.4 Visual Programming with Virtual Markers and Real Markers**

We can get markers sequence from multiple virtual markers and real markers based on the position information of these markers in the AR scenery. We can get the information from Unity Engine.

After that, markers sequence information will be used for visual programming.

# Chapter7

## Preliminary Evaluation

We invited 10 people to experience our system and applications. After that, we asked every participant to finish a questionnaire for evaluation.

### 7.1 Questionnaire

We designed a questionnaire for participants [Figure 76].

Questionnaire

Name:

Gender:

Date:

Age:

Questions

Question 1-8 are based on 5-point scale.

Answer the following questions by marking the most appropriate answer in your own perspective.

1. The system is easy to use.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

2. The system is easy to learn how to use.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

3. Visualization program is easy to understand.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

4. Hand Gesture interactions are nature and intuitive.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

5. Hide marker is convenient as controller for virtual object.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

6. It is easy to interact with AR contents in the system.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

7. Virtual marker is useful for visual programming.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

8. Applications of our system are interesting.

Strongly Disagree   ☐ --- ☐ --- ☐ --- ☐ --- ☐   Strongly Agree

9. Please give some comments of our system, how do you like it, which part you think can be improved?

Figure 76 Questionnaire for participants

In this questionnaire, we use 5-points scale as the scoring standard for evaluation. Q1 and Q2 are used to measure whether users can easily get started with our system. Q3 is used to measure whether our visual programming is explicit and easy to understand. Q4 is used to evaluate hand gesture interactions in our system. Q5 is used to evaluate hide marker controller. Q6 is used for evaluation of interactions with virtual objects. Q7 is used to evaluate whether introducing virtual markers to our system will improve the visual programming part. Q8 is for applications. Q9 is for some comments from participants. We asked every participant to finish the questionnaire after they experience our system and applications.

## 7.2 Result and Analysis

We invited 10 people to experience our system and applications. After that, we asked every participant to finish a questionnaire for evaluation.

We calculated the average of the scores for every question from the participants [Figure 77].

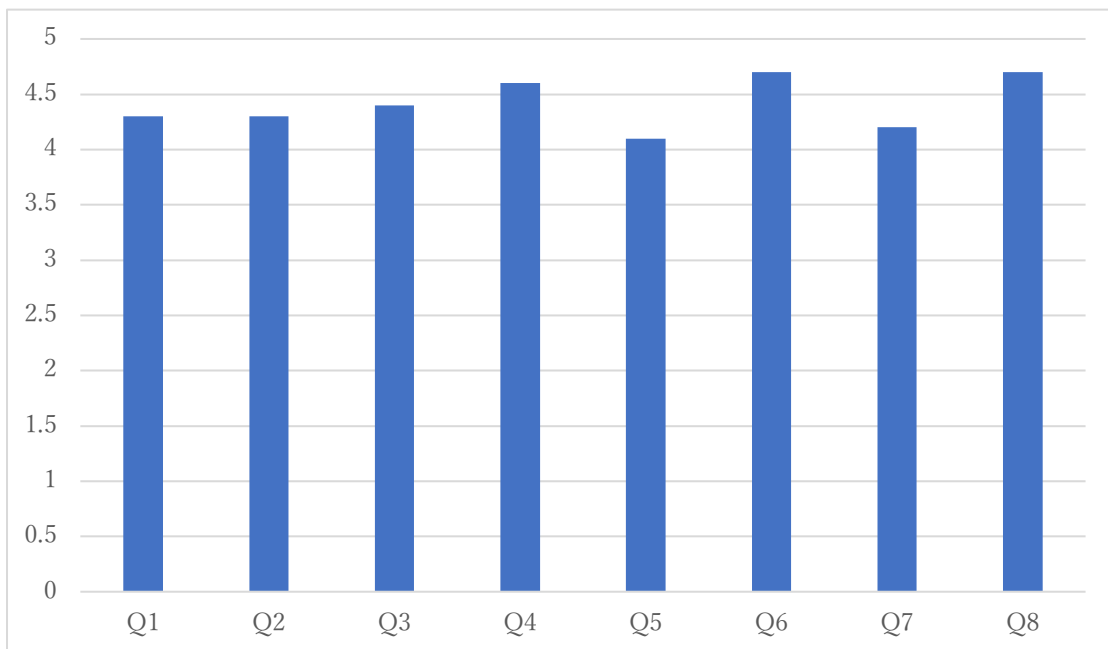


Figure 77 Results from participants

From the results, Participants gave a relatively good evaluation of our system.

From results of Q1 and Q2, we can see that participants think our system is easy to

understand and use. From results of Q3, we can see that participants think visual programming of our system is explicit and easy to understand. From results of Q4, we can see that participants think hand gesture interactions in our system is intuitive and nature. From results of Q5, we can see that participants think hide marker controller is a good way of interaction. From results of Q6, we can see that interactions with virtual object is easy to use for participants. From results of Q7, we can see that virtual markers can improve visual programming part of our system. And from results of Q8, we can see that participants think applications of our system is interesting and appealing.

Participants also give some comments to our system.

In their comments, they think our system is interesting and useful. However, there are some points still need to be improved.

1. Some participants think that hand gesture set can be enriched for more interactions.
2. Some participants think that visual programming part can be simplified to make the visualization program clear.
3. Some participants think that grammar structure of visual programming part can be enriched for more functions.

Nearly every participant gave good evaluation for interactions with virtual object in our system in their comments.

## 7.3 Summary

Generally, participants think our system is interesting and useful. And they give a good evaluation for interactions in our system, especially for hand gesture.

Our system can be improved in some ways. Hand gesture set can be improved for more interactions. Visual programming part can be improved for more functions. And more interesting applications can be made based on our system.

# Chapter8

## Conclusion and Future Work

### 8.1 Conclusion

In this research, we combine multiple kinds and numbers of markers and hand gesture together for some interactions and visual programming in marker-based AR system. And based on the framework, we make some applications.

For markers, we combine multiple kinds and numbers of markers information together in the system and use the combined information for some interactions and visual programming. Different kinds of markers have different attributions. Combining multiple kinds of markers, we can make relatively complex functions than just using one kind of marker.

For hand gesture, we use Leap Motion as the sensor to get user's hand information in real time. After some processing and analysis, we can decide user's hand gesture. With hand gesture recognition, we can detect user's hand gesture in real time. And based on different kinds of hand gesture, we can design different kinds of interactions. Finally, user can make some interactions with AR content or the system using his hands. By introducing hand gesture in marker-based AR system, interactions between user and system will be natural and intuitive.

Combining multiple markers and hand gesture, we can design some interactions with virtual content and the system.

For visual programming, we use multiple markers with some sequence to represent some visualization program logics and use hand gesture to trigger some operations such as executing the program or save the function. After multiple markers recognition and information processing, we can execute the visualization program and get outcome.

We make some applications with the framework. We build a virtual timer with multiple markers and use hand gesture to control the timer. We use hide marker to control some



AR content and use hand gesture to pair input and output markers. We can also use hand gesture to control AR content directly. With visual programming, we can calculate Fibonacci sequence and controlling AR content with a series of markers.

With our framework, there will be more intuitive and natural interactions in marker-based AR system. More functions and applications will also be possible.

## 8.2 Future Work

### *More Kinds of Markers*

Different kinds of markers have their own attributions. If we can combine more markers in the system, it is possible to make more interactions and functions.

### *Improving Hand Gesture*

Hand gesture design should be natural and intuitive for users. We still need some evaluations and feedback to improve current hand gesture set.

### *More Grammar Structures for Visual Programming*

Currently we have loop structure, if-then-else branch structure and function structure in visual programming part. However, as a kind of programming language, there can be more grammar structures for more functions and high-level visualization program. With more grammar structures, the visual programming capability of our system will be extended.

### *More Possible Applications*

Based on our framework, we can make more interesting applications.

Current applications also have some possibilities to modify for a better user experience.

# Reference

- [1] P. Milgram and F. Kishino. A Taxonomy of Mixed Reality Visual Displays. IEICE Transactions on Information and Systems, vol. E77- D, no. 12, 1994, pp. 1321-1329.
- [2] Anuroop Katiyar, Karan Kalra and Chetan Garg. Marker Based Augmented Reality. Proceedings of Computer Science and Information Technology (ACSIT), pp. 441-445, 2015.
- [3] Ivo Aluizio Stinghen Filho, Estevam Nicolas Chen, Jucimar Maia da Silva Junior and Ricardo da Silva Barboza. Gesture Recognition Using Leap Motion: A Comparison Between Machine Learning Algorithms. SIGGRAPH '18: ACM SIGGRAPH 2018 Posters, August 2018, Article No.: 65, Pages 1–2.
- [4] Madhavi Gangurde. Augmented Reality. ICWET '11: Proceedings of the International Conference & Workshop on Emerging Trends in Technology, February 2011, Pages 1363.
- [5] David Chen, William Garrett, Mark Livingston, Andrei State, Gentaro Hirota. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 429–438, 1996.
- [6] Chor-Kheng Lim, Ming-Chih Huang and Fang-Yu Chen. Application AR in field experience education: development of teaching aids in Chinese literature and taoyuan local culture. ICETC '18: Proceedings of the 10th International Conference on Education Technology and Computers, October 2018, Pages 3–6.
- [7] Scott G. Dacko. Enabling smart retail settings via mobile augmented reality shopping apps. Technological Forecasting and Social Change, Volume 124, November 2017, Pages 243-256.
- [8] Gérald Moulis, Alexandre Bouchet. A collaborative approach of augmented reality for maritime domain. VRIC '12: Proceedings of the 2012 Virtual Reality International Conference, March 2012, Article No.: 6, Pages 1–2.
- [9] Jana Pejoska. Social augmented reality app. AcademicMindTrek '15: Proceedings of the 19th International Academic Mindtrek Conference, September 2015, Pages 215–216.
- [10] Amir H. Behzadan and Vineet R. Kamat. Visualization of construction graphics in outdoor augmented reality. WSC '05: Proceedings of the 37th conference on Winter simulation, December 2005, Pages 1914–1920.

- [11] Juri Platonov, Hauke Heibel, Peter Meier and Bert Grollmann. A mobile markerless AR system for maintenance and repair. ISMAR '06: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, October 2006, Pages 105–108.
- [12] Yuna Kang and Soonhung Han. Improvement of smartphone interface using an AR marker. VRCAI '12: Proceedings of the 11th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry, December 2012, Pages 13–16.
- [13] Yang Li, Jin Huang, Feng Tian, Hong-An Wang and Guo-Zhong Dai. Gesture interaction in virtual reality. *Virtual Reality & Intelligent Hardware*, Volume 1, Issue 1, February 2019, Pages 84-112.
- [14] Ram Pratap Sharma, Gyanendra K. Verma. Human Computer Interaction using Hand Gesture. *Procedia Computer Science*, Volume 54, 2015, Pages 721-727.
- [15] Shengyu Fang. “Enhancing User Interactions by Combining AR Markers with Various Kinds of Markers”, Master thesis, Graduate School of Information, Production and Systems, WASEDA University, 2020-02.
- [16] Kazuki Tada and Jiro Tanaka. Tangible programming environment using paper cards as command objects. 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015), July 2015, Pages 4365-4372.
- [17] Chunmeng Lu. “A virtual shopping system based on room-scale Virtual Reality”, Master thesis, Graduate School of Information, Production and Systems, WASEDA University, 2018-07.
- [18] Wei Zeng., Cong Wang and Qinhui Wang. Hand gesture recognition using Leap Motion via deterministic learning. *Multimedia Tools and Applications* 77, 28185–28206 (2018).
- [19] Stafford Michahial. Hand gesture recognition using support vector machine. *The International Journal Of Engineering And Science (IJES)*, June 2015, Volume 4, Issue 6, Pages 42-46.
- [20] Ivo Aluízio Stinghen Filho, Estevam Nicolas Chen, Jucimar Maia da Silva Junior and Ricardo da Silva Barboza. Gesture recognition using leap motion: a comparison between machine learning algorithms. *SIGGRAPH '18: ACM SIGGRAPH 2018 Posters*, August 2018, Article No.: 65, Pages 1–2.
- [21] Rose S P., Habgood J M. P. and Jay T. An Exploration of the Role of Visual Programming Tools in the Development of Young Children’s Computational Thinking. *The Electronic Journal of e-Learning*, Volume 15, Issue 4, 2017, Pages 297-309.

- [22] Leap Motion Developer. <https://developer.leapmotion.com/setup/desktop>
- [23] Accord.NET Framework. <http://accord-framework.net/>