Title	Development of visual programming environment for virtual reality application
Sub Title	
Author	張, 純(Zhang, Chun) 小木, 哲朗(Ogi, Tetsurō)
Publisher	慶應義塾大学大学院システムデザイン・マネジメント研究科
Publication year	2020
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2020年度システムエンジニアリング学 第313号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40002001-00002020-0004

Master's Dissertation 2020

Development of Visual Programming Environment for Virtual Reality Application

Chun Zhang
（Student ID Number：81834548）

Supervisor Tetsuro Ogi

July 2020

Graduate School of System Design and Management,
Keio University
Major in System Design and Management

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81834548	Name	Chun Zhang

Title

Development of Visual Programming Environment for Virtual Reality Application

Abstract

Recent Virtual Reality technology implement is a slice of the field due to the technology provide an interactive three-dimensional computer-generated environment, which can help characters better understand the work in a realistic simulated environment. Given the limitation of the available VR program platform, it still arduous for nonprogrammers to develop a VR project independently. Therefore, we developed a visual programming environment that user can build their own VR project in a three-dimensional setting without code, and it is easily used for nonprogrammers, which provide a simplified and logical programming method. It allowed the user to realize the functions by match the object with the function with the VR controller and headset in the VR environment. The ultimate goal of the research is to allow the non-experts to utilize VR to help them study or work efficiently.

Key Word (5 words)

VR; Visual Programming; Three-Dimension Interactive; User Interface; Immersive Experience

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Research Background

1.1.1 Virtual Reality Background

VR is the abbreviation of Virtual Reality, which is a computer simulation technology that can create and experience a virtual world[1]. It uses a computer to generate an interactive three-dimensional dynamic scene, and its physical behavior simulates. The system can immerse users in the environment[2]. With economic globalization, VR applies in many aspects, such as education, entertainment, and electronic shopping. All in all, VR has great potential and far-reaching research directions in the future. And with the development and improvement of VR technology, VR can provide the user with a better experience.

Virtual worlds are persistent virtual environments in which people experience others as

being there with them - and where they can interact with them[3]. VR technology integrates various scientific techniques such as computer graphics technology, computer simulation technology, sensor technology, display technology, etc. It creates a virtual information environment on the multi-dimensional information space, which can make users feel immersive. Therefore, immersion, interaction, and imagination are the three primary characteristics of the VR environment system. The core of virtual technology is modeling and simulation.

Regardless of whether it is a cultural relic in a museum or a historical relic that has preserved so far, the information currently delivered to the audience is incomplete. VR/AR technology can make up for this regret, restore as much traditional culture as possible. VR is a crucial frontier direction for a new generation of information technology. VR technology integrates technologies in multiple fields such as multimedia, sensors, new displays, the Internet, and artificial intelligence. It expects to become a primary platform for many innovative applications, spawn many new products, new formats, and new models, which might be leading a new round of technology and industry change. After more than ten years of development, VR/AR technology is gradually maturing. At present, VR/AR technology has steadily grown internationally, and the global VR industry ecology has taken initial shape.

VR technology implement is a slice of the field due to the technology provide an interactive three-dimensional computer-generated environment, which can help characters better understand the work in a realistic simulated environment. Given the limitation of the available VR program platform, it still arduous for nonprogrammers to develop a VR project independently.

1.1.2 The Difficulty of VR Application to Non-programmer

In the beginning, I found that this problem exists because there is a group of people who want to use VR, but because of technical limitations, they have to

give up this option. Therefore, I hope to seize the needs of this group of people, so that even if you do not have professional skills, you can use VR well. For example, if students or teachers of non-programming related majors want to use VR technology to help their research or work, they can change the way of image and let beginners or people without foundation can also use VR quickly.

It is arduous for nonprogrammer to develop a VR project-related their work or study. The conventional way for developing a VR project requiring developers to master at least one programming language and use VR engine, but for the people who do not have related knowledge such as English teachers, it a burdensome work to develop a VR demo by themselves. Consequently, how to make it easy and time-saving for a person without a programming background to use VR to help them develop a VR application used in their work? The problem is too significant to solve by myself, so I want to narrow the scope of the problem and change it to how to help a teacher without programming background use VR to help them develop a VR application used in teaching?

1.2. Research Purpose

The purpose of my research is to make users without programming experience to develop VR projects by themselves in a convenient, time-saving, visible way. So the system should provide a simple program method that immerses users in the VR environment and can create an object and write function by VR controller

instead of learning a new programming language from the beginning. It allows the user without programming experience can efficiently and effectively use VR technology to help their work or research.

How to provide a three-dimension and virtual reality environment for nonprogrammers to develop virtual reality applications? The virtual reality technology is implemented in many fields and has full applications in medical, educational, scientific and technological, gaming, and many other areas. The virtual reality system uses computer simulation to create a three-dimensional space of the virtual world, allowing users to immerse themselves in the virtual environment, and have an immersive experience. However, it is still arduous for nonprogrammer to develop a VR project-related their work or study. The conventional way for developing a VR project requiring developers to master at least one programming language and use VR engine, but for the people who do not have related knowledge such as English teachers, it a burdensome work to develop a VR demo by themselves. Consequently, how to make it easy for people who are not good at programming to build VR applications by themselves?

To realize this goal, we first developed a VR programming environment to immerse users in the three-dimensional environment, which is for users to understand better VR and how to create VR. Because users can directly develop their project in the three-dimension climate and see what they create in this virtual world, thus users skip the step of development in a two-dimension

environment and then execute the plan in three-dimension the environment. In other words, users need not transform the code or functions in the two-dimension environment to the result in the three-dimension environment. Besides, it will spend too much time on nonprogrammers learning a new programming language. Therefore, it is indispensable to provide a programming method using easily understood language rather than machine language. The original programming method needs to consider the equipment used in the VR environment as well.

However, when I started my research, I found out that if my target user is someone who doesn't know how to program but wants to do VR development, this research goal is too big for me to start, so I narrowed my research scope. Among the people who want to develop VR but don't have a programming foundation, I have locked one of the them-language education teachers. First of all, language education teachers generally do not have a programming background. Secondly, they have corresponding needs in class and need to use VR technology to assist their work. So I said that my research object is to locate the teacher. At the same time, the problem I hope to solve is also narrowed down to how to enable language education teachers to quickly use VR technology to develop a VR scene to help them use in class?

Overall, the objective lies in providing a simple program method that immerses users in the VR environment and can create an object and write function by VR controller instead of learning a new programming language from the beginning.

But because the scope of this research is too broad, I narrowed the scope of the study to how to enable teachers in the education industry who want to use VR to help them improve their classrooms simply and quickly use VR for development.

1.3. Dissertation Structure

This dissertation includes six chapters, followed by acknowledgment reference and appendix.

Chapter 1 is an overview of this dissertation. In this section, the research background is discussed, and the problem that VR development hard for nonprogrammers is found based on the current situation of VR after that led to the purpose of research.

Chapter 2 analyzes two related studies and compares my research direction with their similarities and differences. The research direction is more explicit.

Chapter 3 contains the requirement analysis of this system and the system structure. The requirements analyze based on the stakeholder analysis of the system. From the elements, the system structure can be designed.

Chapter 4 is the main body of this dissertation. In this chapter, the development process and the functions designed are explained. It shows how users use this

system to create a VR program in the VR environment with headset and controllers instead of a keyboard and mouse.

Chapter 5 is a validation part of the 3D menu and analyzes the questionnaire's data in a statistic way.

Chapter 6 is the last section concluding my research and then talks about this project's future work.

Chapter 2. Related Research

2.1. Programming in Virtual Reality

In October of 2014, Brian released a video of Rift Sketch: a programming interface that uses the Three.js library to render real-time models in virtual reality. Not only can you fully code in VR, but you could simply look to your left and see 3D objects performing live beside the text input window[4].

Another one is Primitive, a startup that pitched at an HTC venture capital accelerator this week in San Francisco. Primitive represents one of the most exciting use-cases I've seen for VR so far—it creates 3D visualizations of source code that can be collaboratively explored and analyzed in VR. It's creators believe that bringing a spatial understanding to otherwise flat code enhances the development process of complex code created by distributed development teams[5].

However, these two research of programming in VR still require user coding, which means that even if the programming environment is changed to a virtual environment, programming is again using code to program. For those without a programming background, they still need to re-learn how to program to use, if they want to do VR program development.

What are the challenges to using VR technology for the design and development of VR-based instructional activities, and what are the recommended approaches? There is a research address the issues regarding identifying the appropriate techniques for integrating VR into traditional instructional design, and the considerations for development for non-technical educators[6]. It spends a lot of time designing in VR, so immerse users into the VR environment can make them better understand what VR is and how to develop their VR world.

In one study[7], they have proposed an operation method using a VR controller, and the estimated usability of our approach comparing with a technique based on hand gesture. As a result, they found that a VR controller had good usability in operations other than moving blocks. Therefore, the VR controller can be a useful interaction tool and programming tool in the VR environment.

From these studies, the VR is a more concrete and easy-to-understand programming environment, but when the programming environment changes, interaction methods and interactive tools will be affected. Therefore, if the traditional programming method is transferred to the VR environment, it will be challenging to operate and learn.

2.2. Visual Programming

In general, programmers describe some source codes using a text editor and compile them to get high performance of a computer. This coding style needs advanced knowledge of grammar and technique of programming. It takes a considerable amount of time to learn programming languages[8].

In the text-based programming languages such as C and Java, programmers generally develop software using a keyboard and text editors. On the other hand, in the environment of the visual programming languages, programmers develop the software by arranging blocks with various functions, mainly using mouse pointing and touch operation. This environment is suitable for beginners of programming to start learning because grammar errors that frequently occur in the text-based programming language are challenging to occur while developing a program[7].

Even there is research found a fun-learning approach to programming[11], it spends a long time studying a new programming language and the logic of how to program.

A 3D virtual space development environment for end-users[9] allows users to visualize and manipulate programs in 3D representation. That visual programming environment is expected to enhance motivation to programming, understanding of applications, and efficiency of programming education[10].

Innovations like the graphical user interface exposed essential elements like the filesystem to a broader audience, and the Internet has become increasingly

democratized as user-friendly tools like WordPress, Youtube, and Soundcloud allow anyone to create, publish and distribute content without writing a line of code[12]. Scratch is a visual programming language that you can use the block to program. Users of the site can create online projects using a block-like interface. Scratch is taught and used in after-school centers, schools, and colleges, as well as other public knowledge institutions. As of April 2020, community statistics on the language's official website show more than 52 million projects shared by over 54 million users, and almost 55 million monthly website visits[13].

Even small children can program, which proves that visual programming has deficient requirements for users and can be used without learning a programming language. Therefore, if visual programming is put in the environment of VR, it can satisfy users that even if they have not learned programming, they can still start to program in the visual language in the VR environment quickly.

Chapter 3. VR Programming Concept Model

3.1. Requirement Analysis

3.1.1 Stakeholder Analysis

In order to solve the target issue, it is essential to map all the stakeholders accordingly. In order to make a proper system of functions, the stakeholders should be taken into account; otherwise, the system can be perceived as incomplete. The different roles of stakeholders in the system should be emphasized to create an understanding of the operations. Also, various stakeholders have different priorities. A proper functioning system should make a difference in preferences into account. We specified the following stakeholders.

Table 3.1: Different Interaction Tools

Stakeholders	**Importance in system**
Teacher	Key user (Primary)
Student	Key user (Secondary)

Teacher: This primary role is given to the most prominent users of our system while the system is in operation.

Student: These stakeholders are the second most important in our system and play a critical role in supporting the primary users.

3.1.2 Stakeholder Requirements

First, I will analyze the needs of the entire system from the needs of the stakeholder. Requirements analysis is a vital activity in research and an essential part of the software development cycle. This stage is to analyze what the system needs to "implement" instead of considering how to implement it. Only when it is clear what the system needs to achieve, can we know the development direction and function realization of the entire system.

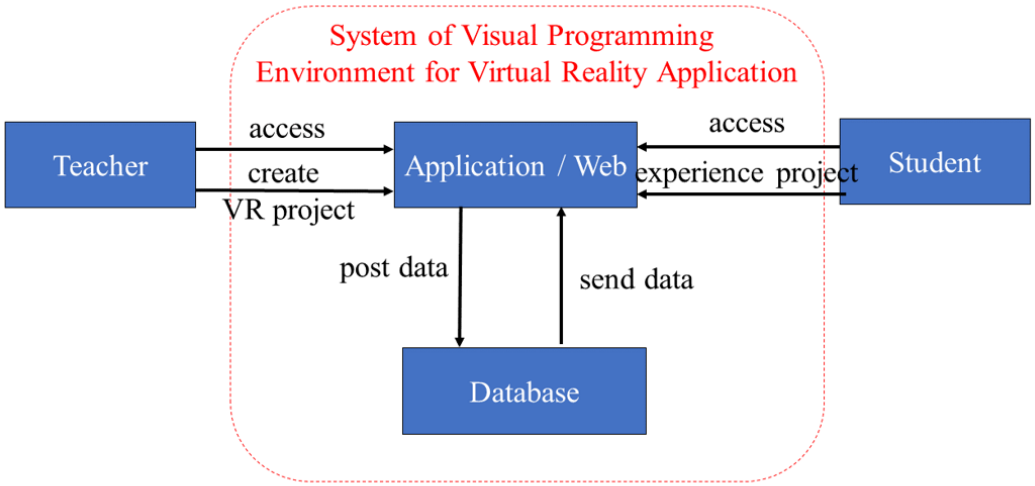

Figure 3.1: Concept Context Diagram

As you can see from the Concept Context Diagram (Figure 3.1), I did a highly generalized context analysis. The reason I do this is mainly to analyze the

relationship between stakeholders and this system, so I can weaken the internal part of the system and focus on the interaction between the user and the system. About the entire system, a detailed description will be given in the system structure chapter.

Teachers need to use this system, creating a VR project and share this project to students, so the teacher account has a higher level than the student account because it needs one more requirement of sharing. Students can access the system and experience the project that the teacher shares with them, but students can not create projects by themselves.

Table 3.2: Stakeholder Requirments

Stakeholders	**Requirments**
Teacher	1. Register teacher account 2. Login 3. Create a project 4. Share project with others 5. Log out
Student	1. Register student account 2. Login 3. Experience project 4. Log out

For teachers, there are five needs, while students have only four. It also shows that different users have different priorities. Teachers have higher priority than students so that they can use more functions of the system.

3.2. System Structure

After analyzing the relationship between the system and the user, I need to analyze the structure of the system and discuss how to design the system.

This system consists of three parts: VR Visualization Interface, Programming in VR Environment, Library. Next, the three parts will be introduced in sequence.

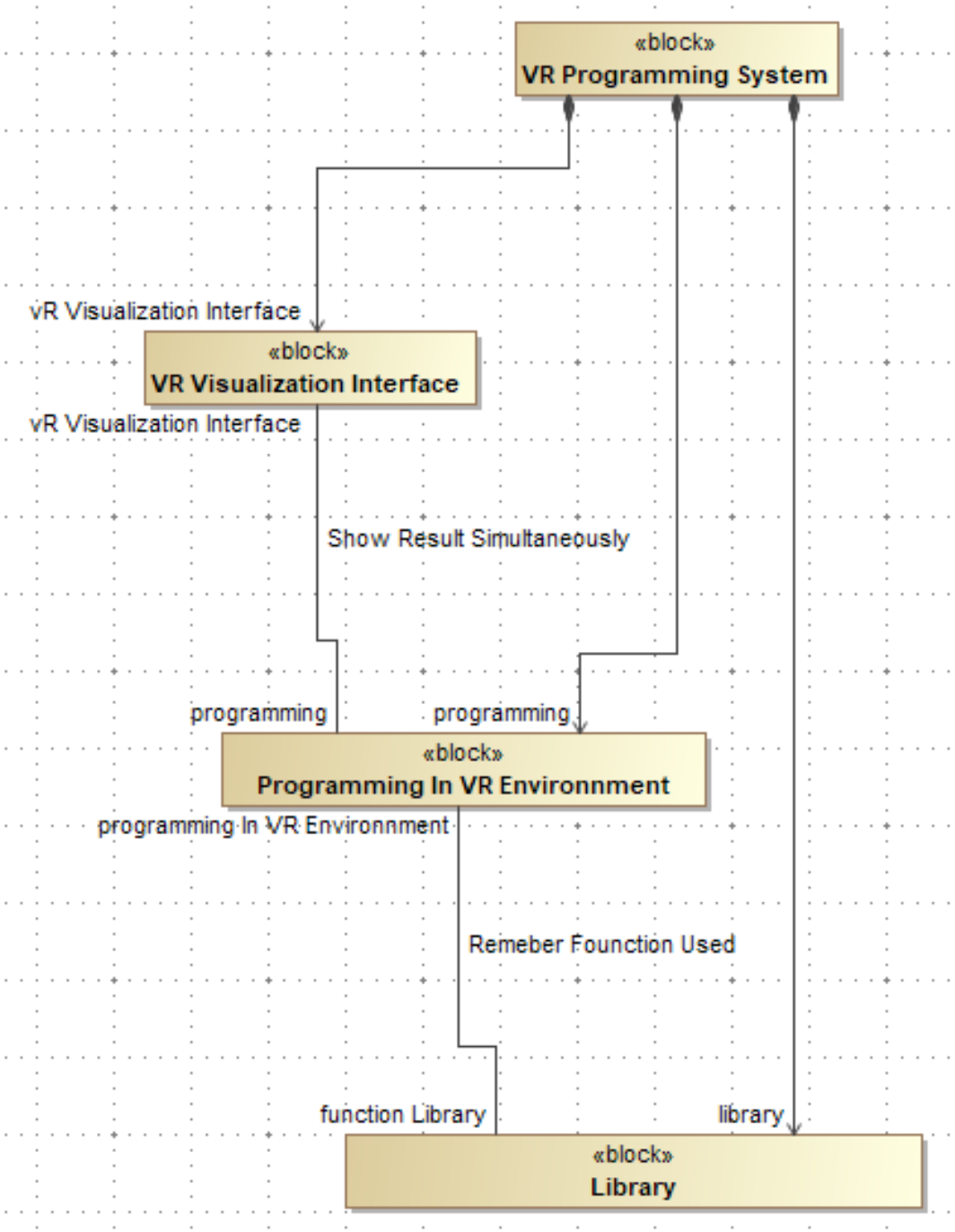

Figure 3.2: System Structure

- Library – ER model

The library is mainly used to store system data, such as user information, functions, and models.

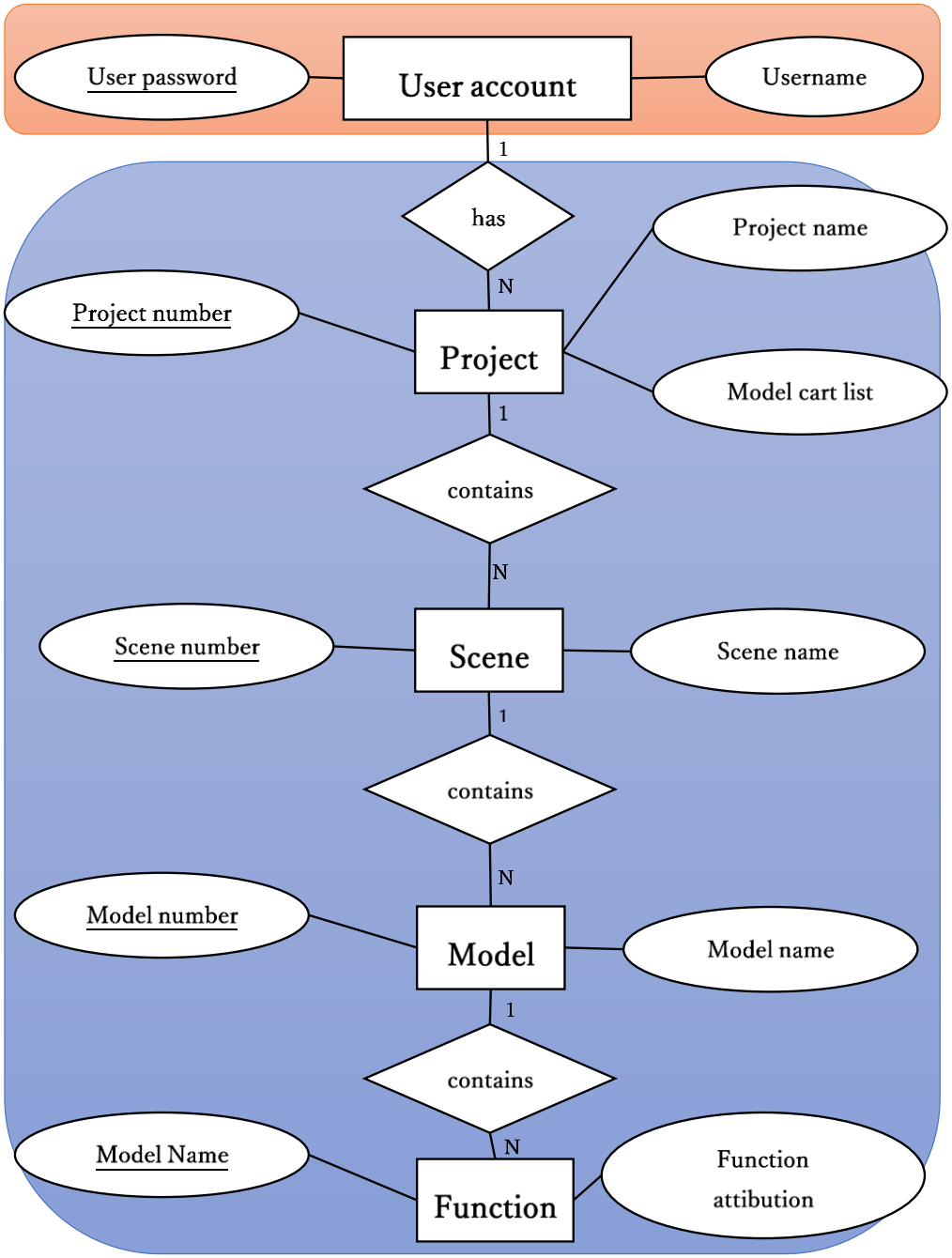

Figure 3.3: ER model

As can be seen from the ER diagram, the data storage is based on each user's account. An account can create multiple projects, a project can contain various scenes, a scene has numerous models, and multiple methods can be added to a model (Figure 3.3). The data of these items will be stored in the database of the

system. When the user wants to access, the user can log in to the account to access the program or share it with other users.

➢ Programming in VR Environment

In fact, in the entire system, the user is not entirely in the VR environment. When users log in and log out and register accounts, users use the mouse and keyboard to register traditionally. But when the project was started, the environment was already in a VR environment.

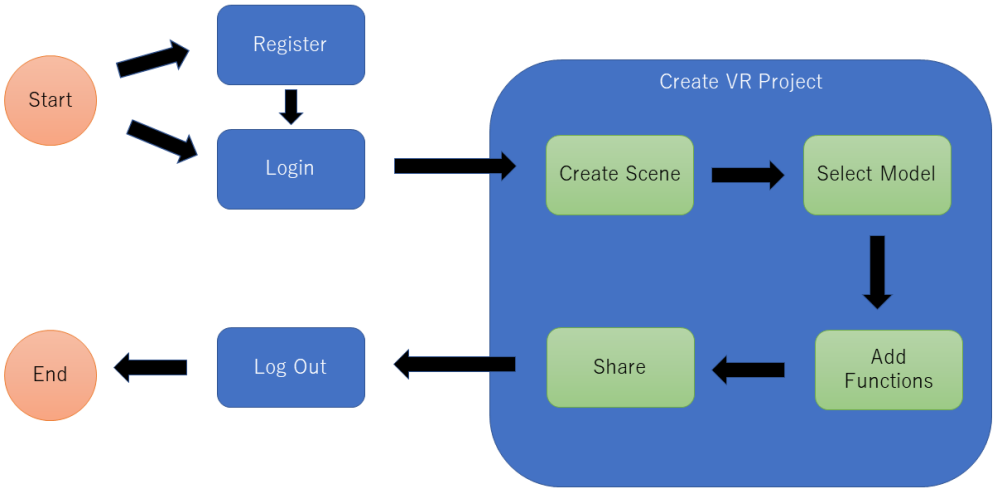

Figure 3.4: Function Flow Block Diagram

So, you can see from the FFBD that all the methods in the Create Project block are all in the VR environment, which also means that all other processes in the generous of the Create VR Project are programming parts.

And how is 3D programming implemented in VR? Selecting a type of object, and this type of object corresponds to specific properties. The properties correspond to particular functions.

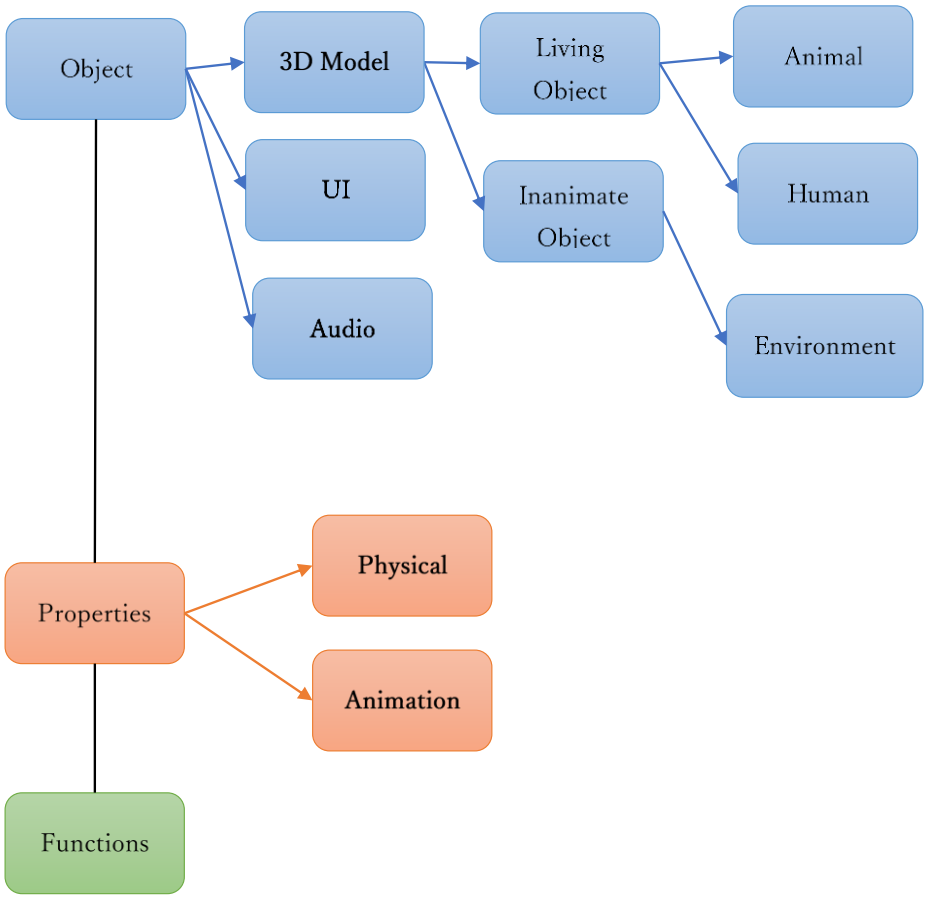

Figure 3.5: Object, Properties, and Functions

There are three kinds of objects that are the 3D model, UI, and audio. Moreover, there are two kinds of 3D models, and one is living objects like an animal or human model, anther one is the inanimate model. The reason to classify models is that different types of models need to add corresponding methods. For

example, if your model is a tree, you cannot add a means of speaking to the tree, because in general, the tree does not speak. Therefore, we need to classify the models here, the same as that, and there are different types of functions so that different methods can match the corresponding models.

The flow of how the user program in VR:

- ✧ Creating an object
- ✧ Adding properties to the object
- ✧ Linking functions in the library to the object
- ✧ Creating new functions and add to the object

➢ Visualization Interface

In VR programming, there are two concepts -model and function. The model is the object user want to use in their scenarios, such as a car or person model developer want to use in their program. Another one is a function that can add in the model to make the model can interact with the model or user.

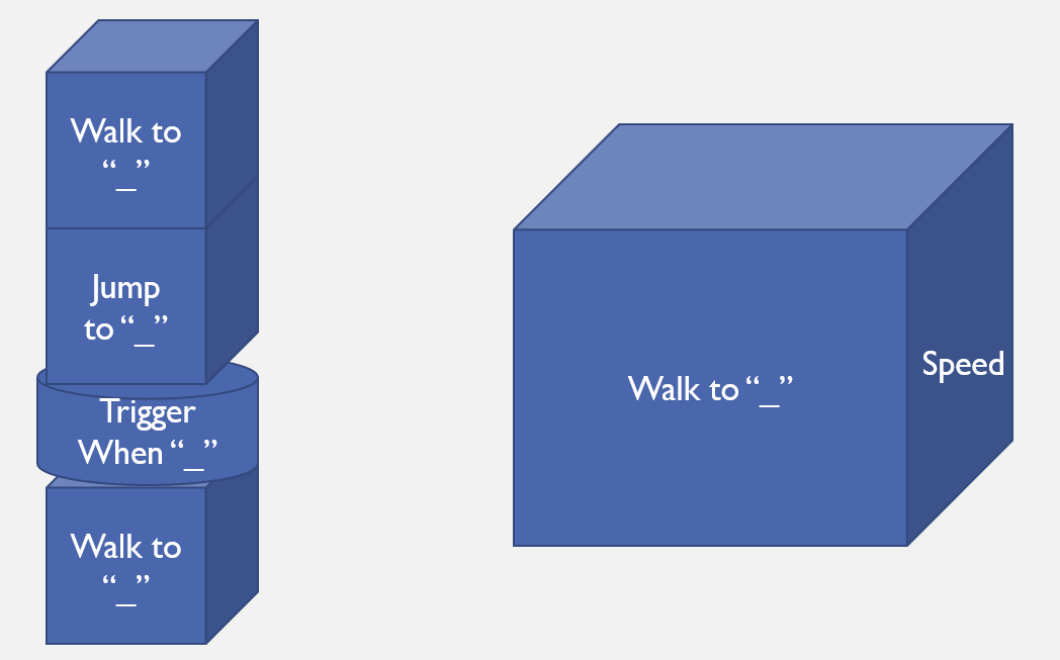

Figure 3.6: Inner Model Program

The inner model visual programming function refers to the Box Type. And the attribution could show in the different faces of the cube. For example, if the function box is walking, I can change the walking speed in another front of this function box.

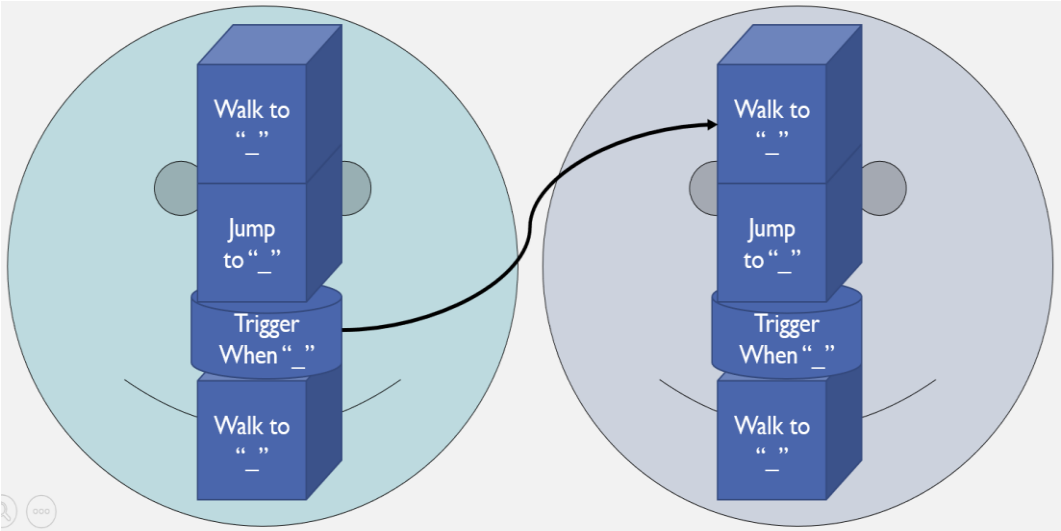

Figure 3.7: Between Models Program

The between objects' visual programming function refers to the Node and Line Type. The two simple faces present two models (Figure 3.7). If you want to model one model to execute the function after another one finished a task, you can connect the flow using a line. Thus, the Walking function would run after the Trigger finished.

Chapter 4. Project Introduction

4.1. Introduction

My application that can be used in the windows platform and users can create a small VR world without code in the VR environment. The application requires a VR headset and controllers because the whole development process is in the VR environment.

Users can use it to create a VR scenario. In the scene, users can firstly create a VR world by VR controllers and handset. There are environment models provided to select, after select environment model, the user would go to the scene and pick the character models. Then the user goes to a new stage that all the models selected in it. In this scene, users can add functions to the model one by one, and the function is also provided in the function library—the detail realization expressed in 4.3.

4.2. Development Tool and Platform

The development tool for this project uses Unity3D, and the final release platform is WebGL. The reason for choosing this tool and platform is because of considerations such as user needs and development difficulty.

When I first started this project, I chose to use Unity3D for development. Unity3D is a multi-platform comprehensive game development tool developed by Unity Technologies that allows players to easily create interactive content such as 3D video games, architectural visualizations, real-time 3D animations, etc. It is a fully integrated professional game engine. Unity is similar to software such as Director, Blender game engine, Virtools, or Torque Game Builder that uses interactive graphical development environments as the primary method. The most important is that Unity can provide VR development function, which is I needed. Therefore, in the early period of my project, I use Unity3D to be the development tool.

In the early period of my research, I planned to make a software that executes in the windows platform. Still, my supervisor Ogi suggest me to change making the computer software to the online page because of the would-be convenience for users. You need not download one more application on your computer, and you can freely use it by just search the web link. I try to develop web pages instead of the application. Then I change my development tool to A-frame.

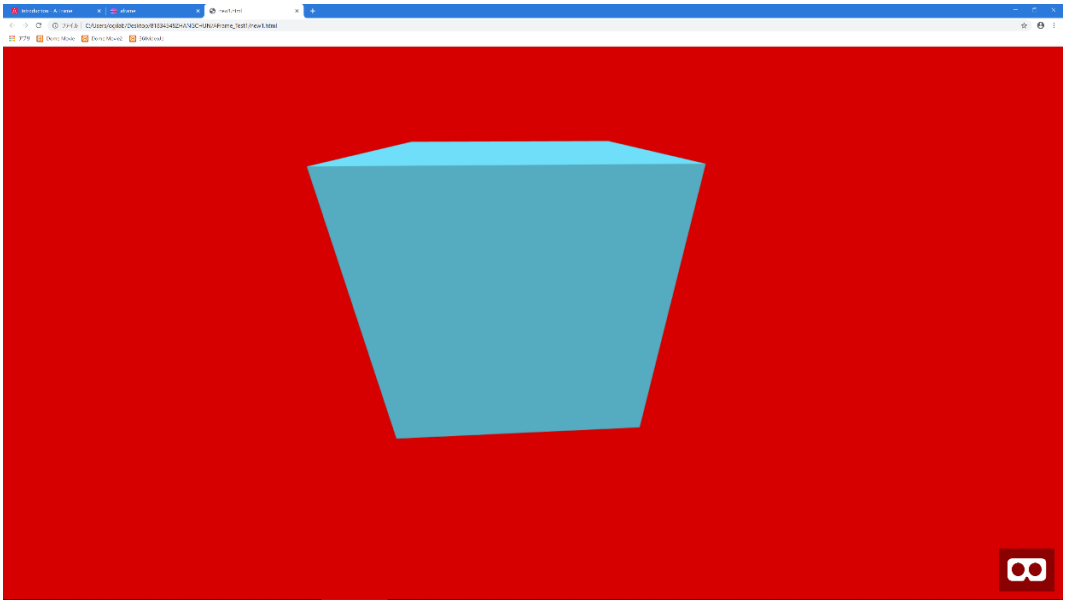

Figure 4.1: A-frame Developed Web Page

As you can see in the Web Page (Figure 4.1), it also allows the user to use VR headset and controllers on the web page so that users can use it by searching from the Internet, and it convenient for someone's first time to use it compare with computer software. Hence, I changed the development tool from Unity3D to A-frame.

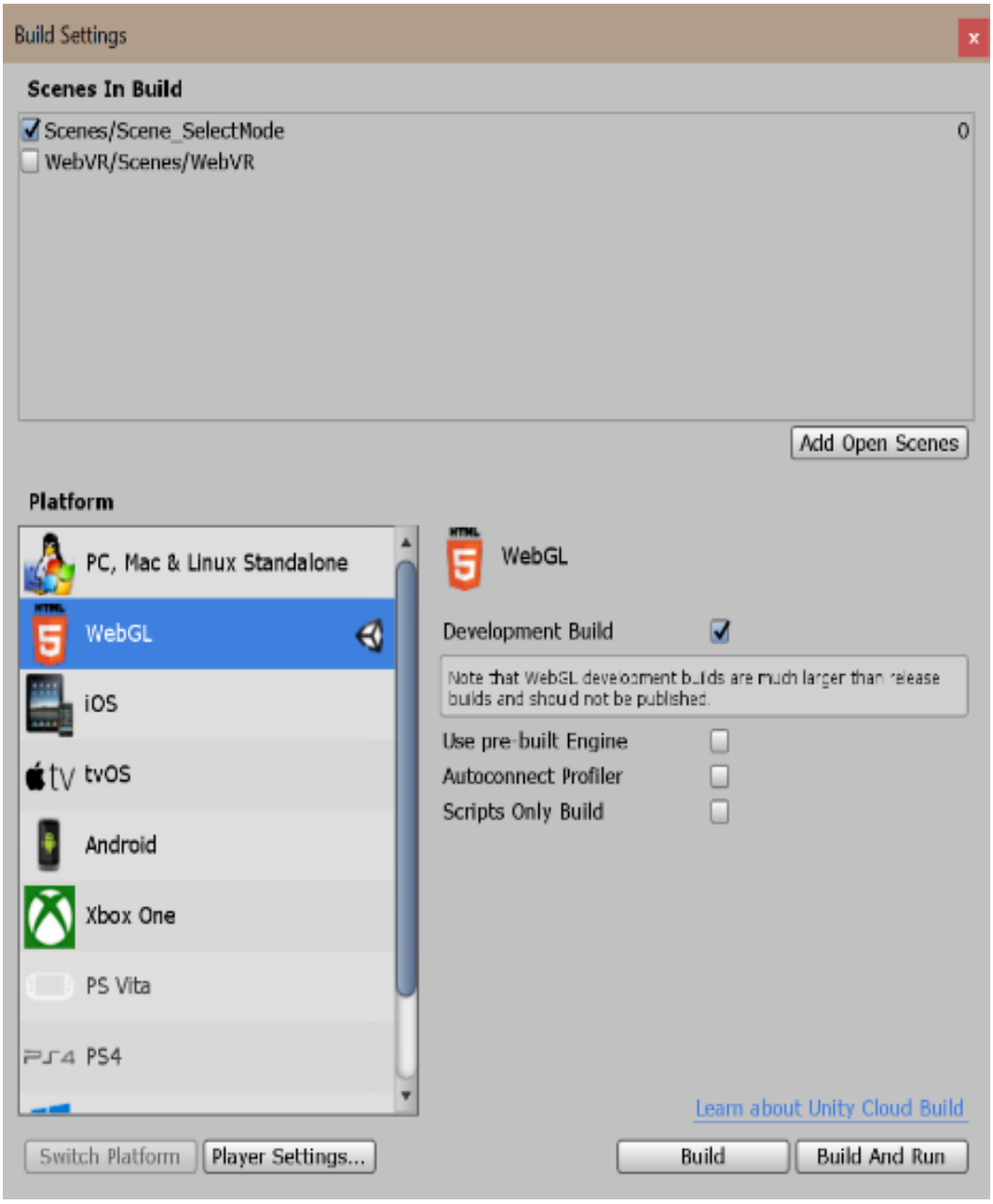

Figure 4.2: Unity3D WebGL Platform

However, as I explained before, it needs a long time to study a new language and master it. A-frame is not such easy to learn for me to use it to develop a project, so I come back to Unity3D and try to find a better way to use Unity3D solving

the issue, then I saw the WebGL might help me. After that, I started to use Unity3D to create my project and export it to the WebGL platform (Figure 4.2).

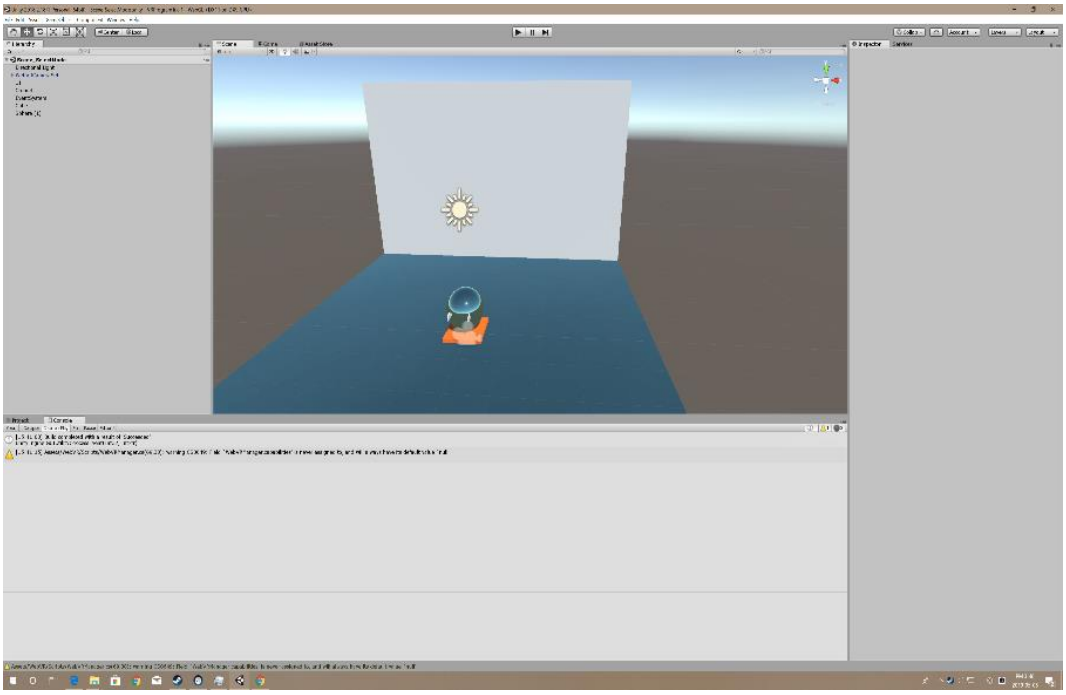

Figure 4.3: Unity3D Developed Scene

As you can see in the Unity3D WebGL Platform (Figure 4.3), I created a demo with a dark blue floor and grey wall. There is a light blue ball in the scene where the user can use controllers to interact with it. The user could press buttons on the VR controllers to pick up the light blue ball and release it to let the ball go. The shape of controllers in the scene using the same models as the real world showed in figure 4.4.

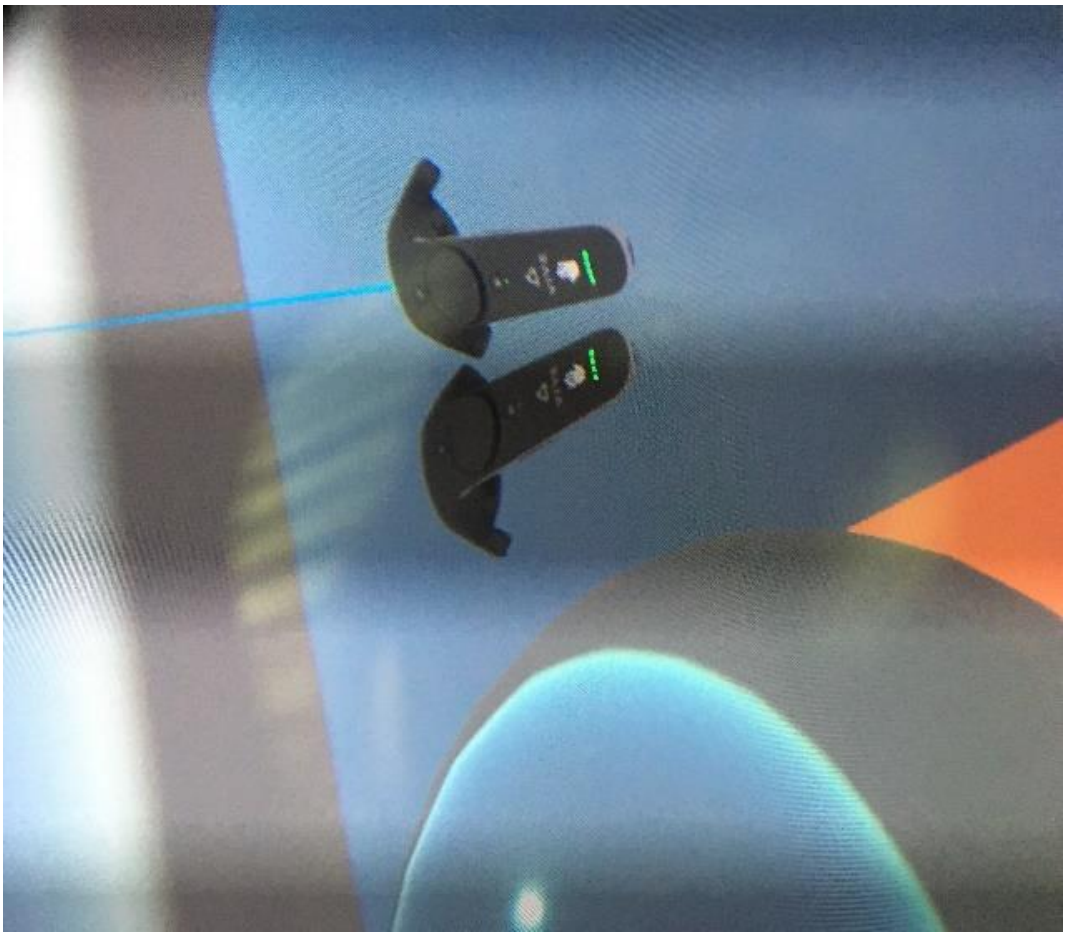

Figure 4.4: Controller Model

Then published this demo into WebGL version, it changed to HTML file and could be open on browse literally. At that time, the problem is the HTML file could not open by IE or google browse, and then I found only the particular browse with VR plug-in could execute this file, so I download Night Firefox browse and added the plug-in. After that, the demo successfully ran in the Night Firefox displayed in the running windows in Night Firefox (Figure 4.5).

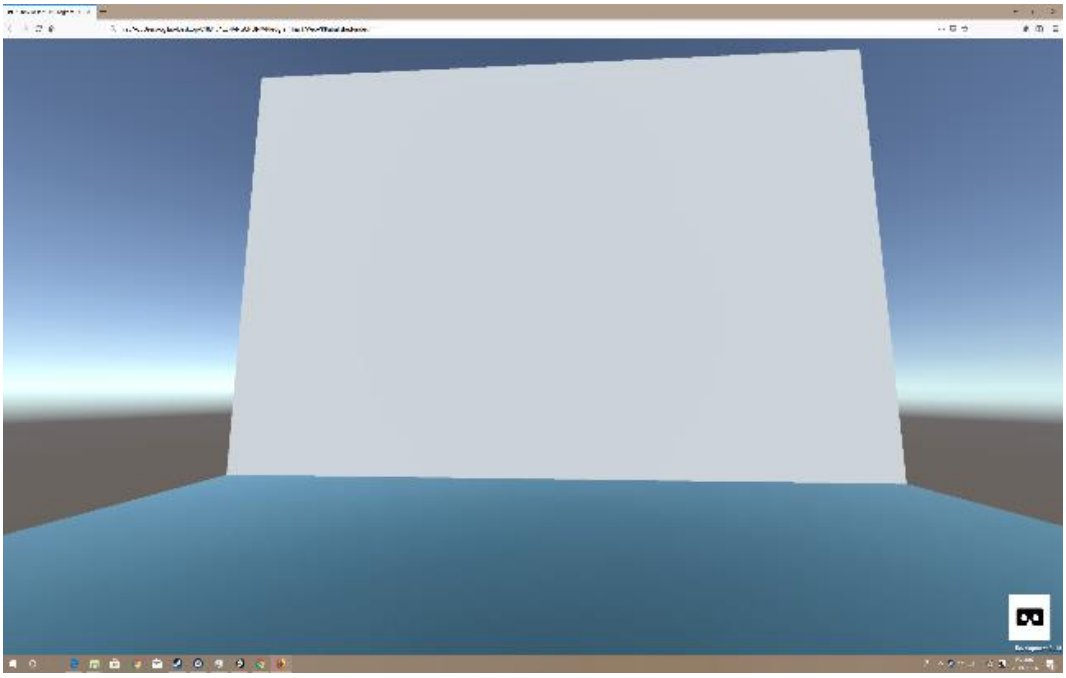

Figure 4.5: Running in Night Firefox

If you click the button on the right bottom corner, you can experience the VR mode, displayed in the figure below, with VR headset and controllers. The original scene divided to two same views, and if you use the VR headset, the 360-degree environment will immerse you into it.

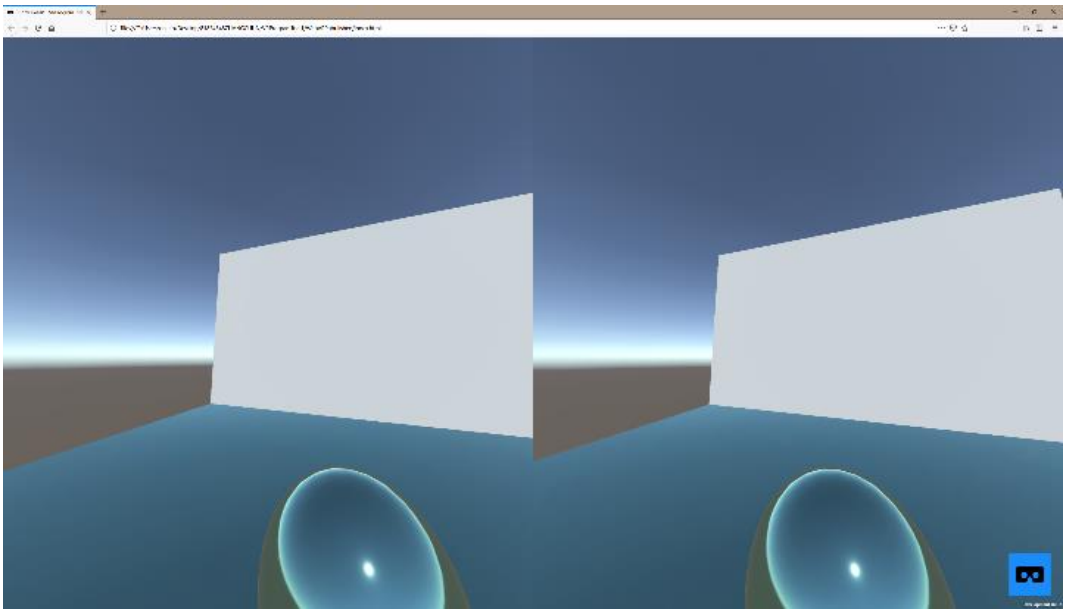

Figure 4.6: Run VR Mode in Night Firefox

The demo successfully ran on Firefox Nightly with VR controllers and headset. It can use these interaction functions wrote in Unity; also, the hand modes can show in browse.

The problem is the controllers did not work when I execute the demo on Night Firefox. The screen changed to VR mode successfully, but the controllers still not work on the Web page. For this reason, my lab member found a new browse totally for VR named Supermedium, then introduced it to me. Supermedium, the team behind the native virtual reality browser that lets you interact with web-based VR content, says its namesake project is no longer in active development, and that it's effectively been put "on ice" in search of something that consumers might use in VR on a daily basis[14].

Figure 4.7: Supermedium

Figure 4.7 is shown on the home page of Supermedium immersing the user into the VR environment and interact with the UI use VR controllers.

Supermedium is available on a range of PC VR headsets, including Oculus Rift, HTC Vive, Valve Index, and Windows VR headsets. You'll find it on Steam and the Oculus Store[14].

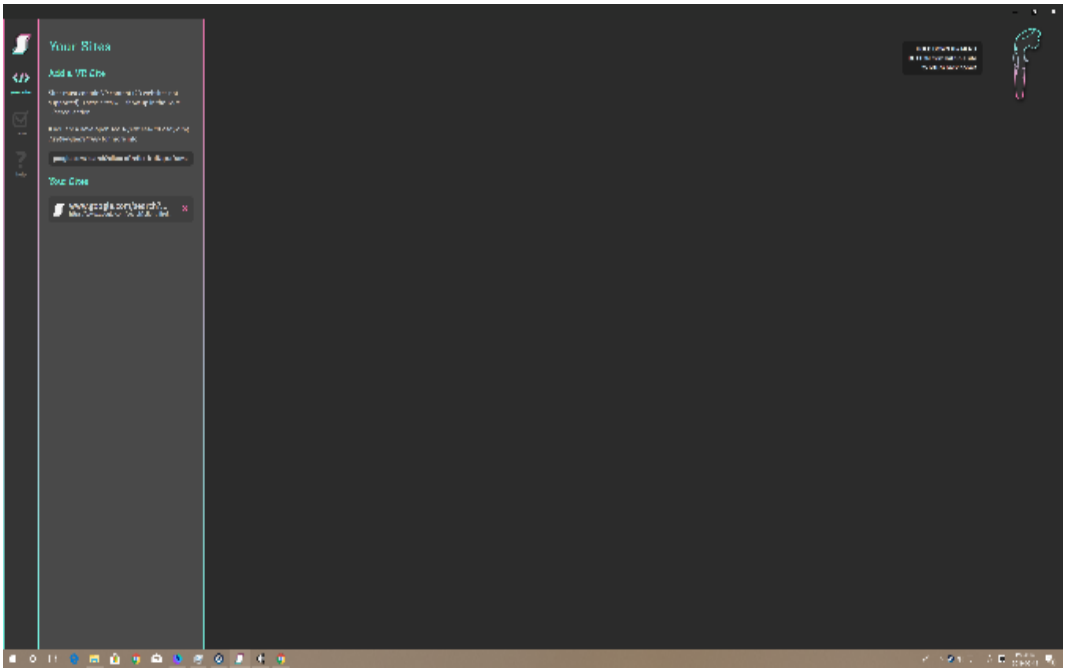

Figure 4.8: Supermedium Editor Page

Supermedium also provides an editor page allowing the developer to make their project better. The unfortunate thing is it also based on A-frame, the language I do not learn before, so I gave up and tried to find another way to solve it.

To solve the problem, I saw a free Unity3D WebVR Assets made by Mozilla when surfing the Internet. It is free to download and available now on the Unity Asset Store. This tool allows creators to publish and share VR experiences they created in Unity on the open web, with a simple URL or link. These experiences can then be viewed with any WebVR enabled browser such as Firefox (using the Oculus Rift or HTC VIVE) and Microsoft Edge (using a Windows Mixed Reality headset)[15].

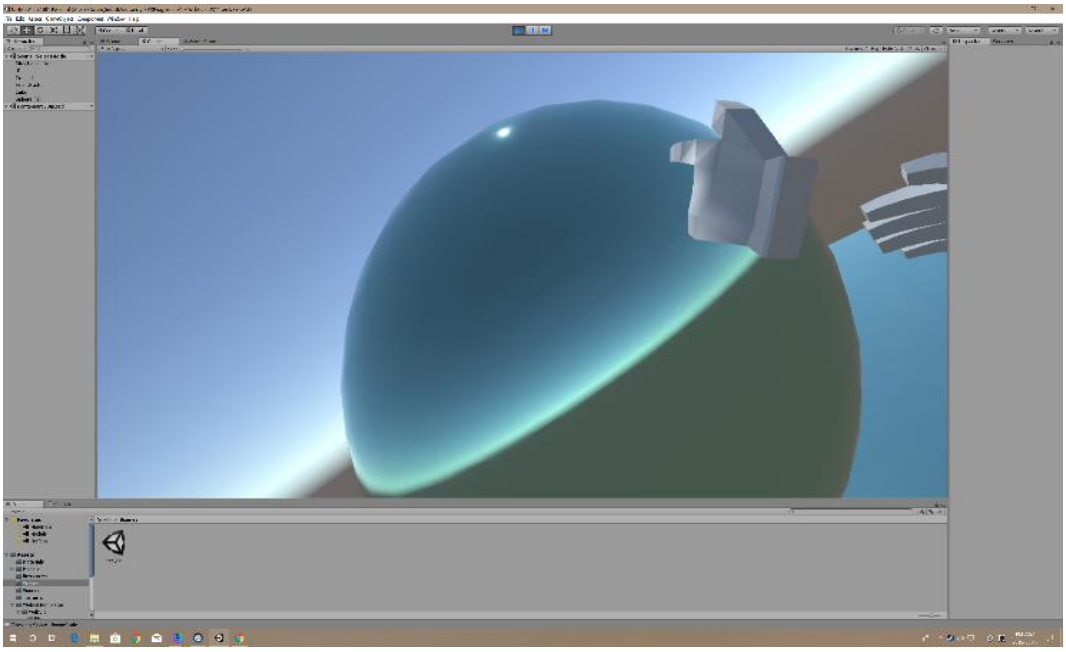

Figure 4.9: A-frame Developed Web Page

I create a new demo displaying in figure 4.9 with the free tool Unity3D Web VR Assets. The assets changed the controller models from the same as the real one to the hand models. I created a new demo displaying in figure 4.9 with the free tool Unity3D Web VR Assets. The assets changed the controller models from the same as the real one to the hand models. Afterward, the demo published on the

WebGL version and successful execution on the browse. It allows us to use the VR controller, running the demo on the web page.

Table 4.1: Compared Development Tools

Tools	**Model Resource**	**Function Support**	**VR Device Adaptability**	**Web VR Adaptability**
Unity3D	Rich model resource library and support external import model	Provides a large number of highly integrated functions for development	It can be well-matched with VR equipment, and there are few problems in the development project	Exporting to the Web VR platform is prone to issues, and the compatibility is weak
A-frame	support external import model	Provide a completed system of development	It is challenging to detect VR devices when the program is running	Very adaptable to Web VR platform, strong compatibility

Comparing the two development tools, we can see that each has its advantages. It is to reduce the difficulty of the development process and the richness of resources. I chose Unity3D, which is rich in model resources and has a complete basic method library, as my development tool.

In the short brief, after various experiments and attempts, the development tool is Unity3D, and the platform of this project is WebGL. Finally, choosing this development tool and publishing platform is a combination of my capabilities and user needs.

4.3. Function Introduction

4.3.1 Interaction in VR Environment

First of all, when the user is in a VR environment, traditional interaction methods are no longer available， which means that users cannot use the conventional way to interact with the system. Instead, users need to immerse themselves in the 3D environment[16].

Table 4.2: Different Interaction Tools

The traditional interaction Tool	The VR interaction Tool
Keyboard	VR Controller
Mouse	

Display	VR Headset

The characteristics of the traditional environment and virtual reality environment are different. If one person wants to create a VR project, he or she has to know at least one programming language among C#, C++ or JS, which is a big challenge for a person without any programming experience. Although there are some VR development engines in the market such as Unity 3D, Unreal Engine 4, and the like, it still not an easy job to learn how to use the engine, even someone who already has programming knowledge. The traditional environment is the usual way programmers use a programming language to create a project, and the virtual reality environment means the user can deep in the virtual reality environment and directly use VR controllers and headset to program without code. Virtual reality is the use of various computer graphics systems combined with multiple display and interface devices to provide the effect of immersion in an interactive three-dimensional computer-generated environment in which the virtual objects have a spatial presence[4]. Thus, there is a massive difference in the traditional program environment and the virtual reality environment.

First and foremost, the noticeable difference is the tool used in these two environments. In a traditional programming environment, users need to use a keyboard, mouse, and screen to create a new project and executed results manifest on the screen. And in the VR environment, the tools are used only hand controllers and headset. Compared to these tools, the programming method has

changed due to the limitation of input information from two-dimension to three-dimension. The button number of VR controllers is far less than the keyboard; accordingly, we change the traditional programming from coding to VR programming by a new visual programming method. And the decrease of the input button results in the program method changes, which is hard to use controller programming as keyboard coding with the programming language. When the programming environment switches to the 3D stereo environment, the traditional flat UI can no longer meet user needs. The interaction of the flat UI in the 360 stereo environment has also become convenient to use in the conventional context example. At this time, we need to make the 2D UI three-dimensional so that users can interact with it in the VR environment[16]–[18].

- Demo One：Scenario Selection

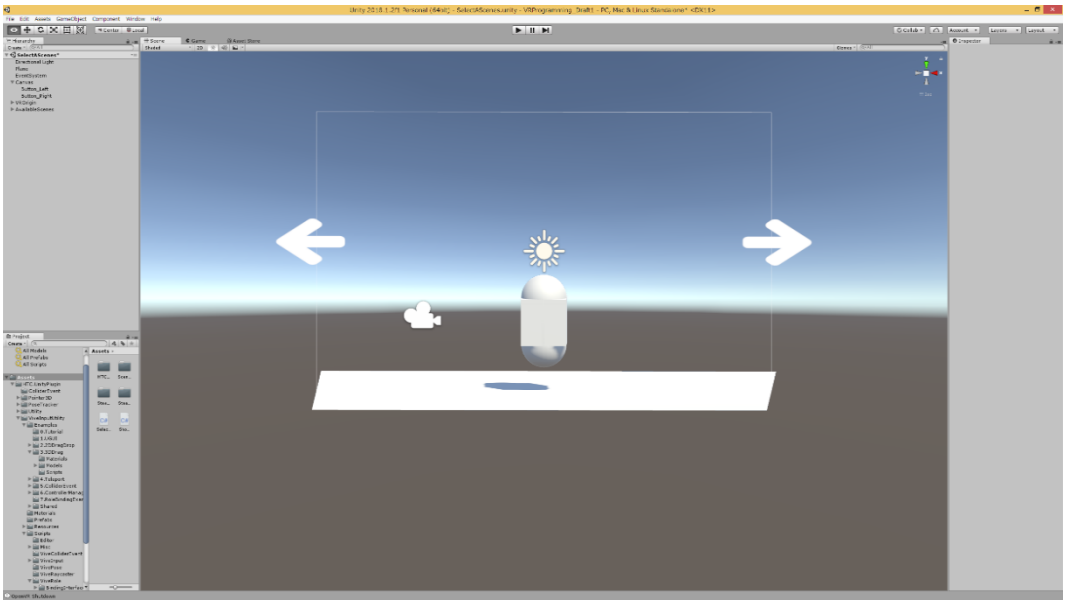

Figure 4.10: Scenario Model Selection

Since the establishment of all 3D scenes is based on a three-dimensional environment, I first tried to implement the function of selecting a scene model. It can be seen in the window of scene model selection (Figure 4.10) that the capsule in the middle temporarily replaces the scene model here. There is a stage-like plane below the model. A model is centered on the left and right are two plane arrows. These two arrows are two buttons used to switch the model.

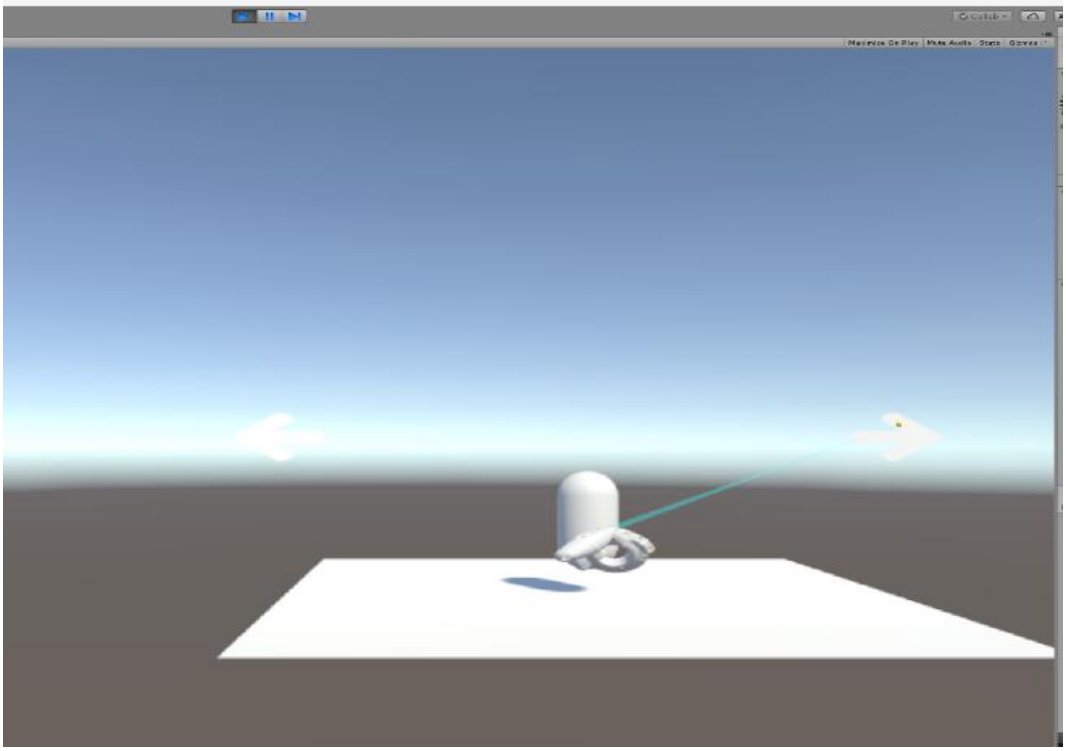

Figure 4.11: Execution Stage of Model Selection

It can be seen in the Execution Stage of Model Selection (Figure 4.11) that when the scene is running, a laser will be emitted from the VR handle and interact with the objects it touches after being ejected.

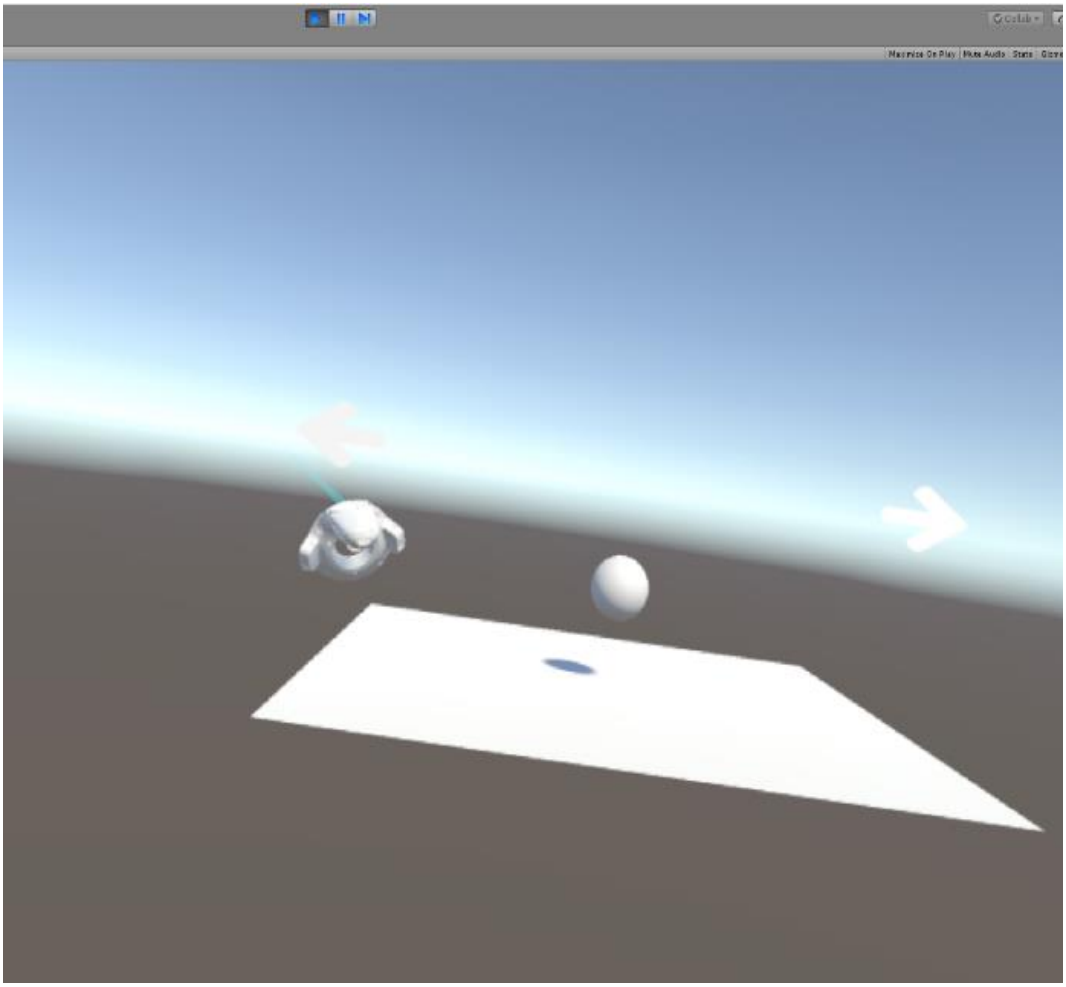

Figure 4.12: Model Changing

When the arrow button is clicked, the model is switched (Figure 4.12). In this scenario, the 2D buttons of the 3D model are combined, but the final effect is not very good. The main reason is that the 2D buttons limit the user's perspective. You can see and use this button only when the user is facing this button. When the angle changes, the buttons would quickly disappear visually in the scene.

- Demo Two：2D Button in VR

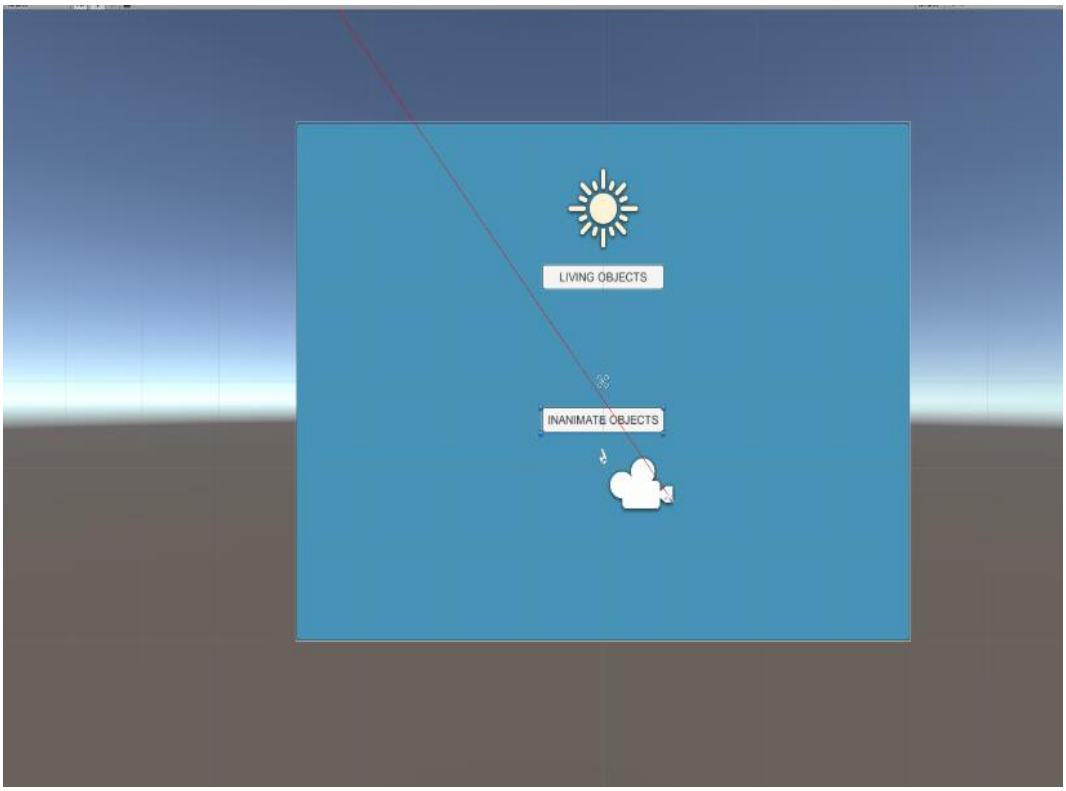

Figure 4.13: Normal 2D UI

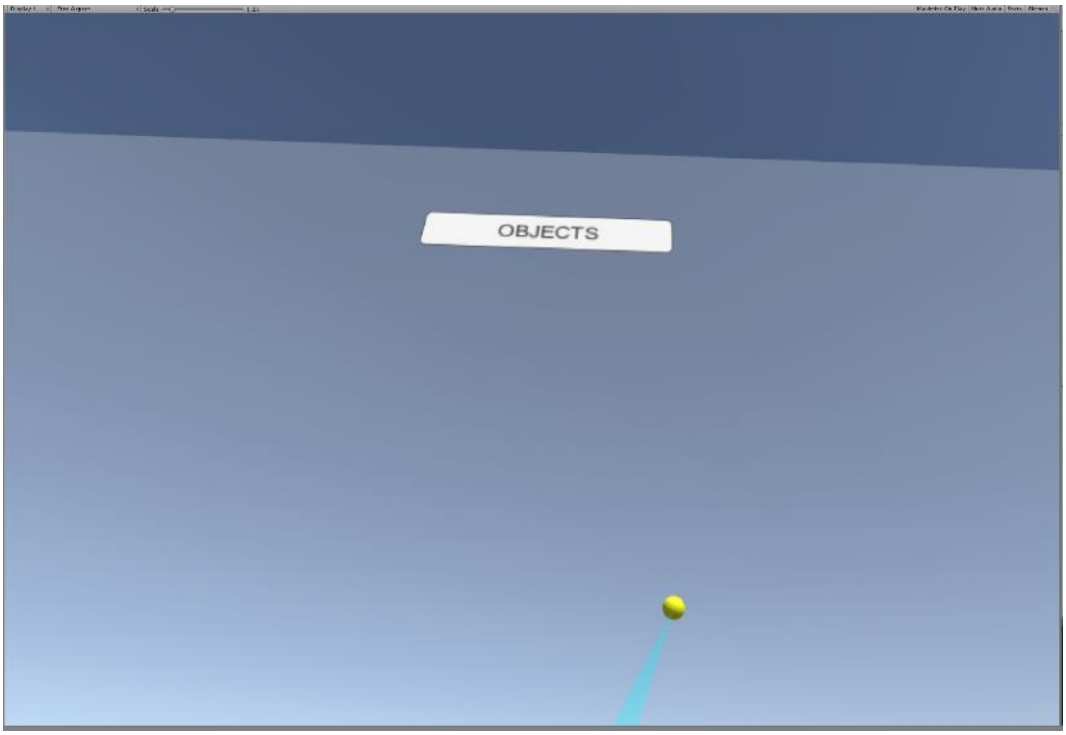

Figure 4.14: Normal 2D UI in VR Environment

For better research and comparison, I created a 2D UI interface in a VR environment (Figure 4.13). As you can see from the Normal 2D UI, when viewed from the front, the entire UI interface can well present to the user's eyes. Still, when the program starts to run, and the user immerses in this scene, a slight angle change will Affect the user's interactive experience (Figure 4.14).

It follows that the traditional UI design can no longer satisfy the use in the VR environment, so the UI needs to upgrade from the original 2D to 3D.

- Demo Three：3D Button in VR

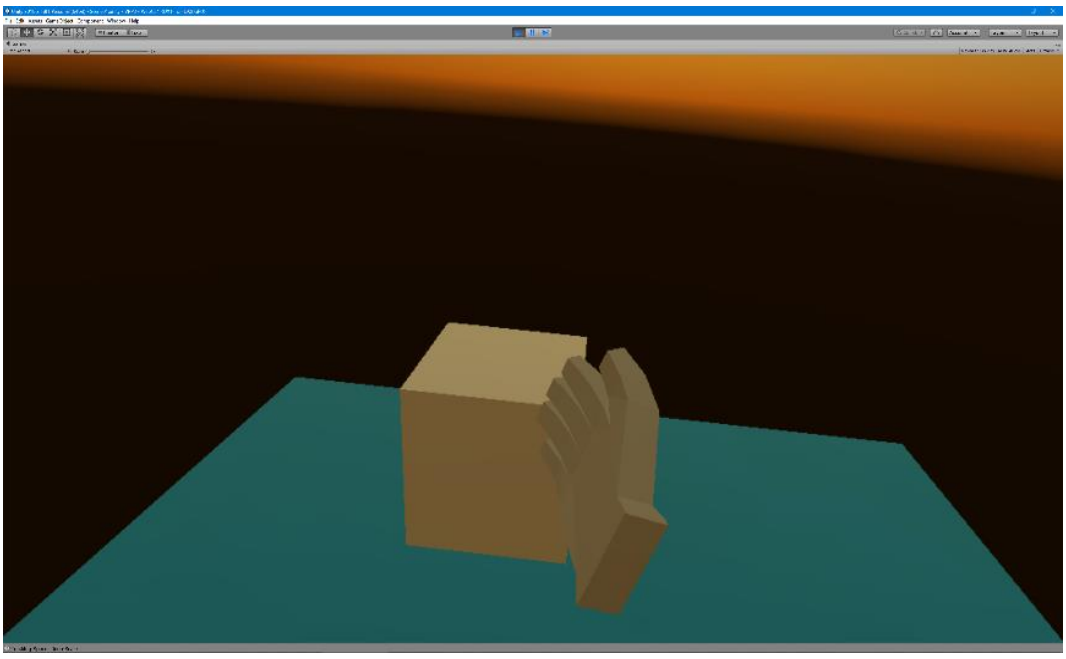

Figure 4.15: 3D Button

I tried to change the button from 2D to 3D. So, I created a new demo that uses some simple 3D models like cube and ball instead of the button in 2D. The user

uses the controller that is a hand model in the VR environment, to interact with the cube (Figure 4.15). When a VR controller in the box collider and click the button down, the ball, cube, capsule, and cylinder would display (Figure 4.16).

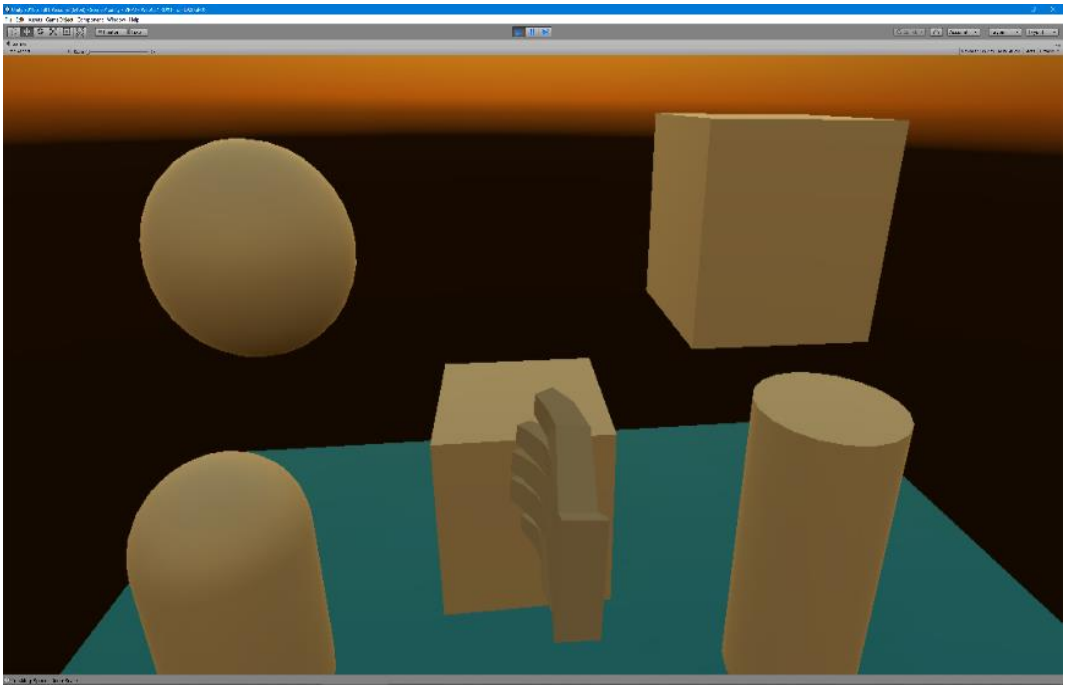

Figure 4.16: 3D UI

The ball, cube, capsule, and cylinder are buttons and the traditional 2D button change to these simple geometries. Nonetheless, these geometries hard to let people know they are buttons for the first time if no other information provided. To make this 3D button easier to accept, I replaced the simple model with other models that contain more information.

Table 4.3: Demos Comparison

Demo	Demo One：Scenario Selection	Demo Two：2D Button in VR	Demo Three：3D Button in VR
UI Design	2D switching buttons with 3D models	2D buttons with 2D UI	3D buttons
Advantages	Close to the traditional UI interface, easy for users to understand, when standing in a fixed position, the operation is simple, the user can quickly realize the scene selection.	The advantage of traditional UI design is that users do not need extra time to understand or explore how to use this UI interface to interact because all traditional UI interfaces are very similar.	Give users a more intuitive understanding and experience. It is very convenient to interact with the VR controller.
Disadvantages	The interaction process will be affected by the change of angle.	It is very difficult for users to use the UI in this environment because of the difference in dimensions and interactive tools changes.	For the first time, users need time to understand and explore.

The difference in programming in a real environment and virtual environment is the programming language difference caused by the limitation of programming tools. In the traditional programming environment, the person creates a VR project typically with a VR development engine. For example, if I want to create a new VR project that is an English course demo, let students can use VR controllers and headset do communication practice. The first work is finding a suitable VR engine after creating a new project and building or downloading some 3D models I needed then using a programming language to realize the functions. Therefore, the traditional environment requires that the creator has mastered a programming language and use code to achieve the purposes. However, the virtual environment provides a visual programming way reducing the time of learning a programming language and engine, which means even you do not have any programming knowledge. You also can create the same project at the same time compared with those who know to program. The advantage of visual programming is it saves time for a beginner but not such flexible because the function created earlier and preserved in a library. So it can be seen the visual programming method more friendly with people without programming foundation, and the master may prefer the traditional programming way.

Due to changes in the programming environment, we have to create a 3D environment that users can visual programming. The 3D environment provides the user-space to create a new scene, arrange models and add function to object by a straightforward method, such as linking objects with service or the user drag

and drop techniques to match tasks with objects. In the 3D environment, we have to provide some more visual programming methods. Simultaneously, because we use VR controllers for development, programmers can also experience this VR program while developing. Users are immersed in the environment to create VR projects. The development process is also the process of debugging, allowing developers to experience VR projects in real-time. If the VR program developed in a two-dimensional environment, the developer must develop in a non-VR environment. Then put on a helmet and debug in the VR environment. So the development process requires many VR helmets and then removes, while VR Development and debugging in the background to avoid this trouble.

4.3.2 3D Menu and Movement

This demo is a 3D menu consisting of 3D buttons. `First, you can see a box in the scene (Figure 4.17), and when a controller in the box and click button down, the menu would display (Figure 4.18).

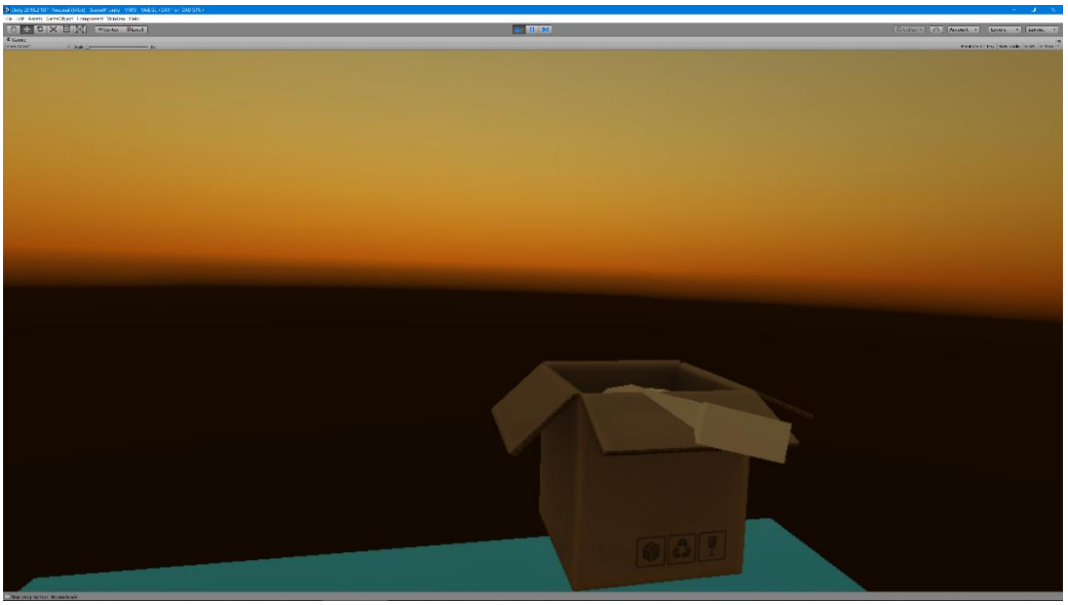

Figure 4.17: 3D Menu-1

In this scene (Figure 4.18), the pig model represents the button to select the animal model. The character model will jump to when interacting with the character model. The palace model indicates that you can choose the desired scene. The apple model is the entrance to select the fruit model.

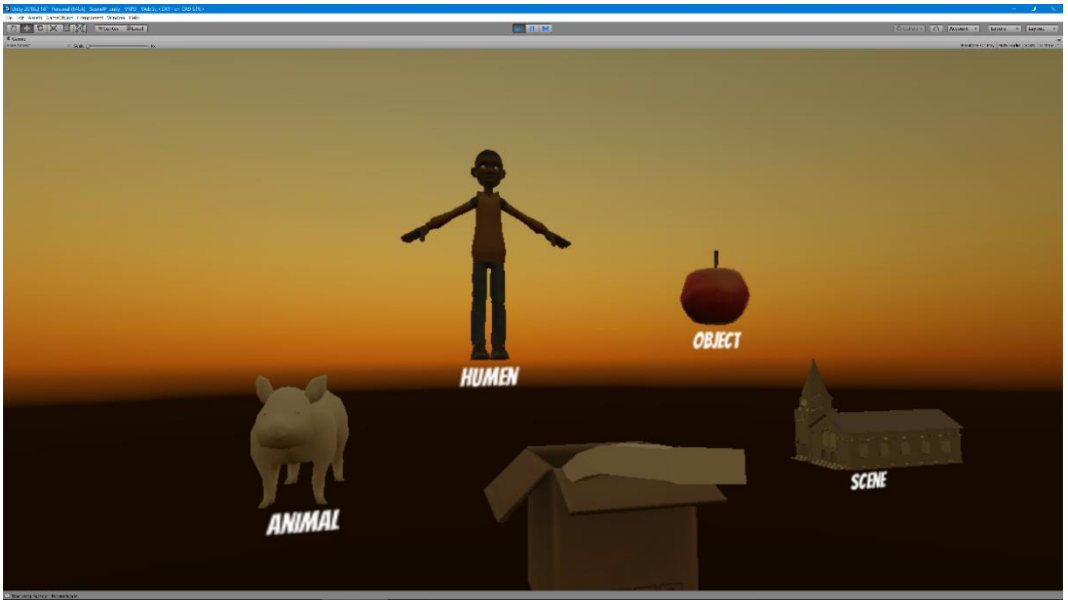

Figure 4.18: 3D Menu-2

Figure 4.18 shows the menu consists of four 3D models with text and one box model in the VR environment. The traditional menu includes text and buttons. However, figure 4.19 shows the 3D menu in the VR environment that users can use VR controllers and headset select the 3D model type. The 3D model with text below instead of the text and button of traditional programming provides users a directly perceived menu in the VR environment. From the menu design, the conventional flat menu that people imagined change into a 3D menu, which allows users to adapt to the VR environment more quickly and develop accordingly.

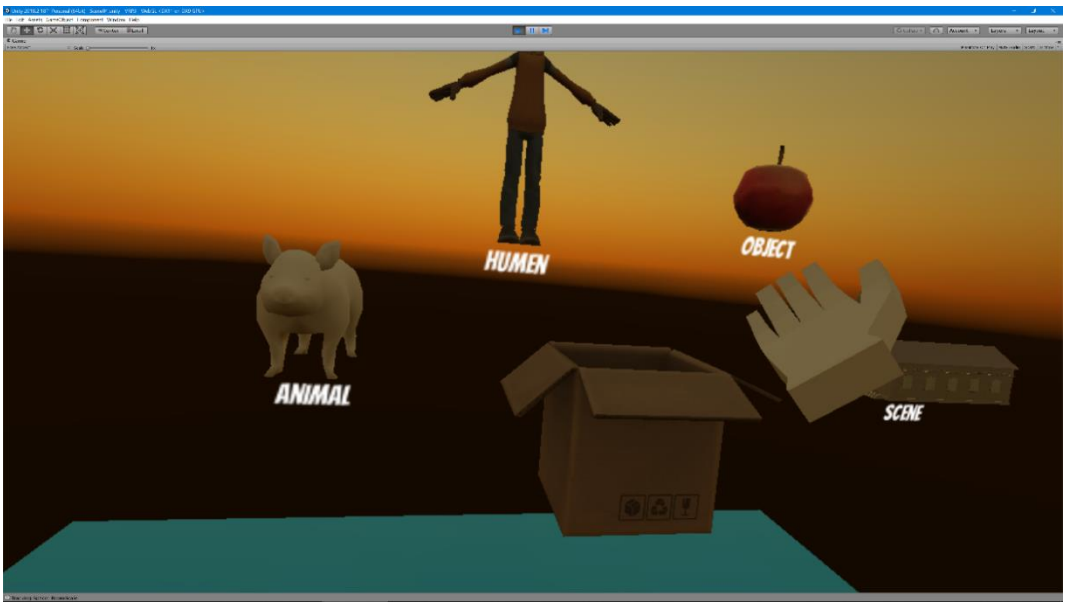

Figure 4.19: Main Menu Scene

The interaction way of this menu is moving a controller in the apple model; meanwhile, clicking the button on the controller, then the scene switches to another one. For example, I put the hand in the object button, and then the scene skips to the object selection scene (Figure 4.20).

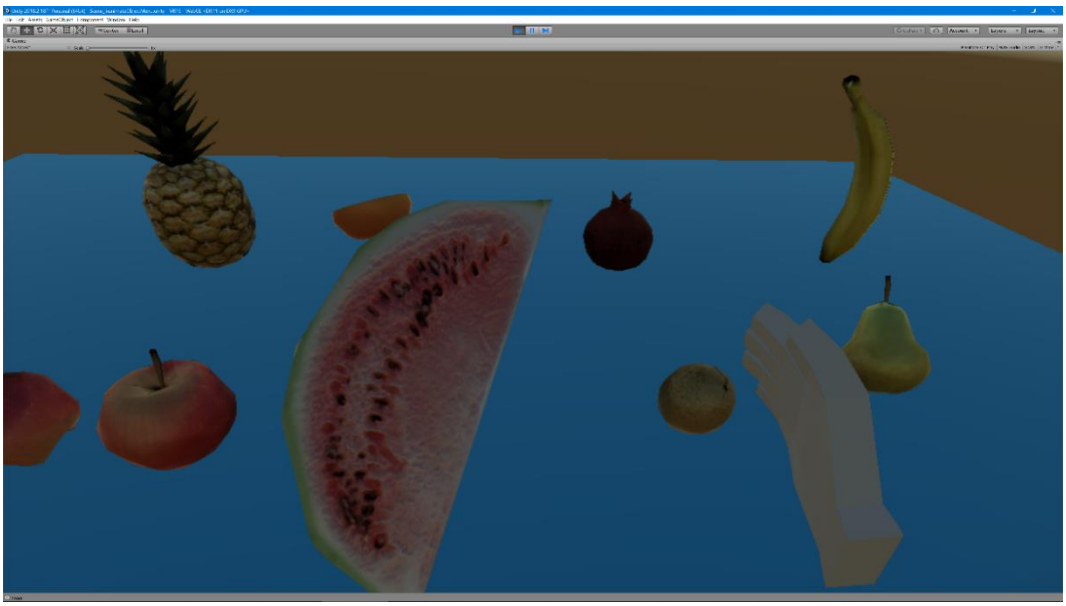

Figure 4.20: Secondary Menu Scene

The whole process presents how the 3D menu works in a VR environment. After the button is converted to 3D, the jump between the main menu and the second menu cannot be the same as the traditional menu we usually use, so here I use scene switching to achieve the switch from the first menu to the second menu.

Another problem is the movement in the 3D environment. When we are programming, there is no problem with action. Still, when we are programming in the VR environment, the entire system rises from two to three dimensions, we need to consider the problem of movement in space.

Figure 4.21: Teleport in VR

Regarding the realization of mobile functions, I used Unity3D Assets. In this auxiliary tool, the Teleport function is included. This function is that users can move freely through VR controllers in the area you want. As shown in Teleport in VR (Figure 4.21), when the user uses the VR controller, press, and holds the trigger button on the handle, and a light green parabola will appear in the scene. When the last drop point of the parabola falls on the user's movable area, a green arrow will appear on the movable drop point. At this time, the user releases the button to move to the drop point.

Figure 4.22: Guidance of the Flow

To make the text prompts fit well in the VR environment, I designed an indicator board (Figure 4.22). The user can move to the dashboard and see how the whole process is going. The user can jump to the corresponding scene by clicking the prompt text on the board.

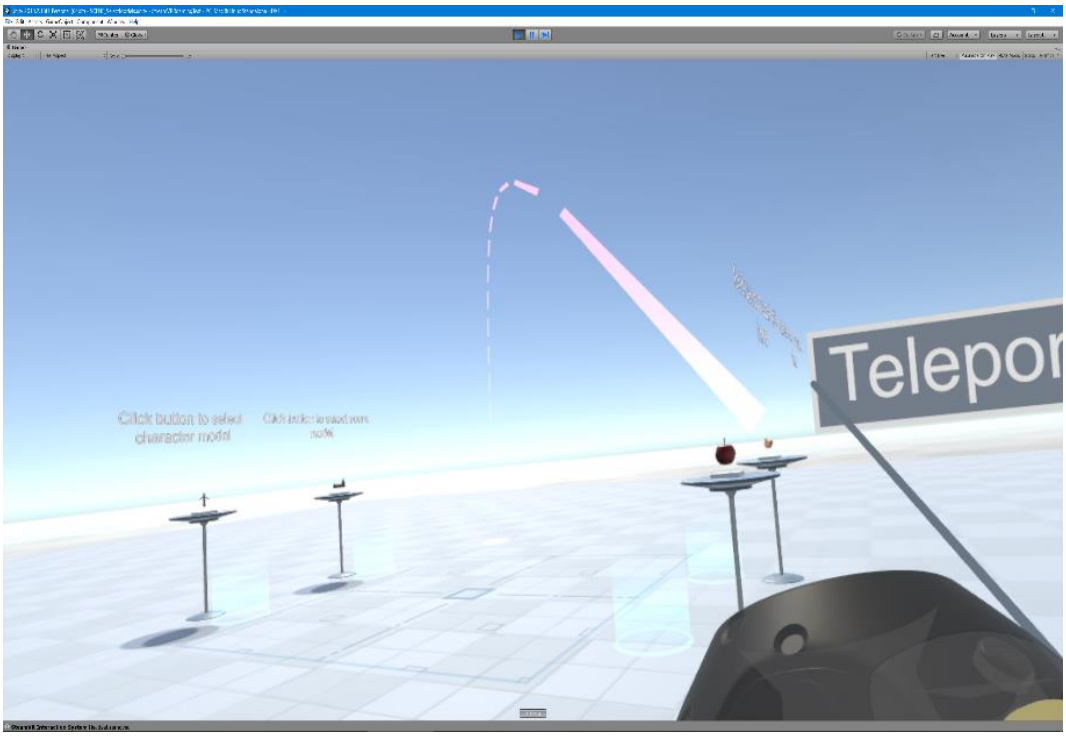

Figure 4.23: Movement and 3D Button Combination

After combining the movement function with the 3D UI, the entire interface has become very different from the two-dimensional interface. The User experience has also changed. In Movement and 3D Button Combination (Figure 4.23), the user first needs to move to the front of the button, then he or she can interact with the button. In the VR environment, the user's interaction is more specific actions than simple choices. But this also makes the entire menu can convey less information, and requires multiple scenes to achieve this function.

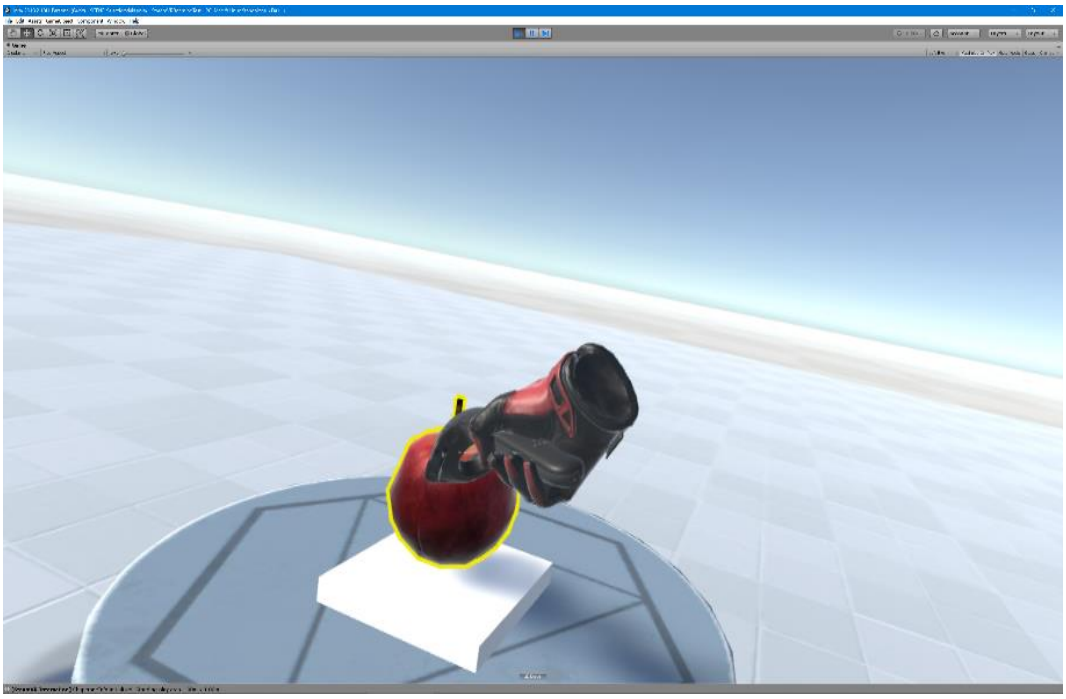

Figure 4.24: Clicking the 3D Button

The user can press the apple model by hand to realize the action of clicking the button (Figure 4.24). It achieves the goal of using the visual language ton let implement functions.

It can be seen from the five demos that using 3D Button can provide better interaction to the user in the VR environment. But there is also a problem: the amount of information transmitted is reduced, and the first time the user uses it, it isn't easy to know how to use it quickly without explanation. So, when I use 3D Button, I also use text prompts to understand how to use this program immediately.

4.3.3 Function Added

Regarding adding functions (Figure 4.25), I refer to the visual programming language Scratch and integrate it into the VR environment. I use the cube to represent different methods. Each model can add methods. Taking Function Added Scenarios as an example, the character model in the scene is the model selected by the previous user. The user wants to add a method to this model. In the view, the model has a translucent Cube area, which is the character model's programming area. The small square on the front table represents the method.

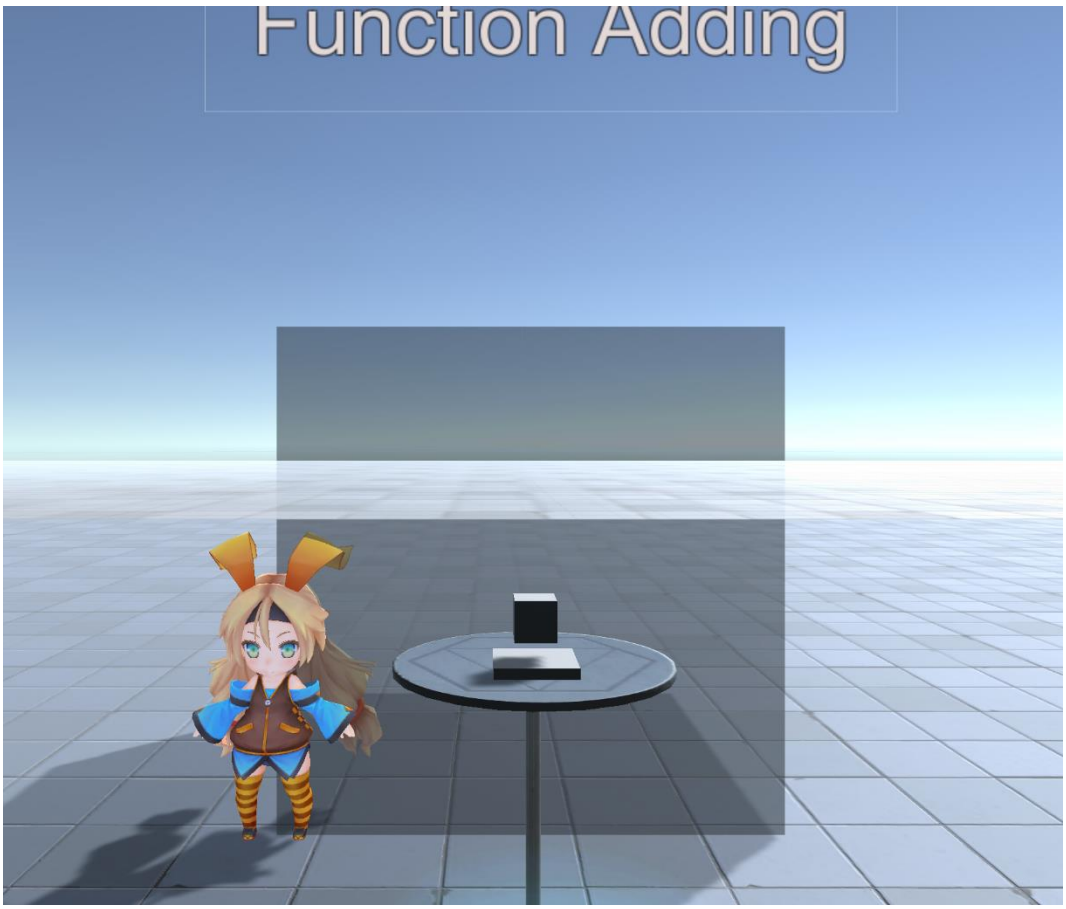

Figure 4.25: Function Added Scenario

When the user activates the method to add a function, the user can use the visual programming method in the VR environment. The user searches the method by voice keywords. For example, if the user wants to add a function of speaking to the character, the user can say the keyword "Speak". If there is a "Speak" Function in the library, the button (Figure 4.26) above the table will be displayed. Then the user uses the VR handle to press the button, and this method is added to the character.

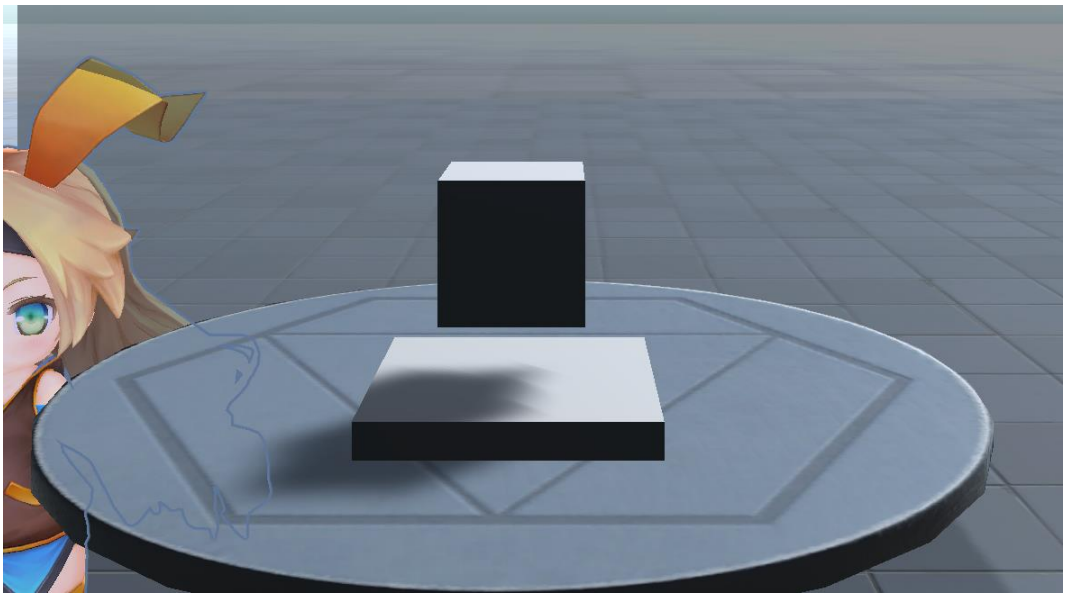

Figure 4.26: Function Button

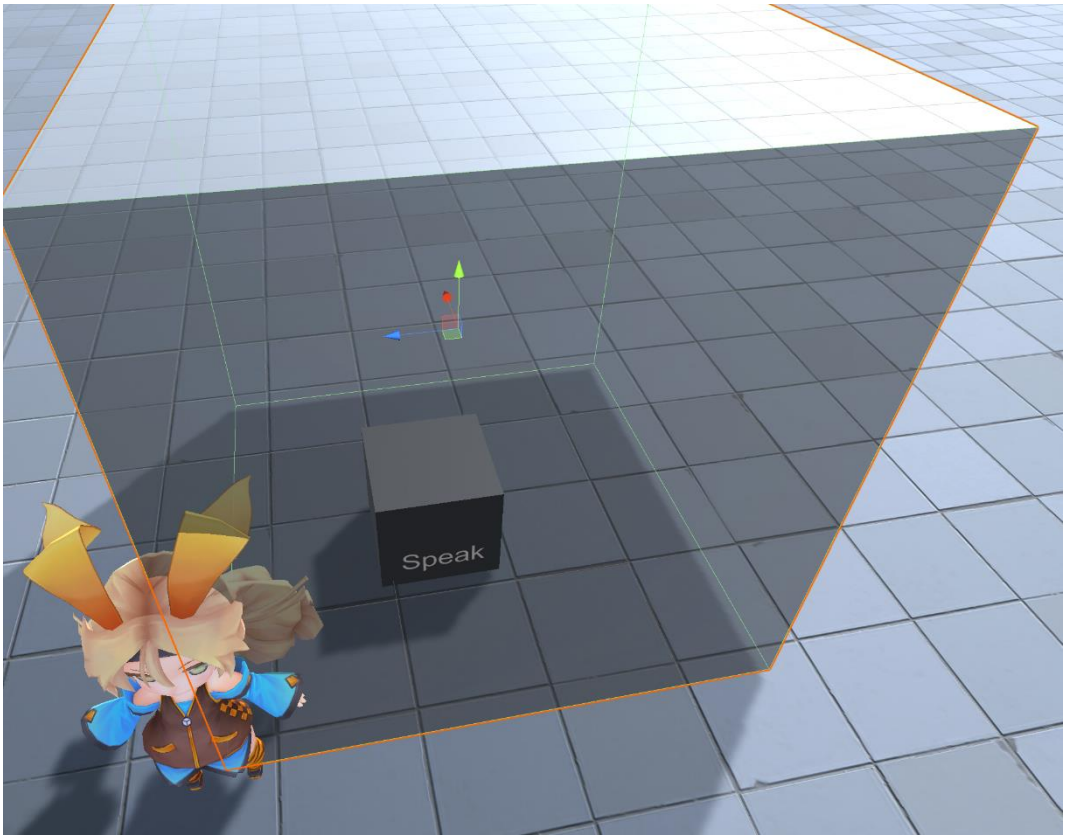

Figure 4.29: Function Space

If the method is successfully added to the model, the speak method will appear in its programming space. If no way is added, the method space for the model will be blank. It can be seen that a method cube appears in the Function Space, which speaks on the cube, proving that the speak function has been successfully added to this model.

Chapter 5. VR Programming Validation

This chapter is mainly discussed about the verification and validation of the system. I primarily focused on the 3D menu part of the system because this part is the basement of the whole system, and it is very important for the simple programming.

As shown in Figure 5.1, the developer operates with the VR controllers, and the entire process of user interaction can be seen on the screen. The user can use the VR controller to implement operations such as grabbing, selecting, moving items, and the like. Since only the 3D menu part is now completed, the further design has not yet been completed. Later, we need to do some complex interactions, such as function implementation.

Figure 5.1: User test Three-dimension menu by VR equipment in the VR environment

To verify and validate the 3D menu, I created a 2D menu with the same content, and then I asked 7 participants to experience both menus and then answer the questionnaire.

The following are the details about the questionnaire and feedback from participants.

- Q1. Do you know the software menu?

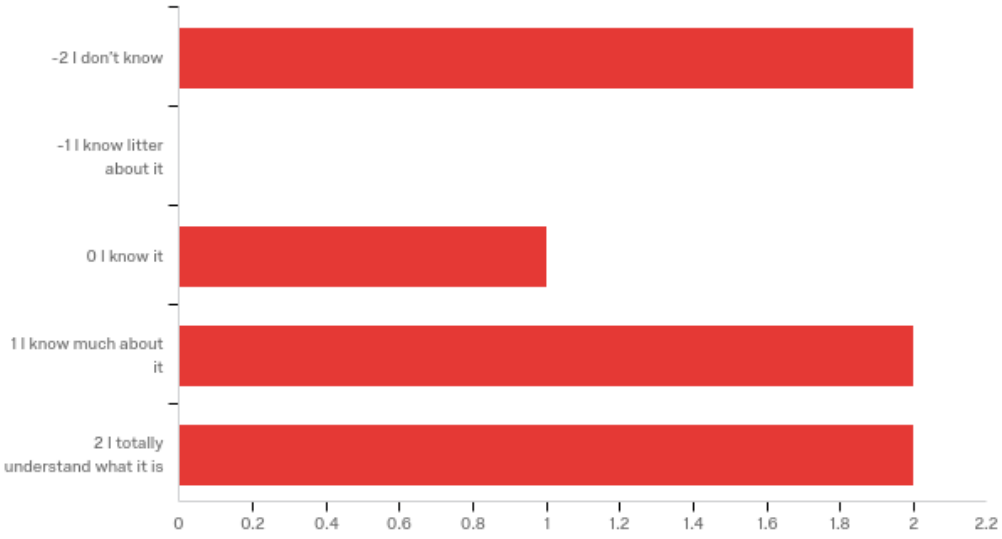

Figure 5.2: Answer Analysis 1

Table 5.1: Data Statistic 1

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Do you know the software menu?	1.00	5.00	3.29	1.58	2.49	7

There are five answers for question 1, and the different answers give different weights range from 1 to 5. The mean of this question is 3.29, and the standard deviation is 1.58. This question is used to ensure every participant can understand what the menu is.

- Q2. Did you use VR before?

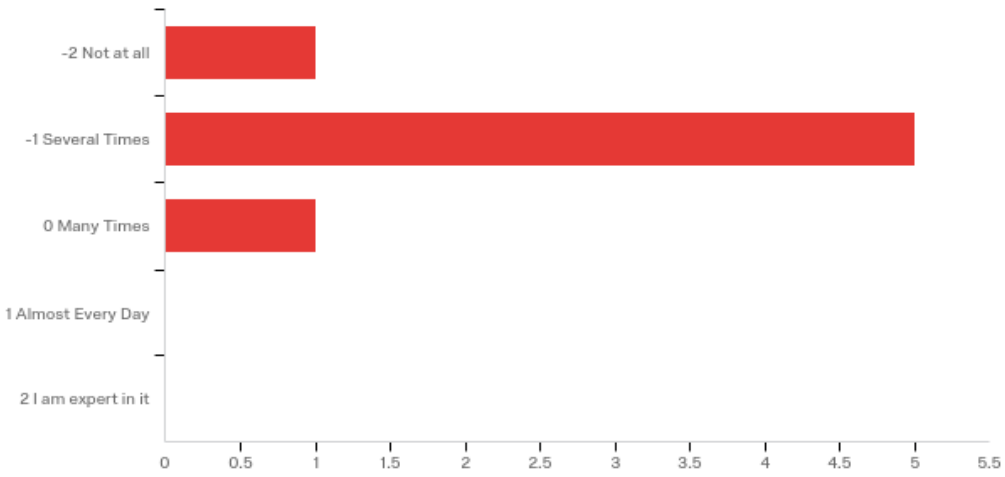

Figure 5.3: Answer Analysis 2

Table 5.2: Data Statistic 2

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
2	Did you use VR Before?	1.00	3.00	2.00	0.53	0.29	7

There are five answers for question 2, and the different answers give a score from 1 to 5. The mean of this question is two, and the standard deviation is 0.58, which means most participants are not familiar with VR.

- Q3. How do you think the 2D menu in you usually used?

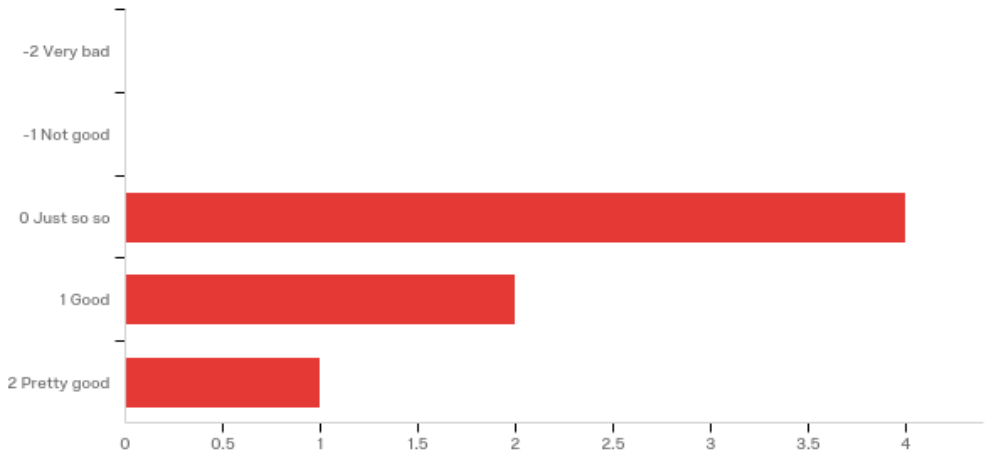

Figure 5.4: Answer Analysis 3

Table 5.3: Data Statistic 3

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
3	How do you think the 2D menu in you usually used?	2.00	4.00	2.57	0.73	0.53	7

There are five answers for question 3, and the different answers give different weights range from 1 to 5. The mean of this question is 2.57, and the standard deviation is 0.73. From selections of participants, they give a high score to the 2D menu used in daily life.

- Q4. How do you think about this 3D menu?

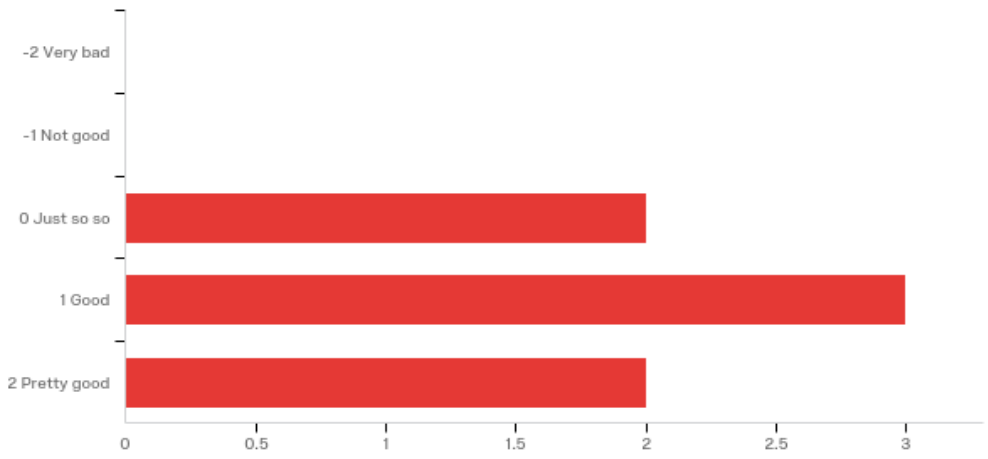

Figure 5.5: Answer Analysis 4

Table 5.4: Data Statistic 4

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
4	How do you think about this 3D menu?	2.00	4.00	3.00	0.76	0.57	7

The mean of this question is three, and the standard deviation is 0.76. Comparing with question 2, participants give a litter higher score to a 3D menu than 2D.

- Q5. How do you think the 2D menu used in VR?

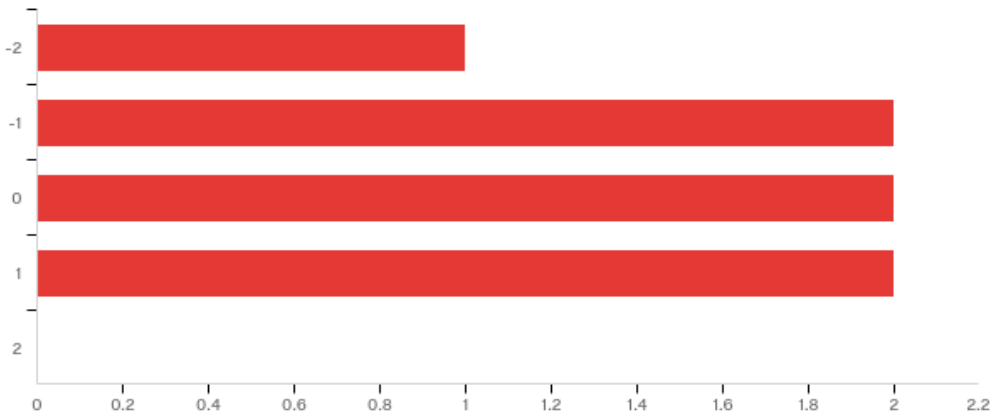

Figure 5.6: Answer Analysis 5

Table 5.5: Data Statistic 5

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
5	How do you think the 2D menu used in VR?	1.00	4.00	2.71	1.03	1.06	7

The mean of this question is 2.71, and the standard deviation is 1.03. This data shows they are not satisfied 2D menu used in VR.

- Q6. How do you think this 3D menu used in VR?

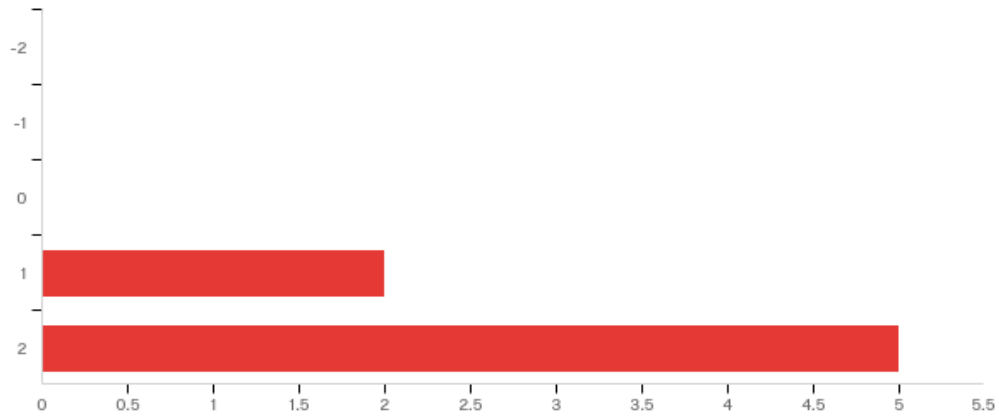

Figure 5.7: Answer Analysis 6

Table 5.6: Data Statistic 6

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
6	How do you think this 3D menu used in VR?	4.00	5.00	4.71	0.45	0.20	7

It is obvious to see that the 3D menu owns a high score in the VR environment, and all the participants give the more than 3 to it.

- Q7. Which one is easier to use in the VR environment compared with 2D and 3D menu?

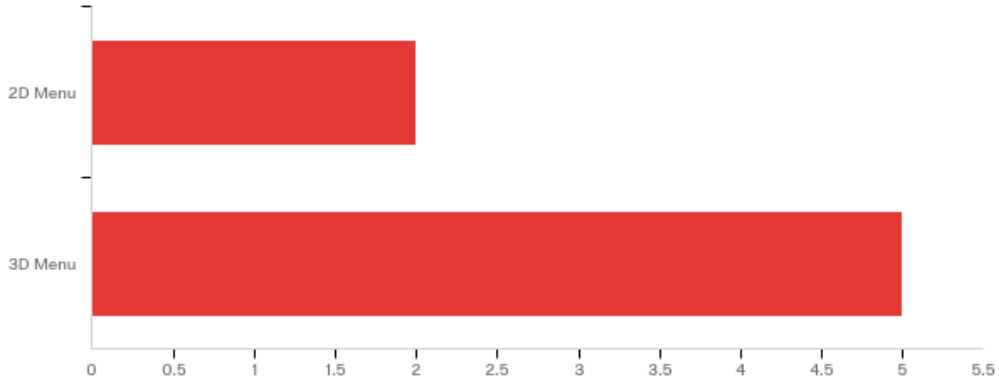

Figure 5.8: Answer Analysis 7

Table 5.7: Data Statistic 7

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Which one is easier to use in the VR environment compared with 2D and 3D menu?	1.00	2.00	1.71	0.45	0.20	7

There are five people who think the 3D menu is more convenient to use in the VR scenario, and two persons still prefer the 2D menu even in VR.

From the Q1 and Q2, all the participants are not the master of VR development, corresponding with my target users. From Q3 and Q4, the participants are satisfied with the general 2D menu and think 3D menu performance good. From Q5, Q6, and Q7, the 3D menu gets a higher score in the VR environment than the 2D menu. 71% of participants think the 3D menu better adapt to the VR environment. At the same time, the comment on the 2D menu in VR is also positive.

After I explain what means 2D Menu, many people said they know what it is. The mean of Q2 is -1, and the standard deviation is 0.53, which shows participants mainly are the people who do not use VR frequently, and my target users are these people who do not know much about VR programming. The result shows. The result shows many participants not satisfied with the 2D menu they used daily life. However, they also did not give a high score to the 3D menu, just a litter higher than 2D. Compared with the 3D menu, most participants think the 2D menu maybe not such suitable for VR. Many participants have a positive attitude toward the 3D Menu in VR.

Chapter 6. Conclusion and Future Works

6.1. Overall Summary

In my research, I mainly designed and developed a program that allows users to develop VR projects more intuitively using VR devices. Based on traditional visual programming, I designed a new 3D programming program. Even people without a programming background can use this simple way to complete project design in the VR environment.

There are still many deficiencies in this research. The first is that at the beginning of the design, the understanding of the target user's needs is not deep enough so that the system can be better in functional design and implementation. Secondly, how to transform the traditional programming method into a three-dimensional programming method in the VR environment still needs a lot of experiments and research.

6.2. Future Work

First, the data storage and user information storage of the method was designed during the design period. Still, this function was not implemented during actual implementation so that the database can be partially realized in the future.

Secondly, in the method search, the methods that can be provided can continue to expand and continue to be added in the future. And can offer more and more sophisticated techniques, making users more convenient to use. Finally, I hope that the whole system can be optimized and adjusted in the future.

Acknowledgments

First of all, I would like to thank my thesis supervisor Professor Ogi. Professor Ogi made guiding opinions and suggestions on the research direction of my thesis. In the process of writing the thesis, I gave thoughtful advice on the difficulties and doubts I encountered, and put forward many useful and improved opinions, and invested a lot of effort. Professor Ogi expressed my sincere gratitude for my help and care! At the same time, I would also like to thank the teachers and all the students of the System Design and Management for helping my project.

Besides, I would also like to thank my friends and classmates for their great support and help in writing the thesis, which has much inspired me. I would also like to thank the authors in the references, through their research articles, for making me a good starting point for the research topic.

Finally, I sincerely thank my family, friends, and classmates. It was really with their encouragement and support that I was able to complete this thesis successfully.

References

[1] J. Steuer, "Defining Virtual Reality: Dimensions Determining Telepresence," *J. Commun.*, vol. 42, no. 4, pp. 73–93, Dec. 1992, doi: 10.1111/j.1460-2466.1992.tb00812.x.

[2] M.-L. Ryan, *Narrative as virtual reality: immersion and interactivity in literature and electronic media*. Baltimore: Johns Hopkins University Press, 2001.

[3] *Defining Virtual Worlds and Virtual Environments*. 2008.

[4] "Programming in Virtal Reality: An Interview With The Creator of RiftSketch." https://cognitive3d.com/blog/programming-in-virtual-reality (accessed Jul. 09, 2020).

[5] E. Craig, "Visualize Code in Virtual Reality," *Digital Bodies*, Jun. 10, 2019. https://www.digitalbodies.net/vr-experience/visualize-code-in-virtual-reality/ (accessed Jul. 09, 2020).

[6] K. Hanson and B. E. Shelton, "Design and Development of Virtual Reality: Analysis of Challenges Faced by Educators," *J. Educ. Technol. Soc.*, vol. 11, no. 1, pp. 118–131, 2008.

[7] A. Onishi, S. Nishiguchi, Y. Mizutani, and W. Hashimoto, "A Study of Usability Improvement in Immersive VR Programming Environment," in *2019 International Conference on Cyberworlds (CW)*, Oct. 2019, pp. 384–386, doi: 10.1109/CW.2019.00073.

[8] H.-J. Joo, H.-B. Song, and M.-K. Park, "A Study on Visual Programming Platform Design for VR/AR SW Education," in *Frontier Computing*, Singapore, 2019, pp. 560–565, doi: 10.1007/978-981-13-3648-5_65.

[9] T. Sugiyama, S. Konno, T. Kinoshita, K. Sugawara, and N. Shiratori, "Interaction techniques for visual programming based design of the 3D object's behaviors and its implementation," in *Proceedings Twelfth International Conference on Information Networking (ICOIN-12)*, Jan. 1998, pp. 722–725, doi: 10.1109/ICOIN.1998.648615.

[10] K. Nagaoka, N. Osawa, K. Mochizuki, H. Takahashi, E. Nishina, and F. Saito, "Evaluation of a 3-D visual programming environment in an introductory course of object-oriented programming," in *30th Annual Frontiers in Education Conference. Building on A Century of Progress in Engineering Education. Conference Proceedings (IEEE Cat. No.00CH37135)*, Oct. 2000, vol. 1, p. T4C/12 vol.1-, doi: 10.1109/FIE.2000.897664.

[11] M. Chandramouli, M. Zahraee, and C. Winer, "A fun-learning approach to programming: An adaptive Virtual Reality (VR) platform to teach programming to engineering students," in *IEEE International Conference on Electro/Information Technology*, Jun. 2014, pp. 581–586, doi: 10.1109/EIT.2014.6871829.

[12] "Visual Programming Guide | 2019 Overview of Available Languages and Software Tools," *Postscapes*. https://www.postscapes.com/iot-visual-programming-tools (accessed Jul. 09, 2020).

[13] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch Programming Language and Environment," *ACM Trans. Comput. Educ.*, vol. 10, no. 4, pp. 1–15, Nov. 2010, doi: 10.1145/1868358.1868363.

[14] S. Hayden, "VR Web Browser' Supermedium' No Longer in Active Development," *Road to VR*, Oct. 29, 2019. https://www.roadtovr.com/supermedium-vr-browser-stops-development/ (accessed Jul. 09, 2020).

[15] C. Y. P. on February 15, 2018 in Featured Article, Games, T. for V. development, and WebVR, "Create VR on the Web using Unity3D – Mozilla Hacks - the Web developer blog," *Mozilla Hacks – the Web developer blog*. https://hacks.mozilla.org/2018/02/create-vr-on-the-web-using-unity3d (accessed Jul. 09, 2020).

[16] A. Cannavò, C. Demartini, L. Morra, and F. Lamberti, "Immersive Virtual Reality-Based Interfaces for Character Animation," *IEEE Access*, vol. 7, pp. 125463–125480, 2019, doi: 10.1109/ACCESS.2019.2939427.

[17] F. Lamberti, G. Paravati, V. Gatteschi, A. Cannavò, and P. Montuschi, "Virtual Character Animation Based on Affordable Motion Capture and Reconfigurable Tangible Interfaces," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 5, pp. 1742–1755, May 2018, doi: 10.1109/TVCG.2017.2690433.

[18] D. Vogel, P. Lubos, and F. Steinicke, "AnimationVR - Interactive Controller-based Animating in Virtual Reality," in *2018 IEEE 1st Workshop on Animation in Virtual and Augmented Environments (ANIVAE)*, Mar. 2018, pp. 1–6, doi: 10.1109/ANIVAE.2018.8587268.

[19] M. J. Schuemie, P. van der Straaten, M. Krijn, and C. A. P. G. van der Mast, "Research on Presence in Virtual Reality: A Survey," *Cyberpsychol. Behav.*, vol. 4, no. 2, pp. 183–201, Apr. 2001, doi: 10.1089/109493101300117884.

[20] E. G. Milano, E. Pajaziti, S. Schievano, A. Cook, and C. Capelli, "P369 Patient specific virtual reality for education in congenital heart disease," *Eur. Heart J. - Cardiovasc. Imaging*, vol. 21, no. Supplement_1, p. jez319.218, Jan. 2020, doi: 10.1093/ehjci/jez319.218.

[21] N. V. Thuc, "Design of Adjustable Software for Fault Detection, Isolation and Recovery of Attitude Determination and Control System in MicroDragon Satellite," p. 122.

[22] K. Wood, R. E. Cisneros, and S. Whatley, "Motion Capturing Emotions," *Open Cult. Stud.*, vol. 1, no. 1, pp. 504–513, Dec. 2017, doi: 10.1515/culture-2017-0047.

[23] L. T. D. Paolis, "Virtual and Augmented Reality Applications," p. 23.

[24] "Programming in Virtal Reality: An Interview With The Creator of RiftSketch." https://cognitive3d.com/blog/programming-in-virtual-reality (accessed Jul. 09, 2020).

[25] L. Blackwell, N. Ellison, N. Elliott-Deflo, and R. Schwartz, "Harassment in Social VR: Implications for Design," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, Mar. 2019, pp. 854–855, doi: 10.1109/VR.2019.8798165.

[26] S. Gunkel, M. Prins, H. Stokking, and O. Niamut, "WebVR meets WebRTC: Towards 360-degree social VR experiences," in *2017 IEEE Virtual Reality (VR)*, Mar. 2017, pp. 457–458, doi: 10.1109/VR.2017.7892377.

[27] K. Habuka and Y. Shinagawa, "Creating VR scenes using fully automatic derivation of motion vectors," in *IEEE Virtual Reality 2004*, Mar. 2004, pp. 225–226, doi: 10.1109/VR.2004.1310081.

[28] M. Zyda, "From visual simulation to virtual reality to games," *Computer*, vol. 38, no. 9, pp. 25–32, Sep. 2005, doi: 10.1109/MC.2005.297.

[29] A. Cannavò, C. Demartini, L. Morra, and F. Lamberti, "Immersive Virtual Reality-Based Interfaces for Character Animation," *IEEE Access*, vol. 7, pp. 125463–125480, 2019, doi: 10.1109/ACCESS.2019.2939427.

[30] B. A. Myers, "Taxonomies of visual programming and program visualization," *J. Vis. Lang. Comput.*, vol. 1, no. 1, pp. 97–123, Mar. 1990, doi: 10.1016/S1045-926X(05)80036-9.

[31] M. Resnick *et al.*, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009, doi: 10.1145/1592761.1592779.

[32] "The Influences of the 2D Image-Based Augmented Reality and Virtual Reality on Student Learning on JSTOR." https://www-jstor-org.kras1.lib.keio.ac.jp/stable/26196123?Search=yes&resultItemClick=true&searchText=VR+development&searchUri=%2Faction%2FdoBasicSearch%3FQuery%3DVR%2Bdevelopment&ab_segments=0%2Fbasic_SYC-5187_SYC-5188%2F5188&refreqid=fastly-

default%3A09ac78343715bbfcb02d5cc3b4d91fdc&seq=1#metadata_info_tab_contents (accessed Aug. 17, 2020).

[33] H. F. Manesh and M. Hashemipour, "A New Software Development Tools with Virtual Reality for Computer Integrated Manufacturing Requirements Analysis," *SAE Trans.*, vol. 115, pp. 908–916, 2006.

[34] D. W. Sari *et al.*, "Virtual reality program to develop dementia-friendly communities in Japan," *Australas. J. Ageing*, doi: 10.1111/ajag.12797.

[35] M. Servera, B. Saez, and J. M. G. Mir, "Feasibility of a virtual reality program to treat children with fear of darkness with nonexpert therapists," *Rev. Psicol. Clin. Con Ninos Adolesc.*, vol. 7, no. 2, pp. 16–21, May 2020, doi: 10.21134/rpcna.2020.07.2.2.

[36] W. R. Sherman, "VR 2012 tutorial proposal title: An in-depth introduction to virtual reality programming," in *2012 IEEE Virtual Reality Workshops (VRW)*, Mar. 2012, pp. 1–2, doi: 10.1109/VR.2012.6180944.

[37] A. Onishi, S. Nishiguchi, Y. Mizutani, and W. Hashimoto, "A Study of Usability Improvement in Immersive VR Programming Environment," in *2019 International Conference on Cyberworlds (CW)*, Oct. 2019, pp. 384–386, doi: 10.1109/CW.2019.00073.

Appendix

A. Local Data Saving Code

```
namespace ProjectLib
{
    public class Project
    {
        private const string projectName = "proName";
        public string proName
        {
            get { return PlayerPrefs.GetString(proName); }
            set { PlayerPrefs.SetString(proName, value); }
        }

private const string projectNumber = " proNumber ";
        public int proNumber
        {
            get { return PlayerPrefs.GetInt(proNumber, 0); }
            set { PlayerPrefs.SetInt(level, value); }
        }
        private const string sceneName = "sceName";
        public string sceName
        {
```

```
            get { return PlayerPrefs.GetString(sceName); }
            set { PlayerPrefs.SetString(sceName, value); }
        }
……
}
```

B. IntoLib Code

```
public class IntoLib : MonoBehaviour {
	private const string Name = "Scene_InanimateObjectMenu";

	// Use this for initialization
	void Start () {

}

// Update is called once per frame
void Update () {

}

	private void OnTriggerEnter(Collider other)
	{
		if (other.tag == "Controller")
		{
			print("aaaa");
			SceneManager.LoadScene(Name);
		}
	}
}
```

C. Show Attribution Code

```
public class ShowAttribution : MonoBehaviour {


    private GameObject attrbutionBox;
    private string attText;


// Use this for initialization
void Start () {
        attrbutionBox = GameObject.Find("AttributionBox");
        attrbutionBox.GetComponent<Renderer>().enabled = false;
        attText = "position:" + gameObject.transform.position.ToString();
    }


    private void OnTriggerEnter(Collider other)
    {
        if(other.tag == "Controller")
        {
            attrbutionBox.GetComponent<TextMeshPro>().SetText(attText);
            attrbutionBox.GetComponent<Renderer>().enabled = true;
        }


    }
```

```
    private void OnTriggerExit(Collider other)
    {
        if (other.tag == "Controller")
            attrbutionBox.GetComponent<Renderer>().enabled = false;
    }

}
```

D. Show Function Box Code

```
public class ShowFunctionBox : MonoBehaviour
{
    public GameObject[] func;
    bool[] funcLine;
    void Start()
    {
        gameObject.GetComponent<Renderer>().enabled = false;
        funcLine = new bool[func.Length];
        for (int i = 0; i < func.Length; i++)
        {
            func[i].GetComponent<Renderer>().enabled = false;
            funcLine[i] = false;
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if(other.tag=="Function")
        {
            gameObject.GetComponent<Renderer>().enabled = true;
            other.GetComponent<Renderer>().enabled = false;
            for (int i = 0; i < funcLine.Length; i++)
```

```
            {
                if (funcLine[i] == false)
                {
                    funcLine[i] = !funcLine[i];
                    break;
                }
            }
            for (int i = 0; i < func.Length; i++)
            {
                if (funcLine[i] == true)
                func[i].GetComponent<Renderer>().enabled = true;
            }
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Function")
        {
            gameObject.GetComponent<Renderer>().enabled = false;
        }
    }
}
```

E. WriteToFile Code

```
public class WriteToFile : MonoBehaviour {


    private void OnGUI()
    {
        GUIStyle buttonStyle = new GUIStyle(GUI.skin.button);
        buttonStyle.fontSize = 25;
        if (GUI.Button(new Rect(10, 10, 450, 80), "Send information to file",
buttonStyle))
        {
            StartCoroutine(sendTextToFile());
        }
    }


    IEnumerator sendTextToFile()
    {
        bool successful = true;


        WWWForm form = new WWWForm();
        form.AddField("DataName1", "Data1balabalabala");
        form.AddField("DataName2", "Data2balabalabala");
        WWW www = new WWW("http://localhost:9000/fromunity.php",
form);
```

```
        yield return www;
        if(www.error != null)
        {
            successful = false;
        }
        else
        {
            Debug.Log(www.text);
            successful = true;
        }
    }
```

F. Controller Code

```
public class Controller : MonoBehaviour {

public GameObject ObjectMenu;


private void Awake(){

ObjectMenu.SetActive(false);

ViveInput.AddPress (HandRole.LeftHand, ControllerButton.Menu,

OnMenuPress);

}


private void OnDestory(){

ViveInput.RemovePress (HandRole.LeftHand, ControllerButton.Menu,

OnMenuPress);

}

private void OnMenuPress(){

//Debug.Log ("menu press");

ObjectMenu.SetActive(true);

}

}
```

G. Select Object Types Code

```
public class SelectObjectTypes : MonoBehaviour {


bool enable;
        public GameObject ObjectTypes;
// Use this for initialization
void Start () {
ObjectTypes.SetActive(false);
}


public void OnClickObjectButton()
{
Debug.Log("SELECT OBJECT TYPES");
ObjectTypes.SetActive(true);
}
}
]
```

H. Select A Scene Code

```
public class SelectAScene : MonoBehaviour {


public GameObject[] sceneModels;
public int length;
private int selectedIndex = 0;
// Use this for initialization
void Start () {
length = sceneModels.Length;
CharacterShow();
}


   void CharacterShow() {
            sceneModels[selectedIndex].SetActive(true);
            for (int i = 0; i < length; i++)
            {
                  if (i != selectedIndex)
                {
               sceneModels[i].SetActive(false);
                  }
            }
      }
```

```
public void OnNextButtonClick()
{
selectedIndex++;
        selectedIndex %= length;
        CharacterShow();
}

public void OnPrevButtonClick()
{
        selectedIndex--;
        if (selectedIndex == -1)
        {
            selectedIndex = length - 1;
        }
        CharacterShow();
    }
}
```

I. Write To File Code

```
public class WriteToFile : MonoBehaviour {


    private void OnGUI()
    {
        GUIStyle buttonStyle = new GUIStyle(GUI.skin.button);
        buttonStyle.fontSize = 25;
        if (GUI.Button(new Rect(10, 10, 450, 80), "Send information to file",
buttonStyle))
        {
            StartCoroutine(sendTextToFile());
        }
    }


    IEnumerator sendTextToFile()
    {
        bool successful = true;


        WWWForm form = new WWWForm();
        form.AddField("DataName1", "Data1balabalabala");
        form.AddField("DataName2", "Data2balabalabala");
        WWW www = new WWW("http://localhost:9000/fromunity.php",
form);
```

```
        yield return www;
        if(www.error != null)
        {
            successful = false;
        }
        else
        {
            Debug.Log(www.text);
            successful = true;
        }
    }
```

J. Drag Object Code

```
[RequireComponent(typeof(MeshCollider))]

public class DragObject : MonoBehaviour
{

private Vector3 screenPoint;
private Vector3 offset;

void OnMouseDown()
{
    screenPoint =
Camera.main.WorldToScreenPoint(gameObject.transform.position);

    offset = gameObject.transform.position -
Camera.main.ScreenToWorldPoint(new Vector3(Input.mousePosition.x,
Input.mousePosition.y, screenPoint.z));

}

void OnMouseDrag()
{
```

```
        Vector3 curScreenPoint = new Vector3(Input.mousePosition.x,
Input.mousePosition.y, screenPoint.z);


        Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenPoint)
+ offset;
        transform.position = curPosition;


        }


        }
```

K. On Click Functions Code

```
public class OnClickFunctions : MonoBehaviour
{
    // Start is called before the first frame update
    public GameObject[] funcBox;

    void Start()
    {
        for (int i=0; i < funcBox.Length; i++)
        {
            funcBox[i].GetComponent<Renderer>().enabled = false;
        }
    }

    public void OnClickStand()
    {
        funcBox[0].GetComponent<Renderer>().enabled = true;
    }

    public void OnClickRun()
    {
        funcBox[1].GetComponent<Renderer>().enabled = true;
```

```
}

public void OnClickWalk()
{
    funcBox[2].GetComponent<Renderer>().enabled = true;
}}
```

L. Show Function Box Code

```
public class ShowFunctionBox : MonoBehaviour
{
    public GameObject[] func;
    bool[] funcLine;
    void Start()
    {
        gameObject.GetComponent<Renderer>().enabled = false;
        funcLine = new bool[func.Length];
        for (int i = 0; i < func.Length; i++)
        {
            func[i].GetComponent<Renderer>().enabled = false;
            funcLine[i] = false;
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if(other.tag=="Function")
        {
            gameObject.GetComponent<Renderer>().enabled = true;
            other.GetComponent<Renderer>().enabled = false;
            for (int i = 0; i < funcLine.Length; i++)
```

```
            {
                if (funcLine[i] == false)
                {
                    funcLine[i] = !funcLine[i];
                    break;
                }

            }
            for (int i = 0; i < func.Length; i++)
            {
                if (funcLine[i] == true)
                func[i].GetComponent<Renderer>().enabled = true;
            }

        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Function")
        {
            gameObject.GetComponent<Renderer>().enabled = false;
        }    }}
```

M. Show Hand Code

```
public class ShowHand : MonoBehaviour
{
        // Update is called once per frame
    void Update()
    {
        Vector3 pos= Camera.main.WorldToScreenPoint(transform .position );
        Vector3 mousePos = new Vector3(Input.mousePosition.x,
Input.mousePosition.y, pos.z);
        transform.position = Camera.main.ScreenToWorldPoint(mousePos );
    }
}
```

N. Function Recognizer

```
public class FunRecognizer : MonoBehaviour
{
    private PhraseRecognizer funRec;
    public string funs = "Play Sounds";
    public ConfidenceLevel    confL = ConfidenceLevel.Medium;

    // Start is called before the first frame update
    void Start()
    {
        print(PhraseRecognitionSystem.isSupported);
        if(funRec == null)
        {
                funRec =    new KeywordRecognizer(funs, confL);
                funRec.OnPhraseRecognized += FunRec_OnFunRecognized;
                funRec.Start();
                Debug.Log("Created Successfully");
        }
    }

    private void FunRec_OnFunRecognized(PhraseRecognizedEventArgs args)
    {
        SpeechRecognition();
```

```
        print(args.text);
    }

    void SpeechRecognition()
    {
        print("SpeechRecognition");
    }

    private void OnDestroy()
    {
        funRec.Dispose();
    }
}
```

O. Function Activation

```
public class FunctionActivation : MonoBehaviour
{
    //public bool funs;
    public AudioSource src;
    public GameObject funSpace;
  // Start is called before the first frame update
  void Start()
  {
      //funs = false;
  }

  // Update is called once per frame
  void Update()
  {
      if(funSpace.GetComponent<FunctionIn>().isFunIn == true)
      {
          src.Play();
          funSpace.GetComponent<FunctionIn>().isFunIn = false;
      }

  }
}
```