UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

School of Telecommunications Engineering

Virtual Reality and People with Functional Diversity

End of Degree Project

Bachelor's Degree in Telecommunication Technologies and Services Engineering

AUTHOR: Li, Weiyao

Tutor: Hernandez Franco, Carlos Alberto

ACADEMIC YEAR: 2023/2024

Undergraduate Project Report
2023/24

Virtual Reality and People with functional diversity

Name:	Li Weiyao
School:	International School
Class:	2020215107
QMUL Student No.:	200977847
BUPT Student No.:	2020213226
Programme:	Telecommunications Engineering with Management

Date: 02-04-2024

Table of Contents

Abstract

This project was carried out in collaboration with the Universidad Politécnica de Valencia. In Valencia, individuals with physical or intellectual disabilities are commonly referred to as 'people with functional diversity'. The Universidad Politécnica de Valencia has been working to develop technology to assist these individuals. 'La Torre' Occupational Centre is a facility in Valencia that serves individuals with functional diversity. The centre provides a learning and activity space for individuals with intellectual or physical disabilities. Some of these individuals may face challenges in recognising objects in the workshops and with basic objects, which can impede their mobility. The project aims to explore the potential of virtual reality technology to assist these individuals in overcoming these challenges. This paper discusses the potential benefits of using Virtual Environments (VE) in VR to aid individuals in recognising and identifying objects. This can help them to adapt more quickly to the various workshop objects in the centre and acquire skills more efficiently. The project employs Unity3D to create a virtual space, where users are encouraged to identify particular objects and place them in assigned locations, based on character prompts. Furthermore, the application features a time tracking function, which can provide staff with useful insights.

Keywords:VR,Unity,people with functional diversity,recognisation

摘要

本项目是与瓦伦西亚理工大学所一起完成的。在瓦伦西亚，people with functional diversity 是一个常见的概念。人们对于生理或智力方面有缺陷的人的关注逐渐增加。瓦伦西亚理工大学一直在致力于用科技去帮助这些特殊人群。"La Torre"occupational 中心是位于瓦伦西亚的一所致力于帮助 people with functional diversity 的中心。它为一些智力或者生理方面有缺陷的人提供了学习与活动的场所。在这里的人们会需要努力辨认不同 workshop 中不同的物品，他们很多人对于认知某些基本的物品具有障碍，所以要实现让他们在 workshop 中正常活动需要很多的实地学习时间。本项目的目的是运用虚拟现实来帮助此中心的人们。本文将会详细介绍如何运用 VR 中的 VE（虚拟环境）来帮助此中心的人们去认识和辨别物体，因此他们可以更快的适应中心中不同 workshop 的物品，以此来完成技能的学习。此项目是基于 unity3D 来开发一个应用，创造一个虚拟的房间（或者说是房子），并且要求用户根据人物提示辨别特定的物品并将其放到指定的位置。另外，这个应用还有记录所用时间的功能，让此中心的工作人员更好的得到反馈。

关键词

虚拟现实（VR），unity，多样化人群，辨别

Chapter 1: Introduction

1.1 Motivation

In Spain, people with functional diversity is often mentioned. People are beginning to pay attention to these special people and to give them care and attention. It is clear that these people are often unable to find a job or have no friends because of their physical or intellectual disabilities, so it is important to provide them with a platform to work and communicate with each other.

As stated in the abstract, Valencia has an occupational centre named 'La Torre' that caters to individuals with physical or mental disabilities. The individuals are referred to as 'users' in the centre, and in this paper, the term 'user' also encompasses people with functional diversity. The centre's objective is to equip them with skills and promote their well-being. To achieve this, the centre has a team of staff who assist and guide them through various tasks. The centre offers various workshops, each equipped with different tools for different activities. For instance, some rooms are stocked with cardboard and scissors, allowing users to create paper crafts with the assistance of staff. However, it is important to note that some users may experience difficulty in recognising and identifying objects, leading to prolonged identification times or even forgetfulness of object characteristics. Therefore, some users who are not familiar with the items in this workshop may require additional time to use them. Additionally, it may require significant effort to familiarise themselves with the objects and identify them, which could negatively impact the experience of other users.

In this way, the aim of the project became apparent: to create a similar environment using virtual reality, providing objects for users to recognise, and after they had become familiar with the objects and the environment, they could begin to use the tools and complete their work in the real workshop.

1.2 Objectives and Scope of the Project

In this project, the users are all users of the "La Torre" centre. There are about 80 of them and they have different physical or mental deficiencies. For example, according to the author's field research and observations, some of them are unable to speak due to brain damage, others have memory and cognitive problems, are unable to use the tools of daily life and have a limited knowledge of the written word.

The plan is to develop an app that provides cognitive skills training through scenario-based exercises. The app will also enhance users' literacy skills by allowing them to read the task text. Additionally, the app will include a recording function to enable the centre's staff to assess the users' progress in both cognitive and literacy aspects. This project utilises virtual reality (VR) technology, which is described in detail in chapter 2. The chapter outlines the differences in hardware requirements between this application and traditional VR applications, as well as the environments and compilation languages used in this project. The language used is clear, concise, and objective, with a formal register and precise word choice. The text adheres to conventional structure and formatting features, with consistent citation and footnote style.

The chapter outlines the differences in hardware requirements between this application and traditional VR applications, as well as the environments and compilation languages used in this project. The content has not been altered beyond improving its adherence to the desired characteristics. Basic programming knowledge, such as data structures and agile development, will be essential in the development and debugging of the application. This project is specific to the design of tasks and target group of users at the 'La Torre' centre. The chapter on further work will discuss the eventual appearance of the application.

1.3 Outline of the Report

In addition to the summary and introduction, this article includes the following sections:

Chapter2:Background

This section introduces the technology used in the project, Virtual Reality (VR), and its specific benefits. It also discusses current gaps in VR software and how the technology has been applied in specific cases.Additionally, the section defines 'people with functional diversity' and provides a brief discussion. The text will cover tools and hardware and provide descriptions. It will also cover the basics that need to be used.

Chapter 3: Design,Implementation and Testing

This section will detail the entire process of completing the application, including: 3D modelling, script implementation for different functions. The scripts for each function will be explained and discussed. More, some design considerations will be mentioned, i.e., why I chose to design the software the way I did.

Chapter 4: Results and Discussion.

This section will discuss the effects of this application, its impact on the user, and the functionality accomplished. There are also some shortcomings and areas for improvement that may be mentioned.

Chapter 5: Conclusion and Further Work

This section is designed to bring the different feedbacks together and to identify what needs to be done in the future based on the shortcomings summarised in the previous section. More, this section will also refer to the development of this application in the longer term (a year or several years from now). This section may also touch on my feelings and thoughts about helping the people with functional diversity.

References

Acknowledgement

Appendix

Risk Assessment

Environmental Impact Assessment

Chapter 2:Background

2.1 Briefly introduction of Virtual Reality and People with functional diversity

2.1.1 Virtual Reality

Virtual Reality (VR), is an emerging technology. It is widely used in games, films and industries that require interaction. It mainly focuses on immersive experience and sensory

realistic stimulation. Since 1989, when Jaron Lanier coined the term virtual reality, the development of virtual reality technology has been at a very high rate until today. With the rapid growth of gaming and entertainment, VR technology is being mentioned again. Nowadays, VR technology has become very popular, the related industries are very developed, and more importantly, with the support of Unity and other development software, the technology has become very friendly to some beginner programmers. This explains why the author was able to use this technology.

Some researchers define VR according to three attributes: (tele-)presence, interactivity, and immersion[]. Presence is usually defined as the feeling of being in a place but not really being there. interactivity is usually defined as the ability of the user to manipulate the environment. Different researchers have different definitions of immersion, e.g., Berg and Vance define it as "the set of technologies that enable people to immersively experience a world beyond reality"[https://ieeexplore.ieee.org/abstract/document/7500674]. Jerald, however, describes it as "a computer-generated digital environment that can be experienced and interacted with as if that environment was real. ".[]Overall, I consider Isabell Wohlgenannt, Alexander Simons' summary is general: “VR uses immersive technology to simulate interactive virtual environments or virtual worlds that allow users to participate subjectively and feel immersed.”[]

Some important terms about VR need to be understood: virtual environments. A virtual environment(VE) consists of "software representations of real (or imagined) agents, objects, and processes; and human-computer interfaces for displaying and interacting with these models" (Barfield). It is software that is created using VR technology, and this software is running on “hardware that may or may not be VR-based.”[]

At the "La Torre" occupational centre, some electronic equipment will be available for use and entertainment, however, VR headsets will not be available because of the cost. In addition, many users have very limited finger dexterity and their neck and head muscles are not as developed as the ordinary person's. Therefore, for these users, VR is a good choice. Therefore, for these users, VR headset devices are not the best option. Me chose to create a virtual environment using VR technology in order to make the software output from this project more accessible to the majority of people, and to have this software run on a tablet. This is different to most VR software, but it is the most suitable option for the users of this centre. One might think that this software is similar to a normal first person game, but there are many differences: normal games are more entertainment orientated, whereas the software produced by this project is more focused on training. Furthermore, the software focuses more on interactivity and is less concerned with entertainment elements such as fighting and puzzle solving.

In summary, this project uses VR technology to create virtual environments and customised tasks for the users of the "La Torre" centre, in order to train their cognitive, memory and literacy skills.

2.1.2 People with functional diversity

"People with functional diversity" is a recently coined term. It focuses primarily on emphasising the different aspects of each person's abilities or potential, rather than just their disabilities or impairments. It focuses on the uniqueness of the individual in some way, and the term is more inclusive and positive. It aims to eliminate discrimination and prejudice in

society and helps to promote inclusion and recognition of diversity in society. More, "emphasises their difference or diversity, values that enrich the world in which we live".[early term report]

2.2 The state of art on apps using VR helping people with functional diversity

VR is widely used in training and rehabilitation because of its ability to provide an immersive experience, and a number of studies have shown VR to be effective as a rehabilitation aid in the treatment of stroke, cerebral palsy, and Parkinson's disease.Anat and Inbal Maidan demonstrated [early] that TT + VR significantly improves gait speed, gait variability, and cognitive function in Parkinson's disease patients during normal walking and dual-tasking conditions, as well as during passage through physical obstacles. These improvements were maintained one month after training, suggesting that TT + VR can provide long-term benefits to patients. Patient satisfaction and enjoyment of the VR system was also high. In conclusion, VR provides a sense of immersion, which has been shown to improve performance and matching accuracy on cognitive tasks.

VR is effective in aiding treatment and rehabilitation.Nevertheless, in the domain of virtual reality and functional diversity, there is a significant need to further explore various avenues. Presently, most applications and methods are primarily focused on "rehabilitation" or "therapy," which are tailored towards individuals with disabilities and do not effectively cater to the functional diversity population. This implies that the majority of applications prioritize medical concerns while overlooking fitness and daily life aspects. At the "La Torre" center, users require more comprehensive training for daily life skills, such as the identification of various objects. However, there is a notable absence of applications specifically designed for the functional diversity population, and the lack of standardized evaluation criteria for user-friendliness further exacerbates the issue. Additionally, there are challenges related to user interaction within the virtual environment, particularly for individuals with limited mobility, as the usage of controls and activation of different buttons can impede complex processing. Furthermore, an analysis by María[Virtual Reality Game Analysis for People with Functional Diversity: An Inclusive Perspective] and others revealed that 92.5% of the applications designed for People with Functional Diversity utilize the English language, which poses a significant barrier for users at the "La Torre" center who primarily utilize the local Valencia language.

2.3 Tools and methods used in this project

unity is a real-time content development platform that allows users to create 3D scenes through rendering and texture mapping. unity supports the creation of apps on different platforms, including Android. The hardware support for this project is the "La Torre" centre tablet, which is in the android environment. Considering the simplicity and compatibility, I used unity to develop and create the application. unity uses C# to provide the scripts for the models, C# as an object oriented language is simple and easy to understand.

More, unity asset store provides many basic materials about 3D modelling. There are many elements available for download, and I used it to download some materials, textures, and lighting effects. These materials were used in the project and made the environment created by the software very realistic. In order to edit and write C# code, Visual Studio Code was used in this project, which is a source code editor that supports many different programming languages. It supports different plug-ins to help people do their programming work faster and easier.

DeepL is used as a very accurate translation software that is applied to translate relevant UI text. The people of the "La Torre" centre speak very little English, so for them, instructions and tasks need to be expressed in Spanish. Therefore, translation software was needed.

Finally, some programming related theories such as data structures, software engineering and other related knowledge, the same was applied to create this software kind.

Chapter 3:Design and Implementation

3.1 Determine the target population

As mentioned before, the target users are the users from “La Torre” occupational center. However, it need to compartmentalised a bit more. Some of the users can perfectly handle tasks like getting a scissors and cut the paper. They may have deficiencies in other areas of their brain species, such as not being able to speak properly, or they may have deficiencies in some of their muscles, such as not being able to control the strength and direction of their hands very well.For these users, it's not so much virtual reality simulations and training that they need, they need surgical or muscle-training equipment more than anything else. Therefore these people are not the main target users of this project.

On the contrary, some users of the "La Torre" centre have problems with memory and cognitive abilities. For example, they may take a long time to remember and recognise an object, which they quickly forget. A good way to train these users is to let them recognise different objects virtually through the software on their tablets and to train them to remember them over and over again, so that when they actually need to use the object, they will have to face the real object to recognise it." Some of the objects in the "La Torre" centre, such as drawing boards, cardboard, or tools for making clay, are managed in designated workshops.

There is only a limited amount of time in the day for users to familiarise themselves with them and use them. Therefore, for users with memory and cognitive problems, without prior training, it may take a long time for them to identify and familiarise themselves with the tools before starting the work they are interested in. This is a waste of time and resources for both them and the centre staff. More, these user groups are quite large, so it is essential for them to familiarise themselves with the site and the tools in advance in order to be efficient when they do use them.

Furthermore, some users may have difficulty in recognising words and diagrams. They can't understand the meaning of words quickly or they are slow readers. The UI of this software contains some task-based text, which can train these users' literacy and graphic skills.

Finally, the documentation of the project can be used as a reference for people with functional diversity in the "La Torre" centre. They can judge, based on the time it takes different people to complete a task, whether a particular individual needs retraining or is already familiar with the assigned tasks and objects and can therefore go on to work in a real workshop.

In conclusion, this product is aimed at users of the "La Torre" centre who have difficulties with cognitive skills, memory and literacy, as well as the staff who assist them.

3.2 Unity Modeling and Design

Below I'll describe how I created a room, and more along with the design of the character model and the UI interface

3.2.1 3D modeling of the entire environment

Object creation:

To create basic models (such as floors or furniture), meshes in unity are often used. A mesh defines the shape and geometric features of an object. In 3D graphics, the shape of an object usually consists of vertices, edges and faces, and a mesh is a collection of these vertices, edges and faces. Each face can be texture mapped and coloured individually so that visually rich 3D models can be created [网格 - Unity 手册].
A mesh can control the size, appearance and structure of an object, for example defining an object as a sphere, square or polygon. In addition, unity provides rendering and display for meshes, which means that objects can be rendered perfectly in applications. What's more, meshes also provide collision volume detection, so that there will be more realistic physics when interacting with other models (e.g. character models). Finally, the surface features and geometry of the mesh have an important influence on light calculation and shadow casting, which means that using a mesh makes the model more realistic.[网格 - Unity 手册]

In addition, the box collider is the base model component. box collider provides collision areas and volumes for items, and it is typically used for simple rectangular or cubic models. Together with the mesh, it provides the collision logic for the model. In this project, when a character model walks towards a model with a collision component, the character gets stuck

instead of going straight through.

A prefab is a special resource in unity that provides universal properties for a class of models. Similar to class definitions in JAVA, prefabs provide modular objects and can be reused. When used specifically, objects can be instance from prefabs and applied to specific scenarios. In addition, the instantiated objects can have attributes or methods added according to specific needs.

Based on the above description, the presentation of each specific model part will be elaborated next.

In terms of floors:
I need to create a house and have the user traverse different areas in the house to complete tasks. The most basic of these is the floor. The first thing to create is a wooden floor for the living room section.

First I create a prefab and define it as a rectangle. Using a mesh filter, we define it as a 2.5m x 2.5m floor and set its height to 0.1m.(screenshot)

In addition, add the mesh renderder component to it so that its network can be rendered. In the Material option, I need to set the material for the rendering.Unity provides some common materials, I chose the wood material and applied it to the mesh so that the model looks like it's made of wood.

Further more, add the mesh renderder component to it so that its network can be rendered. In the Material option, I need to set the material for the rendering.Unity provides some common materials, I chose the wood material and applied it to the mesh so that the model looks like it's made of wood.

Next set up the light. Turn on the shadows option to make the model have shadows. probes in unity stands for light rendering, it acts as a probe that captures light information from the scene and applies it to the surface of the object. light probes are used to capture information about the light in the scene, including the intensity, colour and direction of the light source. When the object is in the area covered by the light probes, the Mesh Renderer will use the light information captured by the light probes to calculate the lighting effect of the object, so as to achieve the rendering. reflection probes are used to collect the reflected light from the other objects, and similarly, when the object is in the area covered by the reflection probes, the Mesh Renderer will use the light information captured by the light probes to calculate the lighting effect of the object. When the object is in the area covered by the reflection probes, the Mesh Renderer will use the reflected light captured by the reflection probes to calculate the lighting effect of the object. Together, the light source and the reflected light shape the lighting effect of the model. Here, I use the blend probes, which allow smooth transitions between probes, resulting in a natural continuum of lighting effects.

Add a box collider to the floor so that the character can step on it without falling off. Set the size of the collider to 2.5 x 2.5 x 0.1, which corresponds to the length, width and height of the floor. The floor also has a Furniture info script that stores the location and name of the floor. This will be useful for later task completion.

In a bathroom, it's obviously not common sense to use a wooden floor. So I modified the material of the wooden floor to make it appear to be made of tiles.

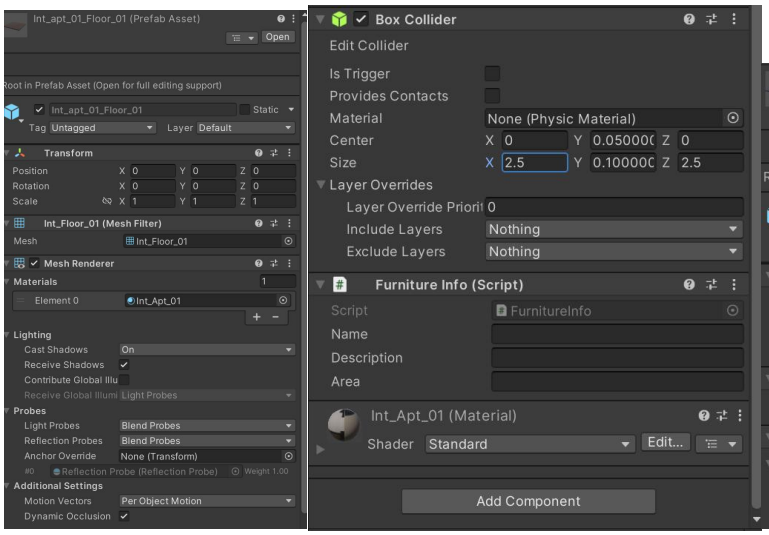
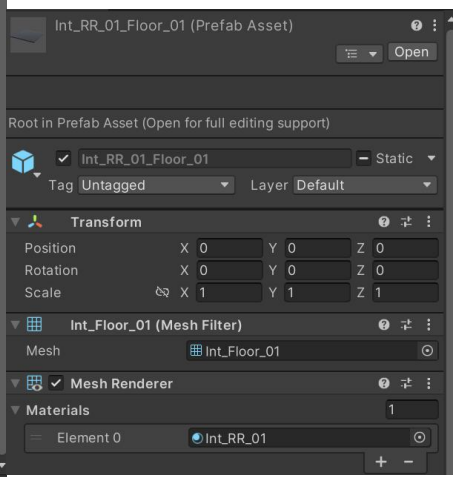


figure: setting of floor(Prefab,normal)



setting of floor(different in materials)

Once the prefab is available, I can reuse it. I flattened 60 floors in the Z=0 plane, giving the house its initial floor. And, in the bathroom I used 4 floors made of tiles.

In terms of wall:

Turn the floor into a wall by placing it vertically. So the setup regarding probes and box collider is the same as the floor. However, since the floor is significantly smaller than the wall, there are still a few details that need to be adjusted. Adjust the length of the wall to 4m so that the height of the house can be made sufficient. Also, the walls are not usually made of wood or tiles, so for the materials choice I chose a solid colour to make the walls look painted.

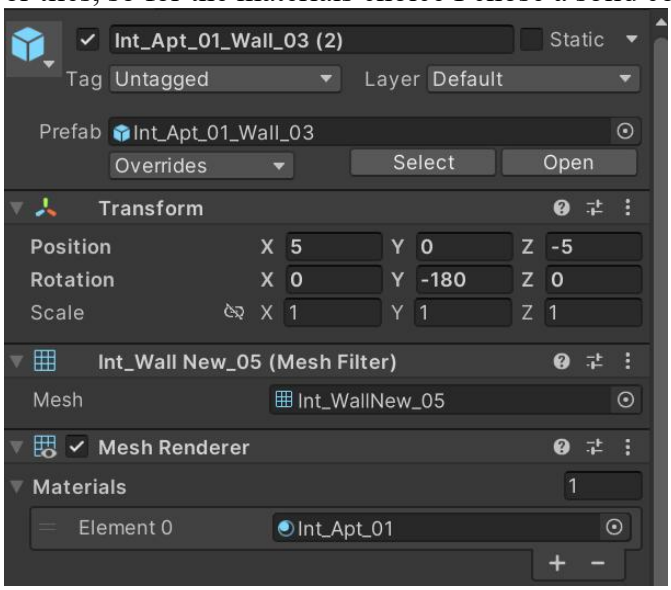


figure:basic wall

The basic wall design is similar to the floor, so it's very simple. However, walls differ from floors in one respect: they are "inlaid" with objects such as windows, doors and other objects. So I also needed some special shapes for the walls, so that doors and windows could fit into the walls.

The construction of the mesh model requires the use of some 3D modelling software such as Autodesk Maya and the output of files in fbx format. This is beyond my ability. Luckily unity

provides some free and open source mesh models, and I found the relevant fbx files in the unity asset store so that I could use this particular shape in my project. [assetstore.unity.com]. For the textures, I chose the same ones as for the basic walls to make the whole environment look harmonious.

Regarding the collider settings, this wall is also different from the basic wall. We need to use a mesh collider, if we use a box collider, the collision volume of the wall will be very different from the volume displayed in the scene. For example, if a wall with a special shape is hollowed out in the middle, in order to put a window model in it, if we use the box collider, the hollowed out part will also be counted as part of the collision volume, and the window won't be able to fit in it.Mesh Collider defines collision zones based on the shape and geometry of the mesh, and it can be applied to arbitrary shapes. objects. Using the mesh collider, the window can be placed without collision volume in the "hollowed out" part of the mesh.

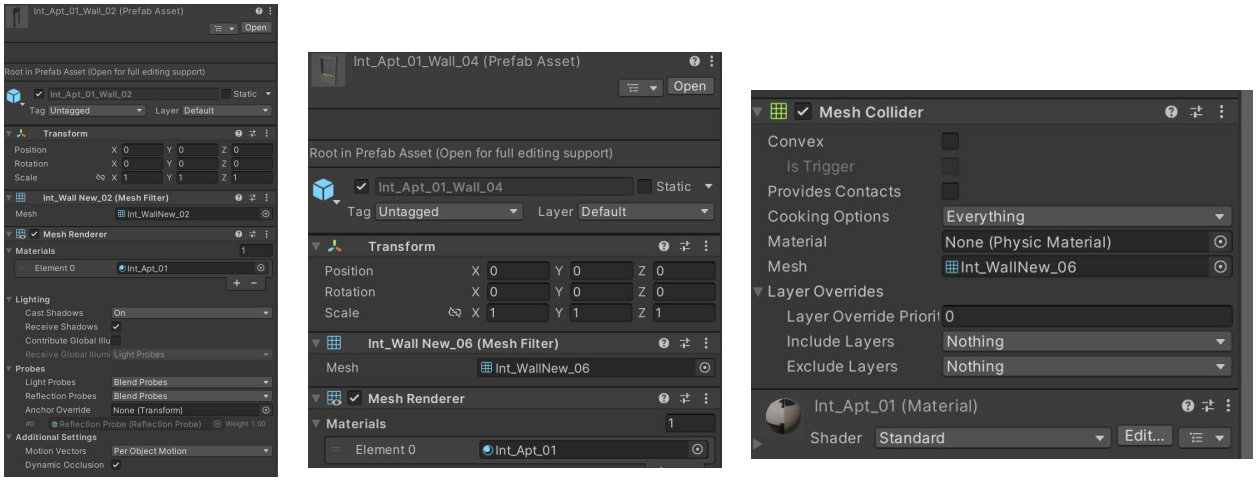


figure:settings of special walls

Ceilings:

The ceiling is very uniform in design; it is the same area as the floor. The advantage of this is that the ceiling can be covered and finished by simply placing it 4m from the floor. The materials used for the ceilings are similar to those used for the walls, in solid colours, so that they look common sense and harmonious.

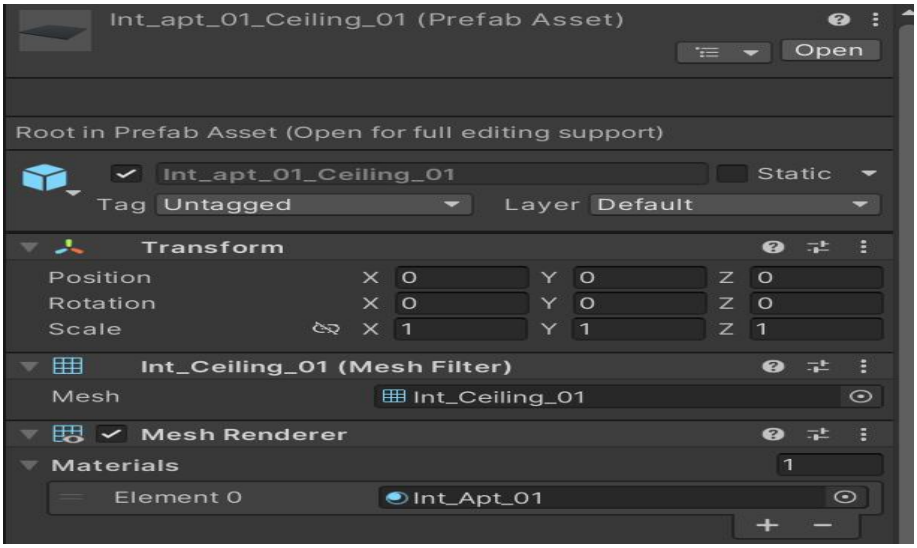


Figure: settings of ceiling.

Doors and windows:
Doors and windows are also one of the basic amenities of a house. Doors consist of a door frame, a door handle and a door, and the reason for this is to fulfil the function of opening the door. Regarding the door frame, it is possible to use a special similar model of the wall and adjust its width small, making it conform to common sense. The door frame is 3m high, 1.6m long and 0.3m thick like the wall. for the material I chose another wooden material, which is different from that of the floor, but also looks like a wooden structure. The body of the door is a little smaller than the frame, 1.4m long and 2.8m high. It uses the same materials as the door frame, making the whole door harmonious. More, the door handle is an irregular shape and its materials are metal.

The Animator component is one of the most important components used in Unity to control the animation of game objects. It allows one to create, manage and control the animation state machine of game objects and play and switch animations at runtime. It is used by Animation State Machine to manage the animation state of the game object. It is used with some furniture that needs to be interacted with, such as the doors mentioned earlier. Again, the animation was beyond my capabilities, so I found the animation in the unity asset store and applied it to the scene. The animation depicts the opening and closing of the door. However, the animation alone is not enough to complete the interaction between the human and the door, a script file is needed to control the timing and movement of the opening and closing of the door.

In summary, the door has an animator component in addition to the mesh related components and box collider for rendering and controlling collisions to be used to incorporate specific animations. More over, there is a script to control the presentation of the animation.

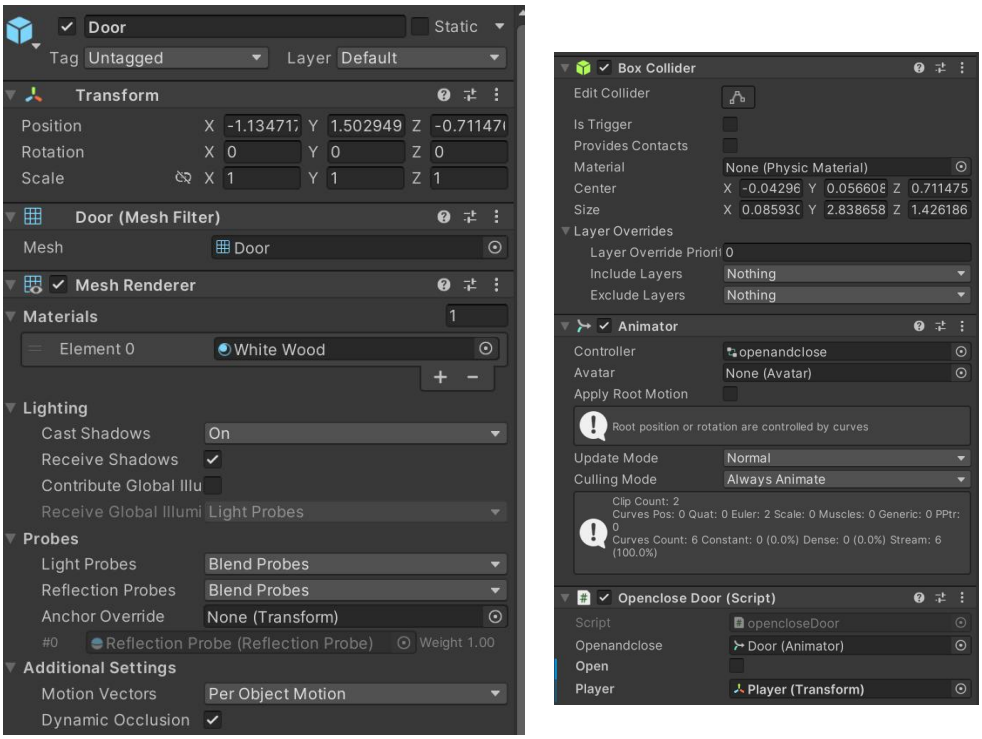


figure:setting of the doors

Similarly, window setups require frames and bodies. The difference is that the window needs to have glass inside so that the user can see the scene outside through the glass. Implementing the glass effect is also very simple, just set the materials of the glass part to glass. It also requires animation and scripting support.

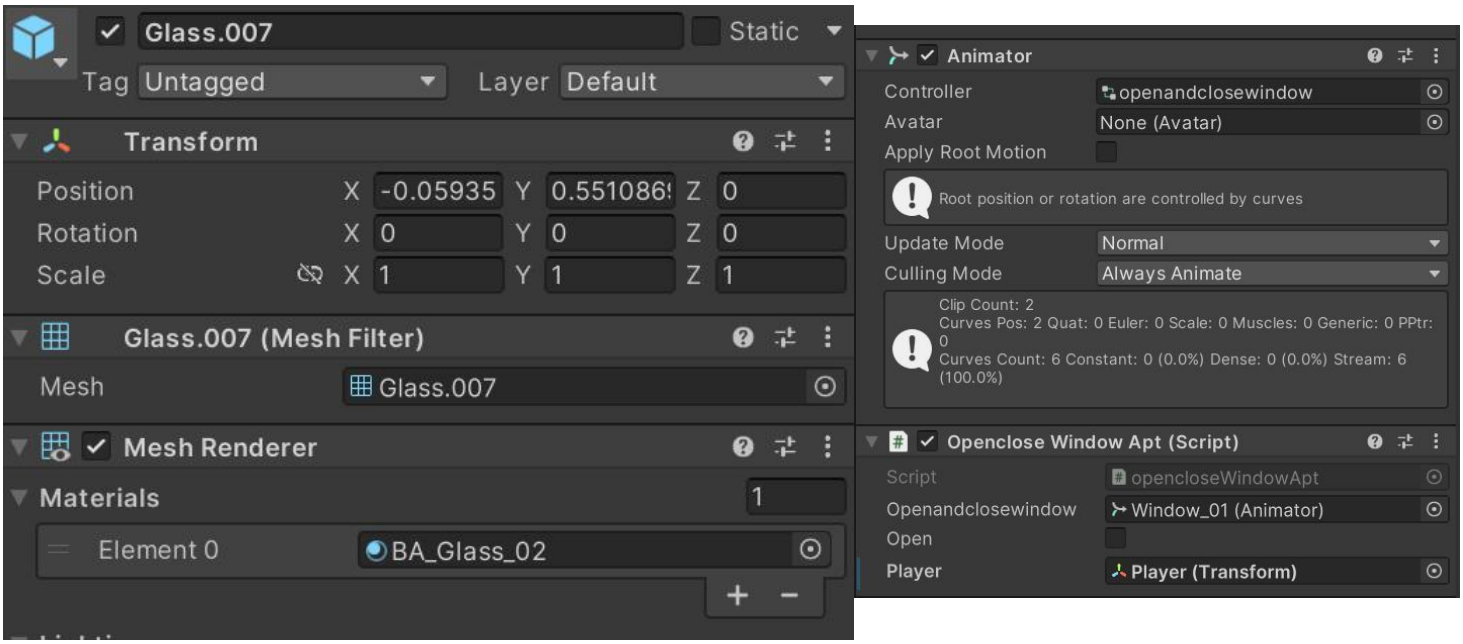


Figure: settings of windows

lights and probe:

To create a light source, I need to use empty objects. An empty object is a game object in unity, it doesn't have any rendering effects, but it can apply components and scripts. For the invisible light source, the null object is needed as a carrier to apply the light component and manipulate the position.

The light component is the component in unity that provides the light source. It creates lighting effects in the scene that affect the visual presentation and rendering of objects. It has the effect of illuminating and creating shadows. the default light source in unity is sunlight, and if there is an occlusion, the light will be dimmed. So I need to set up some light sources inside the house to simulate the effect of real daytime light inside the house. Also, I need to simulate the effect of some lights. I use point light, which simulates the ambient lighting effect around the object, emitting light in all directions. In other words, the objects are illuminated uniformly in all directions to enhance the overall brightness and visibility of the scene. Additionally, objects closer to the light source will be illuminated more strongly, and objects farther away from the light source will be illuminated less strongly. This makes the scene appear more three-dimensional and realistic.

Another important parameter in the light component is shadow. soft shadow makes the shadow edges softer and smoother, and makes the shadows of the objects look more natural and realistic, reducing the unnatural feeling caused by sharp shadow edges. Also, in indoor scenes, soft shadow is more in line with the characteristics of real light. This is good for simulating natural light or for scenes that need a warmer feel. Here I use soft shadow, which gives the room a gentle natural light and softens the scene.

As previously described, a probe is a component for detecting and rendering light sources.The Reflection Probe component is a similar component to the probe component.Reflection Probe captures information about the environment around objects in the scene. The captured environmental reflection information can be applied to the surface of the object so that the object reflects the correct view of the environment at different positions and angles. Moreover, Reflection Probe supports dynamic updating of environment reflection information, so that it can dynamically capture and update the reflection of the surrounding environment at runtime. This enhances the realism and fidelity of the scene, making objects appear more realistic and natural under the light, and making the scene dynamic.

Reflection probes, unlike light sources which require multiple settings, have a high performance overhead and multiple reflection probes may cause shadows to overlap. Therefore, I set up a reflector at the junction of the living room with the kitchen and the corridor to simulate the effect of reflecting light from objects in the house. I also placed a total of 13 light sources in the living room, kitchen, bathroom, and dining room to simulate the natural effect of lights and sunlight coming into the rooms.

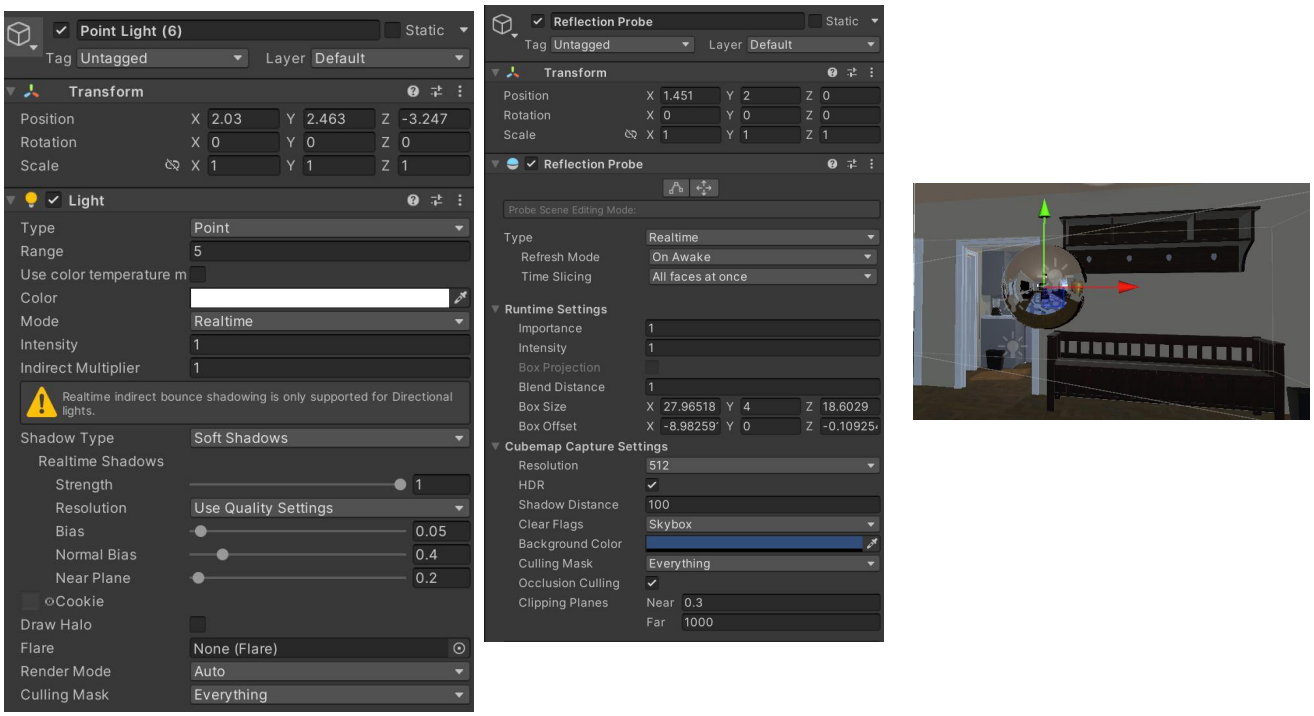


Figure:settings of lights and probe,position of probe

The location of the light source should be on the upper side of the entire room, which will allow the light to spread fully to every corner. To capture the light source, the height of the reflector probe should be in the centre of the room. And this height will make the reflected light from the object uniform in all directions.

At this point, the basic house takes shape and all that needs to be done now is to place specific furniture in specific areas. For example, the sofa is placed in the living room.

Furniture:

Simpler furniture, such as beds, can be modeled in a similar way to floors, by resizing and using different materials, and by adding the box collider component. Other furniture such as

paintings, sofas, sinks and other irregular shapes need to be modeled in the unity asset store by finding the appropriate prefab and applying it. Similarly, furniture with special shapes requires the mesh coillder component for collision simulation.Some furniture that requires interaction, such as closets and TV cabinets, have likewise been added with animations and scripts to control them.

Figure:special furniture:TV set.

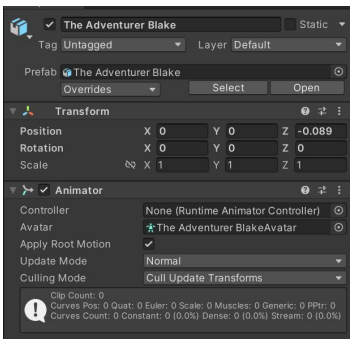
Figure: simple furniture:bed

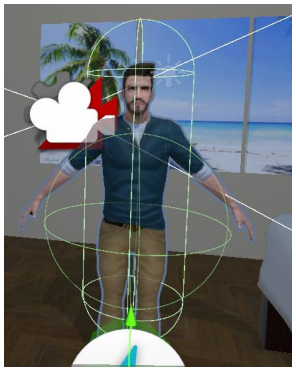
3.2.2 Modeling of characters (main view)

After creating the base environment, the next step is to create a character model that allows the user to manipulate and complete tasks. The character model also serves to allow the game camera to follow, which is the foundation and core of user interaction.

Character:

First create an empty object that contains a root and a skeleton. The root is also an empty object that serves the purpose of giving the following shot a follower. And the skeleton is a model of the rendered character.





As for the skeleton, the character model needs to involve animations related to moving or running, and the character as an uncomplicated shape needs to be created with 3D modeling software. In addition, the details of the face, clothing also need to be modeled and the related art background is required. Therefore, I chose a character model from the unity asset store and modified it to be used in the scene.Running animations are utilized in the character models.|

The Character Controller component in Unity is an important component for controlling character movement and collision detection. It allows to control the character's movement, jumping, crouching and other actions through scripts, realising the basic movement functions of the character. And the Character Controller handles the collision detection between the character and other objects in the scene, including walls, floors, obstacles and so on. When the character encounters a collision, it adjusts the character's movement state, such as stopping the movement, jumping and bouncing, and so on. Some parameter settings need to be understood: the radius determines the size and range of the character in the collision detection, too large a radius will make it difficult for the character to pass through the narrow space, so I set it to 0.28. centre is the centre of the character, and the program calculates the collision based on this and the model volume. Here I set it to the bottom of the character's feet. slope limit and step offset determine the maximum angle of the slope the character controller can climb and the maximum height of the steps the character controller can cross, in this project, the character walks on flat ground, so it makes sense to set them to a higher number. skin width determines the extent to which the character can penetrate an object. In order to prevent penetration, the skin width should be greater than 0.

The character also has a number of scripts that control the input, these enable the character to move in response to user input.

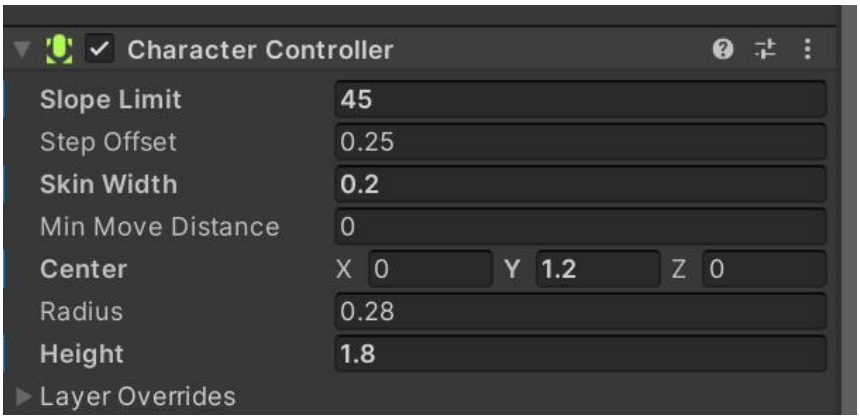


figure:setting of character controller.

Camera:

The lens determines how the user observes the scene. In this project, it should have the following functions: follow the movement of the character model, and rotate the perspective according to the user's input. To achieve this, I used two lenses, a main lens and a follow lens. This allows for more separation between the presented scene and the object being followed,

and give the convenience for further work.

The follow camera uses the Cinemachine Virtual Camera component, which is used to create a virtual camera that controls the viewpoint and camera behaviour in the game. Specifically, it has the following important parameters: follow: it targets the object that the camera needs to follow. The camera will automatically follow the object and maintain the appropriate distance and angle to ensure that the player can get the best visual experience in the game. Here I set it to the "root" empty object I set before, so I can adjust the position of the camera by adjusting the position of the "root". FOV: the width of the viewing angle, too narrow will make it difficult for the user to see the whole scene, too wide may lead to the lens appear too wide may cause the camera to appear to "pass through" the wall. Here, after testing, 45 is the best. body and Aim: these two parameters provide the way the lens moves and the target it faces and observes. Here, for a first person effect, they are set to follow the character and the target is the same as the follow target.

The main camera uses the Chinemachine Brain component to set the live camera to follow the camera so that the screen shows the scene from follow camera. Main camera also have a physics Raycaster component, for casting ray, thus could give information for using.

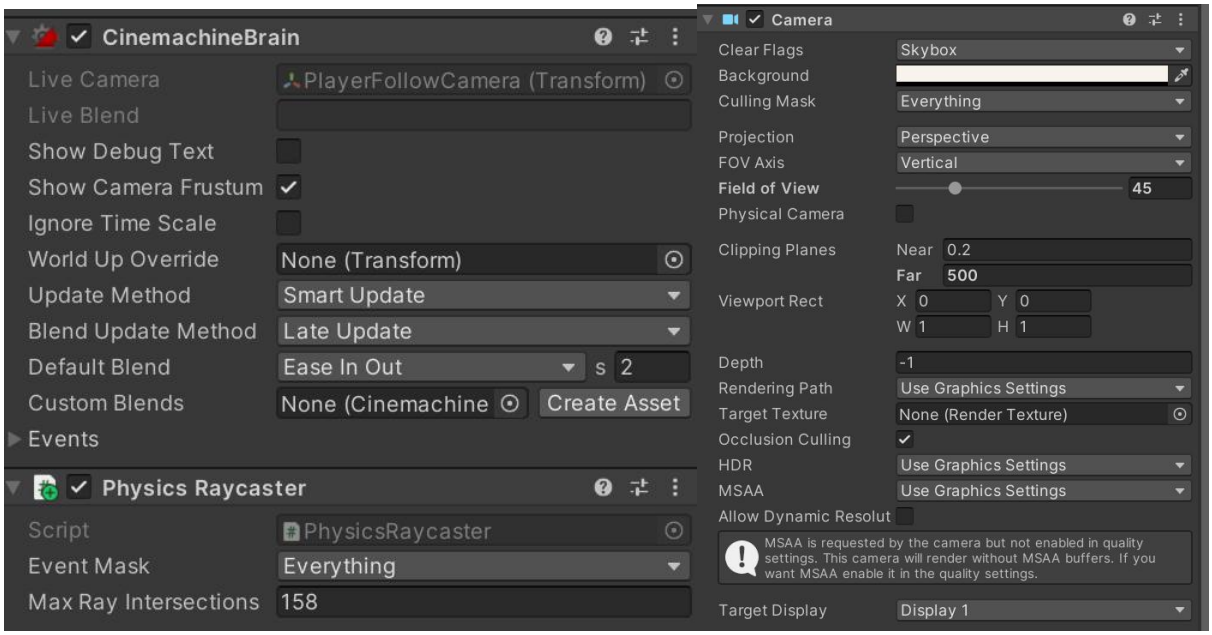


figure: setting of main camera.

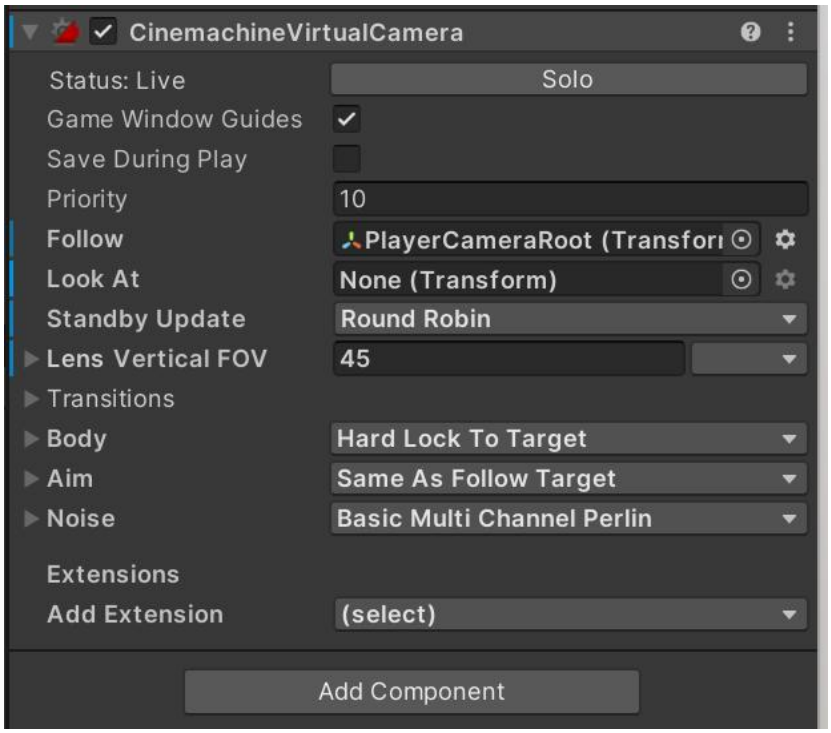


figure:settings of follow camera.

3.2.3 UI design and user manuals

The UI gives the user hints.In this application, there are two scenarios: a game scenario, which includes the main models, and a start scenario, which consists mainly of the UI. The UI of the start scene is a start screen that fills the screen and consists of the following elements: a welcome text, a start button and an exit button.

figure： start page

The UI of the game scene is more complex. Once the user clicks "Start" at the beginning of

the scene, an almost screen-filling text field appears, which describes the task the user needs to complete: bringing the gamepad from the bedroom to the living room. Next to it is an input field to get the user's name for the record. In addition to this there are a number of buttons: quit, restart and accept. When the user clicks on the "Accept" button, the text description field is hidden and the house scene is displayed. While the user is completing the task, there is a button at the top of the screen that displays a prompt: click to view the task (you are now in the + area) and when the user clicks on this button, the text description field will appear again. More, after the user clicks "Accept" for the first time, the name input field will be hidden. When the user completes the task, the text description field will reappear and show the name entered by the user and the time spent. At the same time, the "Accept" button will be hidden, which means that the user can either restart the task or simply quit after completing it.

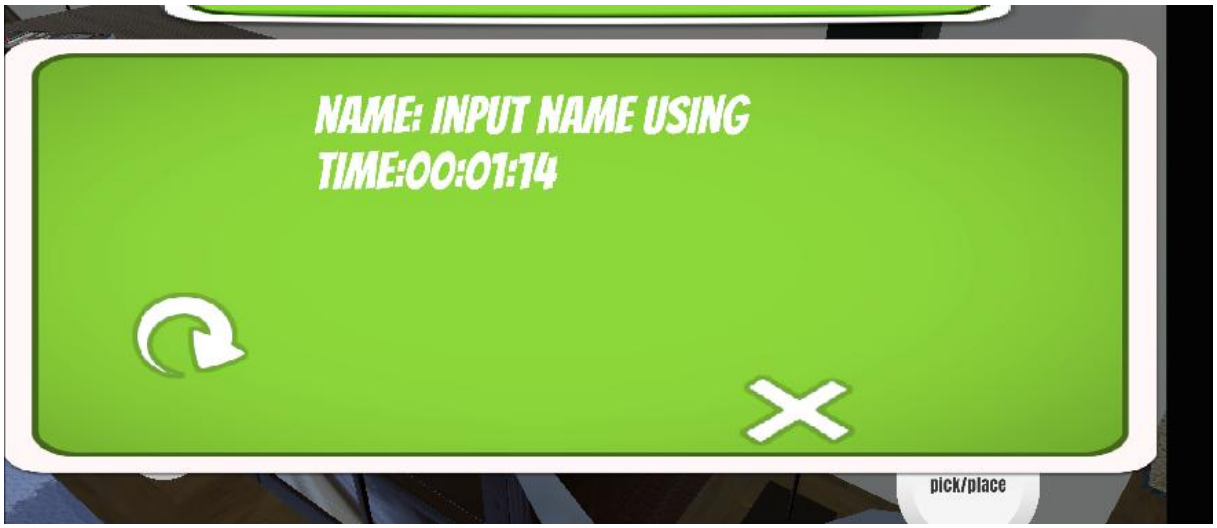


figure:task display UI

The user will also need some manoeuvring areas to control the character's movements. Firstly there will be an alignment centre in the middle of the screen to give the user a reference. Then, to the left and right are the move, watch and hold/put manipulation areas. The user can use these areas to move the character and shift the camera, and to pick up or place props. In the upper left corner, there is a button "Exit" which allows the user to exit the application at any time.

figure: manipulation UI

3.2.4 Result of Unity Modeling and Design

After designing the model and UI, the entire modelling is complete. To summarise, I created a house and character, and through the UI allowed the user to manipulate the character and view tasks.

figure:some screen shot of scene.

3.3 C# Scripting

C# scripting gives the user the ability to communicate with the scene. In this project, there are mainly scripts for several functions: 1. Obtaining user inputs and reacting accordingly. 2. User interaction with props and determination of task completion. 3. Other functional scripts.

3.3.1 Getting user input

The basic logic for getting user input is as follows: first a controller script is created to control the movement of the character model and camera transitions. This script defines the movement speed, gravity, and other parameters and controls the camera movement, in

addition to some logic for calling animations and audio. In addition to the controller script, there is a script called inputs that translates user interface inputs into format that Controller is accepted. Finally, there is the user interface inputs script, whose function is to fetch the user's actions on the user interface and make the relevant actions referable.
The advantage of dividing the input function into three parts is that it reduces coupling, facilitates future modification or updating, standardises the format of the input, and finally facilitates debugging, reducing the possibility of errors.

Controller:
The controller script is the most important one, and all subsequent scripts need to call the methods in it. First we need to define some attributes:

public float MoveSpeed = 2.0f;--Define the move speed
public float RotationSmoothTime = 0.12f;--The smoothing time for the character to turn in the direction of movement.
public GameObject CinemachineCameraTarget;--The target object that the camera is following.
public float TopClamp = 70.0f;--Maximum camera upward rotation angle limit.
public float BottomClamp = -30.0f;--Maximum downward rotation angle limit of the camera.
public float CameraAngleOverride = 0.0f;--Additional offset of the camera angle
public bool LockCameraPosition = false;--Whether to lock the camera position.

We need to initialise the main camera reference and camera angle in the awake() method. The awake() method is run before the start() method; the awake() method is run as soon as the class is instantiated, so we initialise the relevant parameters of the camera in the awake() method:

```
0 个引用
private void Awake()
{
    // get a reference to our main camera
    if (_mainCamera == null)
    {
        _mainCamera = GameObject.FindGameObjectWithTag("MainCamera");
    }
}
```

Then the animation, camera components are initialised in the start() method. These components need to be called at the beginning of the program, so they are placed in the start() method.

```
        private void Start()
        {
            _cinemachineTargetYaw = CinemachineCameraTarget.transform.rotation.eulerAngles.y;

            _hasAnimator = TryGetComponent(out _animator);
            _controller = GetComponent<CharacterController>();
            _input = GetComponent<StarterAssetsInputs>();
#if ENABLE_INPUT_SYSTEM
            _playerInput = GetComponent<PlayerInput>();
#else
            Debug.LogError( "Starter Assets package is missing dependencies. Please use Tools/Starter Assets/Reinstall Dependencies to fix it");
#endif
```

The update() and lateupdate() methods are both functions that run each as the game progresses. So here you need to put some real time update function or code. the code in lateupdate() will run after the code in update() runs, so here you put the camera code in lateupdate() to make sure that the camera moves after the character moves and then follows. Move(),animator component invocations in Update() and CameraRotation(); is in late update();

```
        private void Update()
        {
            _hasAnimator = TryGetComponent(out _animator);


            Move();
        }

        0 个引用
        private void LateUpdate()
        {
            CameraRotation();
        }
```

In AssignAnimationIDs(), the strings "Speed" and"MotionSpeed" are converted to their corresponding hash values and assigned to the variables _animIDSpeed_animIDMotionSpeed respectively. These hash values will be used in the subsequent code to refer to the corresponding state or parameter in the animation state machine, rather than directly refer to the string, which improves the efficiency of the code.

```
private void AssignAnimationIDs()
{
    _animIDSpeed = Animator.StringToHash("Speed");

    _animIDMotionSpeed = Animator.StringToHash("MotionSpeed");
}
```

Next is the move();

First we start by calculating the current horizontal velocity: get the current horizontal velocity of the character, ignoring the vertical velocity.

```
// a reference to the players current horizontal velocity
float currentHorizontalSpeed = new Vector3(_controller.velocity.x, 0.0f, _controller.velocity.z).magnitude;
```

Then we use the lerp function to do linear interpolation to adjust the speed and mix the animation values. currentHorizontalSpeed is the current horizontal speed, targetSpeed * inputMagnitude is the target speed multiplied by the magnitude of the input direction (or the length of the input vector if analogMovement is true or 1 otherwise), Time.deltaTime * SpeedChangeRate controls the rate of speed change. In this way, the Lerp function allows the character's speed to smoothly transition from the current speed to the target speed. In the transition between animation blend and target speed, _animationBlend is the current animation blend, targetSpeed is the target speed, and Time .deltaTime * SpeedChangeRate is used to control the rate of change of the blend. This is similar to the logic used to adjust the target speed.

```
_speed = Mathf.Lerp(currentHorizontalSpeed, targetSpeed * inputMagnitude,
    Time.deltaTime * SpeedChangeRate);
```

```
_animationBlend = Mathf.Lerp(_animationBlend, targetSpeed, Time.deltaTime * SpeedChangeRate);
if (_animationBlend < 0.01f) _animationBlend = 0f;
```

Then we calculate the direction of movement and the angle of rotation: the direction of movement is calculated based on the player's input direction, and the angle of rotation of the target the character should face.

```
Vector3 inputDirection = new Vector3(_input.move.x, 0.0f, _input.move.y).normalized;

_targetRotation = Mathf.Atan2(inputDirection.x, inputDirection.z) * Mathf.Rad2Deg +
                  _mainCamera.transform.eulerAngles.y;
```

We then use the SmoothDampAngle function to smoothly rotate the character in the direction

of the target.

```
                _mainCamera.transform.eulerAngles.y;
    float rotation = Mathf.SmoothDampAngle(transform.eulerAngles.y, _targetRotation, ref _rotationVelocity,
        RotationSmoothTime);

    // rotate to face input direction relative to camera position
    transform.rotation = Quaternion.Euler(0.0f, rotation, 0.0f);
}
```

Finally, the function of moving the character is realized by using the moving direction and speed obtained according to the calculation. And according to the size of the moving speed and input direction, the speed and movement speed parameters in the animation state machine are updated to control the animation playback of the character.

```
    _controller.Move(targetDirection.normalized * (_speed * Time.deltaTime) +
                     new Vector3(0.0f, _verticalVelocity, 0.0f) * Time.deltaTime);

    // update animator if using character
    if (_hasAnimator)
    {
        _animator.SetFloat(_animIDSpeed, _animationBlend);
        _animator.SetFloat(_animIDMotionSpeed, inputMagnitude);
    }
```

The CameraRotation() function serves to control the rotation of the camera based on the player's input and applies the rotation to the Cinemachine camera target to achieve perspective rotation.

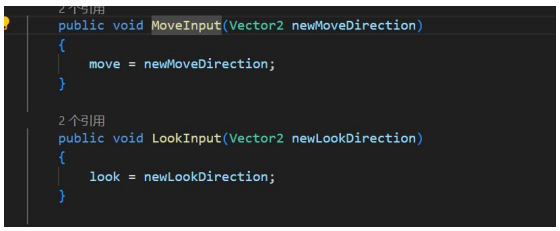
```
    private void CameraRotation()
    {
        // if there is an input and camera position is not fixed
        if (_input.look.sqrMagnitude >= _threshold && !LockCameraPosition)
        {
            //Don't multiply mouse input by Time.deltaTime;
            float deltaTimeMultiplier = IsCurrentDeviceMouse ? 1.0f : Time.deltaTime;

            _cinemachineTargetYaw += _input.look.x * deltaTimeMultiplier;
            _cinemachineTargetPitch += _input.look.y * deltaTimeMultiplier;
        }

        // clamp our rotations so our values are limited 360 degrees
        _cinemachineTargetYaw = ClampAngle(_cinemachineTargetYaw, float.MinValue, float.MaxValue);
        _cinemachineTargetPitch = ClampAngle(_cinemachineTargetPitch, BottomClamp, TopClamp);

        // Cinemachine will follow this target
        CinemachineCameraTarget.transform.rotation = Quaternion.Euler(_cinemachineTargetPitch + CameraAngleOverride,
            _cinemachineTargetYaw, 0.0f);
    }
```

First make sure there is an input and the camera position is not fixed, then adjust the horizontal and vertical angles of the camera target according to the player's input and use the ClampAngle function to limit the horizontal and vertical angles, making sure that their values are within a certain range to prevent the angles from being too large or too small. Finally, the adjusted vertical and horizontal angles are applied to the Cinemachine camera target to achieve the camera rotation effect. Here the Quaternion.Euler function is used to create a quaternion of rotation angles.

Input script:

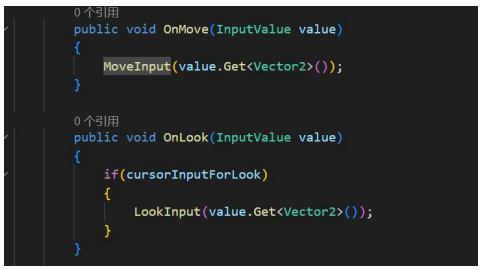
The input script accepts player input and allows the controller to call methods. This has the

advantage of standardizing the format; the public variables move and look defined in the Inputs script are used to store the player's movement and perspective inputs, and the MoveInput and LookInput methods store the received inputs into the corresponding variables defined in the Inputs script.



UICanvasControllerInput and UIVirtualJoystick:

UICanvasControllerInput and UIVirtualJoystick together complete the Virtual Joystick functionality, which allowing the player to simulate movement and perspective control by touching on the screen.The UIVirtualJoystick script is responsible for displaying and interacting with the virtual joystick, including listening to player touch operations and outputting the corresponding joystick position information. The UIVirtualJoystick script is responsible for displaying and interacting with the virtual joystick, including listening to the player's touches and outputting information about the joystick's position. The joystick position information is passed to the outside world by defining an event, joystickOutputEvent, so that other scripts can listen to this event and respond accordingly. The virtual joystick inputs are passed and processed by referencing the Input script in the UICanvasControllerInput and calling their methods.



The ondrag function implements the handling of the player's dragging of the virtual joystick. The screen coordinates are converted to coordinates in the local coordinate system with containerRect as the parent, and they are adjusted to normalize the coordinate values according to the size of the container, so that the coordinate values vary within a certain range. The coordinates are then constrained and inverted, and finally output.

```
public void OnDrag(PointerEventData eventData)
{

    RectTransformUtility.ScreenPointToLocalPointInRectangle(containerRect, eventData.position, eventData.pressEventCamera, out Vector2 position);

    position = ApplySizeDelta(position);

    Vector2 clampedPosition = ClampValuesToMagnitude(position);

    Vector2 outputPosition = ApplyInversionFilter(position);

    OutputPointerEventValue(outputPosition * magnitudeMultiplier);

    if(handleRect)
    {
        UpdateHandleRectPosition(clampedPosition * joystickRange);
    }

}
```

The rest of the functions implement real-time updating of the joystick position, as well as limiting the position of the joystick, and a few functions implement the inversion of individually entered coordinates.

```
public void OnPointerUp(PointerEventData eventData)
{
    OutputPointerEventValue(Vector2.zero);

    if(handleRect)
    {
        UpdateHandleRectPosition(Vector2.zero);
    }
}

2 个引用
private void OutputPointerEventValue(Vector2 pointerPosition)
{
    joystickOutputEvent.Invoke(pointerPosition);
}

3 个引用          void UIVirtualJoystick.UpdateHandleRectPosition(Vector2 newPosition)
private void UpdateHandleRectPosition(Vector2 newPosition)
{
    handleRect.anchoredPosition = newPosition;
}
```

In UICanvasControllerInput are some calls to the input script, which implements the conversion of joystick input to input script input.

```
0 个引用
public void VirtualMoveInput(Vector2 virtualMoveDirection)
{
    starterAssetsInputs.MoveInput(virtualMoveDirection);
}

0 个引用
public void VirtualLookInput(Vector2 virtualLookDirection)
{
    starterAssetsInputs.LookInput(virtualLookDirection);
}
```

3.3.2 Interact with the environment and Determination of the task

The scripts in this section are mainly scripts that control the content of the UI text, and some scripts that control the in-game models.

Script Task1:
This script is used to determine if the task is completed and to give feedback.The task in task1 is to place the gamepad on the specified position, and the code above uses the collision volume determination to determine whether the task is successful or not.

```
private bool IsTaskCompleted()
{
    if (taskProp == null || specifiedFurniture == null)
    {
        Debug.LogError("任务道具或指定家具未设置！");
        return false;
    }


    return specifiedFurniture.GetComponent<Collider>().bounds.Intersects(taskProp.GetComponent<Collider>().bounds);
}
```

If successful, disable the "Accept" button and change the text box to read: "well done".

```
    private void CheckTaskCompletion()
    {

        if (IsTaskCompleted())
        {

            taskCompleted = true;


            infoText.text = "well done!";
            taskBarController.taskBar.SetActive(true);

            GameObject startButton = GameObject.Find("start button");
            if (startButton != null)
            {
                startButton.SetActive(false);
            }
        }
    }
```

TaskBarController is used to control the state of the UI text bar.

```
public void ToggleTaskBar()
{
    gameObject.SetActive(taskBar.activeSelf);
    // 如果任务栏是可见的，则关闭；如果任务栏是不可见的，则打开
    taskBar.SetActive(!taskBar.activeSelf);
}
```

PlayerNameInput script stores the text entered by the user.

```
void ShowInputField()
{
    inputField.gameObject.SetActive(true);
}

0 个引用
public void OnCloseTaskBar()
{

    inputField.gameObject.SetActive(false);


    if (!inputTaken)
    {

        string playerName = inputField.text;
        Debug.Log("Player Name: " + playerName);


        inputTaken = true;
    }
}
```

PlayerPositionTracker script updates the position of the character based on the position information of the furniture on the soles of the character's feet.

```
void UpdateCharacterArea()
{

    Vector3 raycastOrigin = transform.position + Vector3.up * 0.5f; // 经过测试，人物脚底与地板会相交一部分，所以必须要抬高射线，不然就碰不到地板
    Vector3 rayDirection = Vector3.down;

    RaycastHit hit;
    if (Physics.Raycast(raycastOrigin, rayDirection, out hit, Mathf.Infinity))
    {


        GameObject hitObject = hit.collider.gameObject;
        FurnitureInfo furnitureInfo = hitObject.GetComponent<FurnitureInfo>();


        currentArea = (furnitureInfo != null) ? furnitureInfo.area : "Unknown";
    }
```

ItemInteraction implements the pick up or place function.

```
void PickUpItem()
{

    Vector3 screenCenter = new Vector3(Screen.width / 2, Screen.height / 2, 0);
    Ray ray = Camera.main.ScreenPointToRay(screenCenter);
    RaycastHit hit;


    if (Physics.Raycast(ray, out hit))
    {
        GameObject hitObject = hit.collider.gameObject;

        if (hitObject.CompareTag("game props"))
        {
            Debug.Log("Picking up item: " + hitObject.name);
            pickedUpItem = hitObject;
            pickedUpItem.transform.position = offScreenPosition;
            holdingItem = true;
            Debug.Log("Item picked up!");
        }
    }
```

3.3.3 Other scripts

Other scripts is for: exiting or restarting the application, logging time, storing furniture information.

The RecordData script records the amount of time that has elapsed since the user first clicked the "Accept" button until the task was completed. Furthermore, the time and username are recorded and then displayed in a text box.

```
public void OnCloseTaskBar()
{
    nameInputField.gameObject.SetActive(false);
    if (!inputTaken)
    {
        playerName = nameInputField.text;
        Debug.Log("Player Name: " + playerName);


        SaveData(playerName, completionTime - startTime);


        inputTaken = true;
    }
}


1 个引用
private void DisplayCompletionTime()
{

    TimeSpan timeSpan = completionTime - startTime;
```

```
private void SaveData(string playerName, TimeSpan completionTime)
{

    PlayerPrefs.SetString("PlayerName", playerName);
    PlayerPrefs.SetString("CompletionTime", completionTime.ToString());
    PlayerPrefs.Save();

    Debug.Log("Data saved successfully!");
}
```

FurnitureInfo stores the furniture information; ScenceLoader implements the transformation from the start screen to the task screen; RestartButton gives the restart logic; QuitGame gives the exit logic.

```
public class FurnitureInfo : MonoBehaviour
{
    1 个引用
    public string name; // 家具名称
    0 个引用
    public string description; // 家具描述
    1 个引用
    public string area;

    // 静态列表存储所有家具的名称
    1 个引用
    public static List<string> furnitureNames = new List<string>();
```

FurnitureInfo

```
0 个引用
public void RestartScene()
{
    // 获取当前场景的索引并重新加载
    int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
    SceneManager.LoadScene(currentSceneIndex);
}
```

RestartButton

```
0个引用
public void LoadGameScene()
{
    SceneManager.LoadScene("GamePage"); // 加载游戏主场景
}
// Start is called before the first frame update
```

ScenceLoader

```
public void Quit()
{
    #if UNITY_EDITOR
    UnityEditor.EditorApplication.isPlaying = false; //
    #else
    Application.Quit();
    #endif
}
```

QuitGame

3.4Case Design

(This part I will give more details in formal report, because now I have not get much feedback)

The task1 is to get the user to take the controller to the TV stand in the living room. This will work on the users' knowledge of the gamepad and will train their literacy skills. In the process of completing the task, they can also get the hint of the current location, which can also exercise their spatial and location perception.

3.5Getting feed back

(This part I will give more details in formal report, because now I have not get much feedback)

The "La Torre" centre is full of lovely and kind people who were willing to take part in my test. The test consisted of letting them try the app and getting their feedback. There are also many staff members on hand to guide and help the users, and they also give advice.

Feedback from the first test was as follows.

1. The lens is difficult to operate, and the spatial concept is more difficult to understand.
2. Use more voice prompts and picture prompts, less text.
3. It is better to use flowchart to show the task description.
4. It is better to help understand the concept of time, such as: one hour later, ten minutes later.

(later feedback)

Chapter 4:Results and Discussion

4.1 Functions performed

(this part will also be detailed in formal part, because now I still have time to add function)
I trained users in object recognition by creating virtual spaces and giving them tasks to perform by picking up/putting down objects and training them in literacy through textual descriptions.

4.2 Discussion on assistance to persons with disabilities

(this part will also be detailed in formal part, because now I still have time to adjust my app. When my application is mature, I will take photos and describe how work it is)

Chapter 5:Conclusion and Further Work
(this part may be changed in great extend in formal final report, because I still working on application and updating it frequently)

5.1 Challenges Encountered and Lesson Learnt

Having never learnt or used unity before, I spent a lot of time on unity and C# when I first started. And, I would make low-level mistakes on a regular basis, which greatly reduced efficiency. And, the life of studying and working alone in a foreign country also added a lot of pressure to me. In the process of completing this project, I not only learned a lot of unity modelling skills and techniques, I also learned to be persistent and brave. It is with my unremitting efforts that I was able to make the application look like this.

5.2 Future Work

5.2.1 Maintenance and Updates

The main target group of this project is the users of "La Torre", so the maintenance and updating strategy needs to be tailored to their specific situation.Here are some key points for maintenance and updates:

Bug Fixes:Regularly check the operation of the application to find and fix any errors or anomalies that occur.

Address known issues in a timely manner based on user feedback and test results.

Compatibility:Optimise the performance of the application to ensure it runs smoothly on a variety of devices. Currently the project can only support the Android environment, and can be updated to support other environments later as required.

Update content:Continuously improve the functionality and user experience of the application based on user feedback.

Update Unity engine and plugins:Currently using unity version 2022.3.18f1, after which it may be necessary to update the unity version and plugins.

Data Backup:Currently data such as user's name task completion time can only be saved in playerpreb, in the future it can be put into the cloud space, which is the server's database. This will enable the same data to be displayed on different machines.

Security and privacy protection:Ensure application security and privacy protection of user data. For example, user-entered names.

NOTE: The maximum length of the report up to here is 50 pages.

The context of next part will be finished in formal report. Now I have limited feedback and also my application have not be finished. So reference in future may be different from now. In terms of supervisor log and acknowledgement, this need to be finished when all work is done.

References

Everything you cite from other sources should be properly referenced. The QMUL Faculty of Science and Engineering has identified the Harvard and Vancouver referencing styles as the recommended styles for project reports. Details about the referencing style and examples can be found online. https://www.qmul.ac.uk/library/academic-skills/referencing-hub/referencing-guides-and-resources/

Acknowledgement

Give your acknowledgement to people who helped you during the project here. Maximum length of this section is 1 page. You may thank your supervisor but DO NOT MENTION YOUR SUPERVISOR'S NAME HERE.

Appendices

Disclaimer

This report is submitted as part requirement for the undergraduate degree programme at Queen Mary University of London, and Beijing University of Posts and Telecommunications. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

BUPT No.:
QM No.:
Full Name (Pin Yin):
Full Name (Chinese):

Signature:

Date:

Project specification

Include your project specification, part 1 and part 2 here. It must be the final version submitted to QMPlus.

Early-term progress report

Include your project early-term progress report here. It must be the final version submitted to QMPlus.

Mid-term progress report

Include your project mid-term progress report here. It must be the final version submitted to QMPlus.

Supervision log

Include your project supervision log here.

Additional Appendices (as needed)

Information that you think may be helpful or relevant for the reader but that is not directly relevant to the story of your project. Things that might be suitable as an appendix to a report are:

- Large tables of numerical results that have been displayed graphically in the main body of the report.
- Important parts of datasheets for specific devices you have used in your project if you think that they are important enough that the reader should have access to them without finding them off the web themselves.
- Mathematical proofs and results that are important to show but not important to the flow of the story in the report.

NOTE: Full code listings must NOT be included as an appendix, but extracts of code may be included in the body of the report to illustrate particular points. Code should be submitted as supporting documents to QM+.

Risk and environmental impact assessment

Please refer to the project handbook section 3.6.12.