Master's Dissertation 2016

Navigation Digital Signage System based on Interaction of Ultra High Definition Display and Smartphone

ZULIN LIANG
（Student ID Number：81434711）

Supervisor Professor OGI Tetsuro

March 2016

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

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81434711	Name	ZULIN LIANG

Title

Navigation Digital Signage System based on Interaction of Ultra High Definition Display and Smartphone

Abstract

Complex subway station usually has several floors and large numbers of exits. Navigation system is important to help people solve wayfinding problem inside such station. Navigation system is divided into path computation type and mapping guidance type which we focused on this research. In this research, we proposed a new navigation system which combined UHD (Ultra High Definition) display with personal smart phone. Different from conventional 2D map system, 3 Dimensional Map is introduced in our system and could be control by smartphone. This interaction obtains better user control experience and help people understand construction structure more efficiently. In this navigation system, gyroscope and scrolling gesture data from smartphone will be transmitted to UHD display to control 3D map model. The 3D map model can be rotated, zoomed in/out smoothly. Several UHD display and computers are combined into UHD Digital Signage Subsystem, and integrated with groups of Smartphone Subsystem. All the subsystems connect to a Central Server System in order to provide Multi-to-Multi approach. The Multi-to-Multi Approach gain the possibility to serve more people inside station, and hence satisfy mass requirement of wayfinding inside complex subway station.

As a result, the system feasibility had been verified that all subsystems are robust, and the integration of whole system worked reliably. The survey result and experiments showed that whole system had been validated. As a conclusion floor structure was understood by several users more efficiently.

Key Word(5 words)

Digital Signage, Ultra-High Definition, Smartphone, WebGL, WebSocket

Index

List of Figure

1 Introduction

Digital Signage, Smartphone, UHD (Ultra High Definition) TV, and Navigation System make our life colorful and better. Combining these several devices and their sensors to solve real world problem is also a hot issue in terms of (Internet of Thing) IoT and System Engineering. Image you can use your own Smartphone interact with Digital Signage inside a smart city, some researches already showed the future of urban life [1]. IoT helps to build the concept and System Engineering helps to achieve that. Figure 1 showed the possible future life in smart urban area.

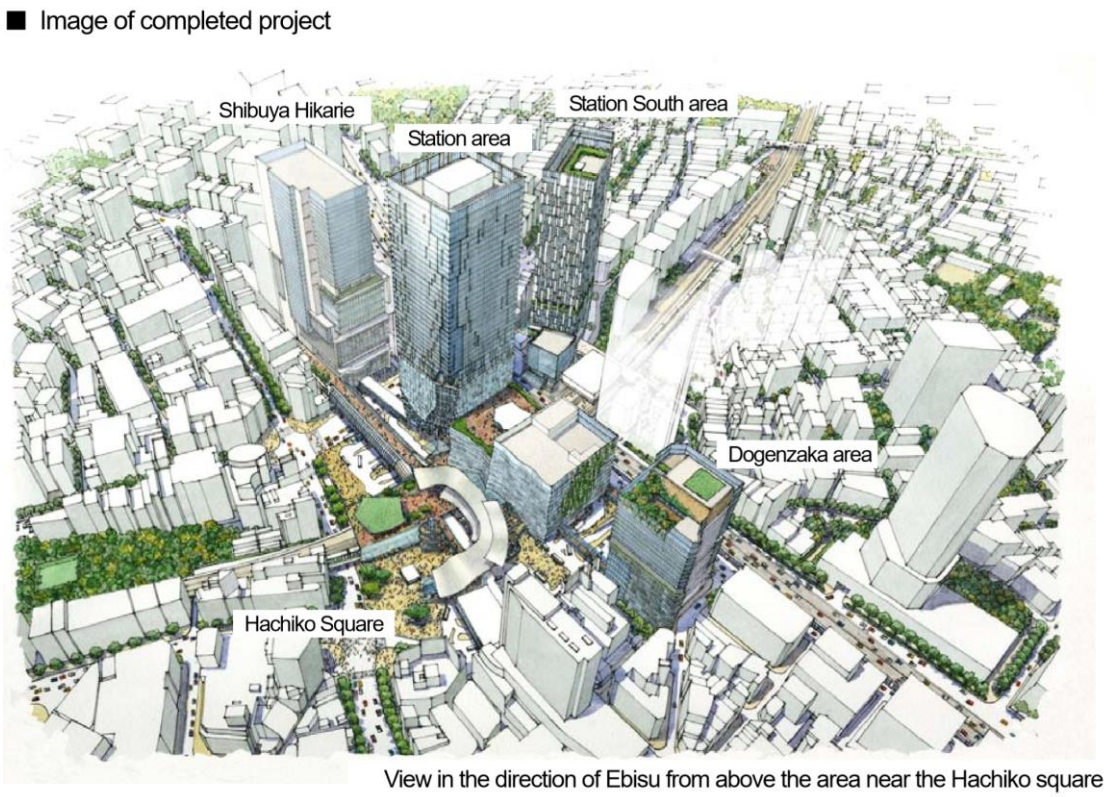


Figure 1: Imagination of Smart City (Urban Planning of Shibuya Station Area, from Tokyu Corporation & East Japan Railway Company: jreast.co.jp/e/press/2012/pdf/20130101.pdf)

As a smart city, the urban traffic system must be able to carry heavily daily transportation. Hence, as an interchange point, complex subway station is built to integrate multiple subway lines and link certain commercial areas together. Complex subway station usually has several floors and large numbers of exits. Inside such station, the wayfinding issue could be a problem for new visitors. Figure 2 shows the example of complexity of Shibuya Station in Tokyo, Japan.

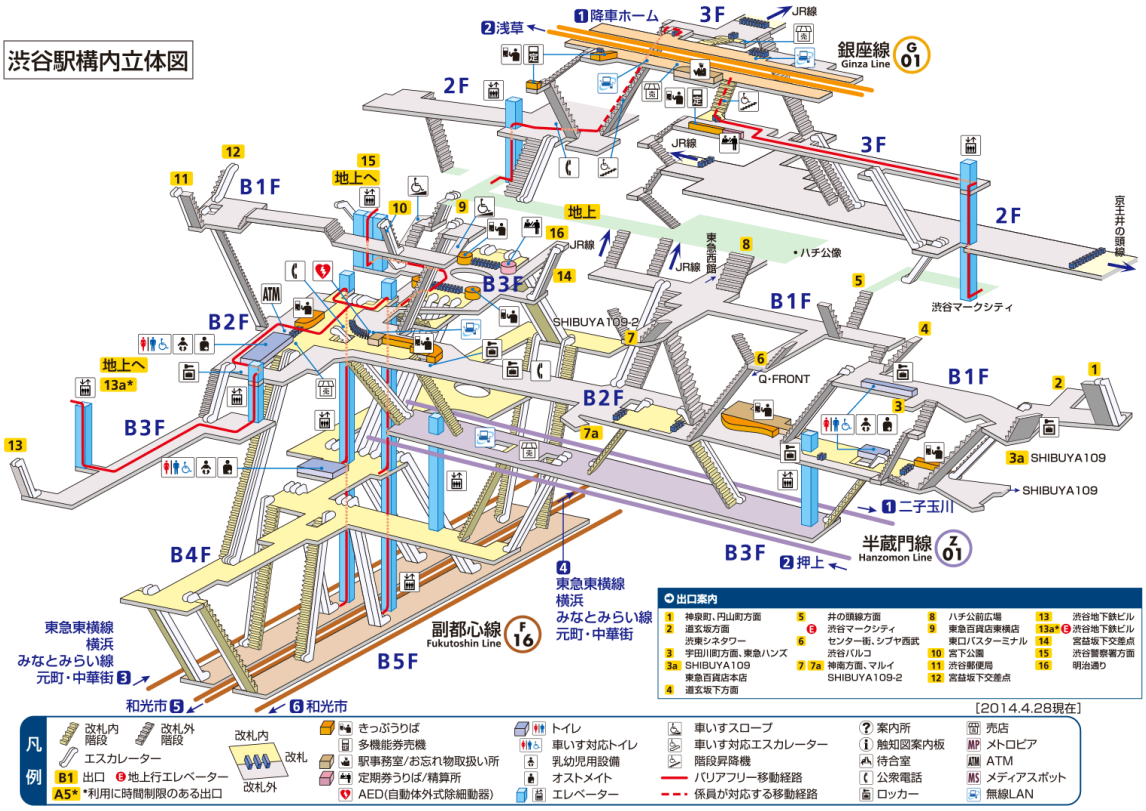


Figure 2: Shibuya Station three-dimensional map

(http://www.tokyometro.jp/station/shibuya/yardmap/)

Navigation system is an integrated system which is designed to help people solve wayfinding problem. It is divided into path computation type and mapping guidance type which we focused on this research.

In this paper, we proposed a novel navigation system which combined UHD (Ultra High

Definition) display with personal smart phone, in order to obtain better user experience and help people understand construction structure more efficiently.

In this paper, the background of this research will be stated. Some literature review will be done to give a glance in the problem (Section 1.2), theory (Section 1.2), solution (Section 1.3), and all the device (Section 0, 0) involve in our system.

1.1 Background

The world changed so fast, and the Tokyo Olympic doesn’t seem like a far future. Large amount of people are expected to visit Japan in 2020 Olympic game. As the introduction in section 1, with the benefit of interchange construction structure, multiple subway lines can interact each other to provide convenient transfer experience. However, this convenience can only be enjoyed by people who are familiar with complex station structure. People find it difficult to find path quickly inside complex station such as Yokohama, Shibuya, Shinjuku, and Tokyo, etc. Below shows some example of complexity:

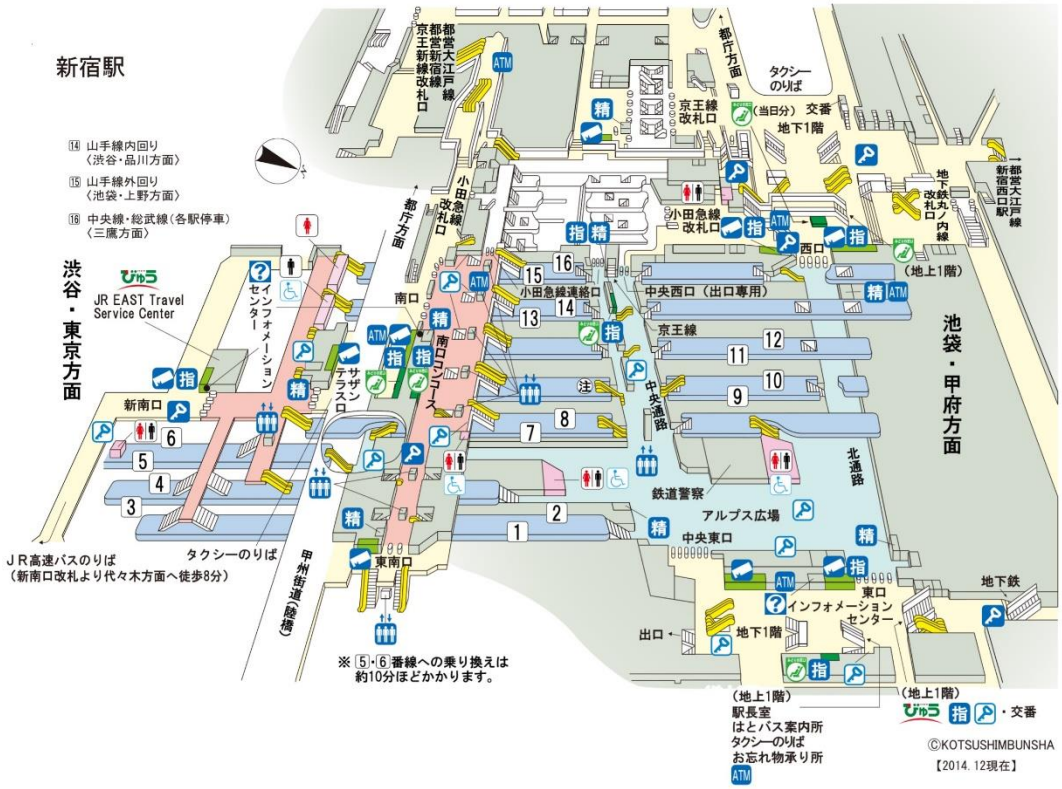


Figure 3: Shinjuku Station map (https://www.jreast.co.jp/estation/stations/866.html)

Figure 4: Shinjuku Station map (http://waral.club/tvmovie/20150214)

Figure 5: Yokohama Station map (https://www.jreast.co.jp/estation/stations/1638.html)

Complexity makes people confuse about station structure. They don't know how to go from one platform to another, where to find the exits and which stair or elevator to take. All these are wayfinding problem. Wayfinding problem is not a new issue, but it still remains when there are so many nowadays advance technology. With Google Maps, GPS, Internet, Smartphone and so many high-tech knowledge device, these problem still occurs, there must be some detail issue haven't been analysis.

1.1.1 Navigation System

Navigation system is usually designed to reduce the recognition cost for wayfinding. The system goes through the data of nodes, links and other informations which abstracted

from the real world, calculates the best way to destination, and gives some advice to users depends on their current location.

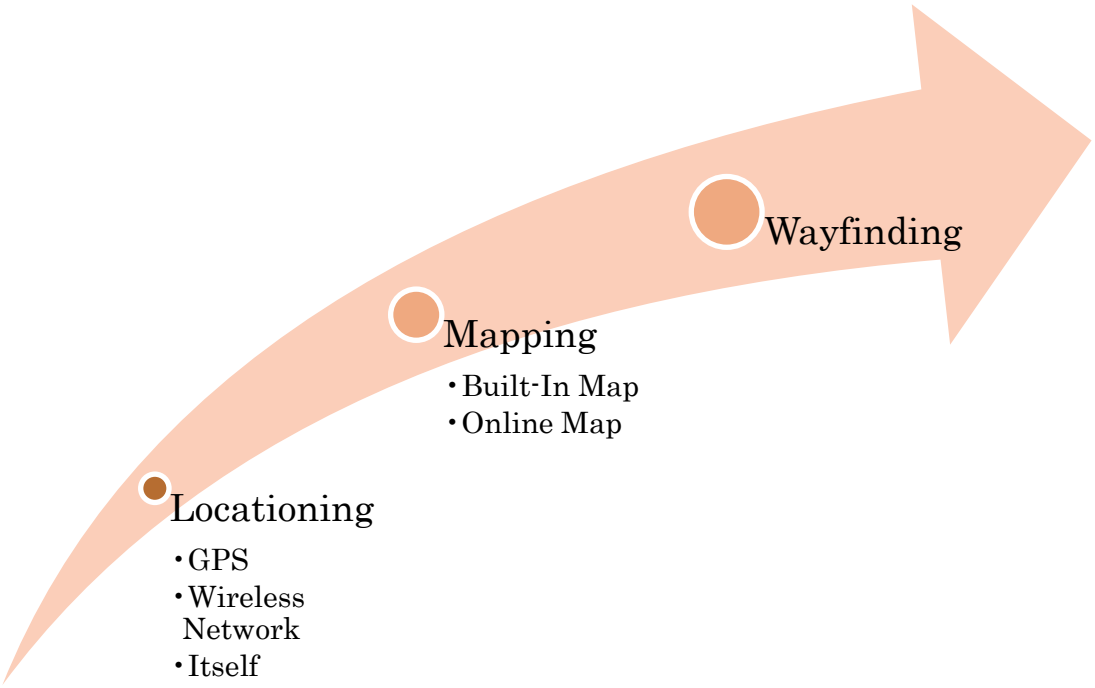


Figure 6: Navigation Process

As Navigation Process showed as Figure 6, Navigation system uses GPS, wireless network or the location of system itself to define the original position, receives information from user by voice or text. After that, the system uses build-in map or online map to detect the best way to desire destination. In terms of hardware, there are portable device, digital signage and voice device, etc. Navigation System is an aggregate concept of devices and location.

Current portable navigation system usually required user to stare at small screen frequency. The location signal which provided by GPS or cellular base station has high probability to be blocked in a closed building environment, especially inside subway station. Some solution use QR code to provide stable location information and avoid signal interference, but it doesn't seem like convenient for users since it requires to open

camera and scan the QR code. Voice devices are usually used in car navigation where driver needs to focus on the roads and could not pay attention to any screen.

There are also large type of navigation system which used digital signage as terminal in order to provide mapping information to end users. Some researches showed that this kind of navigation system had some advantage in campus navigating [2] and autonomous robots navigating [3]. In digital signage navigation system, the locationing is done by the system position itself. The mapping is either by calculating the road in users or inside the signage system.

1.1.2 Digital Signage

In this research, we tried to combine the benefit of digital signage and smart phone, and especially used Ultra-High Definition display and new web technology like WebSocket and WebGL to develop a new navigation system.

Digital Signage has evolved continuously over the years, and it has showed the potential in application of navigation recently. Comparing to conventional usage of advertisement and broadcasting, the interacting with digital signage has gained great interest for researchers due to hot issue of Internet of Thing. In these days, the ubiquity of smart phone and the development of sensor make it possible to apply more intuitive interaction with digital signage. There are widely research on Digital Signage interaction with portable devices [4], [5], [6], [7], [8], [9]. Most of them focus on linking method and interaction method in technique detail.

Digital Signage is an evolution device to broadcast contents instead of traditional static posters. It is widely used in airport (show airline information), exhibition hall (show exhibition info), shopping mall (show advertisement) and so on. Recently it is also famous in vending machine, showed as Figure 7, which adds enjoyment to shopping experience.

Compare to traditional static demonstrate method, digital signage uses electric screen and information system to maintain and update the data. It reduces the cost of labor and always keep fresh data which is quite important nowadays. Digital signage has been tested the possibility as a navigation system for human [2] as well as robot [3]. The interaction between smart device and digital signage has shown better advertisement effect [10].

The interchange subway station provide convenient travel experience among city. Helping people to understand construction structure and find correct path is an essential issue of efficient urban traffic.

Figure 7: A vending machine used digital signage for providing service.

http://www.ad60.com/touch-screen-vending-machines-generation/

Interaction with digital signage gained large among of requirement these years, and one-to-one interaction seems to be difficult to satisfy the requirement. Hence, with the research development, one-to-many interaction had been developed to multicast contents [11]. And Many-to-many interaction has also been developed [12].

Different from multicasting method, in this paper, we proposed a TCP/IP based multi interaction for UHD Navigation Signage System, in order to provide more personal control experience while several persons try to use this system at several place inside station. Traditional server design used a main loop to handle all client problem. In this research, we used listener and handler to connect smartphone link and digital signage together in order to fully use the multi-thread system and increase connection speed. This proposed system deals with wayfinding problem inside complex interchange subway station, and try to give a multi service result to fit all the users' requirement.

1.2 Wayfinding

As introduction at previous section, wayfinding could be a problem inside complex subway station. This problem usually occurs in new visitors who are not familiar with the station structure.

Due to the differential services, wayfinding problem becomes a big issue for not only visitors but also researchers. "Wayfinding is a dynamic affair" [13]. As a cognition process, in this paper, we describe the problem and essential solution as below.

When human-being recognize way-path, it is a continuous task and the task will be renewed moment by moment. Firstly he or she should understand the original point and the destination (E.g. I am in Hiyoshi and I want to go to Akihabara). Then they try to understand the abstract way-path by dividing in several parts (E.g. from Hiyoshi station to Shibuya Station, then from Shibuya station to Akihabara Station). After that, they will try to find out the detail path (E.g. which exchange exit they should take, inside the station whether they should go upstairs or not). This process is showed as Figure 8.

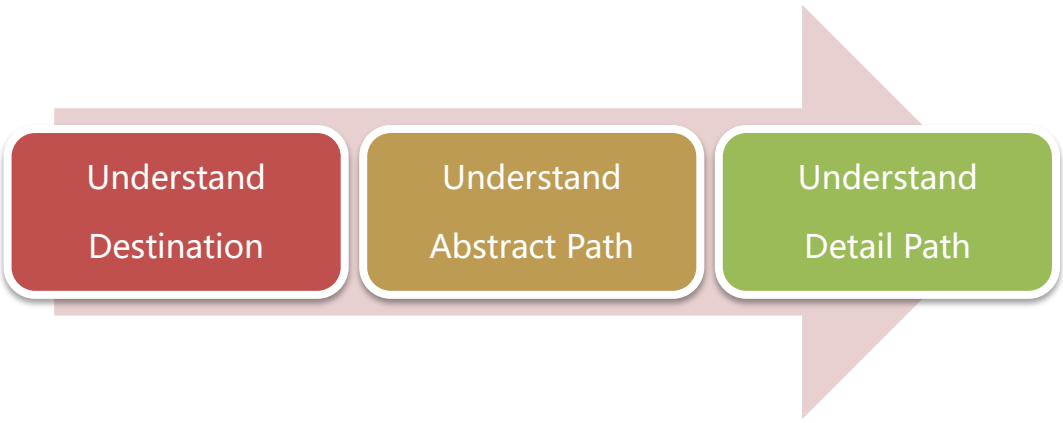


Figure 8: Recognition way-path of a human-being

Wayfinding is also a continuously subdivided-integration process. People will subdivide the way into different parts depend on the cognition requirement. If one person want to go to one platform, for example, he will try to find out whether it is the same floor or not. That is, he or she is going to subdivide the station into floors and stairs. If it is the same, then he or she will continuously consider which road to select. If it is not, he or she will try to find out how many floors they should climb up and whether to take an elevator or a stair.

On the other hand, some station structures seem to be complex but the cognition map inside human could be simple, then people would like to integrate the way in a simpler structure. For example, if all the stair and elevator can help people to reach any floor of the structure, then it is no necessary to remember "which stair or elevator to take". People tend to remember the action by integrating different choices into a simpler action. Another example is, if there are 6 or 7 or more stairs in front of a person, but only the biggest stair allows people to reach the 4th floor directly, then people usually tends to remember the road by the key word "Biggest" other than "the first stair in from of me" or "the stair between red and blue one".

1.3 Current System and their problems

As introduction in section 1, this research mainly focus on wayfinding problem inside complex subway station. The question focus on new visitors who are newly come to complex subway station. As the example showed in previous section, the complexity here could means several floors, hundreds of elevators, stairs and exits.

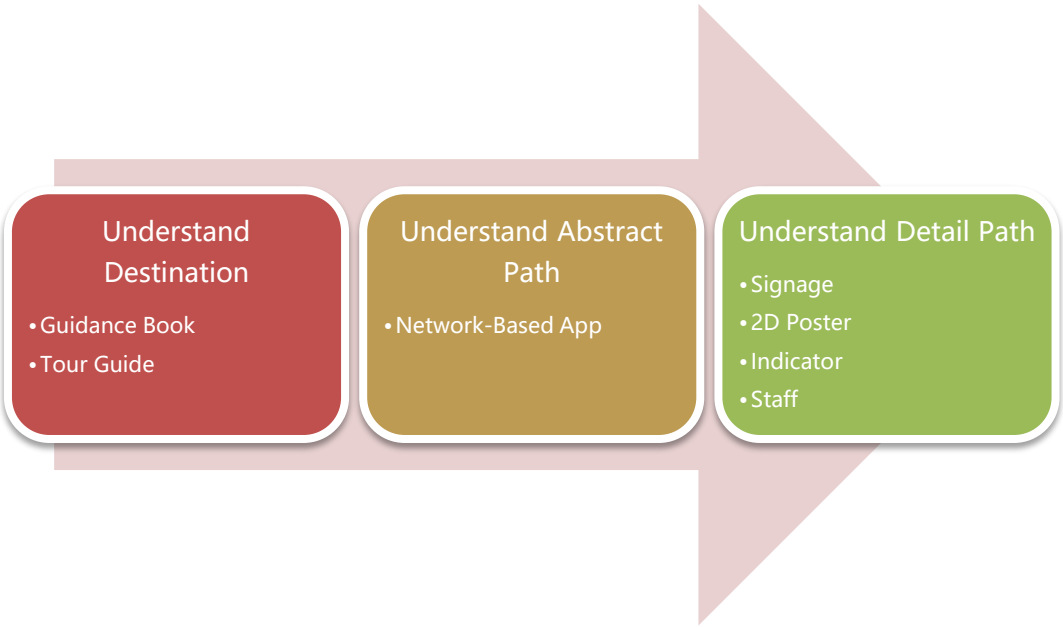


Figure 9: Current Solution in Wayfinding Process

As showed in Figure 9, we state the common process when people deal with wayfinding problem by using current system. When people find a road, he or she needs to firstly understand the original position and destination (I am at Shibuya and I want to go to Akihabara), and that can be solved mainly by guidance book or the advice from tour guide. The understanding of abstract way-path can be solved mainly by the Network-based software or Apps like Google Map (showed as Figure 10) and etc. When inside station, these app lack of floor information. The understanding of detail path is supposed to be solved by signage, 2D poster map, indicator and staff inside the station, but they are not

as effective as they are expected.

Hence, in this research, we mainly focus on wayfinding problem inside complex subway station.

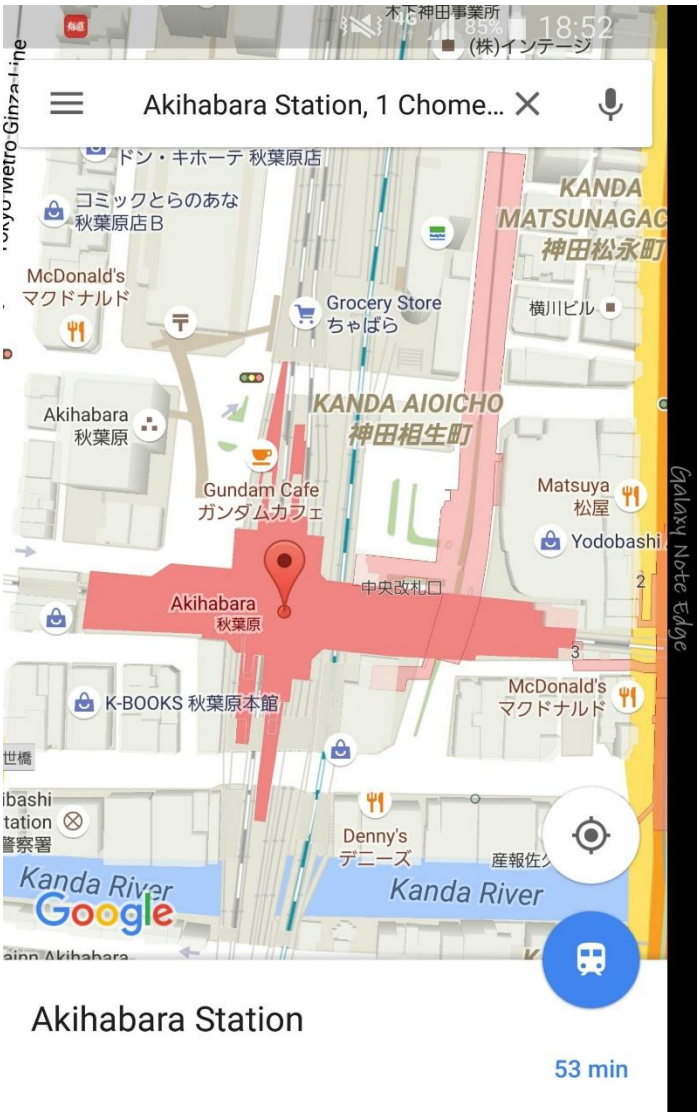


Figure 10: An example of Google Map

Figure 11: a 2D map example of Hiyoshi station (From Tokyu official website: http://www.tokyu.co.jp/railway/station/yardmap/?id=13)

2D poster map, an example showed as Figure 11, is a static map system. It shows the area, important point liked entrance, exit, coin locker and so on. It cannot give enough floor information which is important when coped with wayfinding problem inside interchange subway station.

Indicator (showed as Figure 12) is usually designed as a big arrow with several words to describe direction and the name of that destination. The staffs inside subway station usually use their own language and have lots of other works which are important for maintaining the subway station. When deal with large number of people, the service is difficult to satisfy all the requests.

Figure 12: An example of Indicator

In term of floor information, 3D model is a kind of solution. Setup some entity model inside the station is also a good way. Nowadays, 3D printer makes it possible to develop and build complex model quickly. But as we put the 3D entity model inside station, it still occupies lots of space inside station.

Therefore, designing a new navigation system to solve all these problem together is important. Occasionally some emergency situation happen, it is also required in navigation system to refresh the information quickly. In order to help people understand the path more easily, the navigation system should also give the whole picture of construction structure.

In conclusion, Figure 13 shows the deficiency of current system, or said, navigation method.

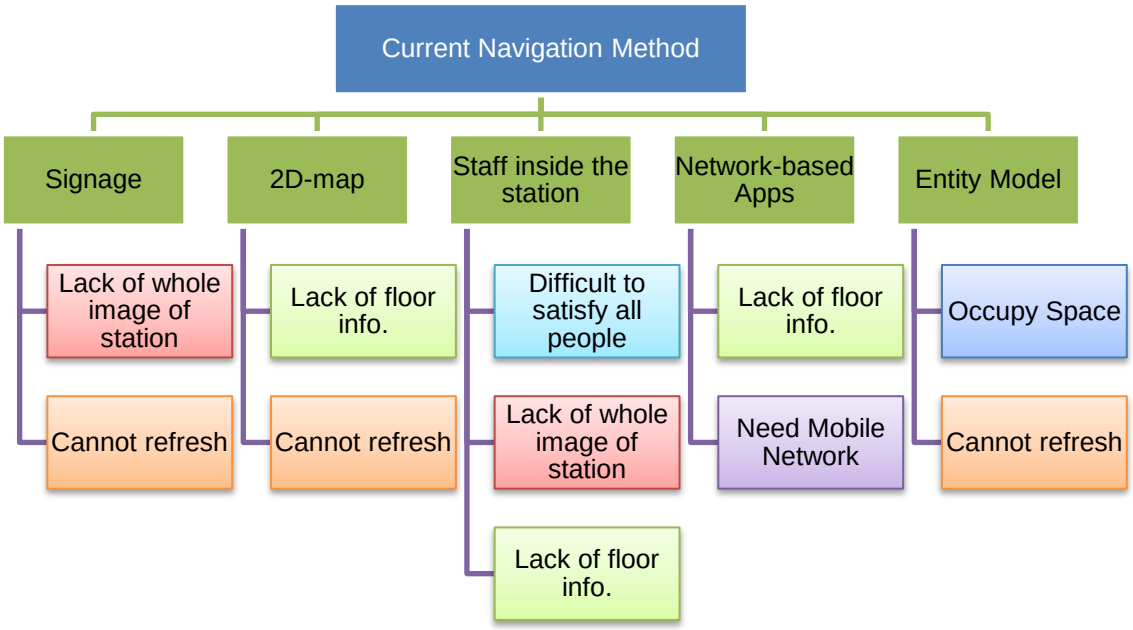


Figure 13: Current Navigation Method and their problem

1.4 Purpose of this Research

In order to solve these wayfinding problem, in this paper, we try to use the proposed system to help users understand the subdivided-integration progress continuously. People from platform A to B, will go through many stairs, branching roads and gates. In this continuous choose-select process, at every essential point, the digital signage, which is a part of our subsystem, will help them to make decision quickly by giving 3D structure. And the multi-to-multi approach helps serve several users at the same time.

The target user is defined as the person who firstly comes to a complex interchange subway station. We are looking forward to a new navigation system which could solve this wayfinding problem under this kind of target user.

2 Proposed System

2.1 System Requirement Analysis

In this section, we try to analysis the system requirement based on end user which are mention in section 1.3. In this research, we mainly focus on wayfinding problem inside complex subway station. The question focus on new visitors who are newly come to complex subway station. In section 1.3, we analysis the current system problem, and each system's problem is collected and showed as Figure 13. Here we list out the problem in center as Figure 14.

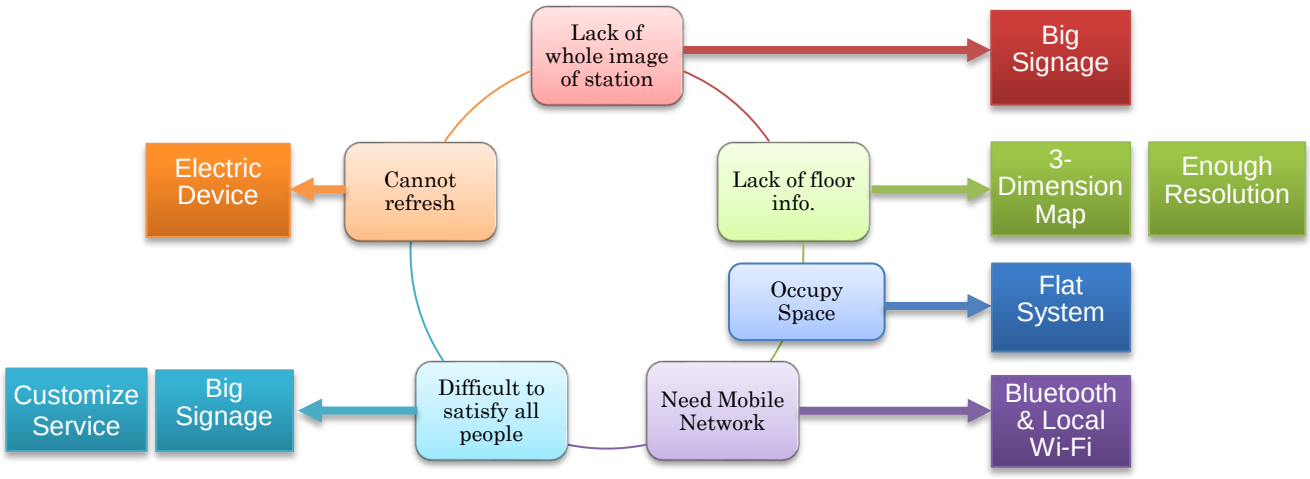


Figure 14: Problems in Current System and Some Solutions.

We analysis the problem and find out some sub-solution for each sub-problem.

Lack of whole image of station is mainly caused by small screen or small media device which whole station structure seems to be difficult to show in that kind of device. This problem can be solved by using big signage system.

Lack of floor information is caused mainly because using traditional 2-Dimemsion Map system. As the 2D image is the projection of 3D model, the complex 3D model could be

difficult to understand since 2D map is only from one viewpoint to look at that 3D model. From the top view is difficult to get floor information, and from left or front view is difficult to get plan metric information. From the Questionnaire 1 (listed in Appendix) which we did, many of subjects (Record Conclusion in Appendix) who took this survey answered that they preferred 3-Dimemsion Map instead of 2D Map. Hence using 3D Map is a better solution. On the other hand, in order to display complex 3D model while remaining enough structure detail, the end user device show also have enough resolution on their screen. In conclusion, lack of floor information can be solved by using 3D Map and big resolution screen.

Cannot refresh, or said, lack or real time information, is a shortness of traditional static media (Post Map, Indicator and etc.). When applied electric devices which are easier to access network could solve this problem.

The problem of occupy space usually happens when applied big entity system inside narrow space. As the developing of digital signage, the screen of digital signage become thinner and thinner which makes it possible to applied flat system attach to the wall in order to make less space usage.

The problem of satisfy all people inside station is usually caused when lack of system capacity and coverage. On the one hand, because of personality of customers, they tend to use their own language and use the system in their own way. On the other hand, there should be enough terminal in this system for users to use. The problem can be solved by applying big signage (many people can use it at the same time) and customize service (each user can use it by their own way).

The problem of requirement of mobile network usually happens when tried to use network based app or software inside underground subway station. It could also happens when visitors try to visit a new place without apply network from local telecom operators (AU,

Docomo and Softbank etc.). It could also happen when enter low signal strength place where lack of GPS signal that is important in locationing.

In this research, the navigation system is designed for new visitors who come to complex interchange subway station deal with wayfinding problem. Hence, analysis of these target users is also important. From the marketing research below (showed as Figure 15, Figure 16, Figure 17), there is high possibility that end user tends to bring their smartphone while newly coming to subway station. They could use build-in app to solve their wayfinding problem, but the interesting thing is many of them are not satisfy with the current system.

Smartphone Users and Penetration in Japan, 2011-2017

	2011	2012	2013	2014	2015	2016	2017
Smartphone users (millions)	**19.0**	**35.3**	**53.0**	**76.5**	**82.2**	**87.4**	**88.9**
—% change	180.1%	85.4%	50.1%	44.4%	7.4%	6.4%	1.6%
—% of mobile phone users	18.0%	33.0%	49.0%	70.0%	74.4%	78.9%	79.9%
—% of population	14.9%	27.7%	41.7%	60.2%	64.8%	69.0%	70.3%

Note: individuals of any age who own at least one smartphone and use the smartphone(s) at least once per month
Source: eMarketer, May 2013

157337 www.eMarketer.com

Figure 15: Smartphone Users and Penetration in Japan. (From http://www.emarketer.com/Article/Smartphone-Use-Japan-Makes-Steady-Gains/1010226)

Number of smartphone users in Germany from January 2009 to February 2015 (in millions)

This statistic shows the number of smartphone users in Germany from January 2009 to February 2015. In May 2014, there were roughly 41 million German smartphone users, an increase compared to February of the same year, at 40.4 million users.

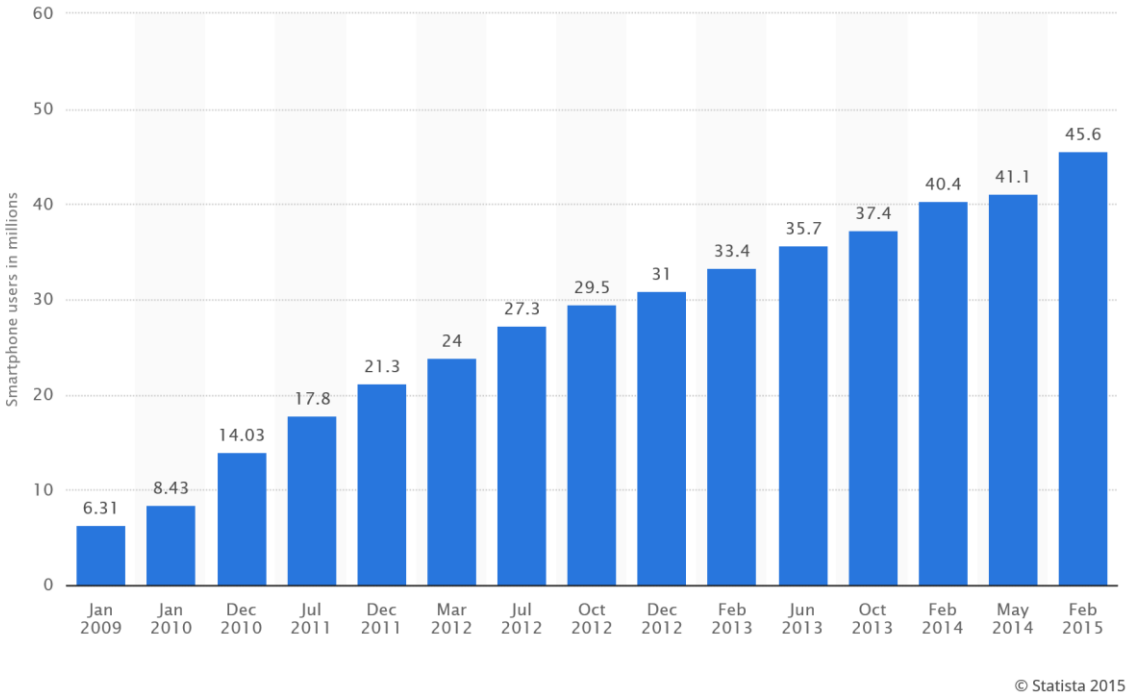


Figure 16: Smartphone users in Germany. (From

http://www.statista.com/statistics/461801/number-of-smartphone-users-in-germany/)

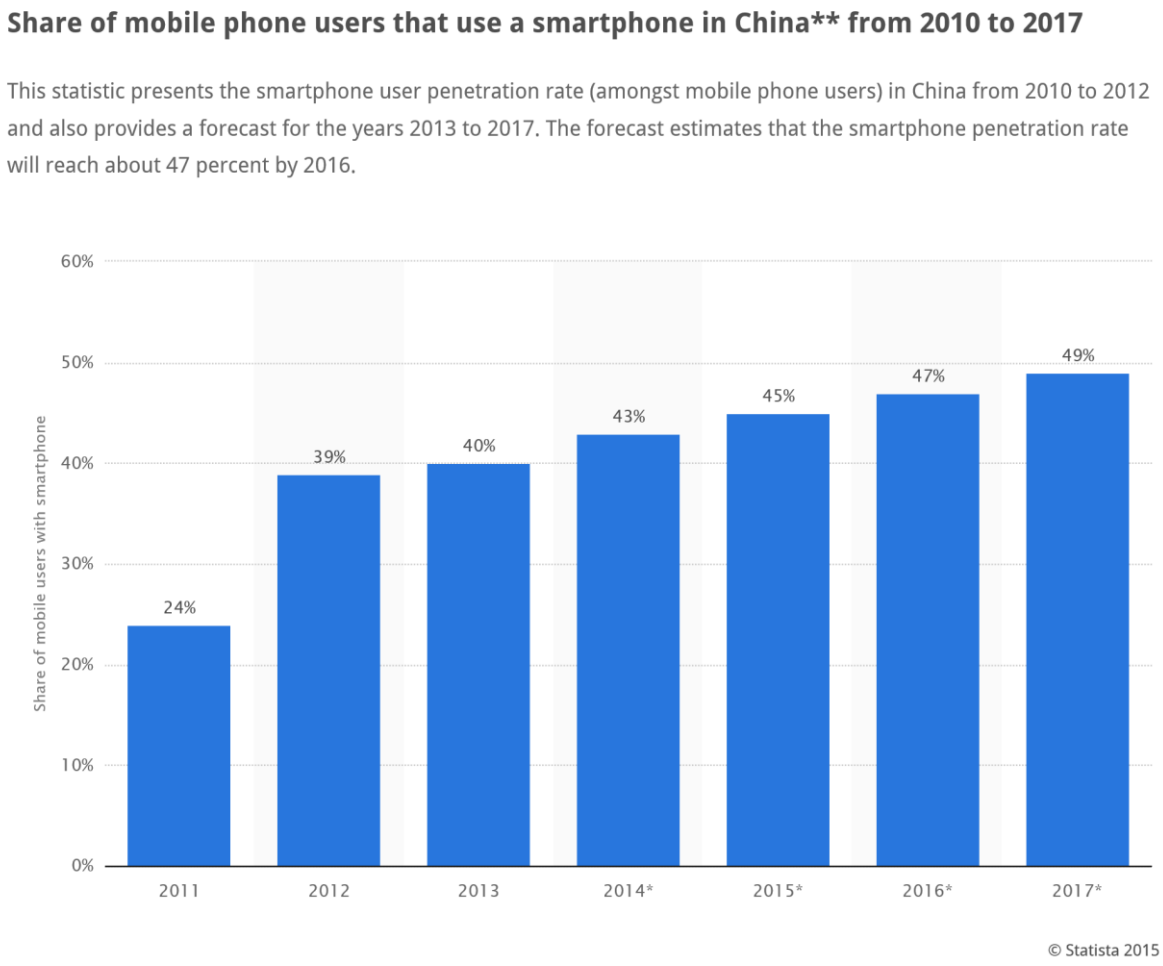


Figure 17: Smartphone users in China. (From http://www.statista.com/statistics/257045/smartphone-user-penetration-in-china/)

With these large amount of smartphone users, nowadays portable navigation app doesn't seem to be difficult to cover end users. The wayfinding problem still occurs because some inherent defect of existing system. The visitors require more advance system which could solve the problem quickly and easily.

By analysis the whole system requirement and target users' requirement, we found out that single improvement approach is difficult to achieve the wayfinding goal and could not satisfy target user. Changing another way, could all the small solution for sub problem combining together become a new system to solve the wayfinding problem?

Hence we list out all the current system problem together and target users' requirement here, showed as Figure 18.

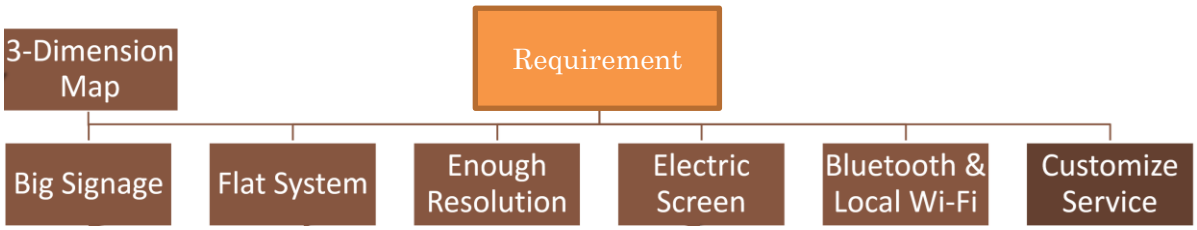


Figure 18: The system requirement overview

Target users want to solve wayfinding problem, and the current system they used has several shortage. Problem solving could be two approach, one is to find a new method to solve the problem, and another is try to improve the existing solution. Problem could be divided into several small problem, on the other hand, several small solutions can also be integrated into final solution. Hence, in Figure 18, we combine all the sub-solution from Figure 14. They are 3D Map, Big Signage, Flat System, Enough Resolution, Electric Screen, Bluetooth & Local Wi-Fi and Customize Service. All the sub-solutions integrated together would be the final solution which target users wanted. In further, we could call this the system requirement, the system required to have these sub-solution to fit the target user's requirement.

2.2 System Concept Development

In Section 2.1, we analysed the system requirement. In this Section, we try to develop system concept over this system requirement. As we mention in Section 0, the navigation system is an effective system to solve wayfinding problem. Considering flat system and big signage requirement, the digital signage, which is thin and flat, is naturally considered

to be involved into our system. Some researches showed that it is efficient to apply digital signage to deal with wayfinding problem [2], [3]. Digital signage generally include a screen system. Recently researches show the power usage of Ultra-High Definition Display (or said 4K) in digital signage [14], [15]. In Figure 15, Figure 16 and Figure 17, the increasing market for smartphone make it possible to combine this kind of ubiquitous portable device into our system. Hence, by analysing these device and comparing their function with our system requirement, the main device choices could be follow, showed as Figure 19.

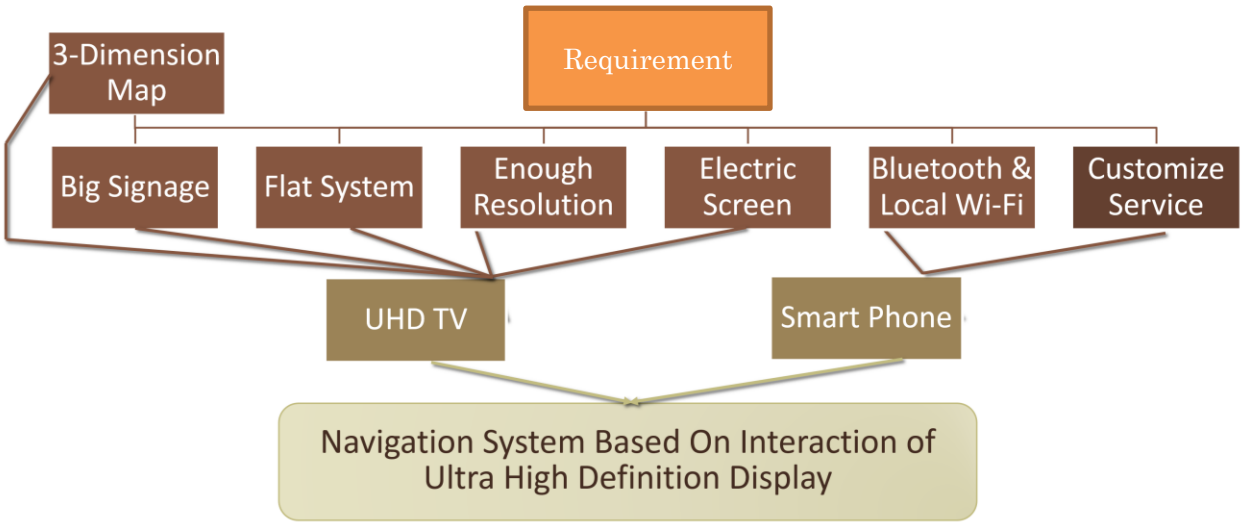


Figure 19: The System Level Solution for Requirement

In Figure 19, a novel concept of Navigation System has been showed. The UHD TV (Ultra-High Definition Television) has the capacity with big screen, flat system and enough resolution. Detail specification for UHD TV will be introduced in Section 3.1. The Smartphone supports Bluetooth & Wi-Fi at the same time. Even without Internet, we can use local network to satisfy these system requirement. Combining UHD TV and Smartphone together could lead to better solution result. User could use Smartphone to interact with UHD TV, try to get more information which is essential to the wayfinding

process. System could also provide real time information to end user. Base on this concept, we develop this system more specific in next Section.

2.3 System Overview

In this section, we proposed a new navigation system trying to help solve wayfinding problem inside the complex subway station. Base on the concept we develop in Section 2.2, the system including UHD TV and Smartphone which we could called them subsystem. In a more general way, we use UHD Display to represent all the UHD TV. The user will use Smartphone to interact our system.

The system architecture prototype overview is shown as Figure 20. There will be several users and several UHD displays. Each user can use a Smartphone to interact with a UHD display in this navigation system. The data transmission is done by local network system. All the data will go through a central server. Inside the central server, there is a database which provide necessary information for Smartphone and UHD Display. The Station 3D model will also install in the server.

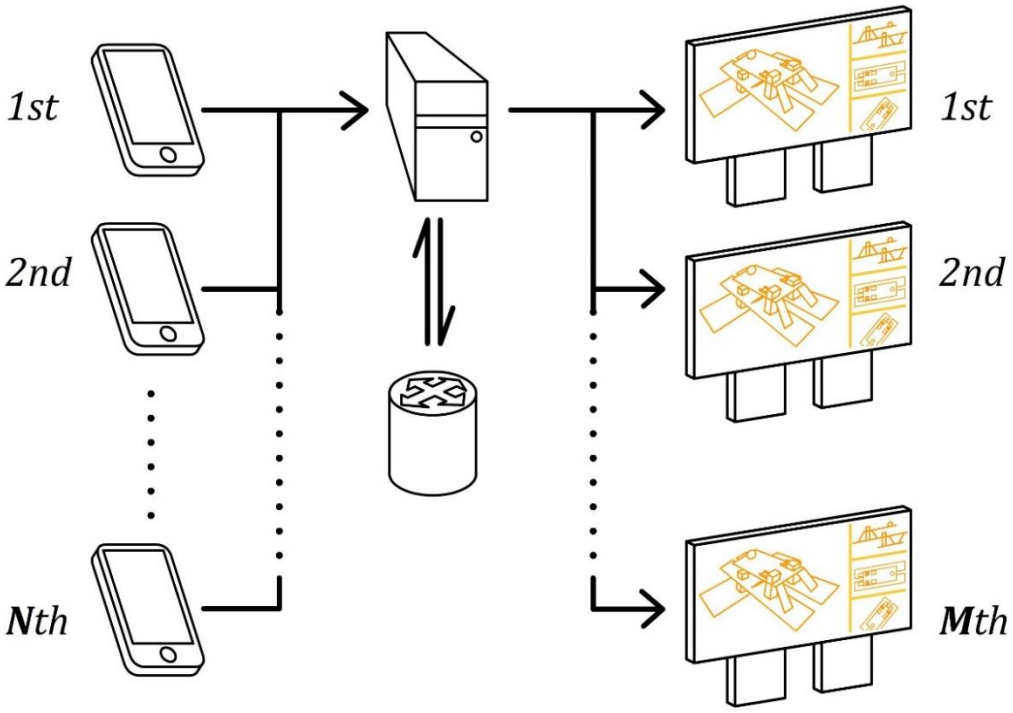


Figure 20: The system architecture prototype overview

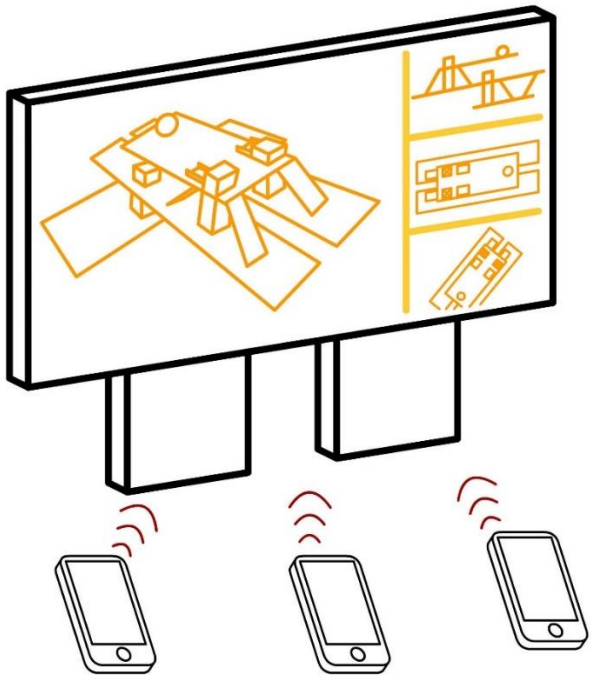


Figure 21: Several users use single Navigation System at the same time.

As Figure 21, the system allowed several users to interact with the 3D-map of station structure through smart phone and UHD display. The interaction data will be transmitted and collected by a navigation server. The data will be delivered to the target UHD client in order to control the contents inside screen. In this research, we designed 1 main screen and 3 sub screens for each digital signage sub system.

For prototype design of this system, there will be a numbers of UHD display as digital signage inside station at every essential position. These positions can be the exits, the place near stairs, and the entrance where people usually trend to make decision. Instead of looking at a small screen only, users interact with this system to find out where they are, where they should go, and how to go there.

The system uses the benefit of web technology like WebGL and WebSocket which expected to reduce development cost and time. And the system also uses gyroscope sensor and touch screen of the smart phone to capture user's hand gesture, in order to give user a better control feeling.

The whole system design overview is showed as Figure 22. The system design overview lists out all the design from abstract level to detail level.

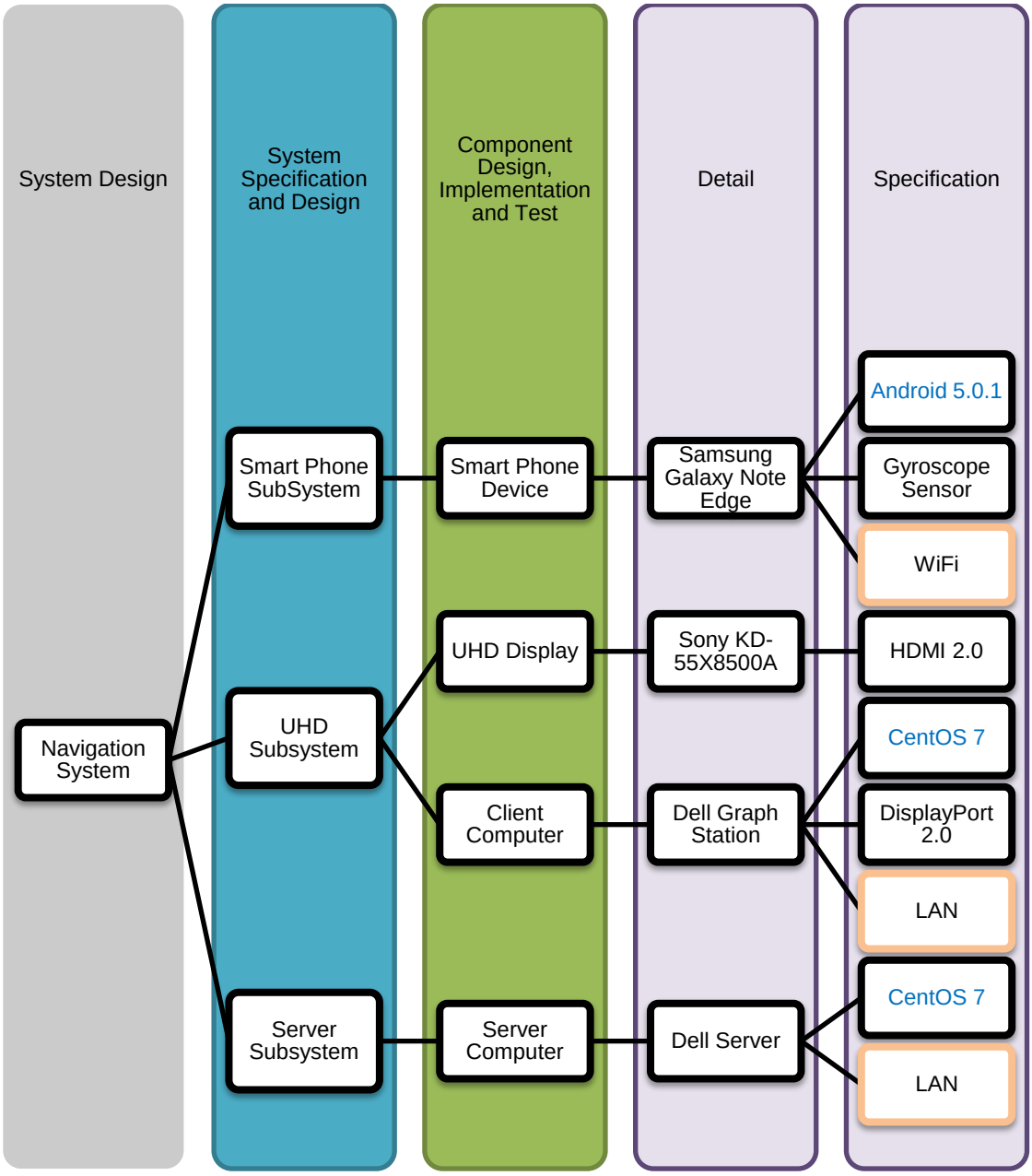


Figure 22: System Design and Detail Design.

2.4 System Architecture

In this section, we transform our system design into system architecture level. Our proposed system combines several subsystem, showed as Figure 23. They are Smartphone subsystems, Digital Signage subsystems and a Central Server. Central Server is responsible for information collection and distribution. Digital Signage subsystems contains a UHD (Ultra-High Definition) display, a BLE Beacon and a client computer. The Smartphone subsystems require the Smartphone has Wi-Fi support, BLE support and gyroscope sensor. The Network environment inside these system is made by applying switch, router and Wi-Fi access point. The client computer and Central Server is accessed to Network by LAN cable. The Smartphone is accessed to Network by Wi-Fi. The BLE Beacon will transmit Bluetooth Low Energy signal for Smartphone to detect the distance between Digital Signage and user.

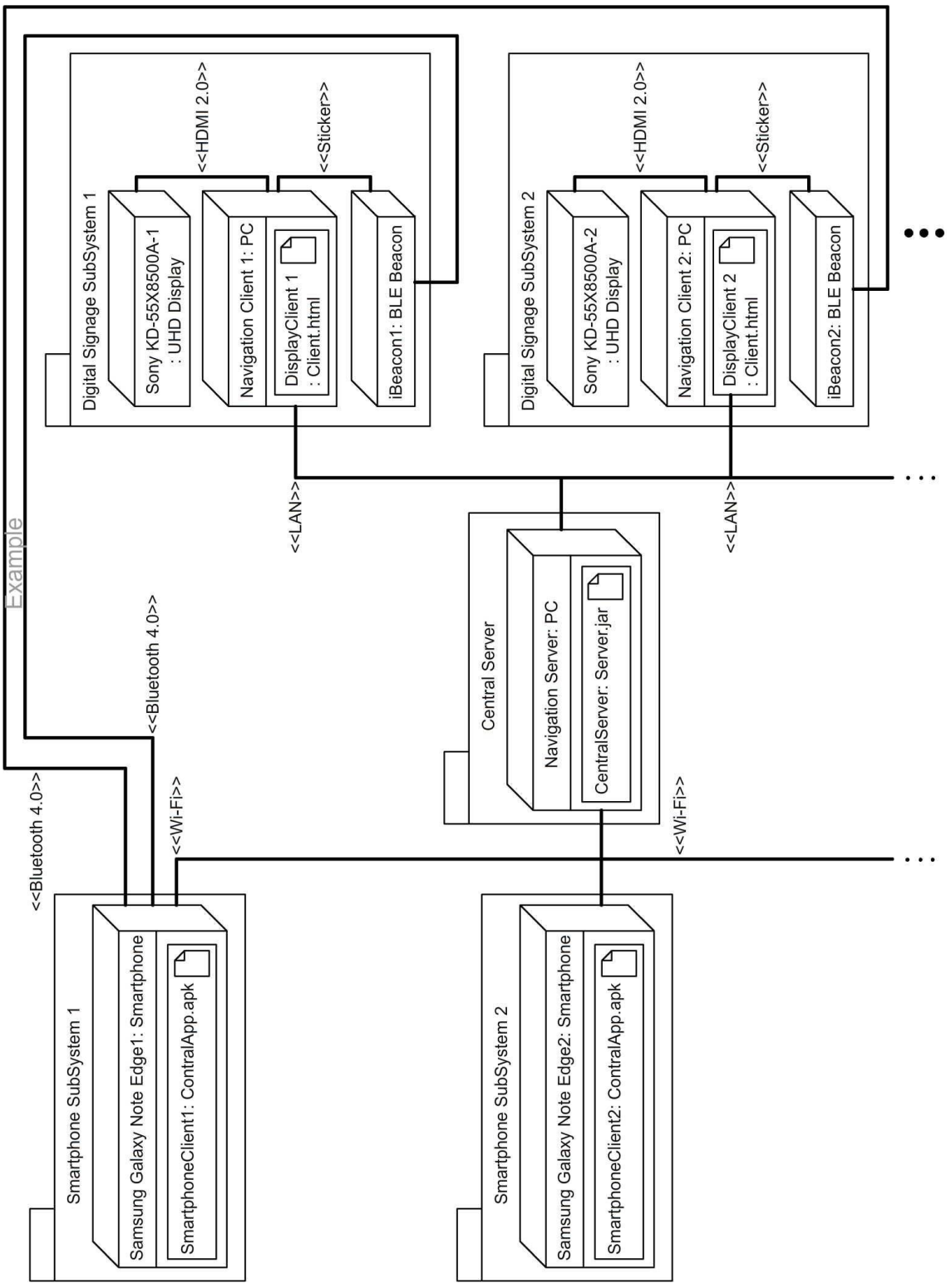


Figure 23: Deployment Diagram of the proposed system

For software implementation, inside smartphone subsystem, each smartphone will be installed an Android APK called ContralApp which is developed for collecting sensor data and managing data sending-receiving task. Inside Central Server, there is an executive server application called Server which is programmed in Java. Inside digital signage subsystem, the client PC will open a webpage called Client which is made by HTML 5 with WebSocket and WebGL to show the content of station model.

The Figure 23 shows two smartphone subsystem and two digital signage subsystem. There is one interact example of smartphone subsystem 1 with digital signage subsystem 1 and 2. Firstly, the Central Server will start listen the client. Then, several Digital Signage inside station will apply connection to Central Server. The Central Server will provide each connection a thread to handle. Each Digital Signage has its own identity code which registered in Central Server. When Smartphone receives remote BLE signal, it will send the identity code that provided by this signal. As Figure 23, smartphone subsystem 1 receives 2 BLE signal, one is from digital signage subsystem 1 and another is from digital signage subsystem 2. Assume that digital signage subsystem 1 is closer to user, the BLE signal strength is stronger, and then the smartphone will try to send this closest BLE identity code to the server. The Central Server receive that signal and try to combine the correct connect path for Smartphone and Digital Signage. After the connect path has been established, the user could choose whether to control or not.

If user chooses to control this digital signage subsystem 1, then inside the server, the smartphone will be put in waiting list. If any of 3 sub screens is available, the available sub screen will be reserved for smartphone subsystem 1. The sensor data, showed as Table 1, collected from smartphone will be transmitted to server and then delivered to destination digital signage subsystem 1. The data, showed as Table 2, include information that is used to indicate which sub screen to control will be packed with WebSocket header

and sent. After that, digital signage subsystem 1 unpacked data and apply motion data to certain sub screen with WebGL Three.js library.

Table 1. The data from Smartphone to Central Server.

Gyroscope X	Gyroscope Y	Scrolling	UUID
Double	Double	Double	String

Table 2. The data from Central Server to Digital Signage Subsystem.

Sub Screen	Gyroscope X	Gyroscope Y	Scrolling
Short	Double	Double	Double

3 System Design Detail

In previous Section 2, we developed the system concept of Navigation Digital Signage System. In this Section, the system design detail will be discussed and implemented. The system is combined by different components, and each component has its own function. All the function integrates together into a whole system.

The interaction is an original feature of the proposed system. In order to provide interaction feature, we used smart phone's gyroscope sensor and touch screen to collect user's gesture data, and transmitted it to the navigation server. After that, the navigation server analyses the data and transforms it into WebSocket data form, and finally transmits this WebSocket package to the UHD display client.

3.1 Ultra-High Definition Display

The Ultra-High Definition display, or 4k display, gains large demand nowadays. The application of 4K TV has been described in a positive prospects [16], and the application of UHD display for scientific and social problem solving is expected to grow. This kind of display provides 4 times resolution than Full-HD display. In such screen, more lines of text and details can be provided to user. The best viewing distance to enjoy 4K screen is around 1.5H (1.5 times of 4K screen's height) according to the NHK researchers [17].

In this research, we choose Sony KD-55X8500A TV as our UHD-display. The screen height is 74.0 cm and the best viewing distance is 111.0 cm. The resolution of this UHD display is 3840×2160 pixels. The display signal is transmitted from computer through HDMI cable. With HDMI 2.0 and NVIDIA graphic card, the display signal could be transmitted in 2160p 60fps format.

3.2 Gyroscope Sensor

The gyroscope sensor is a device which can provide rotation information around three axis (x, y and z, which are orthogonal to each other). With the technology developing, gyroscope sensor is small enough to be integrated as a single chip in many smart phone.

The rotation information provided by gyroscope sensor is measured in rad/s. The direction of rotation follow the right-hand rule, for example, if we rotate the smart phone through x-axis by counter-clockwise direction, then the gyroscope sensor will give a positive value. Otherwise, it will give negative value.

For smart phone device in this research, the 3 rotation axis is shown as Figure 24. The z-axis vector starts from the backside of smart phone, towards to the screen side. The y-axis starts from the center, towards to the top of smart phone. And the x-axis starts from the center, towards to the right side. The rotation relationship is also shown as Figure 24.

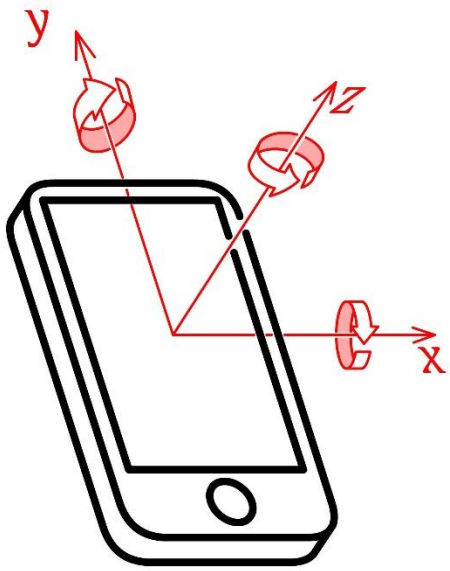


Figure 24: The rotation axis and relationship of smartphone

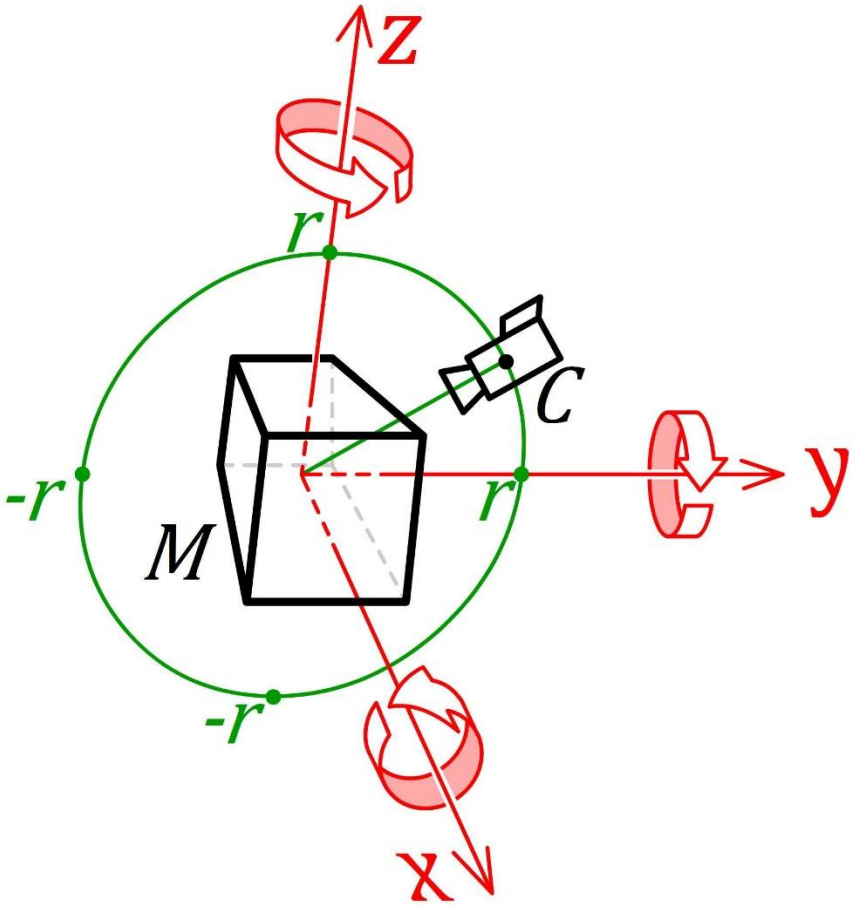


Figure 25: The rotate axis of object (3D map)

Inside the UHD display, the 3D object and the relationship with camera is shown as Figure 25. We also use a 3-dimension coordinate system to describe rotation relationship. In Figure 25, the object is represented as a wired cube in the center. In real application, this cube will be replaced to subway station map. The camera is put in Y-Z plane and will be rotated around x axis (path is shown as path circle in green color). The center of path circle is at the same position of coordinate system center, and the radius of path circle is measure as value r. The camera's viewpoint is from point C towards to circle center. The head of camera always towards to the top, as the same vector as positive Z-axis direction (0, 0, 1).

Because two coordinate systems are different, we describe the motion separately. The coordinate system inside smart phone, which designed by google, is described as SPCS (Smart Phone 3-Dimensional Coordinate System). The coordinate system in UHD display is described as OCS (Object 3-Dimensional Coordinate System).

In smart phone, shown as Figure 24, the rotation data of y-axis, z-axis and x-axis will be used as rotation control. When smart phone is rotated around y-axis or z-axis in SPCS, the object will be rotated upon z-axis in OCS. For which axis to rotate in SPCS as control data, we allow user to choose as they like. When smart phone is rotated around x-axis in SPCS, the camera C will be rotated around x-axis in OCS.

In conclusion, in smart phone, in order to avoid mis-operation, only 2 axis are allowed to be rotated around. As a result, in SPCS, either x-axis and y-axis, or x-axis and z-axis are used to control the object and camera. In OCS, the object is rotated around z-axis, and the camera is rotated around x-axis.

As implementation, in this research, we choose Android Smartphone as our experiment device. The App inside Smartphone is developed in Java programming language. The following code is the essential code for acquiring sensor data from Smartphone:

```
final SensorEventListener myAccelerometerListener2 = new
SensorEventListener() {
       //gyroscope
       @Override
       public void onSensorChanged(SensorEvent sensorEvent) {
          if(sensorEvent.sensor.getType() == Sensor.TYPE_GYROSCOPE){
             gypoX=sensorEvent.values[0];
             gypoY=sensorEvent.values[1];
             gypoZ=sensorEvent.values[2];
             DecimalFormat df = new DecimalFormat("######0.0000");
             tmp2="¥n¥n" + "x:" + df.format(sensorEvent.values[0]) + " ¥ny:"
+      df.format(sensorEvent.values[1])            +      "      ¥nz:"      +
df.format(sensorEvent.values[2]);
          }
       }
       @Override
       public void onAccuracyChanged(Sensor sensor, int accuracy) {
       }
   };
```

The 3 axis rotation data will be saved into Double Type value gypoX, gypoY, gypoZ. After exiting the app, the program will destroy the registration for sensor in order to saving power for Smartphone. The destroy code:

```
public void onDestroy(){
       sensorManager.unregisterListener(myAccelerometerListener2);
       try {
          socket.close();
       } catch (IOException e) {
          e.printStackTrace();
       }
       super.onDestroy();
}
```

3.3 Scrolling for Zooming

In order to allow user to zoom the object, we include scrolling function on smart phone screen. When the App is opened, the whole screen will be a touch sensor. If user uses finger to scroll in negative y-axis direction (0,-1,0) on screen in SPCS, the value r will be reduced and camera will be more closed to the object, vice versa. The zooming function as for:

```
    @Override
    public boolean onTouch(View v, MotionEvent event) {
       return gestureDetector.onTouchEvent(event);
    }
    @Override
    public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
float distanceY) {
       if (e1.getY() - e2.getY() > FLING_MIN_DISTANCE) {
          ySpeed=distanceY;
       } else if (e2.getY() - e1.getY() > FLING_MIN_DISTANCE) {
          ySpeed=distanceY;
       }
       return false;
    }
```

3.4 Link from Smart Phone to Server

From smart phone to navigation server, we used local Wi-Fi to transmit mobile's sensor data. In this research, the test smart phone used Android operating system. Firstly, the smart phone will use TCP/IP to open a communication link to the server. After open the link successfully, the gyroscope sensor and scrolling data will be transmitted to the server and waited for process. In this research, we considered the balance between data flow and smooth feeling. The data flow should not be too high in order to save bandwidth of wireless service. And the refresh rate should high enough to get smooth feeling. The smart phone will transmitted data package in 60Hz that means around 16.67ms one data package will be sent to the server. While linking to the server, the implementation are the following code:

Firstly, open a new Socket for TCP/IP connection:

```
socket = new Socket();
OpenConntThread oct=new OpenConntThread();
sdt=new SendDataThread();
oct.start();
```

Then execute the OpenConntThread for opening the connection:

```
class OpenConntThread extends  Thread{
    @Override
    public void run() {
        try {

            int aaa=0;
            socket.connect(new InetSocketAddress(ipaddr, 5000), 2500);
            isOpen=socket.isConnected();
        } catch (IOException e) {
            isOpen=false;
            e.printStackTrace();
        }

        if(isOpen) {
            try {
                outStr = socket.getOutputStream();
                output2 = new PrintWriter(socket.getOutputStream(), true);
```

```
            } catch (IOException e) {
                isOpen = false;
                e.printStackTrace();
            }

            try {
                bff           =           new           BufferedReader(new
InputStreamReader(socket.getInputStream()));
            } catch (IOException e) {
                isOpen = false;
                e.printStackTrace();
            }
        }

        if(isOpen){
            sdt.start();
        }
    }
}
```

Finally, open a Loop to continuously sending the motion data to the server:

```
class SendDataThread extends Thread {
    private int count=0;
    @Override
    public void run() {
        try {
            Scanner scn;
            scn = new Scanner(socket.getInputStream());
            String msg=scn.nextLine();
            String[] msgs=msg.split(":");
            isOpen = msgs[0].equals("com") && msgs[1].equals("uuid?");
            if(isOpen){
                output2.println(UUID);
            }
        } catch (IOException e) {
            e.printStackTrace();
            isOpen=false;
        }

        while(true) {
            if(isOpen) {
                String sendString;
                if(isYaxis) {
                    sendString  =  AngleX  +  "  "  +  AngleY  +  "  "  +
gypoX*camtotatespeed + " " + gypoY*objrotatespeed + " " + ySpeed+" "+UUID;
                }else{
                    sendString  =  AngleX  +  "  "  +  AngleY  +  "  "  +
gypoX*camtotatespeed + " " + gypoZ*objrotatespeed + " " + ySpeed+" "+UUID;
                }
                try {
                    output2.println(sendString);
                    ySpeed=0.0f;
                }catch(java.lang.NullPointerException e){
```

```
                    isOpen=false;
                }
                try {
                    Thread.sleep(7);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }else{

                try {
                    Thread.sleep(100);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                count++;
                if(count>20){
                    break;
                }
            }
        }
    }
}//End of Thread
```

3.5 Process Inside Server

The navigation server use Java environment to establish network in order to help delivery smart phone sensor data to UHD client. Server will receive the sensor data, analyse it and send it to the UHD client in order to perform the model and camera rotation.

From navigation server to UHD client, the data will be sent by WebSocket technology. Before sending data to UHD client, firstly the client should open the web page which coded by WebGL and WebSocket to connect the navigation server. After WebSocket hand shaking, the link between navigation server and UHD client will be established. Then the motion data will be sent to the UHD client. Detail is in Section 3.7.

3.6 UHD Client, WebSocket and WebGL

The UHD client combines a UHD display and a computer. The display signal is transmitted through HDMI cable from computer to UHD client. Firstly, the UHD client will link to the navigation server through WebSocket technology. After data is sent by smart phone, these data packages will be processed in server, and then target UHD client will receive sensor data.

The WebSocket, different from conventional socket link, is used in web page. Since the Internet service gains large demand in this age, interacting with contents is not only important in conventional computer or smart phone Apps, but also in web page. Hence, in this research, we tried to combine navigation system with WebSocket technology in order to provide better system deliveries. Besides the benefit of embedding in web page, the WebSocket uses JavaScript to design the application, which can be combined with WebGL technology and reduce the learning cost.

The UHD client used WebGL to create a rendering system in order to achieve quick model development purpose. Figure 26 shows original 3D model of Hiyoshi station. In proposed system, the station model was built by open source 3D model software Blender. With the transform plug-in, the model data can be exported into *.js file, which rendered by WebGL through JavaScript on web page as Figure 27.

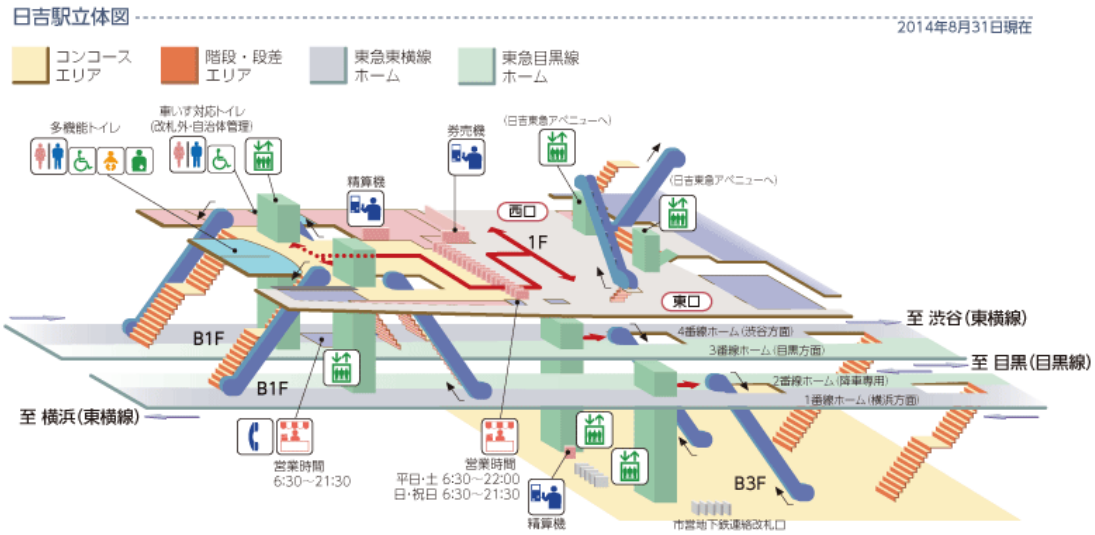


Figure 26: Static 3D model example of Hiyoshi station. (From Tokyu official website: http://www.tokyu.co.jp/railway/station/yardmap/?id=13)

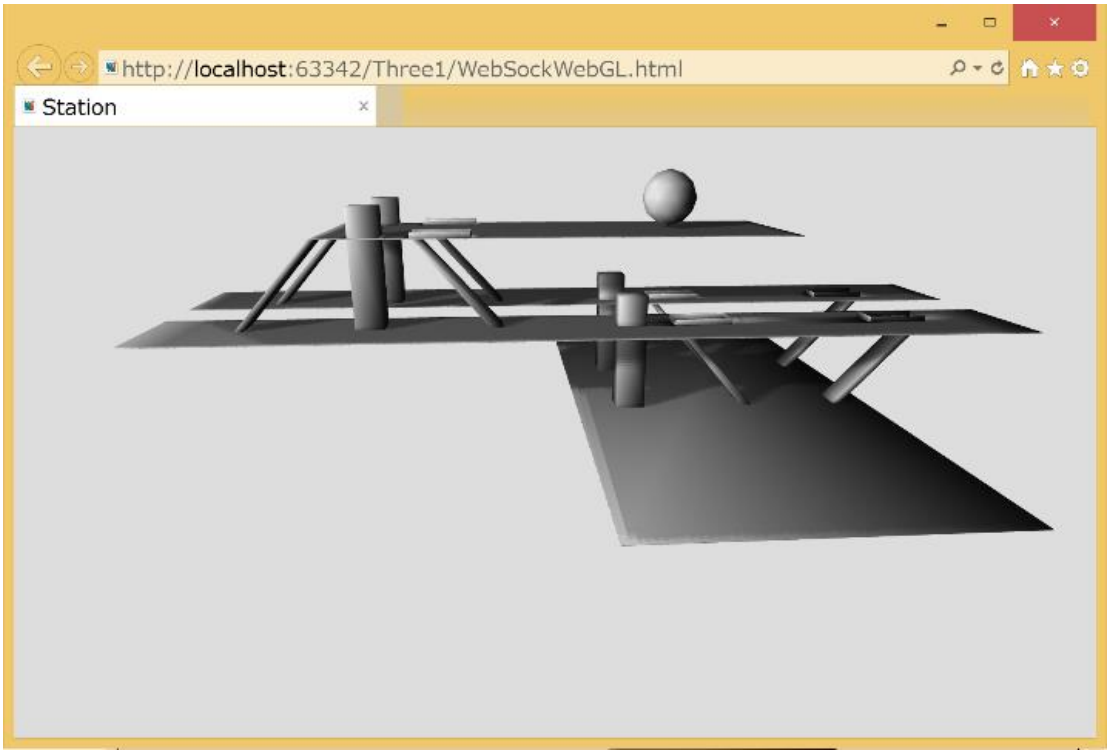


Figure 27: Prototype of 3D station model in web page

The benefit of WebGL makes station map delivery more widely. Not only can this navigation system use station model file, but also station official website can use it for

advertisement and commercial purpose.

In an overview, the UHD client receives the sensor data from navigation server through WebSocket technology in a web page, and then the WebGL gets the data through the same page to apply rotation matrix, in order to change the direction of object model (the 3D station map) and camera. This model and scene will be rendered on web page and transmitted to UHD display through HDMI cable. Finally client will show motion and give interaction feeling to user.

As implementation, the model importing code of WebGL using Three.js library is:

```
    var loader=new THREE.JSONLoader();
    loader.load("hiyoshi.js",
           function (model,material) {
              var mesh=new THREE.Mesh(model,material[0]);
              //mesh.computeTangents();
              mesh.position.x += 16;
              mesh.position.z -= 2;
              mesh.position.y += 10;
              mesh.rotation.x = -Math.PI/2;
              mesh.castShadow = true;
              mesh.receiveShadow = true;
              monkey=mesh;
              scene.add(mesh);
           },"texture/"
    );
```

The essential code of WebSocket is follow:

```
function initWebSocket(){
    webSocket = new WebSocket(wsUri);
    webSocket.onopen = function(evt) { onOpen(evt) };
    webSocket.onclose = function(evt) { onClose(evt) };
    webSocket.onmessage = function(evt) { onMessage(evt) };
    webSocket.onerror = function(evt) { onError(evt) };
}
```

3.7 Multi-Client Handling Method

As we mention in the earlier design (System Overview, Section 2.3), the proposed Navigation Digital Signage System is not only for single user. The system requirement specify the multiple user approach. In this Section, the Multi-Client Handling Method will be introduced to fit the Multi-to-Multi system requirement.

In hardware scope, the Central Server will handle the main task of Multi-to-Multi approach. In software scope, the main server executive program will provide Multi-to-Multi approach by applying multi-thread function. The multi-thread function is like multiple small workers inside the software process. Under this process, each worker will have a handler to deal with the remote connection. The main thread is like a manger to manage all the workers in order to make sure everything works properly. The detail explanation is showed as Figure 28.

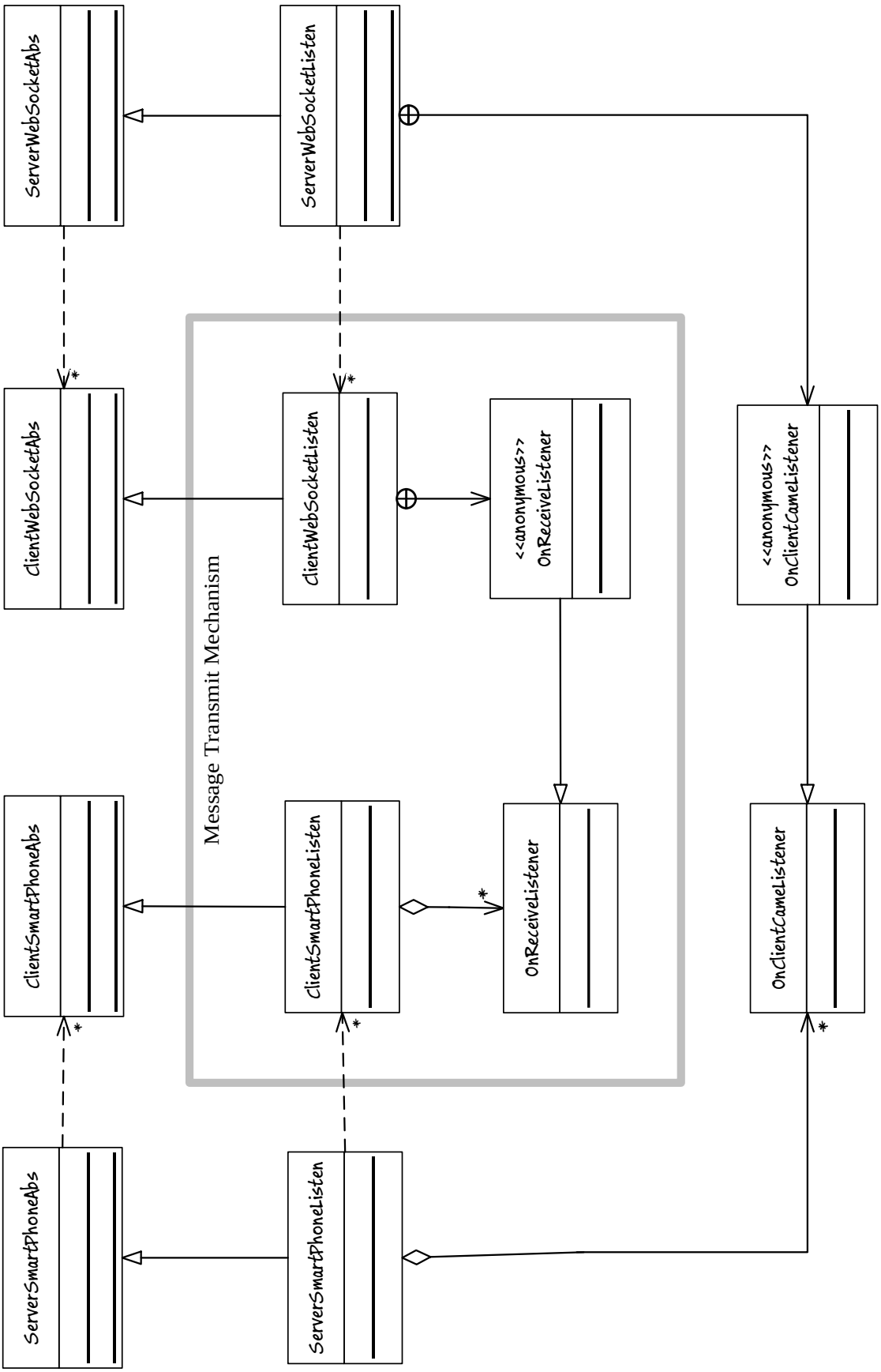


Figure 28: Class Diagram of the proposed Multi-Client Handling Method

Inside the Central Server, different from single link server design, we applied multi-thread client handler to process the incoming request. After launch the server execute file, the server will firstly open two main threads, one is Socket-Server thread for incoming Smartphone connection, another is WebSocket-Server thread for incoming Digital Signage connection. Figure 28 shows the Class Diagram of inner server, ServerSmartPhoneListen is for Socket-Server thread, and ServerWebSocketListen is for WebSocket-Server thread. These two threads are almost the same except the special process function which inside WebSocket-Server. Each main thread will open a port number and create a loop, in order to continuously listen the port and receive any connection at any time. Each main thread will also create a listener, which like a inform interface for other to inform it when necessary. The listener from one main thread will pass on another main thread. Another main thread will hold the listener and will use it to inform opposite thread. In this system, the Socket-Server thread has the listener from WebSocket-Server thread, and the WebSocket-Server thread also has the listener from Socket-Server thread. Currently there are two port numbers have been used, one is for Socket-Server, another is for WebSocket-Server.

Here shows the example code for openning Server of proposed system

```
public class MainClass {
    private static ServerWebSocketListen s;
    private static ServerSmartPhoneListen sspl;
    public static void main(String[] args) {
        sspl=new ServerSmartPhoneListen(5000);
        s=new ServerWebSocketListen(4999);
        sspl.startServer();
        s.startServer();
        sspl.addOnClientCameListener(s.getOnClientCameListener());
    }
}
```

When there is a connection from Digital Signage, the WebSocket-Server will transfer the incoming connection into a WebSocket handler, showed as ClientWebSocketListen in Figure 28 (Also seen at Appendix – B: Programming Code for Central Server). This handler is a client thread that has a loop for receiving incoming data message from Digital Signage‘s client computer. The handler will first evaluate the identity code, and tell the main thread to allocate itself to proper position at the main list. The main list contain lots of handlers and every handler handles one connection. Every digital signage subsystem can define how many sub screens it has. The bigger screen could have more sub screens and the smaller one could have less. A secondary list is created in order to manage those sub screens.

When there is a connection from Smartphone, the Socket-Server will transfer the incoming connection into a Socket handler, which represented as ClientSmartPhoneListen in Figure 28. This handler will first evaluate the identity code from Smartphone, then check the UUID of BLE Beacon and compare UUID to the main list of WebSocket-Server, in order to find out which Digital Signage Subsystem it wants to control. If the certain subsystem (or said, a handler) is found, the ClientSmartPhoneListen (handler) will be put in secondary list inside WebSocket handler. Which sub screen is allowed to control depends on the available slot in secondary list.

In this research, we design 3 sub screens for user to control. If there are 4 users connect to this digital signage at the same time, then the 4th one will be in waiting list. After either of 3 users who quits the control, as 1 sub screen now is available, then the 4th one will get that control. When the control established, the motion data will be sent to the OnReceiveListener, which be established to transmit data from one Socket handler (for smartphone) to destination WebSocket handler (for digital signage).

4 Verification and Validation

In this section, the whole system will be broke down into single link sub system and multi-to-multi system in order to be verified and validated. If the single link sub system could be verified and validated, then only that we continue to do verification and validation in multi-to-multi system. In this verification & validation process, we choose Hiyoshi's and Shibuya's station structures to be our evaluation model. We used frame rate to verify the data transmit speed and model rendering speed. These two items are important to system's performance, and will contribute to the frame rate. The frame rate should be at least 45 fps average to fit the system requirement. In validation process, we invited some subjects to join our experiment, designed some questionnaire to get the feedback from the end user in order to validate the whole system fit the target user's requirement or not.

4.1 Single Link Sub System Verification & Validation

In this reseach, a navigation system based on the interaction of Ultra-High Definition display is proposed. A single interaction subsystem has been built for verification and validation. In this subsystem, only one user is allowed to control the 3D object at the same time. And only one UHD display is used to test. The 3D object of this subsystem is Hiyoshi Station model which showed inside webpage as Figure 27. The initial viewpoint and the position of 3D object is also shown as Figure 27.

In the verification, we used stats.js library to test the frame rate of our system. This library is introduced by mrdoob (http://mrdoob.com/). It can show the frame rate as well as delay in WebGL rendering process. The screen shot of stats.js recorder is showed as Figure 29.

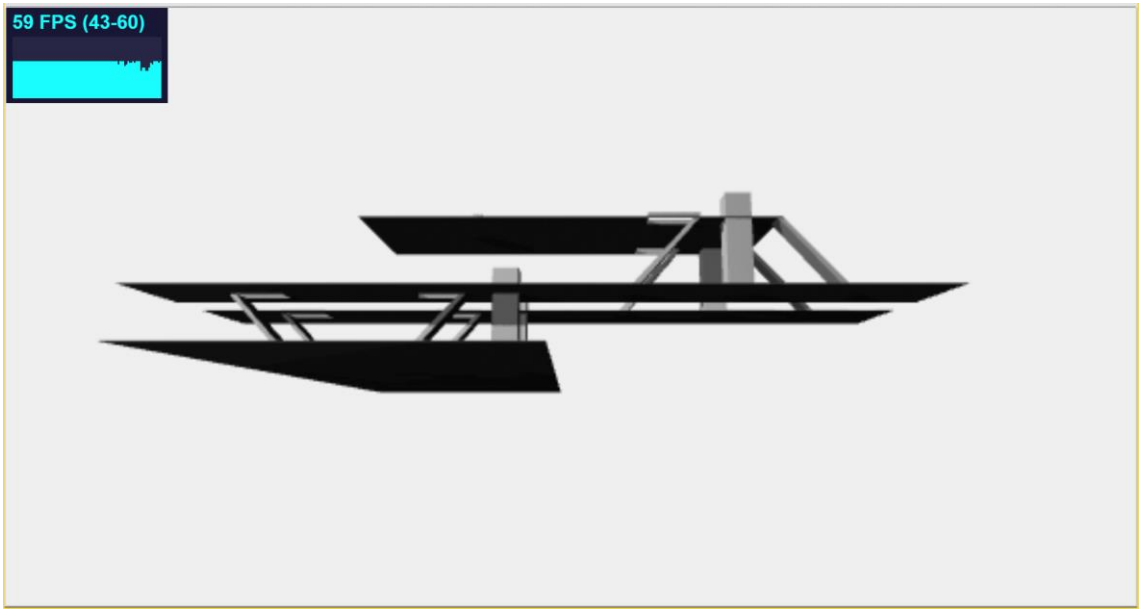


Figure 29: Stats.js Frame Rate Recorder. (In the left top area)

As showed in Figure 29, the frame rate will be fully renewed in every 73 to 76 second. In this research, in order to simulate the real implementation environment, we did the verification and validation at the same time. While test subjects doing validation, the stats.js would also record the frame rate. The frame rate was recorded for every experiment subjects during their test period.

In the validation, 10 university students were invited as the experiment subjects. The subjects were asked to use smart phone to control the 3D object. While controlling object, the subjects were asked to answer a questionnaire. In this system, subjects needed to use one hand or two hand to hold the smart phone. Thus, many of them feel like unpleased to put down the hand and take pen to answer questionnaire. In order to solve this problem and let subject focus on screen and smart phone, the researcher recorded their answer instead of subjects themselves. After experiment, the subjects would check researcher's record of their answer, and gave sign in if there is no error or miss.

The questionnaire is shown as follow:

(Q1). Smoothness of Motion

(Q2). Most Prefer Axis for Object

(Q3). Rotation Speed for Camera

(Q4). Rotation Speed for Object

(Q5). Comment

The Q1 is measured by value 1 to 5. When the subject rotates smart phone, if the 3D object inside the UHD client moves immediately, then it means smooth. If the same motion in smart phone is done, after several delay the 3D object just starts to move, then it means slack or not smooth. 5 represents the smoothest, and 1 represent non-smooth.

In Q2, subjects were asked to rotate around y-axis in SPCS, and then shifted to z-axis. Then they were asked which rotation axis they would like to use.

The Q3 and Q4 are measured by the rate of smartphone rotation and object or camera rotation. The rate here is represented as k in (Equation 1):

$$k = \frac{R_o}{R_s} \qquad \text{(Equation 1)}$$

The R_s and R_o both represent rotation speed. In Q3, the R_s represents the rotation speed around y-axis or z-axis in SPCS according to the result of Q2. The R_o represents the rotation speed of object around z-axis in OCS. For example, if the rate k equals to 1, it means that when subjects rotate smart phone for 360° degree, then the object will also be rotated 360° degree. If the k equals to 2, it means that when smart phone is rotated 360° degree, the object will be rotated 720° degree. In Q4, the difference is that R_o represents the rotation speed of camera around x-axis as Figure 25. In a word, the higher

k it is, faster object or viewing rotation it will be.

As for Q5, the subjects could write any comment if they want. As a free part, the result will be discussed later.

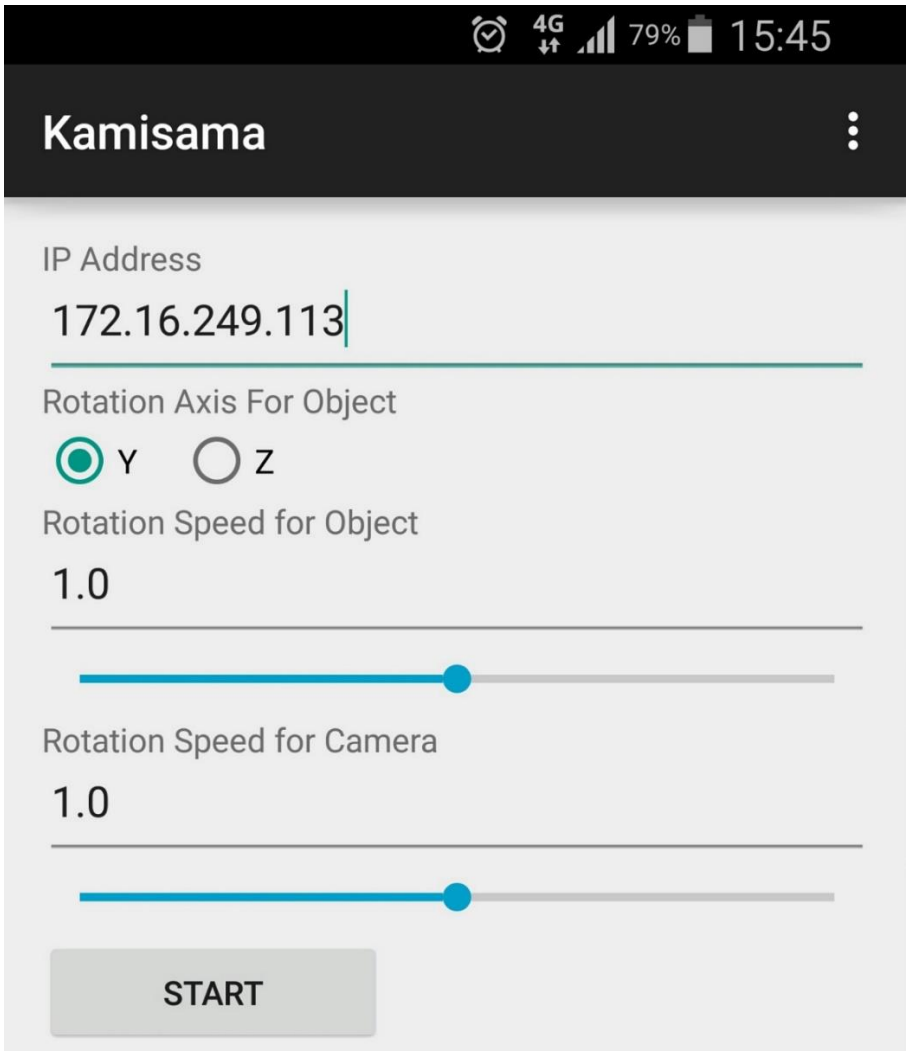


Figure 30: The interface of control App for smart phone

The whole questionnaire time for one subject was around 5 to 7 minutes. Before starting questionnaire, the UHD client and Navigation server already linked together. The subjects could use Apps interface of smart phone to set up the rotation axis and rotation speed as Figure 30. The IP Address here was the Navigation server's address and set by researchers. After set up axis and speed, start button would be clicked to start.

4.2 Result of Single Link System Verification & Validation

In this Section, the result of Verification and Validation will be showed.

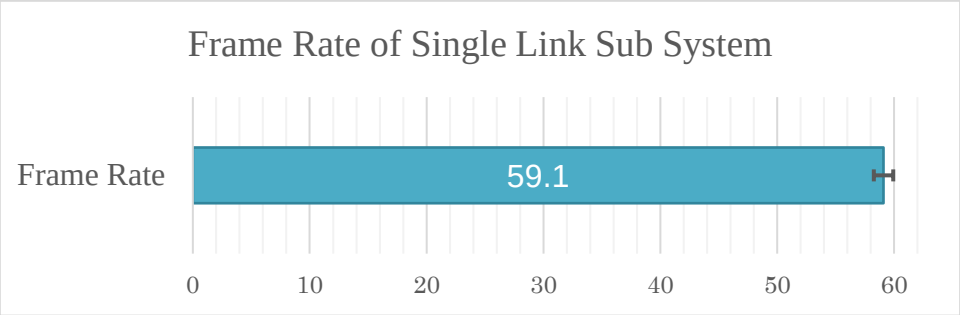


Figure 31: Frame Rate Result of Single Link Sub System

The frame rate result of verification is showed as Figure 31. In this verification process, 10 times of frame rate had been recorded, and each time include 1 minute's average frame rate. In Figure 31, the average frame rate is 59.1 fps.

$$Total = f \times t \qquad \text{(Equation 2)}$$

Then, with the (Equation 2) we could get the total frame which had been record in this verification process. The total frames are $35460 = 59.1(\text{fps}) \times 10(\text{times}) \times 60(\text{s})$. Amount these 35460 frames, the average frame rate is 59.1 fps which is bigger than 45 fps, and it means that the result fit the system requirement of smooth fresh rate. The standard deviation is around 0.83 and because $0.83 \ll 59.1$ which showed the frame rate is stable while verification process.

The result of questionnaire shows as Figure 32, Figure 33 and Figure 34. The average values are shown as bar, and the number of value is shown inside with white label. For Figure 32 and Figure 34, the standard deviations of results are also shown. Figure 32

shows the result of Q1, Figure 33 shows the result of Q2, Figure 34 shows the result of Q3 and Q4.

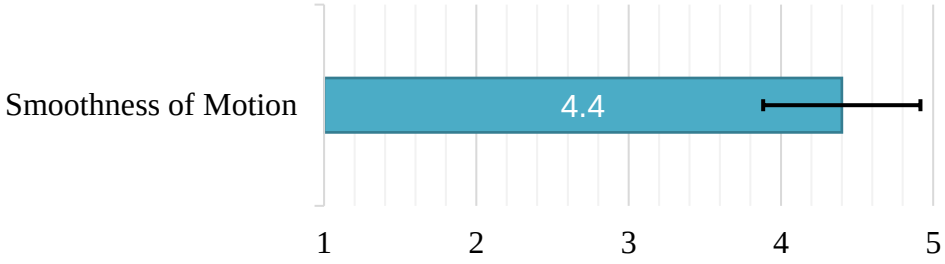


Figure 32: The average score of smoothness of motion

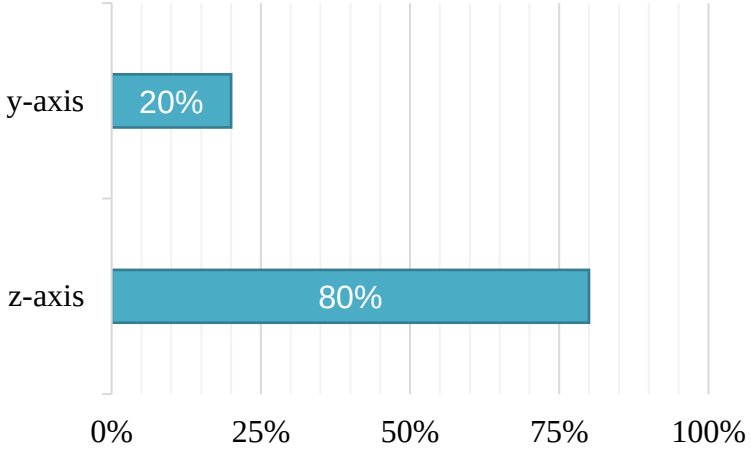


Figure 33: The percentage of most prefer rotation axis in SPCS

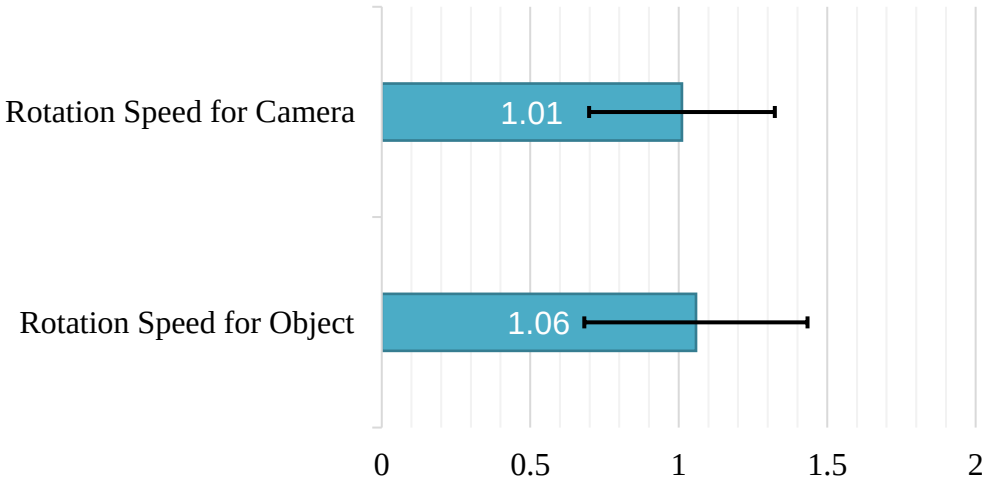


Figure 34: The average prefer rotation speed of camera and object

From the results, subjects got high smoothness feeling of rotating 3D object. The average score of smoothness of motion is 4.4, and the standard deviation is around 0.51. None of the subjects give less than 4 score of this subsystem.

From Figure 33, most of subjects preferred to use z-axis in SPCS. They said it is easier to understand how to rotate than using y-axis.

From Figure 34, the average rotation speed rate for camera is 1.01, and rate for object is 1.06. The standard deviation of rotation speed rate for camera is around 0.31, and as for object, it is around 0.38.

All the validation result showed better system requirement fitting.

From Figure 32, the goal of smooth controlling 3D map had been proved. Most of subjects got smooth motion experience according to the low standard deviation.

From Figure 33, many of subjects preferred rotating around z-axis to control 3D map. At the comment, many of them described that smart phone is like a big plane, when plane-on the desk, it was more like the same situation as 3D map. They said the 3D map inside UHD client was like a plane structure. For the subjects who liked to rotate around y-axis, they agreed that z-axis was more intuitive, but they needed to use 2 hand to hold the smart phone. The feeling was not convenient for them.

For the result and comment, although the majority would like to use z-axis, we will remain that y-axis rotation option to user.

From Figure 34, both average rate is around 1.00. But both the standard deviation is around 30% of standard speed. The prefer rotation speed rate is various from person to person. From the comment of Q5, some subjects who prefer higher rotation rate spoke about experience of playing some high reflect game like FPS game.

From the comment in Q5, most of subjects described a better feeling of understanding the station structure. The feasibility of interaction is shown as エラー! 参照元が見つかり

ません。 and had been proved. Some subjects said that the rotation around x-axis which for controlling camera was not that necessary. Rotation of 3D map in order to see from different angle was more important. Some comments also showed that the zooming function provided by touch screen is not that interesting for them.

As a result, many of them got feeling of holding a real map model on their hand. They felt understanding the station's structure more intuitive, especially for floor structure.

4.3 Multi-to-Multi Approach Verification & Validation

As an application system, not only the structure and performance, it is essential to test the feasibility of proposed system. In this Section, an experiment was built to verify and validate our system.

Two televisions were setup in a single room. One was digital signage subsystem, and another was displaying a static image as a post sign. In this experiment, we developed a subway station model for Shibuya station, showed as Figure 35. The Shibuya station was famous in its shopping mall center, also in its complex construction structure. As a comparison, the static image was set as Shibuya station static map, which provided by Toyoko Subway official website. The Smartphone interface was the same as previous validation and verification process [18].

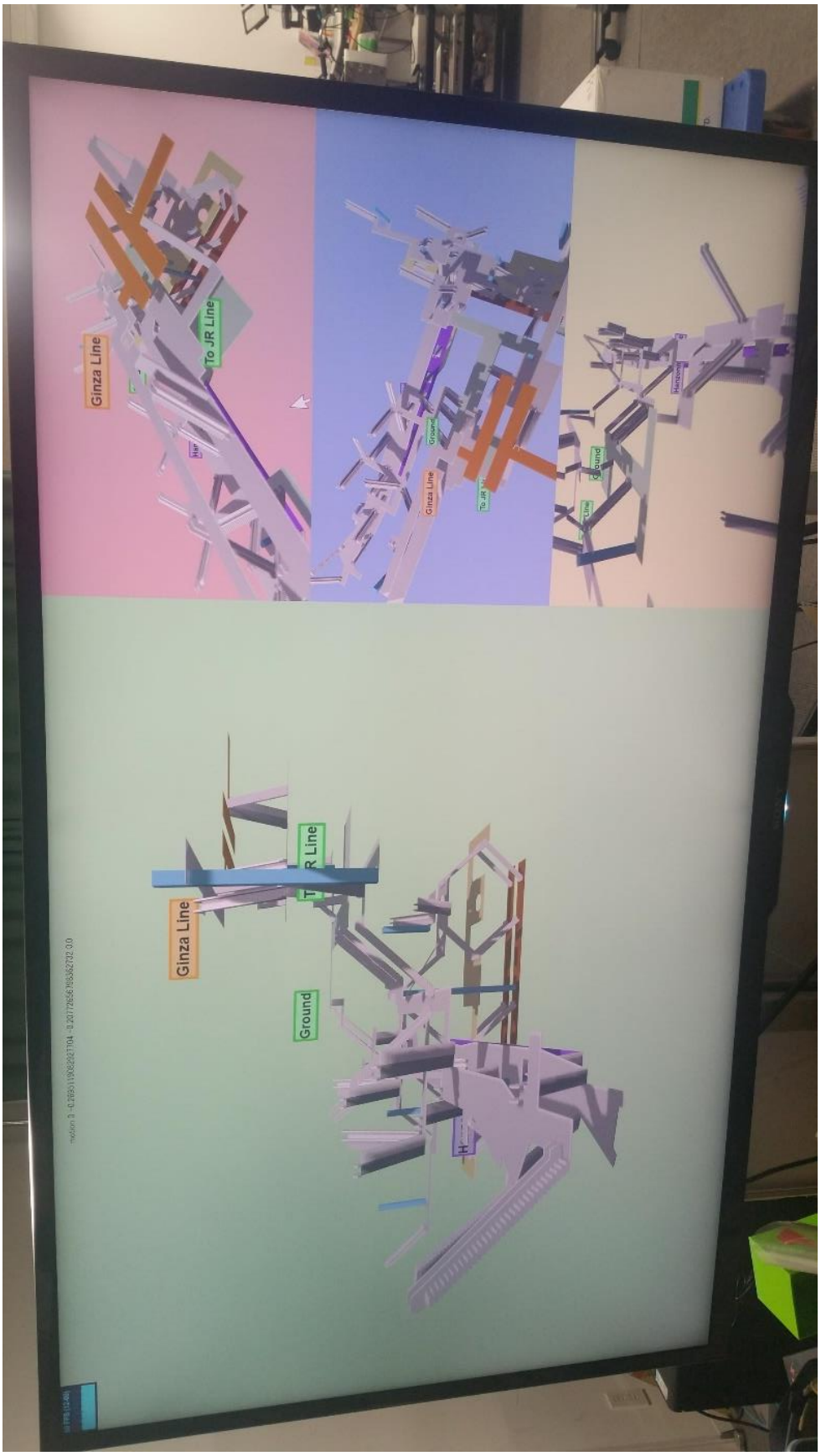


Figure 35: The real system of digital signage subsystem

In verification, we used the same method as Single Link Sub System to verify the system. The purpose of the verification process is to proof that no error/defect/fault during system implementation [19]. Multi-to-Multi approach was combination of multiple Single Link Sub System. In this system, as mentioned in Section 2.4 (System Architecture) and Section 3.7 (Multi-Client Handling Method), there were several factors which influenced the final frame rate of digital signage:

1. Firstly, the system should be run without error. This could be verified during implement of the system. If there is no error happened and the whole system is not shut down by any exception, then it proves that verification result fits the system requirement.

2. Secondly, the 3-Dimensional station model should be displayed properly. As the system design we mention in Section 2.3 (System Overview), there are 3 sub screens and 1 main screen. The station model should be displayed in each screen properly.

3. Thirdly, the message should be transmitted to target digital signage on time. In this research, in order to save dataflow for end user, each motion data will be transmitted in 7 ms (which means the maximum system speed for digital signage could be $142\ (fps) \approx 1000(ms) \div 7(ms)$). The central computer used multi-thread to handle different messages, so the limit of process speed is CPU's speed. After transmitting data to end digital signage sub system, the refresh rate would be decided by rendering speed.

All of these 3 factors finally contributes to frame rate of digital signage subsystem. Hence, by verifying the final frame rate of digital signage while monitoring the system real time log (real time error report) and 3D model's situation (model rendering in display), we

could verify whether the system fit the system requirement or not. As showed in Figure 35, the frame rate will be on left top of screen and by fully renewed in every 73 to 76 second. In this research, in order to simulate the real implementation environment, we did the verification and validation at the same time. While test subjects doing validation, the stats.js would also record the frame rate. The frame rate was recorded for every experiment subjects during their test period.

In validation, People were allowed to use our system freely. Both digital signage and static map were under evaluation. The questionnaire were asked in both system by follow (From 1 to 5, the understanding level becomes higher. 1 represents Not Understand, 5 represents fully Understand):

	UHD Navigation Signage System				
Understanding of floors	1	2	3	4	5
Understanding of Subway Line	1	2	3	4	5
Understanding of Stairs & Elevators:	1	2	3	4	5
Understanding of Way to destination:	1	2	3	4	5

	Static Map				
Understanding of floors	1	2	3	4	5
Understanding of Subway Line	1	2	3	4	5
Understanding of Stairs & Elevators:	1	2	3	4	5
Understanding of Way to destination:	1	2	3	4	5

In order to evaluate the feasibility of system, we also designed an extra questionnaire for our UHD Navigation Signage System as follow (From 1 to 5, 1 represents not satisfy with

the item, 5 represents most satisfy with the item):

	UHD Navigation Signage System				
Smoothness of control	1	2	3	4	5
System access speed	1	2	3	4	5

We also added a space for people to write some comment for us.

In the validation, 15 university students were invited as the experiment subjects. The subjects were asked to use the smartphone control Shibuya station 3D map inside digital signage subsystem, compare it with post map and then try to give satisfaction level to the questionnaire.

For the question of understanding of floors, subjects were asked whether they got a briefly understanding of how many floors and relationship between each floor or not. For the question of understanding of subway line, subjects were asked whether they understood how many subway lines inside station or not. For the question of understanding of stairs & elevators, subjects were asked whether they got a briefly understand of position of stairs & elevators or not. For the question of understanding of way to destination, subjects were asked that assuming they were in some place of Shibuya station and they were supposed to go somewhere, and then whether they could quickly find the way or not. For the question of smoothness, subjects were asked to give the score from 1 to 5 in order to evaluate the satisfaction of smoothness of control. The Access speed is that when subjects decided to control 3D model whether the system response immediately or not.

In this research, as Navigation Digital Signage System is a Multi-to-Multi system, we also asked subjects to fit out the questionnaire for validating the Multi-to-Multi function. As Questionnaire 4 in Section 8 (Appendix – A: Questionnaire), we asked the following

question for subjects:

(Q1). Access Speed	1	2	3	4	5
(Q2). Smoothness of Control	1	2	3	4	5
(Q3). Disturb	1	2	3	4	5
(Q4). Satisfaction of Multi-to-Multi System.	1	2	3	4	5

In this multi-to-multi validation process, 3 subjects would be in a group using the Digital Signage at the same time. While using Digital Signage with Smartphone, they were asked to answer from Q1 to Q4, and all the questions were evaluated in level from 1 to 5. The Access Speed represents whether there is a big delay between clicking connection button and got control. 1 means there is a big delay, 5 means the access speed is fast. The Smoothness of Control represents the smoothness of controlling the 3D Station Model inside Digital Signage. If the answer is 1, means there is a big delay after rotating the Smartphone, and only after that certain delay, the 3D Station Model inside the screen rotates. The Disturb is used to measure multiple user feasibility. While 3 subjects are using this system at the same time, if the subject thinks that he or she is disturbed by other user, or he or she cannot understand which screen is used, then he or she give 1 score to Q3. If he or she thinks fully understand which screen he or she is controlling, then he or she gives 5 score to Q3. The Q4 in terms of satisfaction of Multi-to-Multi system, he or she will give an overall evaluation score to this system.

4.4 Result and Discuss of Multi-to-Multi Approach Evaluation

The frame rate result of verification is showed as Figure 36. In this verification process, 15 times of frame rate had been recorded, and each time include 1 minute's average frame rate. In Figure 36, the average frame rate is 58.2 fps.

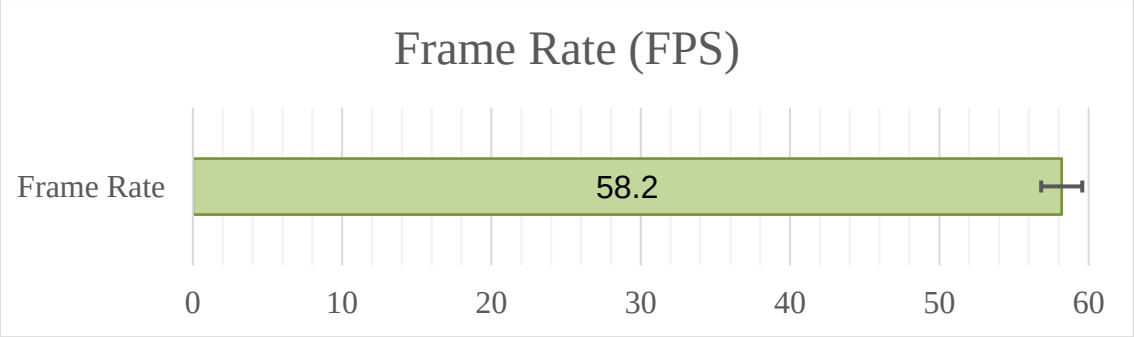


Figure 36: The Frame Rate Result of Digital Signage.

Then, with the (Equation 2) which mentioned in Section 4.2, we could get the total frame which had been record in this verification process. The total frames are $523800 = 58.2(\text{fps}) \times 15(\text{times}) \times 60(\text{s})$. Amount these 523800 frames, the average frame rate is 58.2 fps which is bigger than 45 fps, and it means that the result fit the system requirement of smooth fresh rate. The standard deviation is around 1.38 and because $1.38 \ll 59.1$ which showed the frame rate is stable while verification process.

In the validation process, 15 students participated as subjects doing the evaluation. The result of validation questionnaire is showed as Figure 37, and Figure 38. The average values are shown as bar, and the number of value is shown inside with black label. The standard deviations of results are also shown.

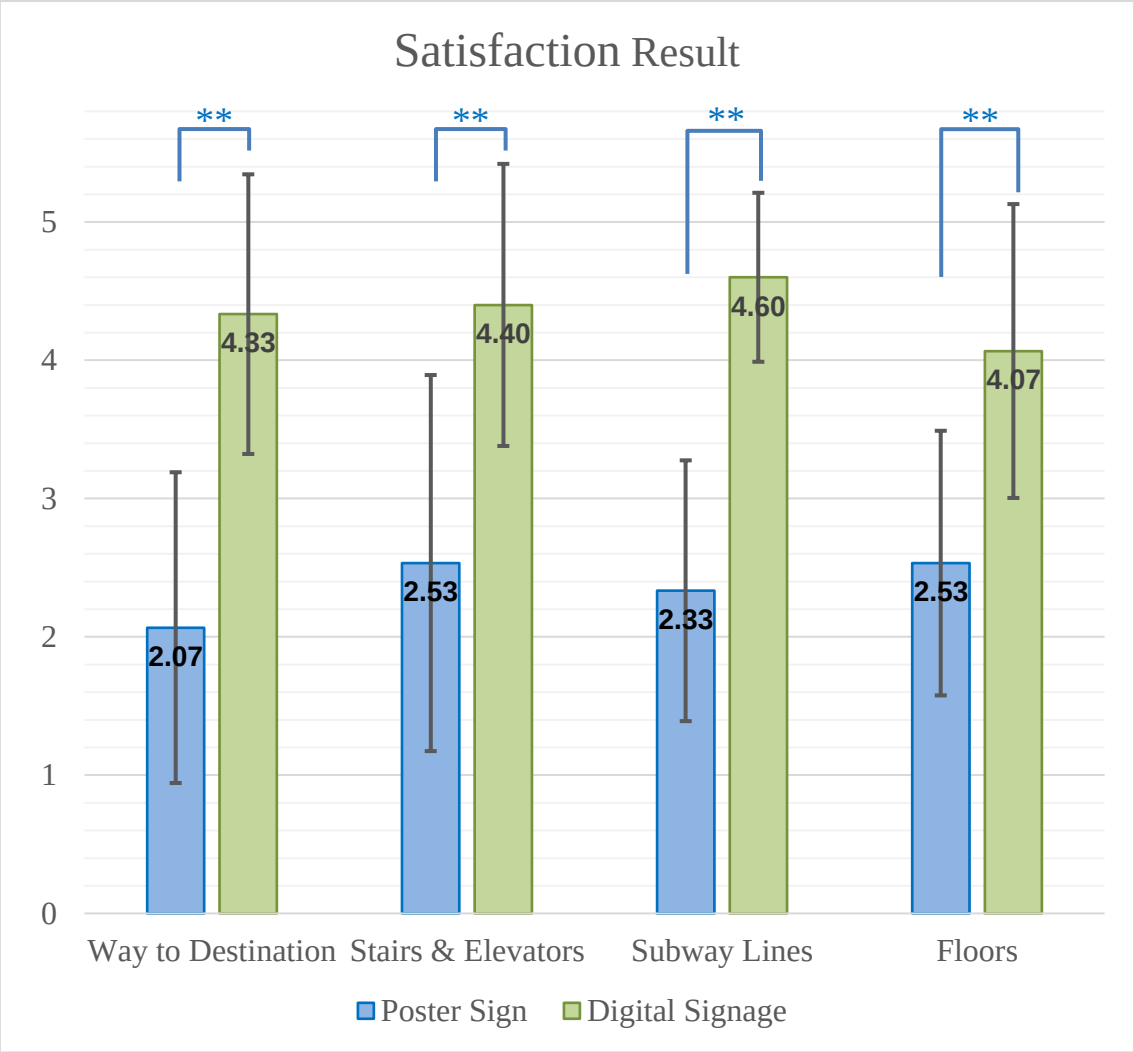


Figure 37: The Satisfaction Result of Post Sign and Digital Signage.

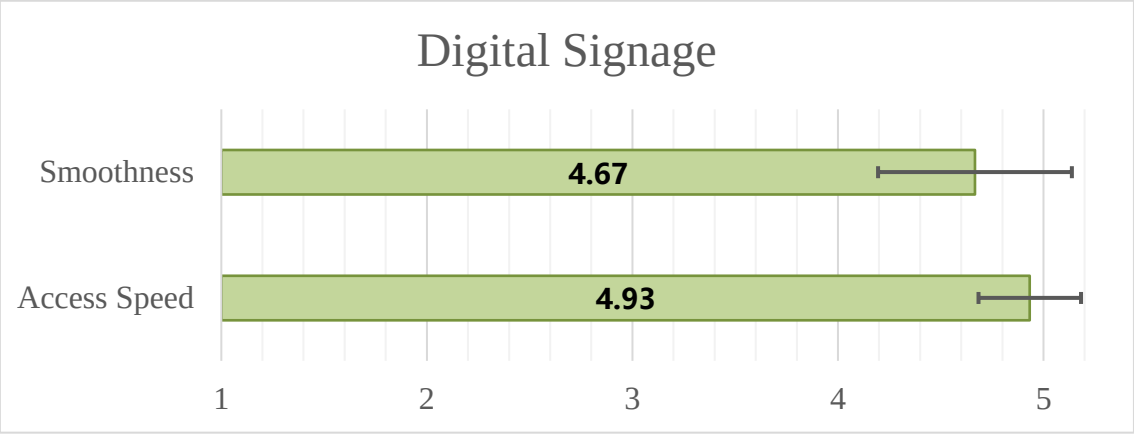


Figure 38: The Smoothness and Access Speed Result of Digital Signage.

As a result, UHD Navigation Signage System shows better performance on understanding the subway station structure and wayfinding. The satisfaction of understanding of floors of digital signage is 4.07, which is 1.54 higher than traditional post sign. The satisfaction of understanding of subway lines of digital signage is 4.6, which is 2.27 higher than traditional post sign. The satisfaction of understanding of stairs & elevators of digital signage is 4.4, which is 1.87 higher than traditional post sign. The satisfaction of understanding of way to destination of digital signage is 4.33, which is 2.26 higher than traditional post sign.

In order to test the difference is significant or not, we also applied T-Test method in both data set. Since one subject was asked to give evaluation level on same question for Digital Signage and Post Sign, we used Microsoft Excel as a tool to do our T-Test. While using Data Analysis function, we chosed “t-Test: Paired Two Sample for Means” as our tool. There are 4 groups of data, Floors (represents satisfaction of understanding the floors), Subway Lines (represents satisfaction of understanding the subway lines), Stairs & Elevators (represents satisfaction of understanding of stairs & elevators), and Way to Destination (represents satisfaction of understanding of way to destination). Each group had two sample, and we assumed each group that the hypothesized mean difference is 0. The result of P value of each group was showed as Figure 37 and Table 3. In Figure 37, if the P value is less than 0.05 and higher than 0.01, then upon the data bar it would be marked by *. If the P value is less than 0.01, then upon the data bar it would be marked **. The less P value it is, in statistic, the more significant difference of two value it will be. The P value of each group was showed in Table 3 at below:

	Way to Destination	Stairs & Elevators	Subway Lines	Floors
P value	0.0001861047	0.0014831045	0.0000014053	0.0006282670

Table 3: T-Test of Paired Two Simple for Means.

For the smoothness of control and system access speed, showed as Figure 38, subjects gave average 4.67 (Standard Deviation is 0.47) and 4.93 (Standard Deviation is 0.25) score on each. The result showed that the rotation smoothness is in good condition and the system access speed was quick.

Some comments suggested that the system could provide some icon for each subway lines. Subjects also suggested that adding some function for customers to make some stair transparent would be better to understanding the structure.

In Multi-to-Multi function validation, the result is showed as Figure 39. For Q1 the access speed, the average score is 4.78, the standard deviation is 0.42. For Q2, the Smoothness, the average score is 4.22, the standard deviation is 0.41. For Q3, Not Disturb, the average score is 4.22, the deviation is 0.92. For Q4, the satisfaction of whole system, the average score is 4.33, and the standard deviation is 0.47.

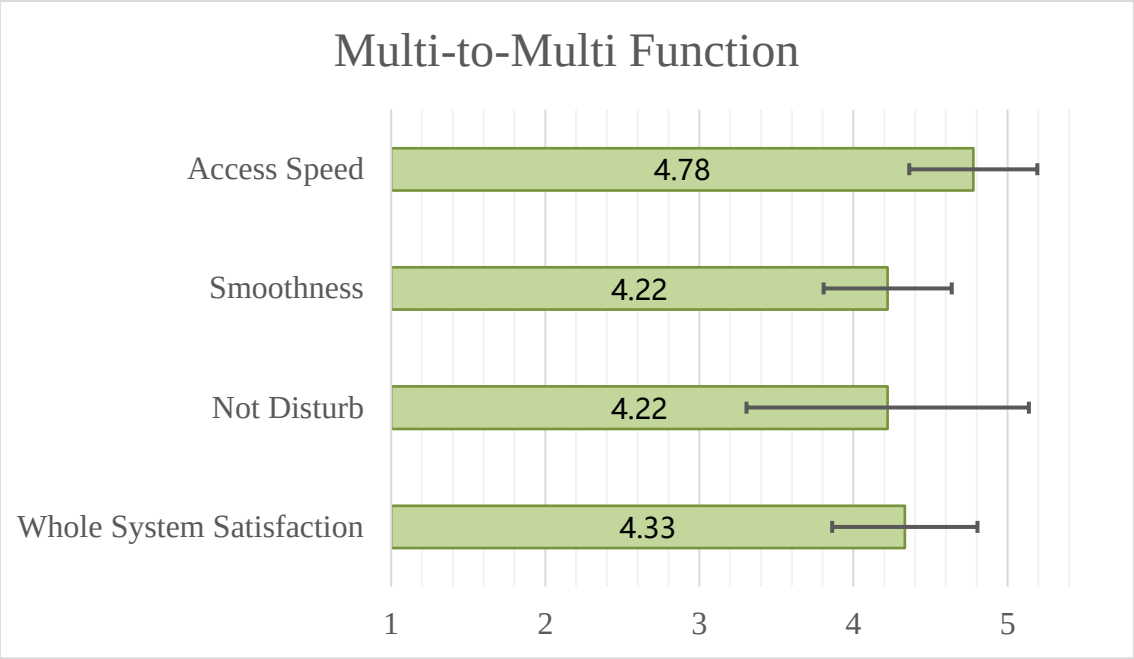


Figure 39: The Multi-to-Multi function validation result of Digital Signage.

The result shows that subjects were satisfied with the Multi-to-Multi system, most of subjects got high system access speed since multiple user uses it at the same time. And the motion data transmitting is fast since smoothness achieve high average score. Most of subjects can cleanly distinguish which sub screen they were controlled. The whole system got satisfaction score of average 4.33 that validate this system fit the system requirement.

5 Conclusion and Future Work

In this research, we proposed a new navigation system based on the interaction of ultra-high definition display. We used Blender to develop a simple 3D model of Hiyoshi station and built a single link subsystem to test the feasibility of controlling. The verification and validation of this subsystem had been done. We also used Blender to build a more complex model of Shibuya Station to test the feasibility of Multi-to-Multi approach of this system. The verification and validation of this system also had been done.

As a result, most of users got a better experience when tried to understand the station structure. The rotation function gave the feeling of holding a 3D map. And the different angle viewing allowed user to understand the floor structure more easily. The big UHD screen can provided the whole picture of construction structure. This system was expected to be a solution of wayfinding problem on future complex interchange subway station.

For future work, the station model should be refined and be redesigned considering more about cartology. E.g. the more reality texture need to be added into the model to provide more precise information. The color and texture should be added properly in order to reduce the gap between real world and virtual map. These color and texture usually come from subway station. We plan to cooperate with Subway Company to test this navigation system.

For real world application, each map's viewpoint should depend on the position of UHD display. E.g. the camera should shift near the stair when this 3D map is shown at a UHD display which near the stair. By this configuration, user's mind can shift from real world to virtual map quickly and understand the virtual path more efficiently.

6 Acknowledgements

First and foremost, I would like to express my deeply thanks to my supervisor, Prof. Ogi Tetsuro, for his kindly and inspirational guidance during my two years' master research life. Without his patient and deep knowledge, this research cannot be achieved to its potential level.

I would like to thank Associate Professor KOHTAKE, Naohiko from SDM Keio and Associate Professor Sisi Zlatanova from TU Delft, for their patient guidance and information on V&V, location system and navigation system.

Further thanks are due to all the lab member of Media System Lab. These two year, we shared our happiness during the discussion, wherever it is at the lab meeting or Gassyuku in Izu Oshima. Special thanks give to my best friend, Bansod Prashant, for his open mind discussion and patient guidance with my English.

I would like to give my thanks to my parents, who strongly support me with the whole oversea study. I would also like to give my thanks to my special, Pin Wen, who deeply and warmly with me.

7 Bibliography

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista and M. Zorzi, "Internet of Things for Smart Cities," *Internet of Things Journal, IEEE,* vol. 1, no. 1, pp. 22-32, Feb 2014.

[2] S. Kim, E. Park, S. Hong, Y. Cho and A. del Pobil, "Designing digital signage for better wayfinding performance: New visitors' navigating campus of university," in *Interaction Sciences (ICIS), 2011 4th International Conference on*, 2011.

[3] A. Corrales Paredes, M. Malfaz and M. Salichs, "Signage System for the Navigation of Autonomous Robots in Indoor Environments," *Industrial Informatics, IEEE Transactions on,* vol. 10, no. 1, pp. 680-688, Feb 2014.

[4] J. Lee and K. Yoon, "The application of digital signage system using smart device," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, 2014.

[5] J. She, J. Crowcroft, H. Fu and P.-H. Ho, "Smart Signage: An Interactive Signage System with Multiple Displays," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, 2013.

[6] H. Sato, M. Urata, K. Yoguchi, N. Arakawa, N. Kanamaru and N. Uchida, "Linking digital signage with mobile phones," in *Intelligence in Next Generation Networks (ICIN), 2011 15th International Conference on*, 2011.

[7] K. H. K. Choi, T. H. S. Chu and H. C. B. Chan, "Dynamic and interactive intelligent signage system," in *2012 IEEE International Conference on Consumer Electronics (ICCE)*, 2012.

[8] H.-J. Suh, "A Wireless Triggered Method of Digital Signage Based on Periodic Scans of a Smartphone," *International Journal of Software Engineering and Its Applications*, vol. 9, no. 9, pp. 197-204, 2015.

[9] J. S. Lee, S. W. Moon, J. W. Lee and K. S. Yoon, "A Study on Digital Signage Interaction Using Mobile Device," *International Journal of Information and Electronics Engineering*, vol. 5, no. 5, p. 394, 2015.

[10] J. Lee and K. Yoon, "The application of digital signage system using smart device," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, 2014.

[11] J. She, J. Crowcroft, H. Fu and P.-H. Ho, "Smart Signage: A Draggable Cyber-Physical Broadcast/Multicast Media System," in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, 2012.

[12] J. She, J. Crowcroft, H. Fu and P.-H. Ho, "Smart Signage: An Interactive Signage System with Multiple Displays," in *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, 2013.

[13] R. Passini, "Wayfinding: A conceptual framework," *Urban Ecology*, vol. 5, no. 1, pp. 17-31, 1981.

[14] J. S. Lee, J. W. Lee, H. Jung, S. Moon and K. Yoon, "The implementation of 4K digital signage system," in *16th International Conference on Advanced Communication Technology*, 2014.

[15] H. T. Jung, J. S. Lee, Y. J. Jeong and K. S. Yoon, "Digital signage system for supporting high quality resolution," in *Computing and Convergence Technology (ICCCT), 2012 7th International Conference on*, 2012.

[16] M. Schubin, "WHY 4K: VISION & TELEVISION," in *Spring Technical Forum of CableLabs-NCTA-SCTE*, 2012.

[17] K. Masaoka, M. Emoto, M. Sugawara and F. Okano, "Presence and preferable viewing conditions when using an ultrahigh-definition large-screen display," in *Electronic Imaging 2005*, 2005.

[18] Z. Liang and T. Ogi, "Navigation System Based on Interaction of Ultra HighDefinition Display," in *ASIAGRAPH 2016 Forum in Toyama proceedinghs*, Toyama, Japan, 2016.

[19] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin and T. M. Shortell, Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, John Wiley & Sons, Inc, 2015.

8 Appendix – A: Questionnaire

Questionnaires 1

1. Do you think the big stations like Yokohama and Shibuya make you confuse about finding the route?

   ☐ Yes. ☐ No.

2. Do you confuse about where the exits it is? _____

   ☐ Yes. ☐ No.

3. Do you confused about changing from line to line? _____

   ☐ Yes. ☐ No.

4. Is there any map about station structure in front of you when you were out of subway and at a platform?

   ☐ Yes, Always there. ☐ Yes, sometimes. ☐ No.

   4.1 Is the station map clear and helpful for you to find the route?

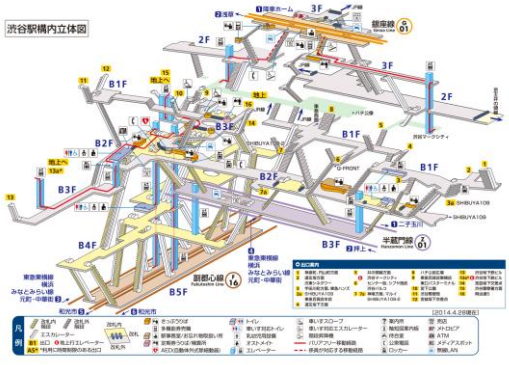
   ☐ Yes, Always help. ☐ Yes, sometimes. ☐ No.

5. Is the indicator which inside station helpful for you?

   ☐ Always ☐ No. ☐ other: _____

6. What kind of map do you think it is helpful for you?

☐



☐

☐

7. Nationality: _____

Questionnaire 2

1. The Smoothness of Motion: Slack **1** **2** **3** **4** **5** Smoothest

2. Rotation Speed: _____

3. Prefer Axis to rotate object: _____

4. Prefer Axis to rotate camera: _____

5. Comment for this navigation system:

_____

_____

_____

_____

_____

_____

Signature: _____

Questionnaires 3

1. While using post map and UHD navigation signage system, please give the score for the following item. From 1 to 5, the understanding level becomes higher. 1 represents Not Understand, 5 represents fully understand.

	UHD Navigation Signage System					Static Map				
Understanding of floors	1	2	3	4	5	1	2	3	4	5
Understanding of Subway Line	1	2	3	4	5	1	2	3	4	5
Understanding of Stairs & Elevators:	1	2	3	4	5	1	2	3	4	5
Understanding of Way to destination:	1	2	3	4	5	1	2	3	4	5

2. After using the system, please give the score for the following item.

	UHD Navigation Signage System				
Smoothness of control	1	2	3	4	5
System access speed	1	2	3	4	5

3. Please give some comments, if any.

Signature: _____

Questionnaires 4

While using UHD navigation signage system with other users at the same time, please give the score for the following items. From 1 to 5, the level becomes higher.

1. After press the connect button, is system access speed quick? From 1 to 5, 1 represents the access speed is slow, and 5 represents less delay of accessing system.

   1 2 3 4 5

2. After connecting the system, are you satisfied with the control smoothness?

   1 2 3 4 5

3. While using the system with other users at the same time, do you feel disturb by others? 1 represents strongly disturb by others, and 5 means you don't feel that much difference at all.

   1 2 3 4 5

4. Please give the satisfaction level for multi-to-multi system.

   1 2 3 4 5

5. Please give some comments, if any.

Signature: _____

9 Appendix – B: Programming Code for Central Server

Example Code for ClientWebSocketListen

```
import Processor.ClientProcessor;
import Processor.OnReceiveListener;
import Processor.OutLog;
import Processor.ReceiveEvent;
import ServerSmartPhone.ClientSmartPhoneAbs;
import ServerWebSocket.ClientWebSocketAbs;
import java.io.IOException;
import java.net.Socket;
import java.net.SocketException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ClientWebSocketListen extends ClientWebSocketAbs{
    private String UUID; //The unique iBeacon ID
    private ArrayList<ClientSmartPhoneAbs> smartphonelist; //Maximum 3
smartphone in the list
    private OnReceiveListener orl;
    public OutLog outlog;
    public ArrayList<Object> accessList;
    /**
     *
     * @param client
     */
    public ClientWebSocketListen(Socket client) {
        super(client);
        this.accessList=new ArrayList<>();
        if (this.isWebsocketLink()) {
            /** Create a receive OnReceiveListener for smart phone
             * when smartphone client receive a msg, then it will
             * inform this OnReceiveListener, to use the send method
             * of WebSocket client to send msg to the remote browser
             */
            orl = (ReceiveEvent receiveEvent) -> {
                if (ClientWebSocketListen.this.isWebsocketLink()) {
                    try {
                        send(receiveEvent.getReceiveMsg());
                        return true;
                    } catch (java.net.SocketException ex) {
                        //
                        try {
                            ClientWebSocketListen.this.close();
                            return false;
                        } catch (IOException e) {
                            if(this.outlog!=null){
```

```
                            outlog.print(new   SimpleDateFormat("yyyy-
MM-dd HH:mm:ss.SSS").format(new Date()));
                            outlog.println("---Client        "        +
ClientWebSocketListen.this.getClientInfo()   +   "   =======   "   +
e.toString());
                        }else{
                            System.out.print(new
SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS").format(new Date()));
                            System.out.println("---Client      "      +
ClientWebSocketListen.this.getClientInfo()   +   "   =======   "   +
e.toString());
                        }
                        return false;
                    }
                }catch (IOException ex) {

Logger.getLogger(ClientWebSocketListen.class.getName()).log(Level.SE
VERE, null, ex);
                    return false;
                }
            } else {
                return false;
            }
          };
          //Set up my processor

          this.myProcessor=new ClientProcessor() {
              @Override
              public void initialProcess() {
                  //Check for WebSocket Signage's UUID
                  try {
                      sendDirectly("com:uuid?");
                  }catch (SocketException ex){
                      //Need Exception!
                  }catch (IOException ex) {

Logger.getLogger(ClientWebSocketListen.class.getName()).log(Level.SE
VERE, null, ex);
                  }
                  String msg=receiveDirectly();
                  String[] data = null;
                  if(msg!=null){
                      data=msg.split("=");
                  }
                  if(data!= null && data.length>1){
                      if(data[0].equals("uuid")){
                          ClientWebSocketListen.this.UUID=data[1];

ClientWebSocketListen.this.println("UUID="+ClientWebSocketListen.thi
s.UUID);
                      }else{

ClientWebSocketListen.this.setIsWebSocketLink(false);
                      }
                  }else{
```

```
ClientWebSocketListen.this.setIsWebSocketLink(false);
                }
                //End of checking UUID

            }

            @Override
            public void sendProcess(String data) {
                try {
                    sendDirectly(data);
                } catch (IOException ex) {

Logger.getLogger(ClientWebSocketListen.class.getName()).log(Level.SE
VERE, null, ex);
                }
            }

            @Override
            public String receiveProcess(String message) {
                ClientWebSocketListen.this.println("Process_recv:" +
message);

                try {
                    send(message);
                } catch (IOException ex) {

Logger.getLogger(ClientWebSocketListen.class.getName()).log(Level.SE
VERE, null, ex);
                }
                return message;
            }
        };

    }
}

/**
 *
 * @return if the link is correct, it will return the
OnReceiveListener. If it is not, it will return a null.
 */
public OnReceiveListener getOnReceiveListener(){
    if (this.isWebsocketLink()) {
        return this.orl;
    }else{
        return null;
    }
}

@Override
protected void onReceive(String message) {
    //System.out.println("recv:"+message);
}
```

```
    @Override
    protected void onClose() {
        if (outlog != null) {
            outlog.print(new                SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            outlog.println("---Client  "  +  this.getClientInfo()  +  "
======= onClose");
        } else {
            System.out.print(new            SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            System.out.println("---Client " + this.getClientInfo() + "
======= onClose");
        }
    }

    @Override
    protected void onOpen() {
        if (outlog != null) {
            outlog.print(new                SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            outlog.println("---Client  "  +  this.getClientInfo()  +  "
======= onOpen");
        } else {
            System.out.print(new            SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            System.out.println("---Client " + this.getClientInfo() + "
======= onOpen");
        }
    }

    @Override
    protected void onPong() {
        if (outlog != null) {
            outlog.print(new                SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            outlog.println("---Client  "  +  this.getClientInfo()  +  "
======= onPong");
        } else {
            System.out.print(new            SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            System.out.println("---Client " + this.getClientInfo() + "
======= onPong");
        }
    }

    @Override
    protected void onPing() {
        if (outlog != null) {
            outlog.print(new                SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            outlog.println("---Client  "  +  this.getClientInfo()  +  "
======= onPing");
        } else {
            System.out.print(new            SimpleDateFormat("yyyy-MM-dd
HH:mm:ss.SSS").format(new Date()));
            System.out.println("---Client " + this.getClientInfo() + "
```

```
======= onPing");
        }
        try {
            this.sendPongFrame();
        } catch (IOException ex) {

Logger.getLogger(ClientWebSocketListen.class.getName()).log(Level.SE
VERE, null, ex);
        }
    }

    private void println(String msg){
        if (outlog != null) {
            outlog.println(msg);
        } else {
            System.out.println(msg);
        }
    }

    public String getUUID(){
        return this.UUID;
    }
}
```

Example Code for ClientSmartPhoneListen

```
import Processor.ClientProcessor;
import Processor.OnReceiveListener;
import Processor.ReceiveEvent;
import ServerSmartPhone.ClientSmartPhoneAbs;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Iterator;

public class ClientSmartPhoneListen extends ClientSmartPhoneAbs{
    private final ArrayList<OnReceiveListener> listeners = new
ArrayList<>();
    private String uuid;

    public ClientSmartPhoneListen(Socket client) {
        super(client);
        uuid=null;
        this.myProcessor=new ClientProcessor(){
            @Override
            public void initialProcess() {
                send("com:uuid?");
                String
uuidrecv=ClientSmartPhoneListen.this.receiveDirectly();
                if(uuidrecv!=null){
                    ClientSmartPhoneListen.this.uuid=uuidrecv;
                    System.out.println(uuidrecv);
                }else{

ClientSmartPhoneListen.this.setIsSmartPhoneLink(false);
                }
            }
            @Override public void sendProcess(String data) {
            }
            @Override public String receiveProcess(String message) {
                return null;
            }
        };

    }

    @Override
    protected void onReceive(String message) {
        //System.out.println(message);
        this.notifyDemoEvent(message);
    }

    @Override
    protected void onClose() {
        //throw new UnsupportedOperationException("Not supported
yet."); //To change body of generated methods, choose Tools |
Templates.
    }

    @Override
    protected void onOpen() {
```

```
       //System.out.println("===New         SmartPhone         Coming!
"+this.getClientInfo()+"===");
    }

    public         void         addOnReceiveListener(OnReceiveListener
onReceiveListener){
       this.listeners.add(onReceiveListener);
    }
    private void notifyDemoEvent(String messageCome) {
       /*
       if (!this.listeners.isEmpty()) {
          for (OnReceiveListener eventListener : listeners) {
             ReceiveEvent   demoEvent   =   new   ReceiveEvent(this,
messageCome);
             boolean isAlive=eventListener.processEvent(demoEvent);
          }
       }
       */
       //System.out.println("Size of Listener:"+listeners.size());
       Iterator<OnReceiveListener> iterator = listeners.iterator();
       while (iterator.hasNext()) {
          OnReceiveListener c = iterator.next();
          if (c != null) {
             ReceiveEvent   demoEvent   =   new   ReceiveEvent(this,
messageCome);
             boolean isAlive = c.processEvent(demoEvent);
             if (!isAlive) {
                iterator.remove();
             }
          }
       }


    }
}
```

Package of Processor

Example Code for ClientProcessor

```
package Processor;

public interface ClientProcessor {
    public void initialProcess();
    public void sendProcess(String data);
    public String receiveProcess(String message);
}
```

Example Code for ClientSMCameEvent

```
package Processor;

import ServerSmartPhone.ClientSmartPhoneAbs;
import java.util.EventObject;

public class ClientSMCameEvent  extends EventObject {
    private static final long serialVersionUID = 3L;
    private ClientSmartPhoneAbs client;
    public   ClientSMCameEvent(Object   source,   ClientSmartPhoneAbs
client) {
        super(source);
        this.client=client;
    }
    public ClientSmartPhoneAbs getClientSmartPhone(){
        return this.client;
    }
}
```

Example Code for ClientWBCameEvent

```
package Processor;

import ServerWebSocket.ClientWebSocketAbs;
import java.util.EventObject;

public class ClientWBCameEvent extends EventObject {
    private static final long serialVersionUID = 3L;
    private ClientWebSocketAbs client;
    public ClientWBCameEvent(Object source, ClientWebSocketAbs client)
{
        super(source);
        this.client=client;
    }
    public ClientWebSocketAbs getClientWebSocketAbs(){
        return this.client;
    }

}
```

Example Code for OnReceiveListener

```
package Processor;

import java.util.EventListener;

public interface OnReceiveListener extends EventListener {

    public boolean processEvent(ReceiveEvent receiveEvent);
}
```

Example Code for OnSMClientCameListener

```
package Processor;

import java.util.EventListener;

public interface OnSMClientCameListener extends EventListener {

    public void processEvent(ClientSMCameEvent demoEvent);
}
```

Example Code for OnWebClientCameListener

```
package Processor;

import java.util.EventListener;

public interface OnWebClientCameListener extends EventListener {

    public void processEvent(ClientWBCameEvent demoEvent);
}
```

Example Code for ReceiveEvent

```
package Processor;

import java.util.EventObject;

public class ReceiveEvent extends EventObject {
    private static final long serialVersionUID = 2L;
    private String message;

    /**
     *
     * @param source usually use "this"
     * @param message the message when a Socket link receive
     */
```

```
    public ReceiveEvent(Object source, String message) {
        super(source);
        this.message=message;
    }

    /**
     *
     * @return get the message which the socket receive
     */
    public String getReceiveMsg(){
        return this.message;
    }

}
```

10 Appendix – C: Programming Code for Digital Signage

Example Code for Digital Signage Client (JavaScript using Three.js “WebGL Library”)

```
<!DOCTYPE html>
<html lang="en">

<head>
<meta charset="UTF-8">
<title>Station</title>
<script src="three.min.js"></script>
<script src="stats.js"></script>
<script src="dat.gui.js"></script>
<script type="text/javascript">
var renderer;
var height,width;
var webSocket;
var wsUri;
wsUri = "ws://131.113.250.176:4999";
//wsUri = "ws://127.0.0.1:4999";
var uuid="1AE18C1C-6C7B-4AED-B166-4462634DA855";
//uuid="00000000-9D6A-1001-B000-001C4D834503";

var degreeX=0.0;
var degreeY=0.0;
var gypoX=0.0;
var gypoY=0.0;
var gravityZ=0.0;
var cameraDegree=0.0;

var cameraDegreeY1=0.0;
var cameraDegreeY2=0.0;
var cameraDegreeY3=0.0;
var cameraDegreeX1=0.0;
var cameraDegreeX2=0.0;
var cameraDegreeX3=0.0;
var cameraLength1=20;
var cameraLength2=20;
var cameraLength3=20;

var cameraDegreeSelf=0.0;
var cameraLength=25;
var mainOutput;

//define of all 3 subScreen
var  gyroX1=0.0;
var gyroY1=0.0;
```

```
var flip1=0.0;
var  gyroX2=0.0;
var gyroY2=0.0;
var flip2=0.0;
var  gyroX3=0.0;
var gyroY3=0.0;
var flip3=0.0;

var df=1.0;
var color1=new THREE.Color().setRGB(0.7/df,0.9/df,0.7/df);
var color2=new THREE.Color().setRGB(0.9/df,0.7/df,0.7/df);
var color3=new THREE.Color().setRGB(0.7/df,0.8/df,0.9/df);
var color4=new THREE.Color().setRGB(0.9/df,0.9/df,0.7/df);

function initThree() {
    mainOutput = document.getElementById("info");
    renderer=new THREE.WebGLRenderer({antialias:true});
    renderer.setSize(window.innerWidth, window.innerHeight );
    renderer.setClearColor(0x000000, 1.0);

    renderer.shadowMapEnabled = true;
    //renderer.shadowMap.enabled=true;

    //try to increase the quality of shadow
    //renderer.shadowMapType=THREE.PCFSoftShadowMap;
    renderer.setPixelRatio(window.devicePixelRatio);

document.getElementById("canvas3d").appendChild(renderer.domElement)
;
}

//======================================
function initWebSocket(){
    webSocket = new WebSocket(wsUri);
    webSocket.onopen = function(evt) { onOpen(evt); };
    webSocket.onclose = function(evt) { onClose(evt); };
    webSocket.onmessage = function(evt) { onMessage(evt); };
    webSocket.onerror = function(evt) { onError(evt); };

}
function onOpen(evt){
    var  serverData='<span  style="color:  blue;">onOpen:  </span>  '  +
evt.data;
    var pre = document.createElement("p");
    pre.style.wordWrap = "break-word";
    pre.innerHTML = serverData;
    mainOutput.innerHTML="";
    mainOutput.appendChild(pre);

}
function onClose(evt){
    color1=color2=color3=color4=new
THREE.Color().setRGB(1.0,0.6,0.6);
}
function onMessage(evt){
    var serverData=evt.data;
```

```
    var pre = document.createElement("p");
    pre.style.wordWrap = "break-word";
    pre.innerHTML = serverData;
    mainOutput.innerHTML="";
    mainOutput.appendChild(pre);

    var dataset=serverData.split(' ');
    if(dataset.length>1 && dataset[0]!=="motion"){
        degreeX=(parseFloat(dataset[0])-80.0)/90.0*Math.PI*0.7;
        degreeY=(parseFloat(dataset[1]))/90.0*25.0-5.0;
        gypoX=parseFloat(dataset[2]);
        gypoY=parseFloat(dataset[3]);
        gravityZ=parseFloat(dataset[4]);
    }else if(dataset.length>1 && dataset[0]==="motion"){
        var rate=0.5;
        switch(dataset[1]){
            case "1":
                gyroX1=parseFloat(dataset[2]);
                gyroY1=-parseFloat(dataset[3]);
                var tmp=parseFloat(dataset[4])*rate;
                if(tmp!==0.0){
                    flip1 = tmp * 0.01;
                }else{
                    flip1*=0.985;
                }

                break;
            case "2":
                gyroX2=parseFloat(dataset[2]);
                gyroY2=-parseFloat(dataset[3]);
                //flip2=parseFloat(dataset[4])*rate;
                var tmp=parseFloat(dataset[4])*rate;
                if(tmp!==0.0){
                    flip2 = tmp * 0.01;
                }else{
                    flip2*=0.985;
                }
                break;
            case "3":
                gyroX3=parseFloat(dataset[2]);
                gyroY3=-parseFloat(dataset[3]);
                //flip3=parseFloat(dataset[4])*rate;
                var tmp=parseFloat(dataset[4])*rate;
                if(tmp!==0.0){
                    flip3 = tmp * 0.01;
                }else{
                    flip3*=0.985;
                }
                break;
        }
    }else{
        var command=serverData.split(":");
        if(command.length>1){
            if(command[0]==="com" && command[1]==="uuid?"){
                webSocket.send("uuid="+uuid);
                pre = document.createElement("p");
```

```
                pre.style.wordWrap = "break-word";
                pre.innerHTML = "uuid="+uuid;
                mainOutput.innerHTML="";
                mainOutput.appendChild(pre);
            }
        }
    }
}
function onError(evt){
    color1=color2=color3=color4=new
THREE.Color().setRGB(1.0,0.0,0.0);
}
function initStats(){
    var stats = new Stats();
    stats.setMode(0);
    stats.domElement.style.position = 'absolute';
    stats.domElement.style.left = '0px';
    stats.domElement.style.top = '0px';

document.getElementById("canvas3d").appendChild(stats.domElement);
    return stats;
}
var camera;
var camera2;
var camera3;
var camera4;
var stats;
function initCamera() {
    camera = new THREE.PerspectiveCamera( 45, window.innerWidth /
window.innerHeight , 0.1 , 1000 );
    camera.position.x = -3;
    camera.position.y = 4;
    camera.position.z = 3;
    camera.up.x = 0;
    camera.up.y = 0;
    camera.up.z = 1;
    camera.lookAt( {x:0, y:0, z:0 } );

    camera2 = new THREE.PerspectiveCamera( 45, window.innerWidth /
window.innerHeight , 0.1 , 1000 );
    camera3 = new THREE.PerspectiveCamera( 45, window.innerWidth /
window.innerHeight , 0.1 , 1000 );
    camera4 = new THREE.PerspectiveCamera( 45, window.innerWidth /
window.innerHeight , 0.1 , 1000 );
}
//setup scene
var scene;
function initScene() {
    scene = new THREE.Scene();
}

//setup light
function initLight() {
    var spotLight = new THREE.SpotLight(0xffffff);
    spotLight.position.set(40, 45, -40);
    spotLight.castShadow = true;
```

```
    spotLight.shadowMapHeight=2048;
    spotLight.shadowMapWidth=2048;
    scene.add(spotLight);
    var ambientLight = new THREE.AmbientLight("#aaaaaa");
    scene.add(ambientLight);
}
//setup mesh
var monkey;
var spritey;
var p=[-6.5,-5,-20];
var ra=0.5;
function initObject(){
    //Loader JS
    var loader2= new THREE.JSONLoader();
    loader2.load("shibuy7.js",
           function (model,material) {
              var mesh=new THREE.Mesh(model,material[0]);
              mesh.scale.set(0.5,0.5,0.5);
              mesh.position.x += p[0];
              mesh.position.y += p[1];
              mesh.position.z += p[2];
              mesh.rotation.x+=Math.PI;
              monkey=mesh;
           },"texture/"
    );
    loader2.onLoadComplete=function(){
       monkey.castShadow = true;
       monkey.receiveShadow = true;
       scene.add(monkey);
    };
    spritey = makeTextSprite("Fukutoshin Line",
                         {  fontsize: 24,
                            borderColor: {r: 149, g: 81, b: 29, a: 1.0},
                            backgroundColor: {r: 222, g: 143, b: 86, a:
0.8} }
    );
    spritey.position.set(p[0]+(1.5)*ra,
                     p[1]+(0)*ra,
                     p[2]-(-4.5)*ra);
    scene.add( spritey );

    var spritey2 = makeTextSprite("Hanzonmon Line",
                         {  fontsize: 24,
                            borderColor: {r: 138, g: 56, b: 202, a:
1.0},
                            backgroundColor: {r: 170, g: 160, b: 200,
a: 0.8} }
    );
    spritey2.position.set(p[0]+(18)*ra,
                     p[1]+(6)*ra,
                     p[2]+(4)*ra);
    scene.add( spritey2 );
    var spritey3 = makeTextSprite("Ground",
                         {  fontsize: 24,
                            borderColor: {r: 1, g: 201, b: 52, a: 1.0},
                            backgroundColor: {r: 125, g: 255, b: 160,
```

```
a: 0.8} }
    );
    spritey3.position.set(1,
                          2.5,
                          -22);
    scene.add( spritey3 );
    var spritey4 = makeTextSprite("Ginza Line",
                          {  fontsize: 24,
                            borderColor: {r: 238, g: 147, b: 34, a:
1.0},
                            backgroundColor: {r: 244, g: 188, b: 119,
a: 0.8} }
    );
    spritey4.position.set(3,
                          5.5,
                          -24);
    scene.add( spritey4 );

    var spritey5 = makeTextSprite("To JR Line",
                          {  fontsize: 24,
                            borderColor: {r: 1, g: 201, b: 52, a: 1.0},
                            backgroundColor: {r: 125, g: 255, b: 160,
a: 0.8} }
    );
    spritey5.position.set(3,
                          2.5,
                          -26);
    scene.add( spritey5 );
}


function SetViewPortCam(x,y,w,h,render0,camera0,scene0){
    render0.setViewport(x,y,w,h+1);
    render0.setScissor(x,y,w,h+1);
    render0.enableScissorTest(true);
    camera0.aspect=w/h;
    camera0.updateProjectionMatrix();
    render0.render(scene0, camera0);
}
function render1(){
    requestAnimationFrame(render1,null);
    //Socket control
    gypoY=0.05;
    if(gypoY!==0){
        cameraDegree-=gypoY*0.016;
    }else{
        cameraDegree=degreeX;
    }
    //UPDATE ROTATION
    if(gyroX1!==0){cameraDegreeX1+=gyroX1*0.016;}
    if(gyroX2!==0){cameraDegreeX2+=gyroX2*0.016;}
    if(gyroX3!==0){cameraDegreeX3+=gyroX3*0.016;}

    if(gyroY1!==0){cameraDegreeY1-=gyroY1*0.016;}
    if(gyroY2!==0){cameraDegreeY2-=gyroY2*0.016;}
    if(gyroY3!==0){cameraDegreeY3-=gyroY3*0.016;}
```

```
    cameraLength1+=flip1;
    cameraLength2+=flip2;
    cameraLength3+=flip3;
    if(monkey!==null){  //Avoid  (Uncaught  TypeError:  Cannot  read
property 'rotation' of undefined)
       //After  checking  the  object  whether  it  is  loaded  or  not,
starting the rotation
       //monkey.rotation.y -=0.001;
       cameraDegreeSelf+=0.01;
    }
    //Left View Port of Camera1 - Method for changing the camera
    camera.position.z = cameraLength*Math.cos(cameraDegree)-20;
    camera.position.x = cameraLength*Math.sin(cameraDegree);
    camera.up.x = 0;
    camera.up.y = 1;
    camera.up.z = 0;
    camera.lookAt( {x:0, y:0, z:-20 } );//-5
    renderer.setClearColor(color1);

SetViewPortCam(0,0,window.innerWidth/3.0*2,window.innerHeight,render
er,camera,scene);
    var xy10;

    //Right View Port of Camera2
    camera2.position.y = cameraLength1*Math.sin(cameraDegreeX1);
    xy10=cameraLength1*Math.cos(cameraDegreeX1);
    camera2.position.z = xy10*Math.sin(cameraDegreeY1)-20;
    camera2.position.x = xy10*Math.cos(cameraDegreeY1);
    camera2.lookAt( {x:0, y:0, z:-20 } );//-5
    renderer.setClearColor(color2);
    SetViewPortCam(window.innerWidth/3.0*2,  window.innerHeight/3.0*2,
window.innerWidth/3.0,
window.innerHeight/3.0,renderer,camera2,scene);

    camera3.position.y = cameraLength2*Math.sin(cameraDegreeX2);
    xy10=cameraLength2*Math.cos(cameraDegreeX2);
    camera3.position.z = xy10*Math.sin(cameraDegreeY2)-20;
    camera3.position.x = xy10*Math.cos(cameraDegreeY2);
    camera3.lookAt( {x:0, y:0, z:-20 } );//-5
    renderer.setClearColor(color3);
    SetViewPortCam(window.innerWidth/3.0*2,    window.innerHeight/3.0,
window.innerWidth/3.0,
window.innerHeight/3.0,renderer,camera3,scene);

    camera4.position.y = cameraLength3*Math.sin(cameraDegreeX3);
    xy10=cameraLength3*Math.cos(cameraDegreeX3);
    camera4.position.z = xy10*Math.sin(cameraDegreeY3)-20;
    camera4.position.x = xy10*Math.cos(cameraDegreeY3);
    camera4.lookAt( {x:0, y:0, z:-20 } );//-5
    renderer.setClearColor(color4);
    SetViewPortCam(window.innerWidth/3.0*2,  0,  window.innerWidth/3.0,
window.innerHeight/3.0,renderer,camera4,scene);

    renderer.setSize(window.innerWidth,window.innerHeight);//The
essential method for setup 2 sub-screen correctly at the started
```

```
    stats.update();

}
var controls = new function(){
    this.lightx = -7;
    this.lighty = -8;
    this.lightz = -18;
    this.rotateSpeed = 0.01;
    this.lx=-7;
    this.ly=-4;
    this.lz=-18;
    //spotLight.position.set(-40, 60, 30);
};
function threeStart() {
    /*
    var gui = new dat.GUI();
    gui.add(controls,'lightx',0,7);
    gui.add(controls,'lighty',-5,15);
    gui.add(controls,'lightz',-5,15);
    gui.add(controls,'rotateSpeed',0,0.25);
    gui.add(controls,'lx',-20,20);
    gui.add(controls,'ly',-20,20);
    gui.add(controls,'lz',-60,0);
    */
    initThree();
    initCamera();
    initScene();
    initLight();
    initObject();
    initWebSocket();
    stats = initStats();
    render1();
}
function onResize(){
    renderer.setSize(window.innerWidth,window.innerHeight);
}

function roundRect(ctx, x, y, w, h, r){
    ctx.beginPath();
    ctx.moveTo(x+r, y);
    ctx.lineTo(x+w-r, y);
    ctx.quadraticCurveTo(x+w, y, x+w, y+r);
    ctx.lineTo(x+w, y+h-r);
    ctx.quadraticCurveTo(x+w, y+h, x+w-r, y+h);
    ctx.lineTo(x+r, y+h);
    ctx.quadraticCurveTo(x, y+h, x, y+h-r);
    ctx.lineTo(x, y+r);
    ctx.quadraticCurveTo(x, y, x+r, y);
    ctx.closePath();
    ctx.fill();
    ctx.stroke();
}
function makeTextSprite( message, parameters ){
    if ( parameters === undefined ) parameters = {};
    var    fontface    =    parameters.hasOwnProperty("fontface")    ?
parameters["fontface"] : "Arial";
```

```
    var    fontsize    =    parameters.hasOwnProperty("fontsize")    ?
parameters["fontsize"] : 18;
    var                          borderThickness                          =
parameters.hasOwnProperty("borderThickness")                              ?
parameters["borderThickness"] : 4;
    var   borderColor   =   parameters.hasOwnProperty("borderColor")   ?
parameters["borderColor"] : { r:0, g:0, b:0, a:1.0 };
    var                          backgroundColor                          =
parameters.hasOwnProperty("backgroundColor")                              ?
parameters["backgroundColor"] : { r:255, g:255, b:255, a:1.0 };
    var canvas = document.createElement('canvas');
    var context = canvas.getContext('2d');
    context.font = "Bold " + fontsize + "px " + fontface;
    // get size data (height depends only on font size)
    var metrics = context.measureText( message );
    var textWidth = metrics.width;
    // background color
    context.fillStyle      =  "rgba("  +  backgroundColor.r  +  ","  +
backgroundColor.g + ","
          + backgroundColor.b + "," + backgroundColor.a + ")";
    // border color
    context.strokeStyle = "rgba(" + borderColor.r + "," + borderColor.g
+ ","
          + borderColor.b + "," + borderColor.a + ")";
    context.lineWidth = borderThickness;
    roundRect(context, borderThickness/2, borderThickness/2, textWidth
+ borderThickness, fontsize * 1.4 + borderThickness, 0);
    // 1.4 is extra height factor for text below baseline: g,j,p,q.

    // text color
    context.fillStyle = "rgba(0, 0, 0, 1.0)";
    context.fillText(    message,    borderThickness,    fontsize    +
borderThickness);

    // canvas contents will be used for a texture
    var texture = new THREE.Texture(canvas);
    texture.needsUpdate = true;

    var spriteMaterial = new THREE.SpriteMaterial(
          { map: texture, useScreenCoordinates: false } );
    var sprite = new THREE.Sprite( spriteMaterial );
    sprite.scale.set(5,2.5,1.0);
    return sprite;
}
window.addEventListener('resize',onResize,false);


</script>
<style type="text/css">
    body{
        margin: 0;
        overflow: hidden;
    }

    a {
        color: #0080ff;
```

```
    }
    #info {
        position: absolute;
        top: 0;
        left: 25%;
        padding: 5px;
        font-size:12px;
    }
</style>
</head>

<body onload='threeStart();'>
<!--Start to run the threeStart()
 IE has some error with camera.aspect method.
 code-->
<div id="canvas3d"></div>
<div id="info"> multiple views - webgl</div>
</body>

</html>
```