Navigation Instruction Validation Tool and Indoor Wayfinding Training System for People with Disabilities

NAVIGATION INSTRUCTION VALIDATION TOOL AND INDOOR WAYFINDING TRAINING SYSTEM FOR PEOPLE WITH DISABILITIES

A Thesis Presented

by

LINLIN DING

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2017

Department of Electrical and Computer Engineering

NAVIGATION INSTRUCTION VALIDATION TOOL AND INDOOR WAYFINDING TRAINING SYSTEM FOR PEOPLE WITH DISABILITIES

A Thesis Presented

by

LINLIN DING

Approved as to style and content by:

_____

Aura Ganz, Chair

_____

C. Mani Krishna, Member

_____

Russell Tessier, Member

_____

Christopher V. Hollot, Department Head

Department of Electrical and Computer Engineering

ABSTRACT

NAVIGATION INSTRUCTION VALIDATION TOOL AND INDOOR WAYFINDING TRAINING SYSTEM FOR PEOPLE WITH DISABILITIES

SEPTEMBER 2017

LINLIN DING

B.S.E.C.E., XI'AN UNIVERSITY OF ARCHITECTURE AND TECHNOLOGY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Aura Ganz

According to World Health Survey, there are 785 million (15.6%) people in the world that live with a disability. It is a well-known fact that lack of access to public transportation is a barrier for people with disabilities in seeking work or accessing health care. In this research, we seek to increase access to public transportation by introducing a virtual pre-travel training system that enables people with disabilities to get familiar with a public transportation venue prior to arriving to the venue. Using this system, users establish a mental map of the target environment prior to their arrival to the physical space, increasing their confidence and therefore increasing their chances of using public transportation.

First, we have to guarantee that all navigation instructions sent to our training system are correct. Since the number of navigation instruction increases dramatically, instruction validation becomes a challenge. We propose a videogame based validation tool which includes a game scene that represents in 2D the physical environment and uses a game avatar to verify the navigation instructions automatically in the game scene. The avatar traverses the virtual space following the

corresponding navigation instructions. Only in case that it successfully reaches the planned destination, the current navigation instruction can be considered as correct.

Then, we introduce a virtual reality based pre-travel wayfinding training system to assist people with disabilities get familiar with a venue prior to their arrival at the physical space, which provides two modes: 1) Self-Guided mode in which the path between a source and a destination is shown to the user from third person perspective, and 2) Exploration mode in which the user explores and interacts with the environment.

In the end, we have implemented visual analytics tools that track and evaluate trainees' performance and help us optimize the game. These tools identify the difficulties faced by the trainees as well as obtain overall statistics on the trainees' behavior in the indoor environment, helping us understand how to modify the system and adjust it to different classes of disabilities.

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Independent navigation in public transportation is frustrating and challenging for people with disabilities [1]. These individuals can get easily overwhelmed by the complexity of large, noisy and crowded transportation hubs and public buildings, resulting in their inability to access them independently.

This research is motived by the need for an effective tool for improving the accessibility of public spaces for people with disabilities. We have developed navigation training tools that will increase the trainees' confidence as well as their skills, making them more likely to navigate through indoor spaces independently.

Computerized simulations and games are a convenient approach to provide trainees with effective and efficient simulated instructional activities [2]. In particular, serious games, i.e., games with non-entertainment goals, bring an opportunity for people to practice and explore an environment in an adaptive, personalized and immersive way for the purpose of education, learning, experimentation or development of certain skills [3].

In this paper, we introduce Unity game engine [4] based virtual reality system that will enable people with disabilities get familiar with the space before arriving to the physical space. We generate a virtual environment that mimics the physical environment. In our system, the trainer will determine the narrative of each landmark in the virtual environment according to the trainee disability and assign navigation tasks to the trainees. Using this virtual environment, the trainee can explore and get familiar with it. In addition to the training game, it is necessary for us to

guarantee all instruction which is going to be sent to our training game are correct. In addition, as the number of navigation instruction is increasing dramatically, it also necessary to develop a tool to evaluate navigation instruction prior to being sent to the training game. In this case, a navigation instruction validation tool is proposed in this paper as well. Finally, we also introduce data analytics and visualization tools that enable the trainer to monitor the trainees' progress, flag areas in which the trainee faces difficulties and modify the assigned tasks according and/or provide assistance in these particular areas. Moreover, the system developer which receives all the training data, can identify areas in the environment that lead to difficulties and can optimize the system accordingly in terms of virtual environment representation, user interface and/or navigation instructions.

The proposed system has the following features:

- The navigation instructions are very detailed – we leverage PERCEPT system navigation instructions generation algorithm we developed and introduced in [5-10]. This algorithm which was designed to assist BVI users in their navigation tasks in the real venue incorporates vocabulary and language rules following orientation and mobility principles [10].
- All the instruction generate by PERCEPT system can be evaluated automatically by the navigation instruction validation tool.
- All the routes can be designed by the trainer to target different disabilities
- The system includes multimodal interaction in the virtual environment such as the sound of fare gates, moving trains etc.
- The virtual environment is simplified representation of the physical venue to reduce the mental overload, fact that is helpful for trainees with a cognitive disability.
- The system includes data analytics tools to determine the trainee progress as well as enable the system developer to adjust the system to different disabilities following experimental data.

CHAPTER 2

LITERATURE SURVEY

In this chapter, we present some literary surveys on existing training system. Then compare these systems with the proposed design.

The idea of Audio-Based Environments Simulator (AbES) [11] is to help blind people move about independently and develop their orientation and mobility skills. The blind user can navigate themselves in the virtual environment by interacting with the audio-based interface, which can help them build a mental representation of the real environment. The AbES includes two modes of interaction: Free Navigation, Path Navigation. The Free Navigation mode allows the blind user to freely explore the virtual space. In path navigation mode, it navigates the user with a task of finding a specific destination through a chosen path. From the experimental report, the system indeed helps the blind develop orientation and mobility skills. But the system only supports 2D presentation of real environment, so a lot of spatial information is ignored.

AudioNav [12] is another indoor navigation tool based on mixed reality. It locates and tracks the user inside the building while there is no GPS reception, finds the most suitable path based on the user's special needs, and provides step-by-step direction to the destination using voice, speech, or haptic feedback. The tool extracts the floorplan and landmarks from the 3D model of a building such as doors and windows. The unique characteristics of this system are that it augments reality, uses the user as a sensor. AudioNav provides information and direction to the user using augmented reality. The proposed system in the paper not only aims to the blind people, it can also be used by the users with cognitive disabilities and sighted people. But the biggest issue in this

system is that it cannot accurately locate user in AR environment, thus, affect the entire experience of user.

Previous researches focus on developing mobility and orientation skills without a device for the people with disabilities, but how are they affected when different assistive tools are used? This paper [13] uses a series of virtual environments and non-visual interfaces to comparatively explore the differences in intuitive navigation: when using the virtual Eye-Cane, when using a virtual White-Cane, when navigating without using a device at all and finally when navigating visually. And they show that using the virtual Eye-Cane increases the user's accessibility in virtual environment. Finally, they demonstrate that navigation with the virtual Eye-Cane takes on patterns relatively similar to those of navigating visually. But the virtual environments in system didn't mirror the real space and it is only designed for visually impaired disabilities. So, the people with other disabilities cannot benefit from it.

Compared with the above existing training systems, the proposed training application reflects target building's blueprint, accurately tracks the player in a virtual environment, provides suitable path based on people's special needs, gives step by step landmark based navigation instructions to direct the user to a specific destination. We also set up a web console to keep track of player's performance in real time. Both the senior and people with disability can quickly access it, which helps them build a mental map of target environment and get familiar with the PERCEPT system.

CHAPTER 3

NAVIGATION INSTRUCTION VALIDATION TOOL

With the increasing use of navigation tools, the number of navigation instructions is also increasing dramatically. Most navigation instructions are generated automatically through a navigation algorithm. Since the number of navigation instructions generated by an algorithm can be very large, validating navigation instructions automatically becomes a challenge for developers. The validation tool proposed in this work is designed to solve this issue.

Simulation technology has been widely used by academic research as a cost-effective way to verify some practical problems before actual test. The basic idea of this validation tool is to simulate an avatar in a virtual environment which has exactly same blueprint as a target building. The avatar can understand navigation instructions and follow them to traverse a particular route from source to destination. Only in case that the avatar successfully reaches the desired destination, the instructions for this route are evaluated as correct, otherwise the system will report this incorrect instruction to the developer.

The system includes two modes of interactions: Regular Speed Mode and Fast Validation Mode. Regular speed mode visualizes the entire validation process, which enables a developer to check each action of the avatar in virtual environment and how the avatar reacts to each navigation instruction. In the fast validation mode, the tool generates all possible combinations of source and destination pairs in the target space and validates all instructions for those routes automatically. The validation speed in this mode has been obviously improved, which is forty times quicker than the regular speed mode. If the current validation task is completed, the validation result for the current task will be reported to the developer's local machine immediately.

In addition to the available validation modes, another feature of this tool is constructing virtual environment dynamically. The virtual environment is created through decoding a target building's floorplan instead of manually creating it, which significantly saves lots of developing time. As a result, the whole system serves as an automatic validation of instruction before the instructions are exposed to users with the navigation tool.

3.1 System Architecture

In this chapter, the architecture overview of validation tool is presented.

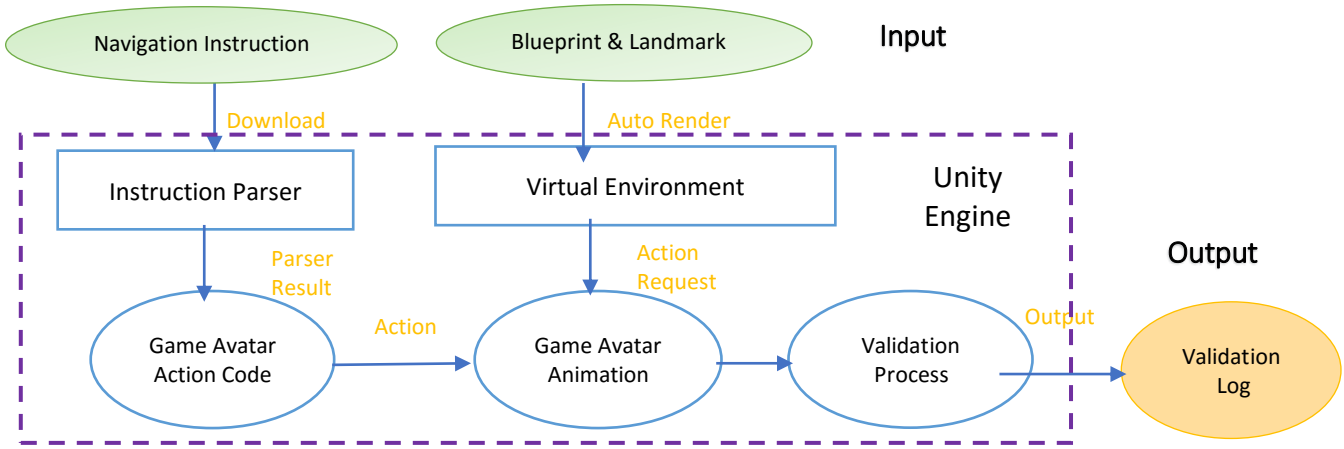

Figure 3.1 System Architecture of Navigation Instruction Validation Tool

The validation tool is developed by Unity game engine which is used to create a board range of 2D and 3D interactive virtual environment, thus becoming an ideal platform to visualize the physical indoor environment. This proposed tool has three major parts: input module, validation module, output module.

The input module is composed of three parts: target building's blueprint, landmark information, and navigation instruction. By decoding the original building's blueprint—a CAD

format file—the virtual environment is rendered in the scene without having to create it manually. The PERCEPT web server stores landmark information as well as navigation instructions on the cloud. Our validation system has to load this information from the PERCEPT server and save them to the local machine, which is also an essential step for constructing a virtual environment.

The validation module consists of several different components: 1) the virtual environment in the game reflects the blueprint of real space. All elements and game objects in the scene are attached with box colliders which can be triggered by the virtual avatar. Once the user chooses a specific validation mode, the system will automatically help the developer configure associated parameters for the current mode; 2) the instruction parser is used to extract action codes sequence from the current navigation instruction, and then 3) the animation module will transmit these action codes to the movement and animation the avatar, finally 4) the validation controller is responsible for monitoring the validation process and recording the validation result for instructions on each route automatically. For the output module, validation log is printout to the user's local machine in real time. User can check it at any time.

3.2 Implementation Details

3.2.1 Representation of the Virtual Environment

Initially, the construction of the virtual environment relies upon that developers generate game objects in the scene according to the building's blueprint manually. In this parper, we put forward a new approach to generating the virtual environment automatically. We figured out a third-party library called .Net dxf, which is developed to read and write AutoCAD dxf files. With the help of .Net dxf, our system retrieves each object's location information from the building's blueprint which is in form of AutoCAD dxf and then converts those information from image format

into text format. In the end, through parsing the object text information, our game engine successfully generates the virtual environment automatically.

The main object that shapes the virtual environment is the wall of a building. There is an example illustrating how do we represent the virtual environment through .NET dxf library.

Step 1:

Group all wall objects in the blueprint to a certain layer in CAD. Utilize Microsoft .netDxf API to read wall objects' information from CAD file and decode them into text format.

Step 2:

Unity Engine reads the specific text file that contains all walls' information in the blueprint. By parsing this file, the wall object corresponding to the real environment is generated in the virtual environment.

Step 3:

Attach the generated wall objects with box collider and set their tag value to wall. With this procedure, the wall object in the scene can be detected by the game avatar and stops the avatar from crossing through them.

Since the scene is made up of different kinds of objects such as stairs, elevator, pillar, windows, doors, and landmarks, we have to render these entities as well so as to better reflect the environment. We group/categorize all objects which have same attributes from the building blueprint and repeat the same procedure as generating wall for them.

Figure 3.2 Generated Virtual Environment of UMass Amherst Campus Center

This approach achieves rendering the virtual environment automatically. Without modifying the source code, it can be applied to any other buildings, which also means it allows a large-scale implementation and has a good scalability.

3.2.2 Parser Algorithm

The ability to understand navigation instructions is critical for intelligent avatar that can navigate the game environment. In order to make the avatar traverse the environment based on the instruction, the instruction parser is designed which can translate instructions into an action code sequence for the virtual avatar. The input of the parser is landmark based navigation instructions generated by PERCEPT algorithm. Those navigation instructions follow a certain grammar format. There are several examples of navigation instructions generated by the PERCEPT.

Sentence1: "Your current location is: Room D Door 1, with the Room D Door 1 to your back"
Sentence 2: "trail the wall on your left side, until you reach an intersecting hallway, 89 feet away"
Sentence 3: "There is Room F to your 2 o'clock direction, walk across to the Room F to your 2 o'clock direction, 28 feet away, you will reach: Room F"

Sentence 4: "With the wall to your right, walk past the opening"
Sentence 5: "With the East Entrance to your back, there is Harvest Market to your 2 o'clock direction, walk across to the Harvest Market to your 2 o'clock direction, you will reach: Harvest Market"

From the above examples, we found out that PERCEPT instructions describe routes using landmark (Room D Door 1, intersecting hallway, Room F), action command (trail the wall, walk across) and some amount of detailed information (89 feet away, 2 o'clock). We also noticed that many instructions' format is similar. Based on this, the parser we introduced in this research applies these features to analyze and process instructions for the game avatar. The basic idea of the parser is first classifying the navigation instructions by their semantic similarity, figuring out which type the current instruction belongs to, next extracting keywords such as "action command" or "landmark name" from it, finally, generating executable action code sequence for game avatar. Here we present the detailed steps of how the interpretation parser works.

Step 1:

Retrieve all navigation instructions for a specified source and destination, segment each instruction into several individual sentences. Here we illustrate an example of segmenting a given instruction.

Original Instruction is:
Your current location is: Room D Door 1, With the Room D Door 1 to your back. Turn left. Trail the Wall on your left side, until you reach an intersecting hallway, 89 feet away

After partition, the original instruction becomes three independent sentence:

1) *Your current location is: Room D Door 1, With the Room D Door 1 to your back*
2) *Turn left*
3) *Trail the Wall on your left side, until you reach an intersecting hallway, 89 feet away*

Step 2:

Figure out which type each navigation sentence is in. In the backend, there is a container that categorizes all possible instruction sentences by the semantic similarity.

Table 3.1 Instruction Type and Navigation Instruction

Type #	Navigation Sentence
0	Your current location
1	Turn
2	There is / Walk across…. on your…clock
3	Trail the wall / Continue and trail / Continue and follow / follow the wall……until reach…
4	Cross the / walk past…...until reach….
5	With the wall…...to your…
6	You have reached your destination

Each row in the above table illustrates a type-number of an instruction and its corresponding navigation sentence. Only a subset of the navigation categories' container is listed here. For the previous example, the parser figures out the type-number of for each sentence based on the container we just mentioned.

1) *Your current location is: Room D Door 1, With the Room D Door 1 to your back → Type 0*
2) *Turn left → Type 1*
3) *Trail the Wall on your left side until you reach an intersecting hallway, 9 feet away → Type 3*

Step 3:

Extract keywords from instruction sentence and generate executable action code sequence based on those keywords. Once the parser knows the type of the current instruction, it knows where to extract keywords from it. Keywords usually means “action command”, “landmark information”

and other navigation details in the sentence. The extracted keywords from each navigation sentence are as follows.

1) *Your current location is: Room D Door 1, With the Room D Door 1 to your back*
   Extracted keyword: Room D Door 1
2) *Turn left*
   Extracted keyword: Left
3) *Trail the Wall on your left side, until you reach an intersecting hallway, 89 feet away*
   Extracted keywords: Left side, intersecting hallway

Step 4:

Convert each action code into the movement or animation of the game avatar. The supervision of this tool is in form of observing how the avatar performs when following the navigation instruction. The below table shows action codes and their associated animations for the game avatar.

Table 3.2 Action Code and Avatar Animation

Action Code	Avatar Animation
Turn right / left	Turn right / left
Continue/ Walk straight	Continue/ Walk straight
Trail the keyword until reach keyword	Ray cast from avatar to find the nearest landmark, Trail the certain landmark to trigger next node
Walk across/past the keyword	Walk straight to trigger next node
With the keyword to your back / left / right	Find closest node to avatar, Turn avatar back/left/right to it

Walk to keyword direction	Ray cast from avatar to find the landmark at the provided direction, walk straight in that direction to trigger next node
You have reached your destination	Finish current task, Trigger next task

3.2.3 Validation Controller

For each source-destination pair (s, d), the instruction parser translates its associated navigation instructions into action code sequence which is used by the virtual avatar to perform corresponding animation and traverse the virtual environment from source s to destination d. If the avatar reaches destination d following these action codes successfully, the navigation instruction for (s, d) are considered as correct. Otherwise, mark the navigation instruction for (s, d) as incorrect. The tool validates navigation instruction for all source-destination pairs and generate a validation log in real time.

The validation controller provides two validation modes for user: Regular Speed Mode and Fast Validation Mode.

- Regular Speed Mode:

This mode visualizes the whole process of how the game avatar traverses the environment from source to destination based on associated navigation instructions. In this mode, the game avatar clearly demonstrates how it executes each action code according to the corresponding instruction. Validating instructions for one source-destination pair takes around thirty seconds. Only in case that the virtual avatar successfully triggers the planned destination, the system considers associated navigation instructions is correct.

- Fast Validation Mode:

Since visualizing the movement of the game avatar is a time-consuming process, Regular Speed mode is not suitable for testing large amounts of data. Based on this, Fast validation mode is designed, which allows the user to quickly validate a bunch of navigation instructions at one time. According to the statistics, validating instructions for one source-destination pair, in average, takes 0.3 seconds, thirty times faster than the visualized mode. This mode relies upon the avatar's quick-travel over two landmarks on each path. We evaluate this approach at the main floor of UMass Amherst Campus Center which including 80 landmarks and 1000 instructions, it successfully found out incorrect instruction sentence, and report the result to user local machine automatically.

3.3 Test Case

Now, we will have a picture based example to illustrate navigation instructions, how they are converted to action codes, how they are used by the game avatar to traverse the virtual environment that represents UMass Campus Center Building and how the validation module works in the system. We assume that the source is the East Entrance and the destination is the UStore as marked in Figure. The instructions are presented to the user in multiple chunks of instructions. Next, we show step by step how to the system generates and validates each instruction chunk.

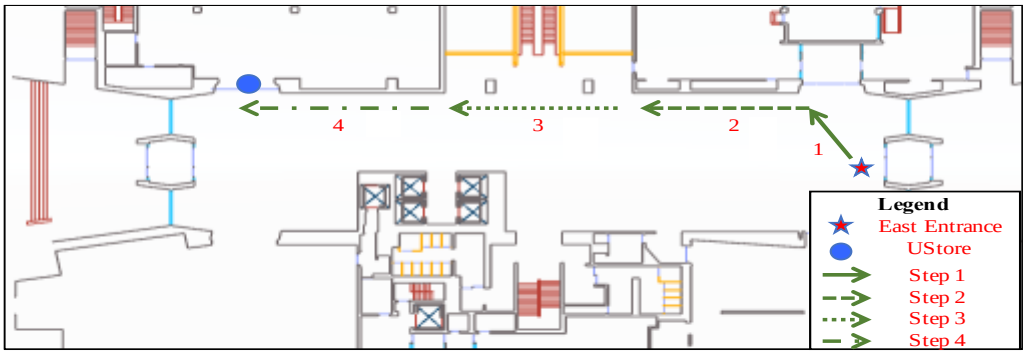

Figure 3.3 Campus Center Virtual Environment

Step 1 (Chunk 1) – from East Entrance (Source Point) to Harvest Market

"Your current location is: East Entrance. With the East Entrance to your back, there is Harvest Market to your 2 o'clock direction, walk across to the Harvest Market to your 2 o'clock direction, you will reach: Harvest Market. Press next Instruction Button."

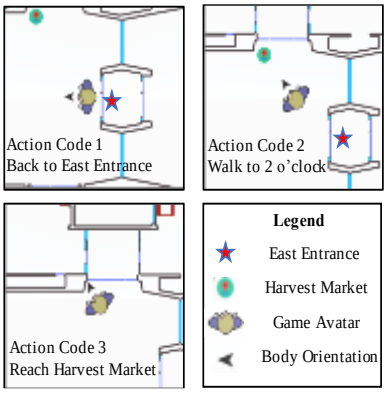

Figure 3.4 Animation of Step 1

Action codes generated by the instruction parser: "back to East Entrance, walk to 2 o'clock, reach Harvest Market". The corresponding game avatar traversal is shown in Figure. Following the action codes, the game avatar will go from East Entrance to Harvest Market at 2 o'clock direction. Then the next chunk of instructions is presented. Since there is no collision occured during the traversal of avatar, the current instruction is marked as correct.

Step 2 (Chunk 2)– from Harvest Market to opening

"Continue, trail the wall on your right side, until you reach an opening. Press next Instruction Button"

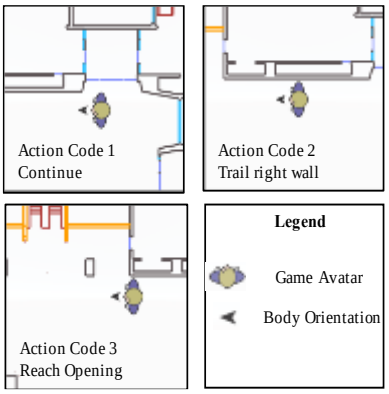

Figure 3.5 Animation of Step 2

Action codes generated by the instruction parser: continue, trail right wall, reach opening. Game avatar traversal is shown in Figure 6. Following the action code, the game avatar will go from Harvest Market to an opening following the wall on its right side. Then the next chunk of instructions is presented. Additionally, validation module evaluates current instruction as correct.

Step 3 (Chunk 3) – walk past the opening

"With the wall to your right, walk past the opening. Press next instruction button".

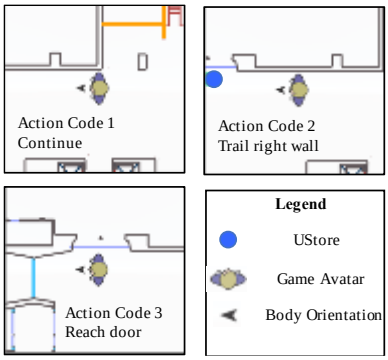

Figure 3.6 Animation of Step 3

Action code generated by the instruction parser: right to wall, walk past opening, stop. Game avatar traversal is shown in Figure. Following the action code, the game avatar will walk past the opening and stop at that point. Current instruction passes the validation module. Then the next instructions chunk is presented.

Step 4 (Chunk 4) – from opening to UStore (Destination)

"Continue, trail the wall on your right side, until you reach the first door, you will reach: UStore. Press finish button to end the journey."

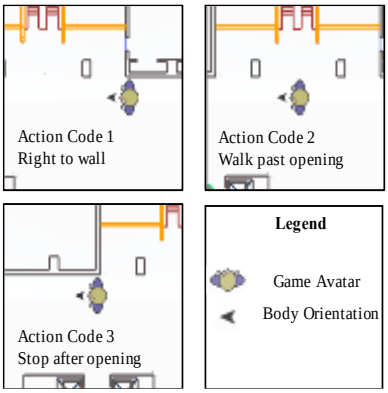

Figure 3.7 Animation of Step 4

Action code generated by the instruction parser: continue, trail right wall, reach door. Game avatar traversal is shown in Figure. Following the action code, the game avatar will continue walking and follow the wall on its right side until it reaches a door. Then it will reach the destination, which is UStore. Until now, all the instructions on the routes have been tested, and they successfully navigate avatar to reach the planned destination. Thus, all of them has been marked as correct by validation module.

Overall, the validation framework developed in this work servers as both a template and a prototype for validating any navigation instruction and inside any building.

CHAPTER 4

INDOOR WAYFINDING TRAINING SYSTEM

In this chapter, we present a novel indoor navigation tool called 3D indoor wayfinding training application for seniors and people with disabilities. This application has been developed with virtual reality as well as audio-based technique. It enables users to get familiar with physical space and navigation instruction in the convenience of their own home. As a result, users establish a mental map of the target environment prior to their arrival to the physical space, increasing their confidence and therefore increasing their chances of using public transportation.

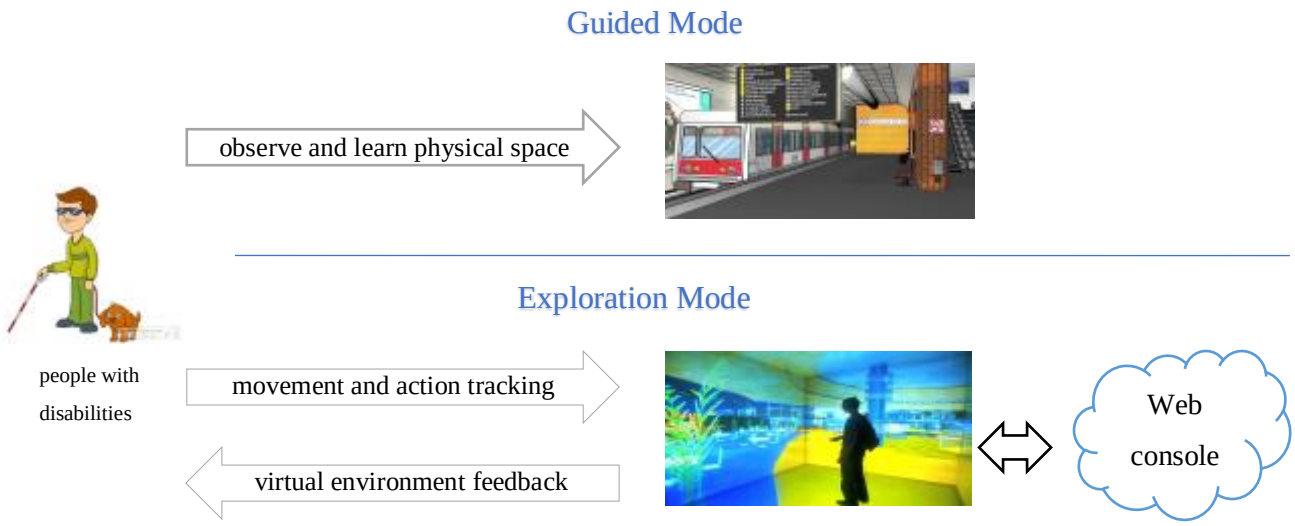

Figure 4.1 System Overview of Training Game

This training system provides two modes: 1) Guided mode in which the path between source and destination is shown to the user automatically from third person perspective. The user can observe and learn the physical space by following game character in the scene, and 2) Exploration mode in which the user explores and interacts with the environment by role-playing a virtual character. This mode is designed for learners who are not able to access the information as quickly as others. The user controls a principle character which has to carry out a series of

wayfinding tasks. When users are encouraged to fulfill a task, they are more willing to become familiar with the environment and listen to the instruction to identify details around them.

We also introduce a web console to keep track of user performance that would allow for a more precise evaluation of our system.

4.1 System Architecture

In this section, the training system architecture overview is shown as below. The system is composed of several components. Next, I will go through them one by one.

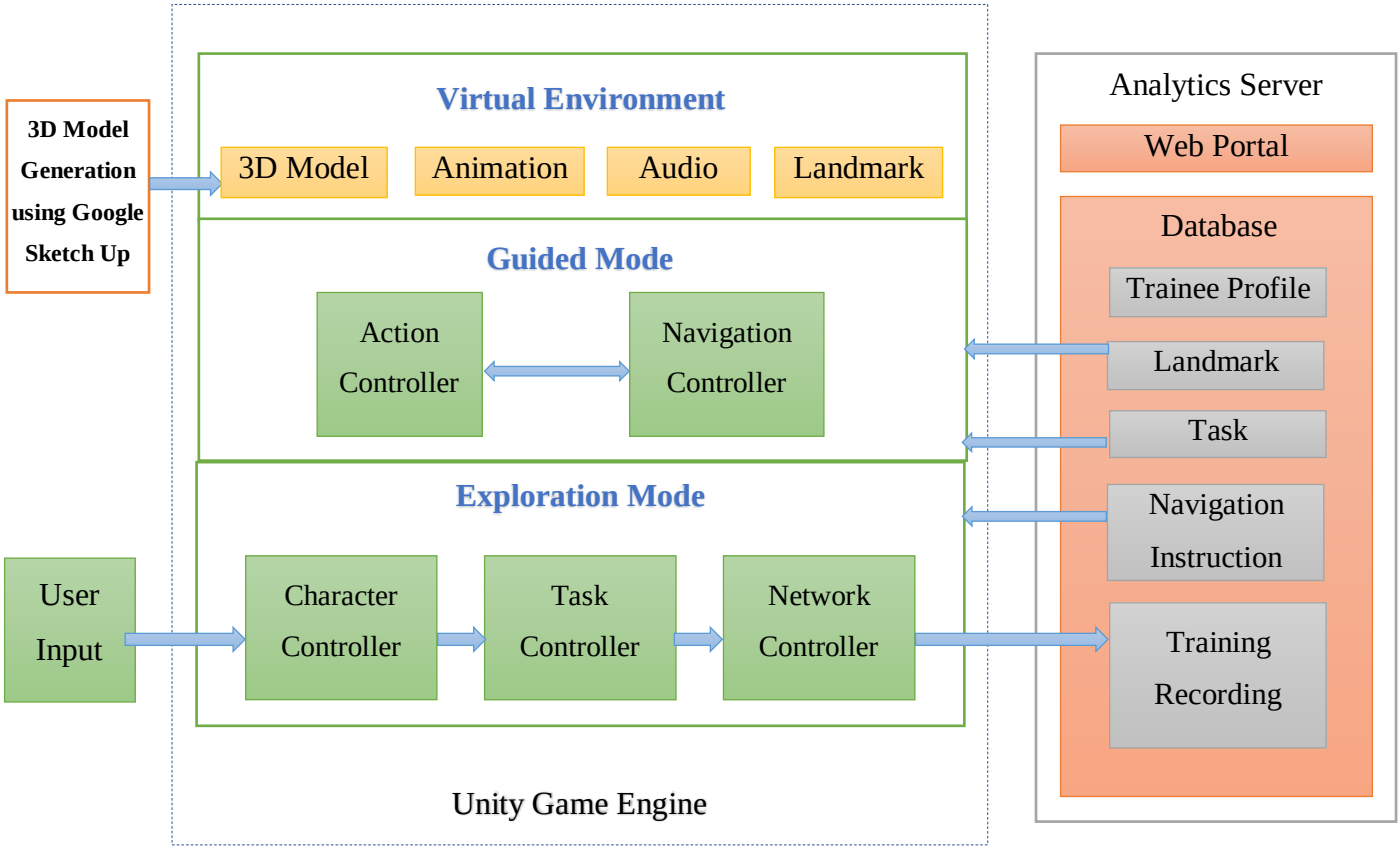

Figure 4.2 System Architecture of Training Game

Create Environment Tier:

- 3D Model Module: just like the training game, there is a 3D virtual environment which can reflect the shape of target building.
- Animation Module: we customize different animations for various objects in the game so that it can enhance the immersion experience of the player.
- Audio Module: the game incorporates various sound audio cues and voice instructions to fit the needs of blind users. users can know their approximate location by audio feedback.
- Landmark Module: this module enables to generate all landmark objects dynamically in the scene.

Self-Guided Tier:

- Navigation Controller: with the help of unity built in navigation system, the game character knows how to find a path to the next point and move accordingly in the scene.
- Action Controller: Through the event system, we can register different kinds of action events for the game character. And the character should respond an action event when it is triggered.

Exploration Tier:

- Character Controller: this module controls the movement of the virtual character in the game according to the player's input.
- Task Controller: we designed a task manager to store a chain of tasks that help player go through all important locations in the environment and get familiar with their description.

4.2 Implementation Details

In this section, we will introduce the implementation details of this training system, which will cover the following areas:

- Development Tool: Unity 3D
- Creating Virtual Environment
- Self-Guided Development process
- Exploration Development Process

4.2.1 Development Tool

Our training system is based on Unity 3D that is a powerful and versatile tool to achieve interactive-experience for users. Unity 3D is a modern game engine, offering a variety of built-in features such as materials, texture, lighting, physics effect, animations and audio support, auto-navigation scheme, as well as powerful user interface for development. Additionally, Unity 3D takes advantage of scripting programming. The behavior of game object in the game is controlled by script components that is attached to them. Script is usually created within Unity directly, and it allows developer to trigger game events, modify component properties over time and respond to user input in any way they like. Two different kinds of programming languages support in Unity: C# and JavaScript, which many software major students may already be familiar with.

4.2.2 Creating Virtual Environment

The virtual game environment represents the real-world environment of target building. For the current version, our target building is Boston North Station. Creating Virtual Environment is composed of four components: Construct 3D model, Generate Landmark, Add Animation and Add Audio Sound. The rest of this section will go into the implementation details of creating this virtual environment.

1) Construct 3D Model

In order to provide a 3D virtual environment, we need to create a 3D model according to the venue's Blueprint. We generate this 3D model in Google Sketch Up and import it into Unity.

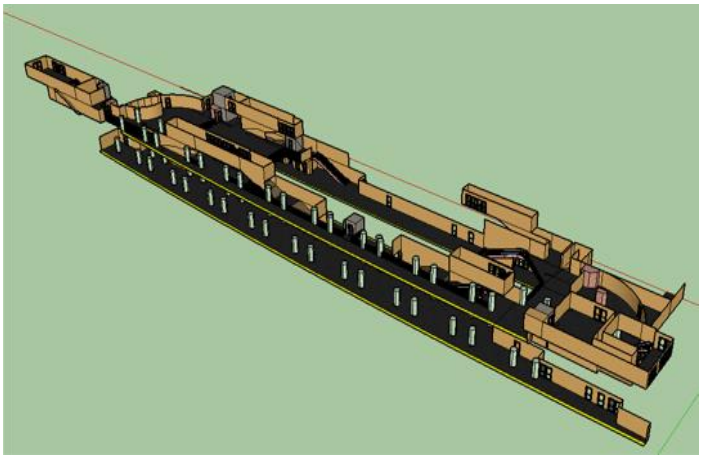

Figure 4.3 3D Virtual Environment Model

- Generate Landmark

Navigation instructions in our system use landmarks which represent points of interests in the venue. Landmarks are crucial for people with disabilities in performing wayfinding tasks in indoor environments. For example, we set up fare gates, escalators, and elevators in the environment as landmarks. We implemented a script that downloads the landmarks information from a remote server, converts the landmarks location given in world coordinates to Unity coordinate system, and creates landmark game objects in the scene.

- Add Animation

In this application, all the original models imported from external are static. To enhance the reality of the virtual environment, all special models in the game will show various animations according to different situations. Unity has a rich and sophisticated animation system allowing

developer to create animations directly within it. we describe and explain and the animation development process with an example of subway train below.

Step1: create animation clips

Assume we want a subway train comes from one side of the scene to the platform. To achieve it, we should create a corresponding animation clip for the train model. Here we named this this animation as “TrainComingAnim”. The figure below is the Unity Animation Window. The timeline showed in the right-side panel is where we are able to establish the timing of our animation clip.

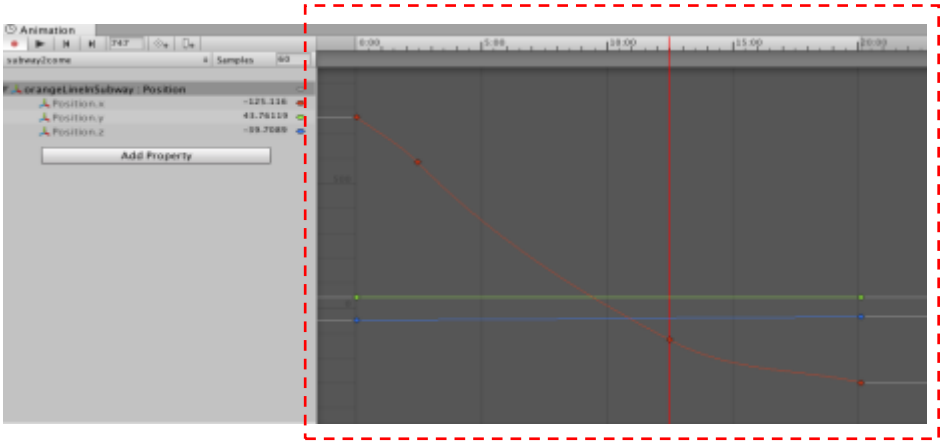

Figure 4.4 Create Animation Clip

We use the three curves in the figure to define the x, y, z location values of the train over the length of the animation. As we can see, the red curve becomes smoothly when it closes to the end of animation which means the subway train slows down as it approaches to the platform.

we generated other animations clips for the subway train through the same method, those are “TrainDoorOpenAnim”, “TrainDoorCloseAnim”, “TrainLeavingAnim” and “TrainIdelAnim”.

Step 2: manage and control animation sequence

We use Animator Controller to arrange a set of animation clips. We drag all animation clips associated with subway train into the Animator Controller. All those animations clips are then organized manually into a structured flowchart system as it shown in figure below.

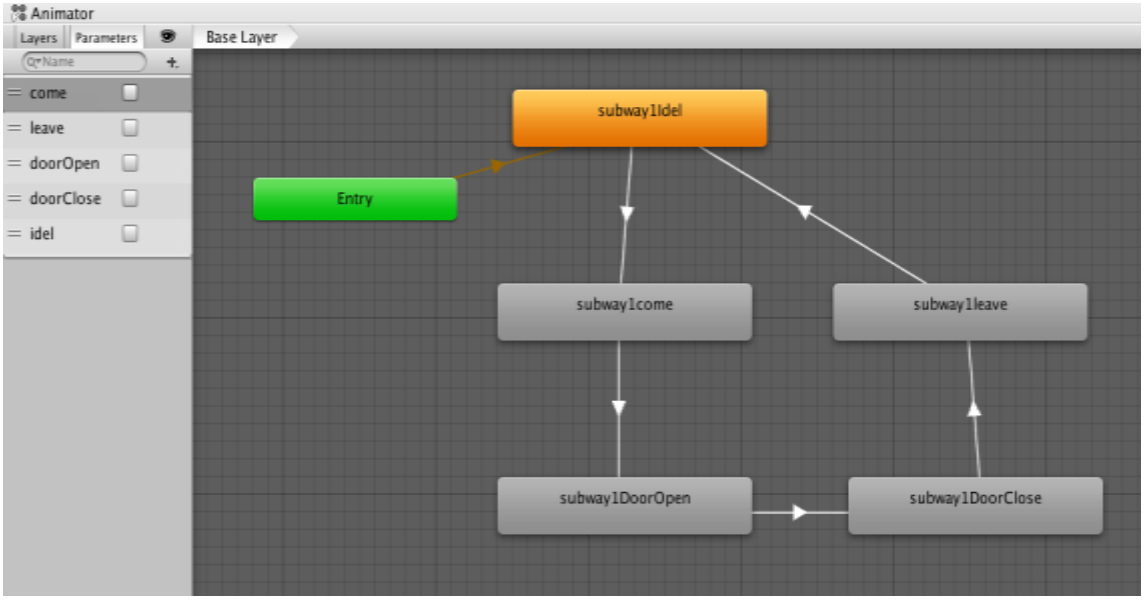

Figure 4.5 Animator Controller

The left-side panel shows five trigger parameters we created: “idle”, “come”, “leave”, “doorOpen” and “doorClose”. The Animator Controller uses those parameters to affect the transition between two animation clips and determines which animation should play in the next frame.

Step 3: create animator scripting

We set those trigger parameters in script to tell the train when it should switch to a new state and play a specific animation. For instance, if the "come" parameter is set as it's shown in below, the subway train will enter "come" state and play "TrainComingAnim" animation.

```
if (anim.Equals ("come"))
{
    subwayAnim.SetBool ("come", true);
    arriveSound.Play ();
    subwayAnim.SetBool ("idel", false);
    StartCoroutine (WaitForAnimation ("Base Layer.subway1come"));
}
```

Figure 4.7 Write Animator Script

- Add Audio

We provide audio sounds to help player interact with the virtual environment and help them get spatial information from surroundings. There are three types of sounds: Environmental Sounds, Indicator Sounds, and Navigation Sounds.

Type1: Environmental Sounds

The volume of the environmental sound will increase/decrease in proportion to the objects' distance from the user.

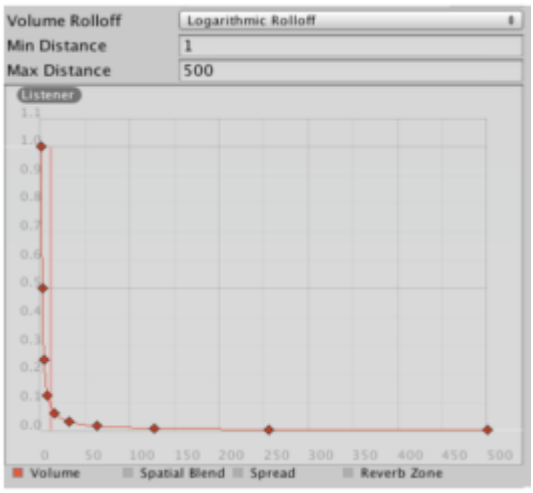

Figure 4.8 Environment Sound

Figure 4-8 shows their relationship. The horizontal direction reflects the distance between object and the user. And the vertical direction shows the percentage of the normal volume. We modify the volume distance function to Logarithmic curve. When the player approaches a particular object which generates audio, the player can get increasingly accurate sound from the object. For example, the closer to the platform, the more clearly the player can hear the trains' sound.

Type 2: Indicator Sounds

We have access control mechanism in the game, i.e., if a trainee tries to cross a wall or enters an escalator through a wrong entrance, a warning indicator sound plays to deny the trainee's forward moving. When the virtual character goes into a landmark area, this training application will provide the description of the current place to the trainee, e.g. "Fare Gate nearby". So, the trainee can get useful spatial information from his/her surroundings.

Type 3: Navigation Sounds

Our system provides navigation instructions according to the trainee's current location in the virtual environment as well as his/her chosen destination. When the trainee arrives at a landmark, the next voice navigation instruction will play.

4.2.3 Development Process in Guided Mode

In Guided mode, the virtual character navigates automatically from a selected source-destination through a planned route. The planned route corresponds to the path described by navigation instructions. There are two main modules in this mode: Action Controller and Navigation Controller.

1) Action Controller:

allows the trainee to register different action events for the virtual character. With the event system, the virtual character will be able to perform pre-defined actions when needed, e.g. take elevator to a different floor, or scan a fare ticket to pass the fare gate. Examples of such action events: Push button of elevator event

- Scan ride ticked event
- Enter escalator event
- Exit escalator event
- Reach platform event

2) Navigation System:

Unity build-in navigation system allows us to create a character that can intelligent move around the game world. The process to build an intelligent agent is as follows.

Step 1: define walkable area in the scene

After creating the scene in Unity, select the desired walking ground and mark them as "Walkable" with navigation mesh component. From the figure below, you can see all walkable area are highlighted in blue.

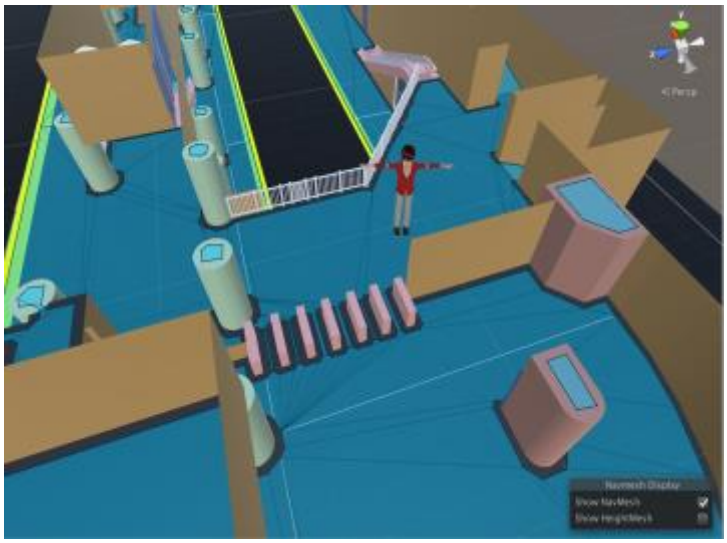

Figure 4.9 Walkable Area in The Scene

Step 2: add "Navmesh Agent" component to character

Navmesh Agent component helps us create a character which can avoid obstacles while moving to its goal and give the virtual character the ability to find a path between two locations in the scene.

Step 3: attach the following function to the agent

1. push all planned passing points for the current route in an ordered queue

2. get the path-finding component for the player, which can give the agent the ability to automatically calculate the required travel path in path-finding process and move accordingly.
3. if agent reaches its goal, set the next point of the queue as next destination
4. do the above process repeatedly until the character reaches its target destination.

4.2.4 Development Process in Exploration Mode

In Exploration mode the trainee explores the environment by role-playing a virtual character. There are three main unique components in this mode: Character Controller, Task Controller, and Network Controller described below.

1) Character Controller:

The exploration mode is played from a first person perspective. The trainee can move freely throughout the environment. The character controller manages the orientation and movement of the virtual character following the trainee's inputs, including forward, backward, turning around, or looking up and down.

2) Task Controller:

The task controller is designed to train the trainee to go through all important locations in the environment and get familiar with their description. The trainee will be assigned a sequence of navigation tasks, T_i (i=1,…m): Task i is identified by a source-destination pair (Si, Di). If the character reaches destination Di, successfully next task will be presented to the trainee. The process

terminates when the trainee finishes all assigned tasks. The trainee can press “Lost” button to return to the starting point of the current task.

3) Network Controller:

The navigation instructions of each task are divided into a number of chunks which we denote as subtasks. Every time, a trainee fulfills a subtask or gets lost in the game, the network controller can detect it and submit the trainee’s behavior information to the analytics server. Typically, each behavior data contains each action the trainee made, the location where the action took place, the time and walking distance it took the trainees to complete the current action, and associated instructions.

4.3 Case Study in North Station

We have implemented the game for Boston North Station subway where all elements in the physical environment have corresponding locations in the virtual environment as shown in Figure 4-10. We depict all three floors of the station including the entrances, escalators, elevators, stairs and fare gates. We illustrate two examples of the training game: Guided mode and Exploration mode.

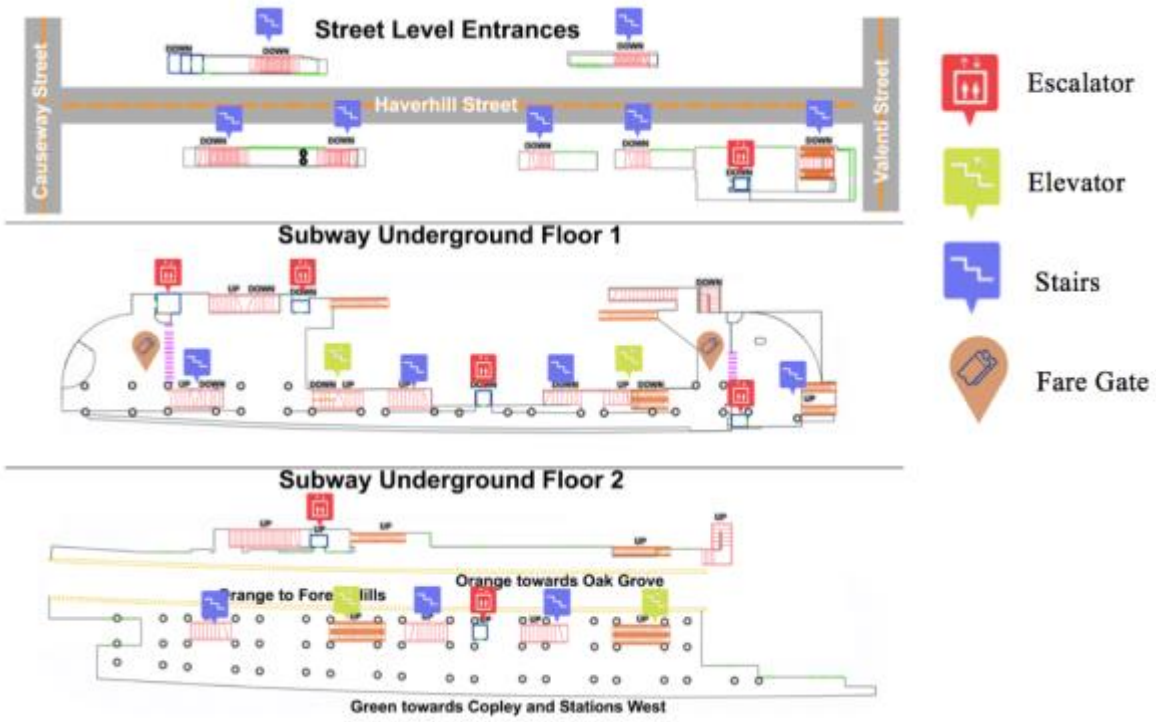

Figure 4.10 Blueprint of Boston North Station

4.3.1 Guided Mode Scenario

We assume that the trainee chose the following guided task: Valenti Way Entrance (source) -> Green Platform to Copley and West (destination). The 3D view of the trainee while performing each one of these subtasks (Figure 4-11).

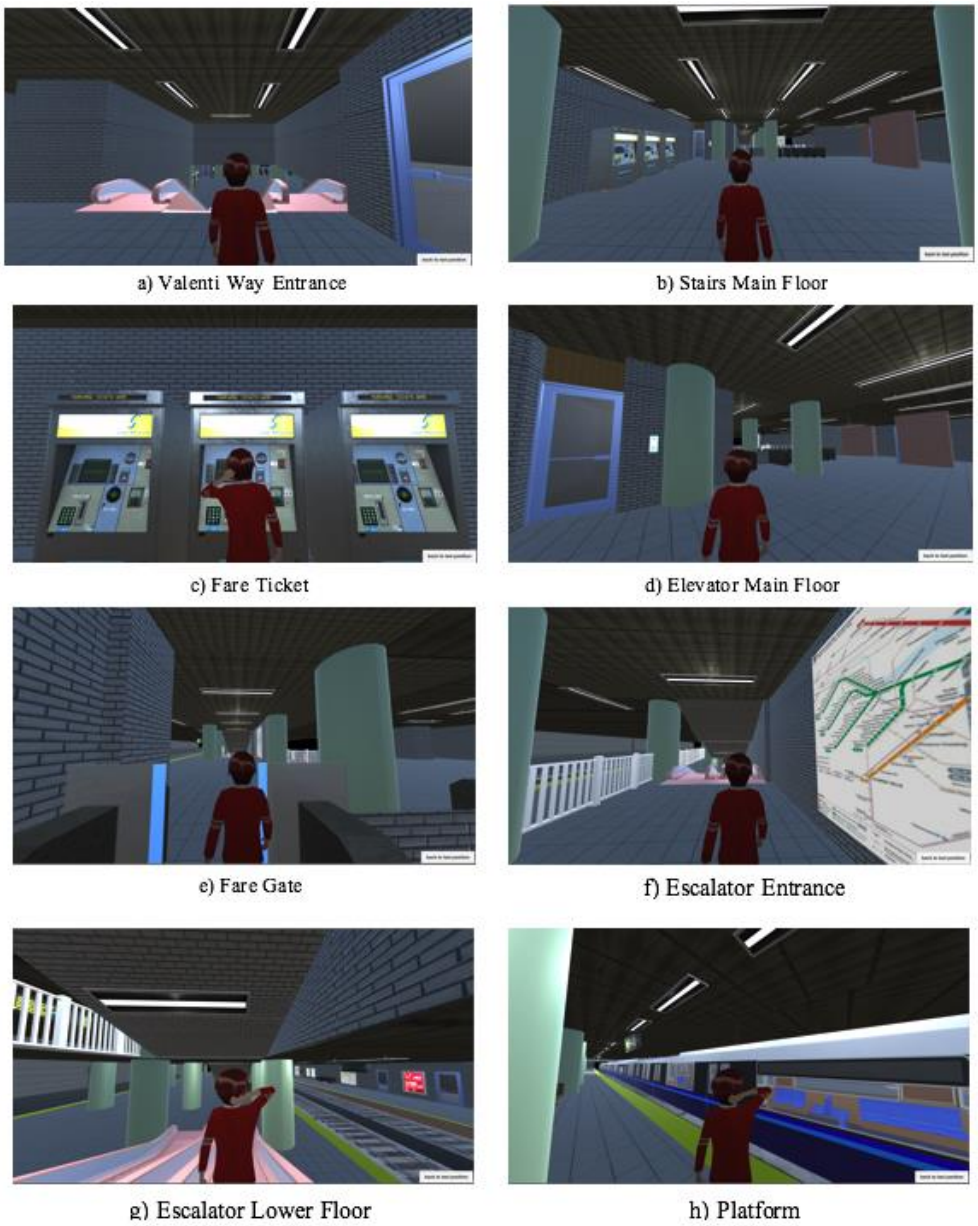

Figure 4.11 Guided Mode Scenario in 3D

The subtasks are described below:

1. subtask 1 (landmark 1): current location is: Valenti Way Entrance, go down two sets of stairs, (figure 4-11. a)

2. subtask 2 (landmark 2): reach stairs main floor, (figure 4-11. b)

3. subtask 3 (landmark 3): turn to 10 o' clock direction, walk across the opening, and buy a fare ticket (figure 4-11.c)

4. subtask 4 (landmark 4): pass elevator main floor (figure 4-11. d)

5. subtask 5 (landmark 5): scan the fare ticket and go through the fare gate (figure 4-11. e)

6. subtask 6 (landmark 6): walk past the opening until reach escalator main floor (figure 4-11. f)
7. subtask 7 (landmark 7): take the escalator down and reach escalator lower floor (figure 4-11. g)
8. subtask 8 (landmark 8): turn right and walk straight ahead until reach platform (figure 4-11. h)

4.3.2 Exploration Mode Scenario

We assume that the trainee chose the following exploration task: Haverhill Street Entrance (source) –> Orange Line Inbound (destination)

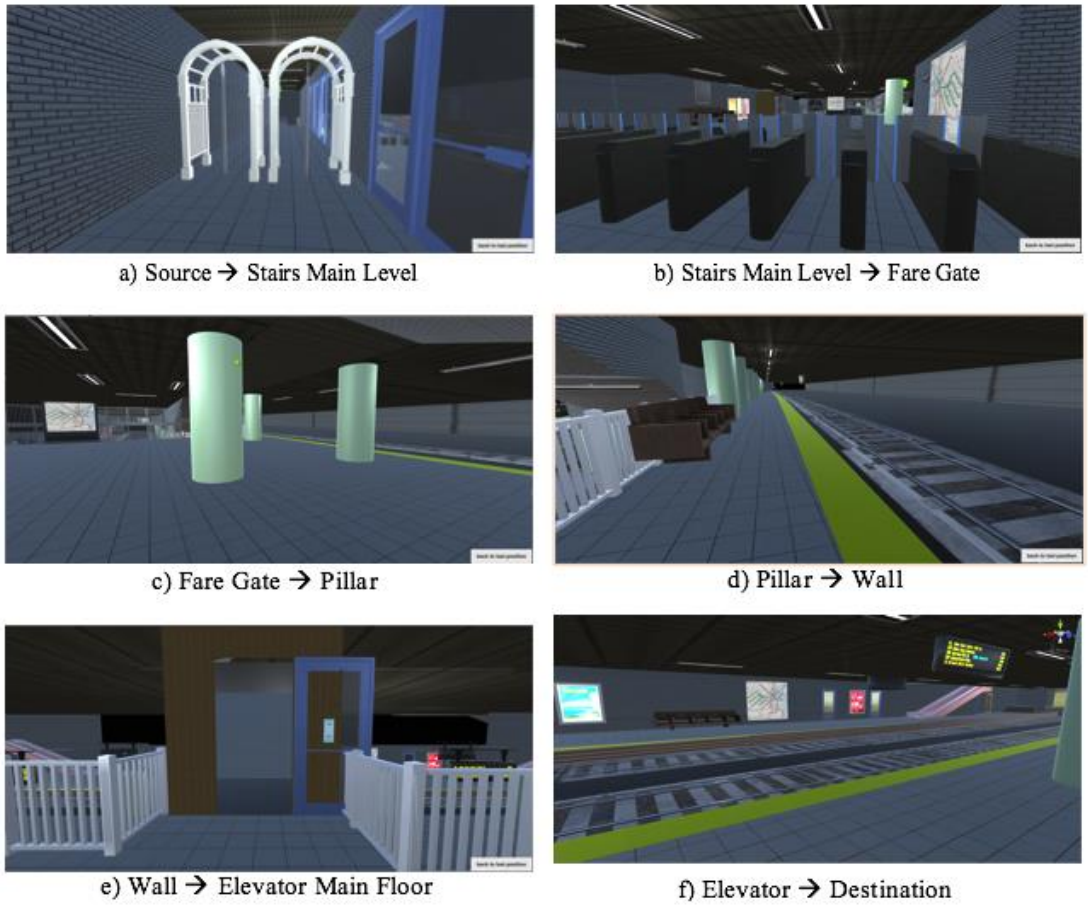

Figure 4.11 Exploration Mode Scenario in 3D

The exploration task includes the following subtasks:

1. Subtask 1(landmark 1→landmark 2): Your current location is: Haverhill Street Entrance, go down two sets of stairs, until you reach the Stairs Main Floor, (see in figure 4-11.a)

2. Subtask 2 (landmark 2→ landmark 3): Your current location is Stairs Main Floor, turn right, heading northeast, reach the Fare Gate to your right side, (see in figure 4-11.b)

3. Subtask3 (landmark 3 → landmark 4): Go through Fare Gate, trail the Wall on your right side, heading north, until you reach the Pillar (see in figure 4-11.c)

4. Subtask 4 (landmark 4→ landmark 5): There is Wall to your 1 o'clock direction, walk across to the Wall to your 1 o'clock direction, heading south, 45 feet away, (see in figure 4-11.d)

5. Subtask 5(landmark 5→ landmark 6): Continue, trail the wall on your left side, until you reach the Elevator Main Floor, 180 feet away, you will pass by 1 opening, Take the elevator down (see in figure 4-11.e)

6. Subtask 6(landmark 6→landmark 7): Your current location is: Elevator Lower Level, With the Elevator Lower Level to your back, there is Orange Line Inbound to your 6 o'clock direction, walk across to the Orange Line Inbound to your 6 o'clock direction, heading northeast, until you reach the Orange Line Inbound (see in figure 4-11.f)

CHAPTER 5

VISUAL ANALYTICS TOOL

The previous chapter has introduced how the virtual reality game assists people with disabilities to get familiar with the target space and PERCEPT system. Certainly, any training process requires information feedback in order to allow monitoring the learning progress. In this chapter we introduce game based visual analysis tools that tracks players' information in real time. Using this information we will provide feedback to the trainees, the trainers as well as help the system developer optimize the wayfinding training system.

We developed the visual analytics tools using the Microsoft Asp.net framework as shown in Figure 5-1. By implementing ASP.net Web API, our server can handle multiple unity requests at the same time.

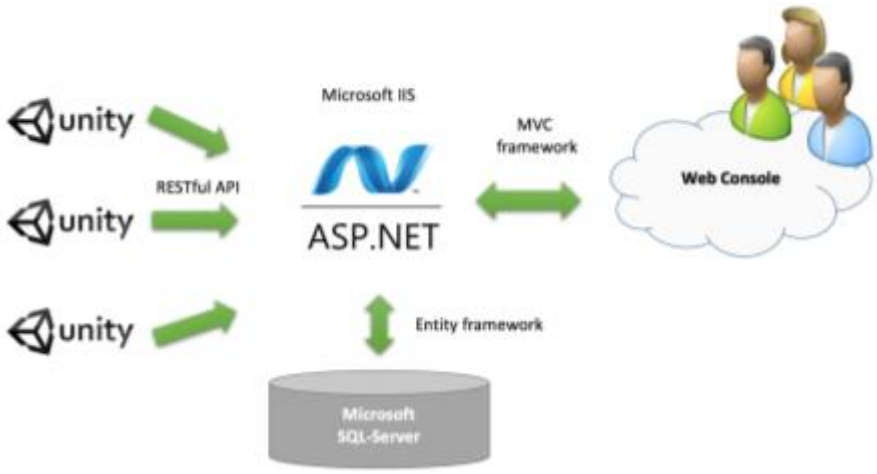

Figure 5.1 Visual Analytics Framework using ASP.NET

Before we introduce the visual analytics, we define the following terms:

- **Trainee:** each registered trainee in this training system. The trainee data includes: age, gender, vision impairment level, computer familiarity, and navigation skills.

- **Task:** a mission asking a player to explore from a certain source to a specified destination according to the planned route. Each task has a specific source, destination, and corresponding navigation instructions.
- **Landmark:** each landmark is an interest point in the target space and has its own unique Latitude and Longitude.
- **Subtask:** the traversal mission from one landmark to another in the game. Each subtask has attributes such as previous node (landmark), current node (landmark), time cost, walking distance etc.
- **Attempt:** each training session of a player in the game. Each attempt has an associated task and a certain number of related subtasks.

The development process of such tool includes the following phases: 1) model creation, 2) data collection, 3) data analysis and 4) metrics visualization. We describe below each one of these phases.

5.1 Model Creation

The server is responsible for managing trainees' profiles, training records as well as task and landmark data. The database of the server built with SQL-Server includes: User, Task, Landmark, Subtask and Attempt. The database diagram is shown in Figure 5-2.

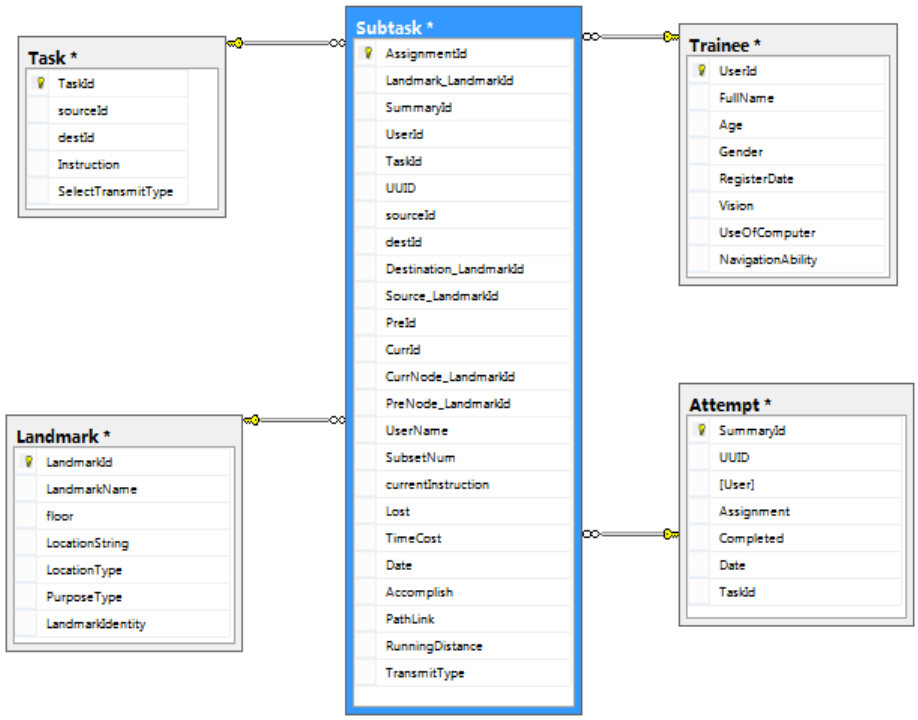

Figure 5.2 Database diagram

5.2 Data Collection

Since data collection is crucial to the analysis process, we describe how the data is collected:

- **Trainee-** A new registered account will be mapped to a trainee entity in database
- **Landmark-** All landmark information is merged from Percept Server
- **Task -** All landmark information is merged from Percept Server
- **Subtask -** Every time a player fulfills a subtask or gets lost in the game, the action information along the player identity is sent to the server as a subtask object. The information includes user ID, Task ID, previous node, current node, walking distance, time cost and associated instructions. At the server side, the received subtask object will be mapped as a new entity in the subtask-table and saved in the database.

- **Attempt -** Each new time the user starts the game will be sent to the server and is stored as an attempt object in the database

5.3 Data Analytics

Once all needed information is gathered, the system starts to evaluate the collected and stored data. Before we define the system performance metrics, we define the following terms:

$\boldsymbol{t}$ *: a certain attempt*

$\boldsymbol{j}$ *: a certain trainee*

$\boldsymbol{k}$ *: a certain subtask*

$\boldsymbol{i}$*: a certain task*

$\boldsymbol{m_j}$*: number of different tasks taken by trainee j*

$\boldsymbol{n_i}$*: number of trainees who traversed task i*

$\boldsymbol{p_j}$*: number of subtasks taken by trainee j at attempt t*

$\boldsymbol{n_{jk}}$*: number of attempts taken by trainee j for subtask k*

$\boldsymbol{l_k}$ *: number of trainees who traversed subtask k*

$\boldsymbol{C_{jk}(t)}$*: trainee j succeeded or failed subtask k at attempt t (succeed: 1, fail: 0)*

$\boldsymbol{T_{jk}(t)}$*: number of times subtask k was traversed by trainee j at attempt t*

$\boldsymbol{A_{ij}(t)}$*: trainee j succeeded or failed task i at attempt t (succeed:1, failed:0)*

$\boldsymbol{N_{ij}}$*: number of attempts taken by trainee j for task i*

- **Performance metrics**

Using the definitions above we define the performance metrics in two categories: efficiency and success.

$\boldsymbol{EST_{jk}(t)}$*: efficiency of subtask k of trainee j at attempt t*

$$EST_{jk}(t) = \frac{C_{jk}(t)}{T_{jk}(t)}$$

$\boldsymbol{ET_{ji}(t)}$*: efficiency of task i of trainee j at attempt t*

$$ET_{ji}(t) = \frac{\sum_{k=1}^{p_j} C_{jk}(t)}{\sum_{k=1}^{p_j} T_{jk}(t)}$$

$\boldsymbol{EST_{jk}}$*: efficiency of subtask k of trainee j for all attempts*

$$EST_{jk} = \frac{\sum_{t=1}^{n_{jk}} C_{jk}(t)}{\sum_{t=1}^{n_{jk}} T_{jk}(t)}$$

$\boldsymbol{EST_{k}}$*: efficiency of subtask k across all trainees*

$$EST_{k} = \frac{\sum_{j=1}^{\boldsymbol{l_k}} \sum_{t=1}^{\boldsymbol{n_{jk}}} C_{jk}(t)}{\sum_{j=1}^{\boldsymbol{l_k}} \sum_{t=1}^{\boldsymbol{n_{jk}}} T_{jk}(t)}$$

$\boldsymbol{SU_j}$*: success rate of trainee j over all tasks,* $\boldsymbol{m_j}$*-*

$$SU_j = \frac{\sum_{i=1}^{m_j} \sum_{t=1}^{N_{ij}} A_{ij}(t)}{\sum_{i=1}^{m_j} N_{ij}}$$

$\boldsymbol{ST_i}$*: success rate of task i - over all trainees,* $\boldsymbol{n_i}$*-*

$$ST_i = \frac{\sum_{j=1}^{n_i} \sum_{t=1}^{N_{ij}} A_{ij}(t)}{\sum_{j=1}^{n_i} N_{ij}}$$

Table 5 depicts the performance metrics of interest for the trainee, instructor and developer.

Table 5.1 Performance metrics viewed by Trainee, Instructor, and Developer

Type	*Metrics*	*Trainee*	*Instructor*	*Developer*
Efficiency	$EST_{jk}(t)$		*yes*	*yes*
	$ET_{ji}(t)$		*yes*	*yes*
	EST_{jk}		*yes*	*yes*
	EST_{k}			*yes*
Success Rate	SU_j	*yes*	*yes*	*yes*
	ST_i		*yes*	*yes*

CHAPTER 6

DATA VISUALIZATION ON ANALYTICS TOOL

So far, all the issues regarding the backend of the analytics tool have been addressed. In this section, we will move to the front end of the analytics tool. First, we present a hierarchy of the website and then go into the details of each section on the web.

The diagram shown in Figure 6-1 depicts the hierarchy for our web console:

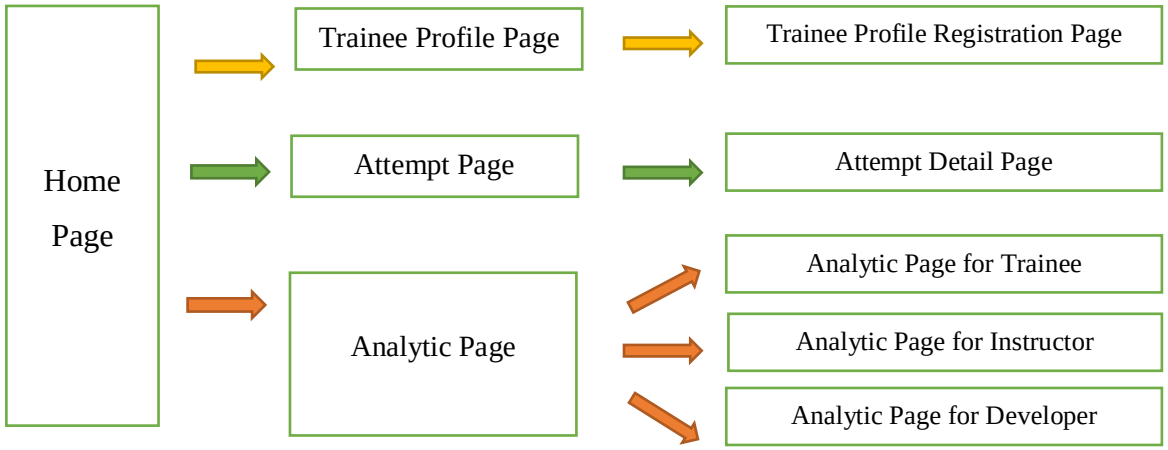

Figure 6.1 Hierarchy of Web Console

Web Console Interface:

1) Home Page (see Figure 6-2)

The navigation bar at the top contains the links to “Trainee Profile Section”, “Attempt View Section” and “Analytic View Section”. With the navigation bar, viewers are able to easily navigated to a particular section of interest.

Figure 6.2 Home page

2) Trainee Profile Section (see Figure 6-3)

Trainees who want their training performance to be tracked have to register. The registration page allows them to set some personal information like age, vision impairment level, computer familiarity, and navigation skills.

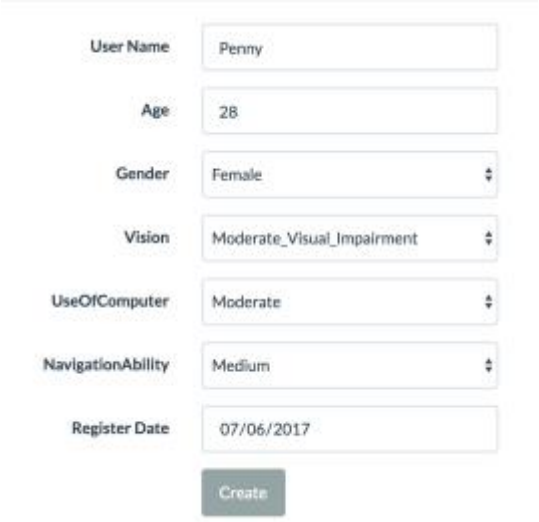

Figure 6.3 Trainee Profile

3) Analytic View Section

6.1 Analytic Page for Trainee

Objective: the trainees monitors their own progress and can identify subtasks where they have difficulties (they can focus in these subtasks).

The trainee menu page lists all registered trainees in our training system with links to each user profile and analytics details (see Figure 6-4).

Figure 6.4: Trainee Analytics Menu

Trainee analytic details page includes the following contents:

Content 1: Task efficiency per attempts for multiple tasks is depicted in Figure 6-5 for each

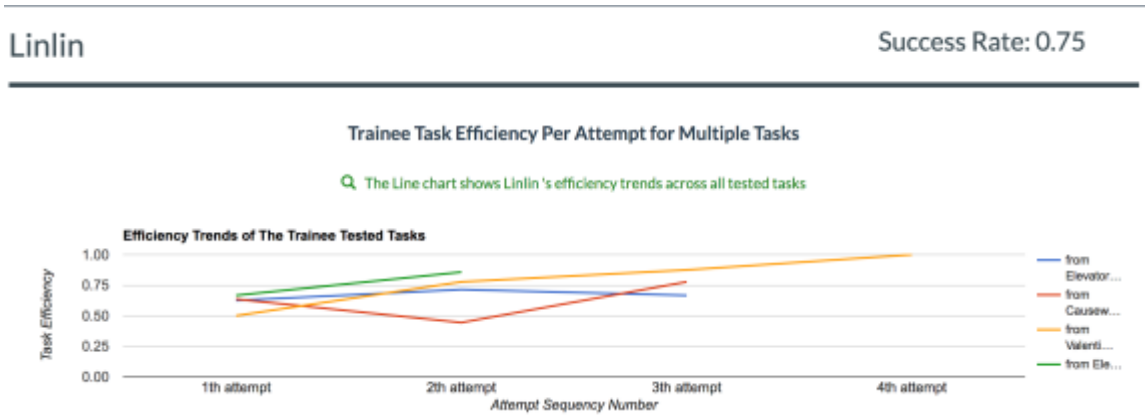

Figure 6.5: Task Efficiency per Trainee over Multiple Tasks and Multiple Attempts

As expected, we observe that the task efficiency increases as the number of attempts increases, i.e. the trainee proficiency increases with increased number of attempts.

Content 2: Blueprint analytics map (see Figure 6-6) in which we depict with different colors to indicate different subtask efficiencies. This statistics can help the trainee identify situations where he/she fails repeatedly.

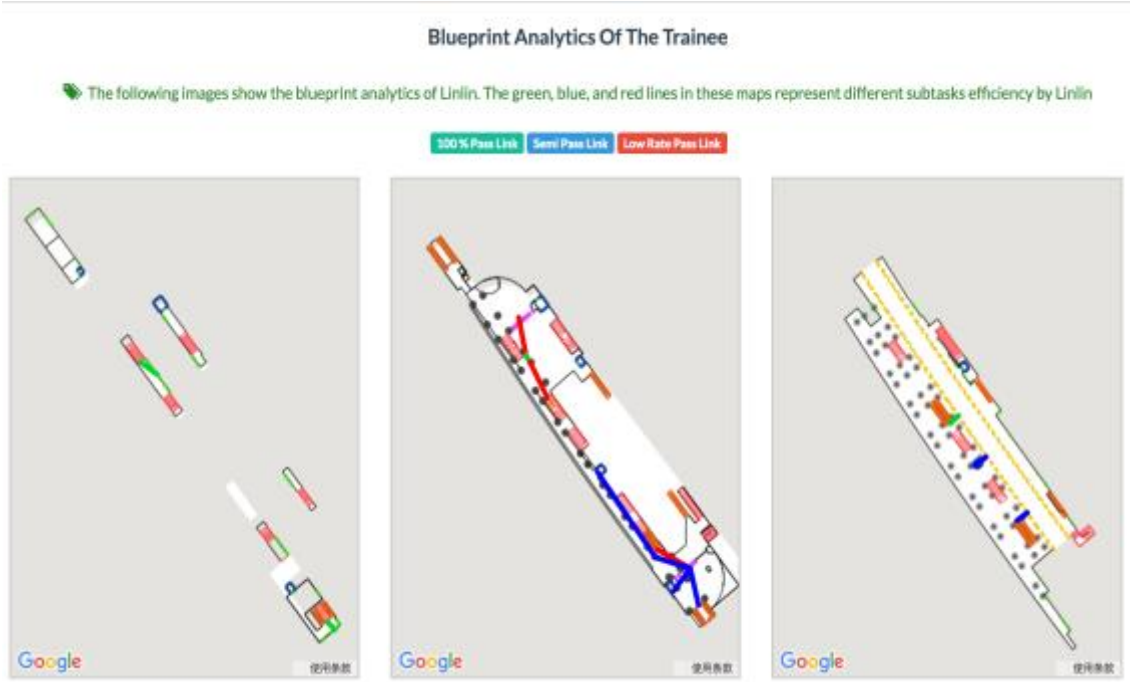

Figure 6.6: Blueprint Analytics per Trainee

6.2 Analytic Page for Instructor

Objective: This section allows the instructor to follow individual trainees performance and average trainees' performance and provide feedback to both the trainees and the system developer.

Figure 6-7 provides two types of data analytics for instructor: task analysis and user analysis.

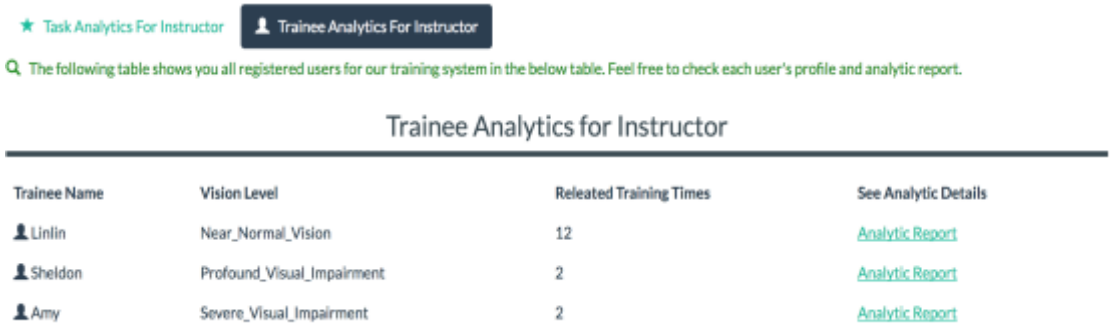

Figure 6.7.1 Data Analytics for Instructor: Trainee Analysis

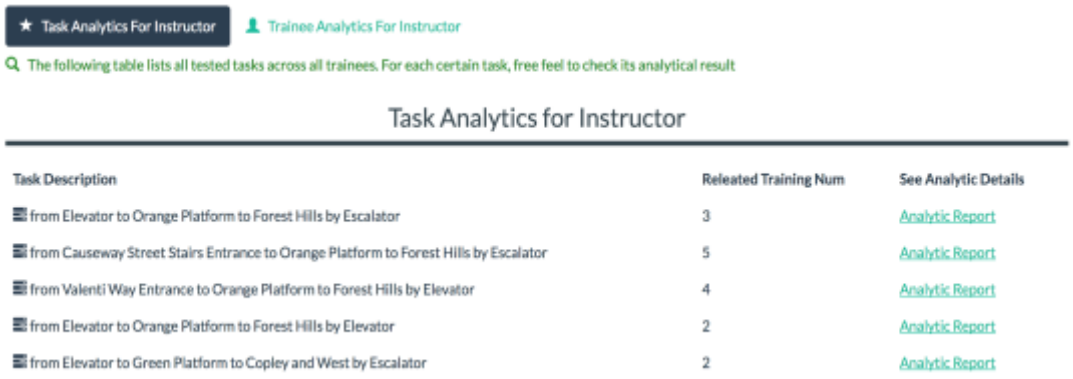

Figure 6.7.2 Data Analytics for Instructor: Task Analysis

The “analytic details” button at each row in the table is used to explore an analytic report of a certain task or user. Those details button in trainee’s table will jump to the same page as the “Analytic Details for User”. Therefore, in the following part, we only talk about the task analytic details page. The content of the task analytic page includes:

Content 1: success rate of navigation tasks $\boldsymbol{ST_i}$

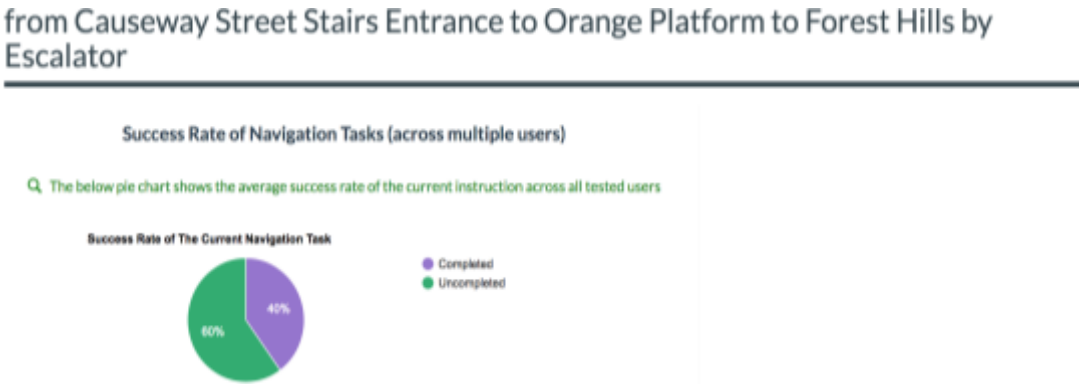

Figure 6.8 Success rate of navigation tasks

Content 2: subtask efficiency $\boldsymbol{EST_k}$

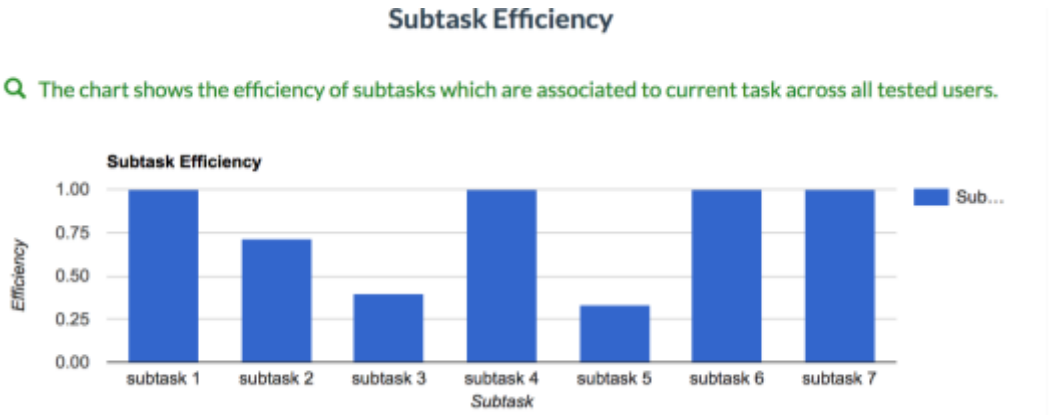

Figure 6.9 Subtask efficiency

6.3 Analytic Page for Developer

Objective: The developer focuses on improving the instructions. Thus, this section allows developer to diagnose issues with the instructions and address them accordingly.

Two types of dashboards have been developed in this section. The first analytics dashboard is Blueprint Analytics across all trainees, the second dashboard uses pie charts to visualize the Population Distribution.

Content 1: Blueprint visual analytics (Figure 6-10)

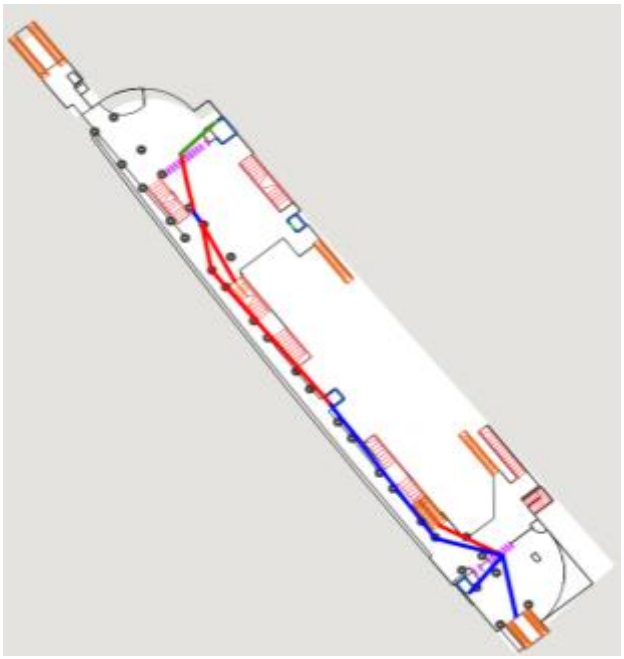

Figure 6.10 Blueprint Visual Analytics across all Trainees

Figure 6-10 depicts the blueprint visual analytics result across all trainees. Each subtask is represented by a line interconnecting the adjacent landmarks. Subtasks with 100 % efficiency are depicted in green lines, 60-99% in blue lines and the rest in red lines. Developers should first pay attention to the red lines (subtasks) which require further investigation and subsequent improvement of the navigation instructions.

It is important to mention that these results are also influenced by factors such as age, computer skills, and navigation ability.

Content 2: Population distribution (Figure 6-11)

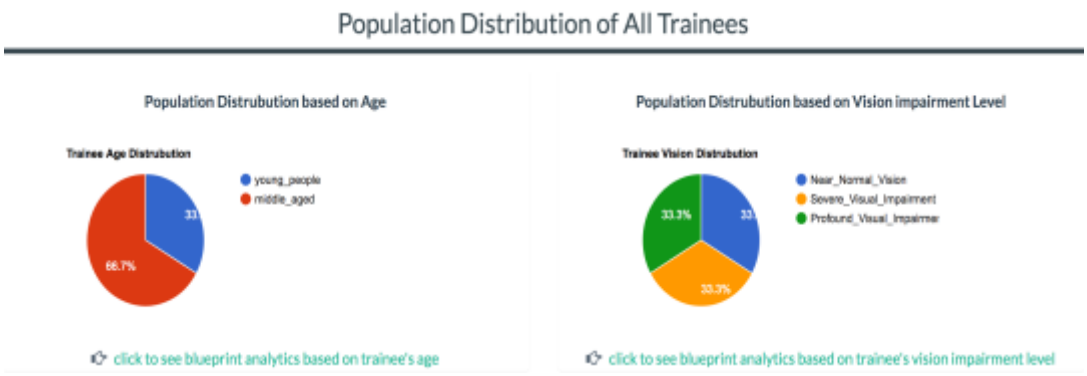

Figure 6.11 Population Distribution

Figure 6-11 depicts the population distribution across all trainees based on different attributes such as: age, vision impairment level, computer familiarity, and navigation skills. The developer can obtain further information about blueprint analytics for specific trainees' attributes by clicking on the green link (see Content 3 below).

Content 3: Blueprint Analytics for specific trainee population attributes

Figure 6.12 Blueprint Analytics for different Vision Impairment Levels

Figure 6-12 depicts the blueprint visual analytics for different visual impairment levels. The developer may conclude that the instructions on specific links/subtasks may need to be modified to meet the requirements of a specific population with a given visual impairment.

CHAPTER 7

CONCLUSION AND FUTURE WORK

In this paper, we introduced a virtual reality training system that enables a person with disability to perform pre-travel training. We have also developed data analytics and visualization tools that will allow the trainers to follow the progress of the trainees as well as inform the developers on the necessary changes in the system design. Our next step is to test the system with human subjects.

BIBLIOGRAPHY

[1] J.Scheer, T. Kroll, M.T.Neri, P.Beatty, “Access barriers for persons with disabilities: the consumer’s perspective”, Journal of Disability Policy Studies, vol. 13, no. 6, pp. 221-230, 2003.

[2] K. Choi, P. Wong, W. Chung , “Using computer-assisted method to teach children with intellectual disabilities handwashing skills”. Disability and Rehabilitation: Assistive Technology, vol. 7, no. 6, pp. 507–516, 2012.

[3] P. Standen, and D. Brown, “Virtual reality in the rehabilitation of people with intellectual disabilities”, Cyberpsychology & Behavior, vol. 8, no. 3, pp. 272–282, 2005.

[4] “Game Engine,” Unity, accessed on Jul. 20, 2017. [Online]. Available: https://unity3d.com/

[5] A. Ganz, S. R. Gandhi, J. Schafer, T. Singh, E. Puleo, G. Mullett, and C. Wilson,“PERCEPT: Indoor navigation for the blind and visually impaired,” in Proc. IEEE. Int. Conf. Eng. in Medicine and Biology Soc., Boston, MA, USA, Aug. 2011, pp. 856-859.

[6] A. Ganz, J. Schafer, E. Puleo, C. Wilson, and M. Robertson, “Quantitative and qualitative evaluation of PERCEPT indoor navigation system for visually impaired users,” in Proc. IEEE. Int. Conf. Eng. in Medicine and Biology Soc., San Diego, CA, USA, Aug. 2012, pp. 5815-5818.

[7] A. Ganz, J. Schafer, S. Gandhi, E. Puleo, C. Wilson, and M. Robertson, “PERCEPT indoor navigation system for the blind and visually impaired: architecture and experimentation,” Int. J. of Telemedicine and Applicat., vol. 2012, pp. 1–12, 2012.

[8] A. Ganz, J. M. Schafer, Y. Tao, C. Wilson, and M. Robertson, “PERCEPT-II: Smartphone based indoor navigation system for the blind,” in Proc. IEEE. Int. Conf. Eng. in Medicine and Biology Soc., Chicago, IL, USA, Aug. 2014, pp. 3662-3665.

[9] A. Ganz, J. Schafer, Y. Tao, L. Haile, C Sanderson, C. Wilson, M. Robertson, "PERCEPT based interactive wayfinding for visually impaired users in subways", in Proc. 14th. Int. Conf. Technology and Persons with Disabilities, San Diego, CA, USA, Mar. 2015.

[10] Tao and A. Ganz, “Scalable and vision free user interface approaches for indoor navigation systems for the visually impaired,” M.S. thesis, ECE Dept., UMass Amherst, Amherst, MA, 2015.

[11] Sánchez, Jaime, Matías Espinoza, Marcia de Borba Campos, and Lotfi B. Merabet. "Enhancing orientation and mobility skills in learners who are blind through video gaming." In Proceedings of the 9th ACM Conference on Creativity & Cognition, pp. 353-356. ACM, 2013.

[12] Navid Fallah. “AudioNav: A Mixed Reality Navigation System for Individuals Who Are Visually Impaired”. ACM SIGACCESS Accessibility and Computing, 24-27.

[13] Mandelbaum S.; Levy-Tzedek S.; Chebat D.-R.; Amedi A. “Vision-deprived Virtual navigation patterns using depth cues & the effect of extended sensory range”. In Conference on Human Factors in Computing Systems - Proceedings, (2014 01 01): 1231-1236