# Democratizing Mobilities Design Tools Using Virtual Reality

## Aiding Designers Simulate Crowd Flow on Rejsekort Validator Placement in Copenhagen's Nørreport Metro Station

Ivan David Tovar Siso, Kasper Amstrup Jensen, Stefan Nordborg Eriksen

Medialogy Master Thesis - May 2022

# Democratizing Mobilities Design Tools Using Virtual Reality

## Aiding Designers Simulate Crowd Flow on Rejsekort Validator Placement in Copenhagen's Nørreport Metro Station

Ivan David Tovar Siso, Kasper Amstrup Jensen, Stefan Nordborg Eriksen

Medialogy Master Thesis - May 2022

MED-10 Master Thesis

Democratizing Mobilities Design Tools Using Virtual Reality Spring Semester 2022

Authors: Ivan David Tovar Siso, Kasper Amstrup Jensen, Stefan Nordborg Eriksen
AAU supervisor: Claus Brøndegaard Madsen

Ivan David Tovar Siso
<itovar17@student.aau.dk>

Kasper Amstrup Jensen
<kaje17@student.aau.dk>

Stefan Nordborg Eriksen
<sner17@student.aau.dk>

# Contents

# Terminology

This chapter provides an overview of the terminology which is used throughout the report. The definitions provided here are made based on the use for this specific project.

| Term | Definition |
|---|---|
| Democratizing | Making the tools for Mobilities Design more available and easy to use for people without technical programming knowledge. |
| Simulation | An artificial representation of problems or events used to mimic the real life counter part. |
| Affordance | The possibilities offered by everything with which you can interact. |
| Feed-Forward | A signal that implies the affordances of anything interactable. |
| Signifier | A form of sign that explains the affordances of anything interactable. |
| AI | Artificial Intelligence. The simulation of intelligence, generated by computer systems. |
| User | A person either currently using or intended to use the application. |
| Agents | Any actor or entity that interacts with the environment. For this specific project it is mostly referring to computer generated silhouettes, roughly mimicking human shape, with simulated behaviour for traversing the metro. |
| Entity | The single independent existence of a thing apart from other things. |
| Spawn | The act of instantiating or placing new objects into the virtual environment. |
| VR | Virtual Reality. A computer generated simulation of a three dimensional space or environment, within which a person can interact by using special electronic equipment, such as an HMD and a pair of controllers. |
| VE | Virtual Environment. A computer generated three dimensional model representing a realistic or stylized environment in which a user can interact, for example by using VR. |
| HMD | Head Mounted Display. A monitor that can be worn on the head, showing the virtual environment from a real life perspective through a pair of goggles. Often referred to as VR headset. |
| Oculus Quest 2 | A specific brand of VR headset. Used in this project as the preferred model for evaluation. |
| Immersion | The sensation of tricking the senses to believing that you are in another place by providing visual-, auditory- and haptic feedback. |
| Presence | The sense of feeling that you are part of an artificial world by being engaged in the world or the narrative. |
| Cybersickness | A feeling of nausea, dizziness or the likes, caused by the mismatch between the feeling of movement in the VE while staying still in real life. |
| Haptic | A simulated sensation of touch in a virtual world provided to the user through forces such as vibrations or motion. |
| Avatar | Virtual character representing the user in the VE and in our case, controlled by the user. |

| Term | Definition |
|---|---|
| Iteration | A repetition of a process done in order to further evolve or build upon the original design. |
| Prototype | A proof of concept, or early example of an idea used to explain or showcase the concept of a larger project. |
| Metroselskabet | A Danish company in charge of running the metro stations in Denmark, and collaborators on this project. |
| Experts | Users with better prerequisites for judging the prototype, based on personal experience within the given field. For this project, the experts was mainly gathered from Metroselskabet, as they knew how the metro stations function. |
| Concourse Level | The name of the upper level inside Nørreport metro station. |
| Platform Level | The name of the lower level where the trains arrive inside Nørreport metro station. |
| Diegetic | Existing or occurring within the simulation, and abiding to the laws of the simulated world or environment. |
| UI/GUI | User Interface/Graphic User Interface. A form of control panel with which a user can interact from within the simulation. |
| Logging data | The storing of different kinds of data during the simulation, such as agent position, ID, etc.. |
| Collider | An invisible field surrounding objects in the virtual environment that enables collision detection. |
| Trigger | A collider that objects can pass through, which is capable of executing code as another collider enters or exits its perimeter. |
| Locomotion | Refers to the specific method of user movement within the simulation. |
| Raycasting | The visual representation of a 3D vector in virtual space from point of origin (the user's hand in the VE) to point of collision. Used in this project for selecting and placing validators and for aiming where to teleport. |
| Teleportation | A type of locomotion. The act of moving instantaneous from one place to another. |
| Steering | A type of locomotion. The act of moving continuously through a space at a constant pace. |
| BT | Behavior Trees. A form of AI based on nodes which are arranged in a tree like structure. |
| Zones | Areas of interest scattered across the traversable space, used by the agents as target locations when navigating, and for logging of data. |
| Utility Function | A mathematical function, in which the "usefulness" of an item or action can be scored, and the resulting utility score can be used to decide between multiple options. The concept comes from Utility Theory. |
| Navigation | A form of simulated behavior used by agents to traverse around the VE. |
| Baking | The process of saving data related to a 3D model into a texture file. Used in this project for generating lighting and the NavMesh. |
| NavMesh | A computer generated traversable area used by the agents for navigation. |
| Validators | Check in-/check out stands used by passengers carrying a "rejsekort". |
| Rejsekort | A card containing credit, used by passengers to gain permission for riding the trains. |

# Abstract

The term mobilities design is broadly used throughout the design industry as a way of streamlining the flow of people traversing through urban areas. In this digital age, many of the tools used in the workflow of urban designers, to visualize the potential effects of different design choices are already well established, with tools such as Legion or PTV Viswalk. However, as the concept of virtual reality is still gaining popularity and is finding its way into many different industries, an opportunity was seen in trying to incorporate the many qualities of virtual reality into the design process of urban designers. Virtual reality allows for a real-life perspective which cannot be gained on a normal 2D monitor. Furthermore, having the agents react to changes in real-time, removes the chore of having to restart the simulation every time a change in the setup of the simulation is made. Instead, the effects of changes to the virtual environment are immediately shown inside the simulation.

This project takes its point of origin in how virtual reality can support designers in visualizing the flow of a crowd traversing through an urban area. As a specific use-case, a collaboration was formed with Metroselskabet in Copenhagen, Denmark, with the specific task of visualizing the effect from rearranging the validators at Nørreport metro station.

This project consists of three main elements which make up the final prototype. First is crowd simulation, utilizing Unity's NavMesh system for pathfinding, and behavior trees for scripting the agent behavior. Secondly, interaction is used in order to allow users to navigate and interact in the prototype. Finally, a virtual environment is created to bind it all together.

The prototype was evaluated using an expert heuristic evaluation along with Cybersickness and presence questionnaires. The participants in this evaluation, were four experts from Metroselskabet and one virtual reality expert from Aalborg University. The results show that although some interactive problems remain to be corrected, the prototype can be considered a sufficient and valid tool for performing design tasks in the given virtual environment. During the evaluation, some experts at Metroselskabet found agents in the environment to be disruptive and distracting, and noted discomfort from being surrounded by virtual crowds. They also reported high levels of induced Cybersickness when evaluating the prototype with steering locomotion, but were very comfortable with teleportation. They also felt the fidelity of the graphical elements of virtual environment were sufficiently representative of its counterpart in the real world.

# Chapter 1

# Introduction to the Problem Space

In the grand design of urban society, there is a field of research called mobilities design, which is there to help study what elements are considered when designing urban areas that naturally guide people to traverse smoothly through their daily life. The study of mobilities design is significant as it influences many parts of society, from smaller use-cases such as the flow of people through a space and daily transportation, to large-scale movements of capital and information across the world. In the ever-growing society that we live in, where cities, and especially populations are growing day by day, it becomes even more important to abide by the principles of mobilities design in the workflow of urban designers [22, 26, 30, 32].
One of the challenges designers are facing when incorporating mobilities design, is managing crowd flow. When designing a public area, it can be hard to guess how it will be traversed in the end, as peoples behavior is hard to predict, and many different factors such as culture or gender, can have a big impact. In the digital age we currently live in, it is no surprise that designers have tried to alleviate this problem using digital aides. These aides are often in the form of computer simulations such as Legion, PTV Vissim or PTV Viswalk, which are described in section 1.4. In short, these aides are used to simulate crowds or traffic, with the purpose of visualising the flow of people traversing the space. However, at the time of this project, no aides have yet been observed which can manage to deliver a real-time Virtual Reality(VR) simulation in which the designer can modify the environment, and see immediate results.

As such, the main focus of this project will be to develop a proof of concept of such a tool, using crowd simulation centered in mobilities design to help designers visualize crowd flow in real-time. The application will make use of VR in order to immerse the designer in a Virtual Environment(VE) in which the simulation can be spectated from a real-life perspective.
A specific use case was decided upon for the creation of a prototype, in collaboration with Metroselskabet(MS). They are in charge of running the metro stations of Copenhagen, Denmark. Through multiple meetings and discussions about the challenges occurring in the metro, the problem statement was formulated as follows:

*"How in-situ VR crowd simulations can support designers in visualizing the flow of a crowd when rearranging the validators of the Nørreport Metro Station in a VE"*

This problem statement meant that the solution could be tested on the basis of a real location, Nørreport Metro Station in Copenhagen, and with a real design problem in mind, the placement of validators. This was stated as a current problem from the people at MS because it was affecting the flow of passengers, and where a bad design could result in congestion, meaning that it was a relevant target for the project.

## 1.1 Initial Requirements

Having a particular use-case, makes the grand and abstract idea more trackable, and so the prototype described throughout this report, was designed with five requirement specifications in mind:

- The application should simulate a crowd of pedestrians moving through Nørreport Metro Station.
- The user should be able to experience the application in VR.
- The simulated pedestrians should interact with validators (both check-in and check-out).
- The user should be able to place, move, and remove validators in the VE.
- As requested from MS, they, as user, should be able to extract data from the simulation. This data should afford the visualization of crowd flow.

These requirements were made to guide the design through an iterative process towards the final iteration which can be seen in figure 1.1.

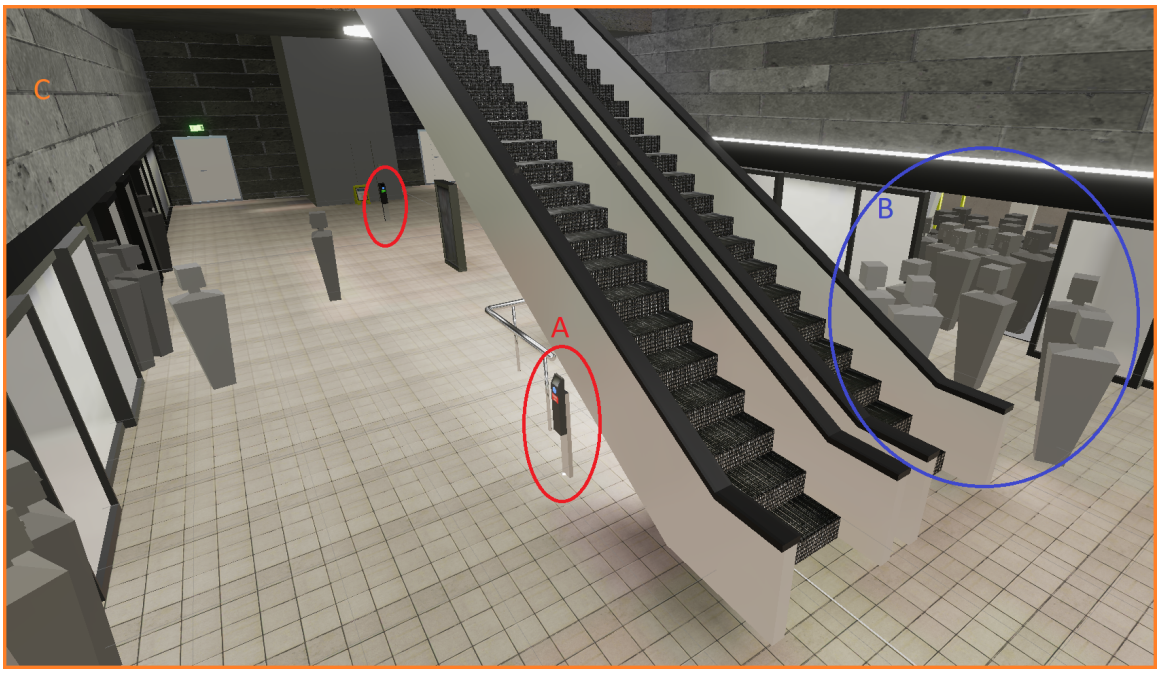



**Figure 1.1:** Main elements of the application. "A": the validators. "B": the crowd simulation. "C": the VE.

The main elements of the prototype will be presented and discussed throughout this report. However, a short overview will be provided here in order to present the main topics and challenges of the project.

## 1.2 Main Elements of the Final Prototype

In order to understand the main components of the final prototype covered by this project, we divide the components into three main components, some of which have sub-components themselves.

### 1.2.1 The "Crowd Simulation"

First of all the prototype uses *crowd simulation* to simulate the flow of passengers in the metro station. These simulated passengers will so forth be referred to as agents and are represented as the gray box silhouettes (see figure 1.1.B). The crowd simulation consists of the following two **sub-elements**:

- The first being **Unity's NavMesh system**, which is the main component for handling the pathfinding and steering of the agents. This means that it will create the area which can be traversed by the agents while also making sure that the agents will not bump into obstacles or each other.
- For the second sub-element, an **artificial intelligence(AI)** model known as **behavior trees(BT)** was used to simulate the behaviour and program the actions afforded to the agents in the metro station. This can be behaviour that affords interacting with the VE, e.g. using the validators, escalators, elevators, and boarding or unboarding the trains.

### 1.2.2 The Interaction

Another main element of the prototype is the *interaction* part. The interaction is what allows a person or designer to use the prototype, in our case, by creating a virtual representation of that person through the use of VR. The person using the prototype will so forth be referred to as the user. VR can afford them to be present inside the simulation, where they will be able to move around and perform specific design tasks tailored to the chosen scenario. A **sub-element** of the interaction element is **locomotion**. This is what allows the user to traverse the environment, with teleportation chosen as the preferred type of locomotion used for this prototype. This affords the user to teleport to any visible area which is marked as traversable.

In the current version of the application, the user can create, delete, move or rotate validators, as well as log data from the agents to an external CSV file for further analysis. All these actions are presented through a holographic wristwatch menu as seen in figure 1.2, which acts as the diegetic GUI to avoid breaking immersion.

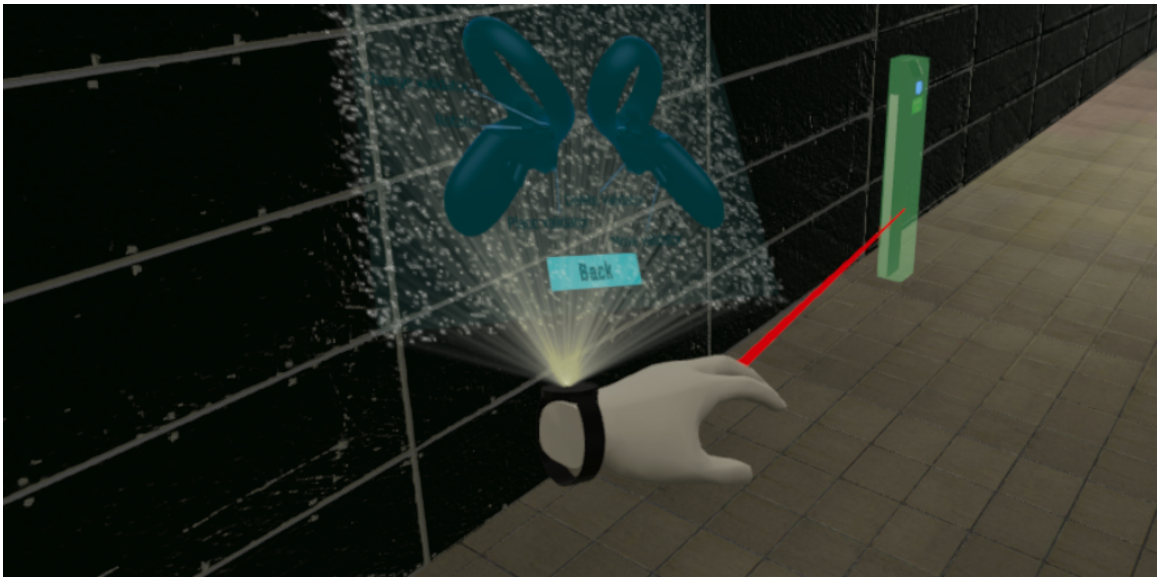



**Figure 1.2:** Embodied diegetic GUI for the user to interact with the application.

### 1.2.3 The Virtual Environment

For the crowd simulation and the interaction to work, the prototype contains a 3D model representation of Nørreport Metro Station, which acts as the *VE* in which the simulation takes place (see figure 1.1.C). This VE was modelled after the blueprints of the real Nørreport Metro Station, as well as multiple reference photos taken from the actual location. The VE is relevant as it provides the user with a real scale environment, seen from a first person point of view, allowing for a more realistic experience, as well as providing better conditions for performing design tasks.

## 1.3 Addressing the Multidisciplinary Aspect of this Project

This section presents research from different fields that contributed to the development of the application. In short, some of these elements are related to each other, and through this project, we attempted to merge them together as a proof of concept for using VR for aiding mobilities design.

Summarizing, the research fields described below served as building blocks for the main elements previously mentioned in section 1.2. They are presented in the following manner:

- **AI** and **Simulations** contributed so that the user could experience the crowd and liveliness of the environment for which they are designing. In the case of Nørreport Metro Station, we see this application as an opportunity to "rent" out the whole metro station as a **simulated VE**; its entrances, elevators, trains, and connection to the main train station without having to disrupt the commuters daily usage of this space. **AI** allows this project to simulate the crowd, so that the user can see how their design changes of positioning validators, affect the crowd flow in real-time.
- **VR** affords the user to feel present in a virtual environment that can feel more natural to those presented in a 2D screen. State of the art that affords crowd flow simulations, by the time this project took place, did not afford the user immersing themselves in an environment that is to-scale (1:1), and require the designer to go through a steep learning curve to use their solutions, as explained in section 1.4.
- **Mobilities and immobilities design** provide a theoretical need for the inclusion of new technologies into the design processes. Mobilities design encompasses culture in the way the urban spaces are designed. Technological changes being a part of culture are necessary to keep mobilities design techniques relevant.

These elements make the application multidisciplinary in nature, based on the requirements mentioned in section 1.1. Simulating a crowd moving through a VE requires AI to be responsive to the environment; the validators are a part of the environment, and the crowd interacting with the validators can cause clusters of people, which could have a dampening effect on the flow. Designing the distribution in space, and placing these validators, fall into the category of mobilities (and immobilities) design.

In this project we use Computer Graphics Imagery (CGI) in order to create a realistic as possible representation of reality from a visual stand point while keeping performance under consideration. Similarly, we use Interaction principles to make for a meaningful experience for the user, such as locomotion methods used in VR.

Detailed information on the theory and authors that shaped the research of Mobilities design, AI, simulations, and VR can be found in appendix A.

## 1.4 State of the Art

The State of the Art for crowd flow simulations by the time this project took place, had not been explored in VR. However, there were several different models used in the urban design, as well as the development & construction industry. Some of those existing tools are *Legion*, *PTV Vissim*, and *PTV Viswalk*, which are described in Appendix A under section A.5.

In short, these state of the art applications provide very thorough and detailed solutions for visualizing flow. However, while they might be detailed, they also have a steep learning curve, as they include many different features which can result in an overcrowded and unmanageable UI. Furthermore, some of these applications are better suited for big scale simulations, meaning that for something like the traffic in a metro station, the data would be less viable.

Another thing to note, is that these applications do not provide any real-time feedback, which means that the user would have to restart the simulation every time they want to make an adjustment. Furthermore, adding VR to the solution could give the users the opportunity to see the results from a real world perspective, which might provide the urban designers with a new point of view.

# Chapter 2

# Design

This chapter presents the process and considerations for the final iteration of the application. With a clear use-case, and a set of requirements that served as guidelines for the research, this application was designed as a proof of concept for the use of VR in the process of designing mobilities design. In order to aid the design and implementation process, a high level abstraction of the application was created as shown in figure 2.1. For details on how the previous iterations of this application were designed and tested, please refer to appendix B.

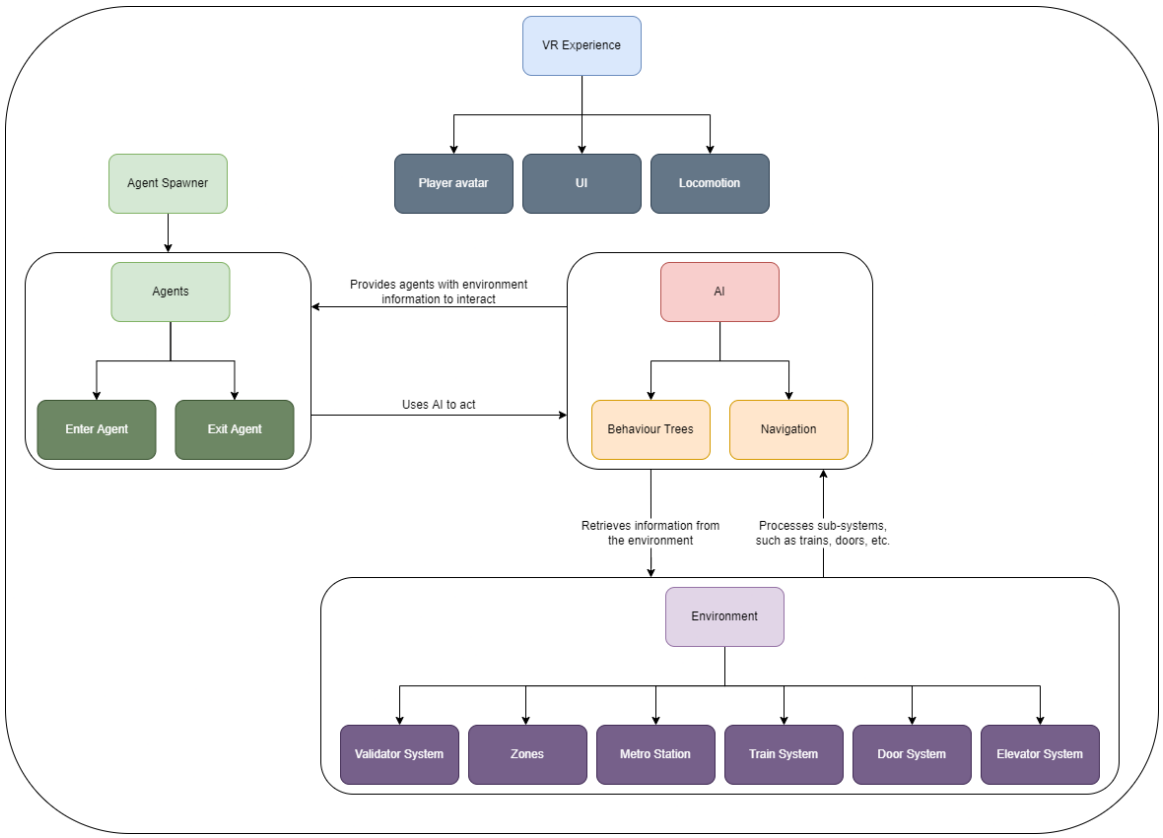



**Figure 2.1:** High level abstraction of the system, to help the design and development process

A quick overview of figure 2.1 can be interpreted as follows:

- The **agents spawner** creates exit and enter agents in the VE from every entrance of the metro

station. The spawner uses data provided by MS to populate the VE from the entrances that are the most frequented in the peak hours of the metro station. The distribution function for spawning agents is presented in section 3.4.

- **Enter** and **exit** agent behaviours were modelled using BTs, enabling them to act quickly to changes in the VE. An example of this, could be that when the trains are arriving to the platform, agents would hurry to catch it.
- **Enter** and **exit** agents use Unity's NavMesh system to navigate the VE.
- The **AI** retrives information from the environment, and provides the agents with that information, so that they can interact with the VE.
- The **environment** contains and processes all sub-systems of the simulated metro station. Examples of these sub-systems can be, the arrival of the trains and its relationship with the doors opening and closing when the train leaves. This information is retrieved by the AI module in order to tell the agents when they can board the trains (agent's can only board the train when the doors are open).
- The **VR experience** module affords the user an interface to exist in the VE, as an embodiment of self through the player avatar. The UI sub-module allows the user to extend the range of affordances they are capable of in the VE. Finally, locomotion allows them to traverse the VE.

## 2.1 Designing the Final Prototype

This section presents the considerations and process for designing the final iteration of the application. During a field trip to MS, an early version of the application was presented to the experts. Additionally, a tour of the station was provided by MS to observe passenger behaviour that could be modelled in the simulation.

### 2.1.1 From the Field Trip

As the experts at MS tried the early version of the application for the first time, they provided us with feedback that served as opportunities to enhance the user experience and realism of the simulation. Some of the most important elements to mention are listed in section 2.2.

The functionality of this early version contained steering-based locomotion, agents traversing the environment, as well as boarding and leaving the train, and agents were visualized as capsules for simplicity. Emphasis was put on the behavior of the agents, as the experts showed interest in this topic in particular, and could provide valuable inside knowledge of some typical behavior that they would like to see in the prototype. This was behavior such as places people tend to gather, leaning up against escalators and walls, people traveling in groups and the use of the elevator.

#### On Location Observations

During the tour of the station, it was observed that Nørreport Metro Station is not just packed during rush hour, but with a varying flow of people arriving steadily from the surface, and additionally with trains every 2.5-5 minutes. This means that train arrivals result in bursts of people needing to navigate what becomes a cramped space. This is extra chaotic when trains arrive on both platforms at roughly the same time. Furthermore, the influx of passengers that are boarding the trains is more stable, with occasional waves of people coming in from the larger Nørreport

Station which is connected to the metro through a tunnel.

Especially the large groupings are a concern, as many people tend to wait at the first door they come across to wait to board the metro. This is especially true for the doors closest to the passageway that connects the metro station with the rest of Nørreport Station, where passengers can connect and transfer with S-trains, regional trains, and long-distance trains. In order to circumvent this issue, only one out of the four escalators, connecting the lower platform level with the upper concourse level, have been directed downwards. This helps to guide passengers, coming in from the street, to move towards the far end of the metro, and thereby taking more advantage of the space. This also helps to quickly move passengers out of the metro, since there are three escalators moving upwards, which especially is a great help at times when the two trains arrive simultaneously.

In addition to gathering information, time was also spend on taking pictures and videos of the metro in order to document all the visual details, such that the virtual environment could be updated to look as closely as possible to the real environment. These pictures were matched to the areas in the blueprints provided by MS. The concourse level blueprints, matched with the pictures taken from the field trip is shown in figure 2.7.

## 2.2 MoSCoW - Requirement Specification

With all the new information from the trip fresh in memory, it was decided to make use of the MoSCoW method in order to design the features to implement in the application. This was to sort out the priorities for all the features which had been discussed, in relation to the remaining time period for the project.

### 2.2.1 Must Have

As such, it was decided that the prototype must have:

- Teleportation locomotion.
- Updated Zones that match feedback from MS.
- Chance for validators to fail.
- Validator sounds.
- Elevator usage.
- Passengers in a rush if the train is arriving/leaving and they are relatively close.

#### Locomotion

The prototype showcased to MS during this trip used steering locomotion. Here, it was discovered that this specific type of locomotion turned out to cause nausea for the testers involved. Based on thorough research and considerations of locomotion in VR, as explained in section 3.1, it was ultimately decided to change to a teleportation movement system instead.

#### Zones

According to the feedback from the showcasing in Copenhagen, and the observations from the field trip, the waiting zones for the agents also needed an update. The observations showed that passengers tend to accumulate along the escalators and walls, instead of taking advantage of the open areas. This behavior is due to the fact that people like to lean against railings and walls to conserve energy, without thinking about the consequences of blocking up passage ways, according to the stewards at the metro. This lead to the decision of updating the current zones in the prototype.

#### Validators

The decision of adding a chance for the validators to fail came in relation to the decision of adding sound effects to the prototype. The additional modality of sound would further improve the feeling of immersion for the users, while also making the behavior of the agents more identifiable. These sounds would help indicate the state of actions performed by agents without having to use animations. It was thereby decided that since sound effects would be added to the validators, they would also have to have a chance of failing, as this also is a common occurrence at the metro.

#### Elevators

During the tour through the metro at rush hour, it was discovered that the elevators play a big part in the flow of the passengers, as they are not only used by disabled or people with strollers, but where observed to be constantly at full capacity. Even outside of rush hour, some people proffered to use the elevator out of convenience, which led to the decision that they had to be part of the prototype.

It is important to note that due to time constraints, and the difficulty of the task, this was not accomplished.

#### Agents' Speed and Variety

Lastly, it was agreed upon to add a variation to the speed parameter, such that not all agents would be moving at the same speed. This was in order to mimic the passengers who are in a rush, and have to run in order to get on the train in time, which also was observed multiple times at the metro.

### 2.2.2 Should Have

Moving on, these requirements are the ones which were decided to be a should have:

- A way to visualize the data, f.x. heat maps of congestion.
- Enable extraction of data regarding average time spent in the metro for each agent, from spawn to despawn.
- Different types of agent's profiles to represent different types of passengers, f.x. young passenger could have a faster walking/running speed than passengers of advanced age.

- "Pro" passengers, which would represent the passengers who take no time to use the validators.
- Metro ambience, to make the simulation closer to reality.
- Train sounds.

#### Extraction of Data

On request from the experts at MS, data extraction from the simulation was needed in order to analyse the large scale crowd flow, since the user could only be in one place at a time when experiencing the simulation from a first person perspective. They proposed extracting data that could yield heat maps, showing the congested areas, as well as some statistical data regarding the time agents spent traversing the VE from spawn to despawn. Heatmaps can be a great way to visualize how the outcome of the validators affected the area in the metro, and how the validators may be used to direct people around in the metro. Moreover, having some sort of statistical output, such as an average time spend in the metro for the agents, should prove valuable to have, as it may show how effective the validator set up might be in comparison to other setups, which is why these was chosen as a should have.

#### Different Types of Passengers

The different types of passengers were considered according to the information provided by MS, who informed us about the multiple types of tickets and access cards, which should have an effect on the flow of the passengers. This, along with the fact that it may be relatively easy to implement, let this feature be a should have. Furthermore, in relation to different types of passengers, the "pro" passengers were also considered. These should resemble passengers who might be able to use the validators without any delay time.

#### Sound

In the end, the ambience of the metro, as well as the sound of the trains, should be of great value for immersing the user in the experience, as well as adding an extra layer of realism to the simulation.

### 2.2.3 Could Have

These are the requirements chosen as a could have:

- Humanoid agents, to update the visuals of agents from being capsules, to being something more human-like as shown in figure 2.2. This serves as a partial implementation to what would be an iterative process of making the agents more human-like. An example of this would be adding walking animations.
- Escalator customs, f.x. holding to the right and walking on the left side of the escalators.
- Authority agents, to represent the stewards working at the platform level that indicate people to move along.
- Passenger groups, to simulate families or group of friends travelling together who have a different behaviour than isolated agents.

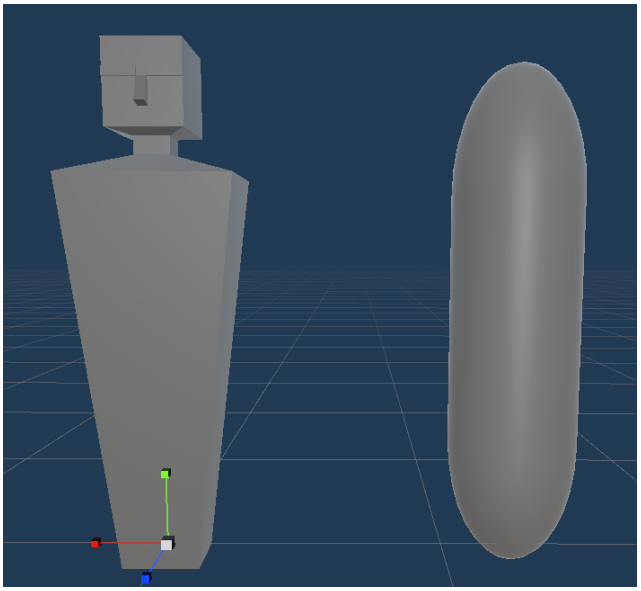


**Figure 2.2:** Humanizing the agents, while considering the polygon count for performance

- More realistic movement, f.x. not walking so close to the walls all the time, and not always walking on a straight line.

The requirements listed above were all things that could have potentially heightened realism of the simulation. With the exception of the humanoid agents, these were considered to have an impact on the crowd flow. However, considering the time frame, it was ultimately decided that these were all things that might require a more thorough and time consuming implementation, while simultaneously not adding the most value in comparison to the previously mentioned requirements.

### 2.2.4 Will-Not Have

Lastly, these were the requirements that was ultimately decided not to be part of the scope of this specific project:

- Tourists.
- Leaning against surfaces.
- Announcements over the speakers.

### 2.2.5 Tourists

Tourists were believed, by the stewards, to be one of the biggest challenges in the metro, as they had no knowledge about danish customs, and so would often go against regulations, or in other ways accidentally hinder the crowd flow in the metro. However, the challenge of implementing such behavior was still considered to be too high of a risk, and was therefore saved for potential future work in the project.

### 2.2.6 Leaning against Surfaces and Announcements

Leaning against surfaces, as well as announcements over the speaker, were also saved for future work, as it was believed that the gain of implementing these was too minuscule to be worth the effort and time when comparing to other features of higher priority.

## 2.3 Designing Unpredictable but Rational Agents

This section presents the process to transition from the MoSCoW model and the requirement specification to the actual design of the agents. Some of the most important traits of the agents, such as the use of validators and the need for queuing when a validator is occupied are discussed.

### 2.3.1 Usage of Validators and Queuing

In earlier iterations, agents entering the station would traverse to the platform, navigate to a random zone, wait for a train to arrive, then attempt to enter the train in time. Agents exiting were even simpler, traversing directly towards a random exit. There were plenty of details to be refined here, many of them connecting to giving the agents a heightened sense of individuality and variability.

First and foremost was the usage of validators. A majority of agents must seek out a validator before boarding a metro wagon or exiting the station. But not all agents should have to use validators, as some passengers posses some form of pendler-card, or would purchase tickets through an app. A prototype variation of this behaviour was in the works by the time the initial presentation took place, but developing it demonstrated the need for handling multiple agents attempting to use validators at the same time, especially when they targeted the same validator. For the final iteration agents need to choose between a queuing-like behaviour, or re-navigating to another nearby, available validator, if their original target is occupied. Figure 2.3 shows an example of two enter agents walking towards a validator they have targeted for check-in (indicated by a green line from the agents' position to the validator). When the validator becomes occupied by one agent, the second agent can decide to queue, or move towards another validator to satisfy the need for checking-in.

This queuing behaviour simulates the perception of agents using the validators and their availability to other agents that could be in need of checking-in/-out. Agents that are navigating towards their destination and have to check-in/out, were designed to have a "patience"-relationship towards having to queue to finalize the checking-in/out action, versus continuing in their path and delaying the action. Agents with pendler-tickets would not be in need of this queuing behaviour when nearby validators. The mathematical reasoning behind the queuing-patience trade-off can be found in section 3.5.2.

Validators in the real world occasionally fail to scan Rejsekort. This is sometimes at random, but at other times it is due to the user not holding the card close enough to the validator, pulling the card away too early, or trying to scan through a wallet or similar. This was modelled as part of the validator interaction with a fixed, but adjustable rate of 5%.

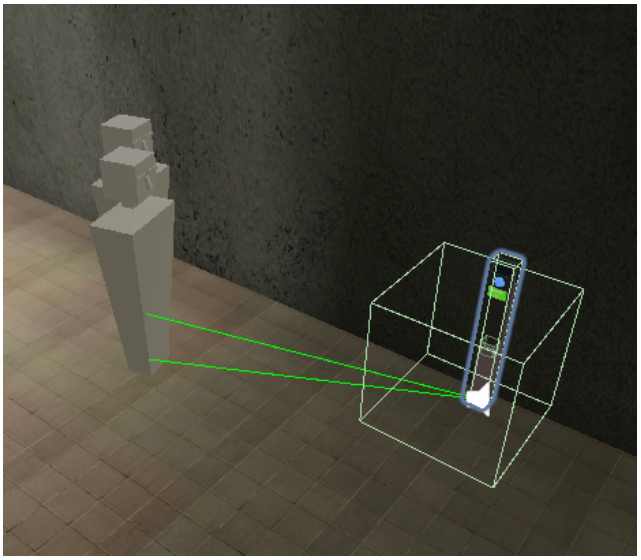


**Figure 2.3:** Two enter agents targeting the same validator for check-in. The closest agent will check-in and the second agent might queue or choose to use a different validator on the way.

### 2.3.2 Creating Two Different Agent Profiles

The motion of agents was also improved significantly. Instead of all having the same base speed, agents must have variations in speed. Not only can their base "walk"-speed have variations between agents, but agents could be able to change between walking and running. At Nørreport we observed several passengers in different stages of "hurry", at worst running down escalators to catch the train that was just about to leave. For this, scriptable objects containing agents stats were created.

Two initial profiles were created, one for "young" agents and one for "older" agents, where young agents would have a higher speed range. Figure 2.4 shows an example of the profile for young agents. Another consideration for the agents, where they might differ, could be the chance of acquiring a ticket through an app, or possessing a pendler-card; however, as no data regarding this was acquired, all agents were given the same chance to posses alternative valid ticketing of 30%.

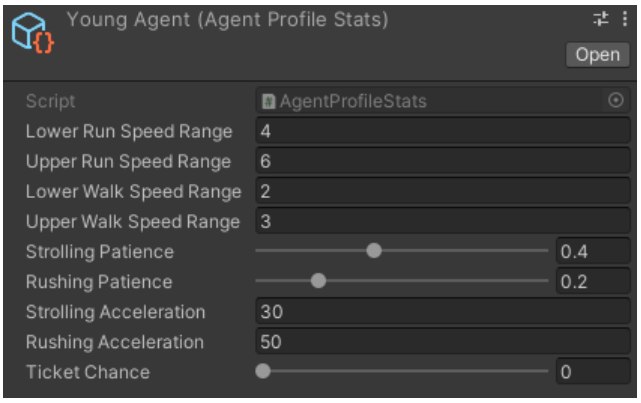



**Figure 2.4:** Young agent profile. At spawn, agents would be either young or older agents and their speed (and other stats) will differ depending on their profile.

### 2.3.3 Agents Targeting Zones "Conveniently"

Target zones were initially picked at random by agents. As noted from our observations at Nørreport (see section 2.1.1), passengers tended to move to the closest available gate, and to a smaller degree also towards something to lean against while waiting. Adding more logic to how and why a zone were chosen was a must, and it was modelled by adding weights to the waiting zones, where agents would be more likely to choose a zone located close to their point of entrance, depending on the amount of people already occupying that zone. The specific action of leaning against objects was not implemented, as it was assumed as less impactful for evaluating crowd flow.

Figure 2.5 shows the zones in platform one, in which the agents arriving to the platform (from the left of the image) would have a higher chance of choosing the zones closest to them, which in turn could cause clustering of people. This behaviour would match the observations from the field trip.

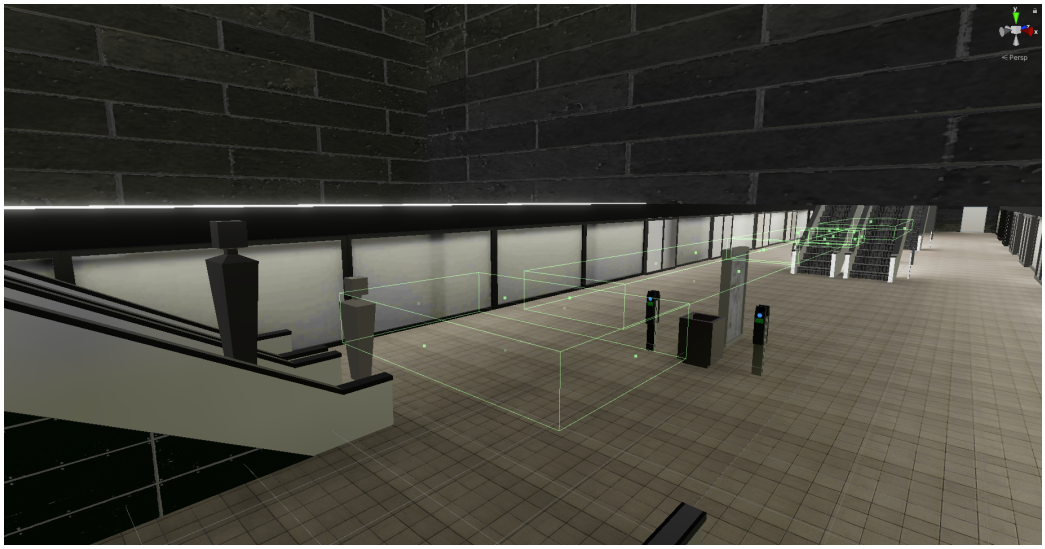


**Figure 2.5:** Sample zones in platform 1 where agents would have higher chance of choosing the closest zone to their point of arrival to the platform level.

The details on the mathematical implementation on how zones are chosen can be found in section 3.5.2.

## 2.4 Elements that were not Implemented due to Time Constraints

Although all these elements were considered, their particular implementation will not be a part of the scope of this project due to time constraints. These are some of the tradeoffs that could improve agent behaviour, for example for the pathing of the agents, but are very expensive/complicated to implement:

- Elevator usage: Even though this feature was not implemented, section 2.4.1 presents the plan we had to implement this very important aspect of the simulation, that we consider to largely affect the crowd flow of Nørreport Metro Station.
- Pathing away from walls and escalator customs.

### 2.4.1 Elevator Usage

An important environment-agent interaction that was not implemented due to time constraints, was the usage of elevators. During the tour described in section 2.1.1, we observed heavy usage of these by all sorts of passengers at Nørreport. Agents needed to be able to use elevators as navigation waypoints to traverse in and out of the metro station. With the limitations of the NavMesh system in mind, a strategy for implementing functional elevators would be to make them traversable in a similar way to how train access is managed, using doors with a NavMeshObstacle component, and broadcasting their state to listening agents. For the actual usage, we built reusable sub-behaviours, and create a model that, given the elevator's current status and individual agent opinion, estimates the utility of using an elevator so that an agent can chose between using the elevator, or traversing as usual. Figure 2.6 shows an example of the navigatable path for agents being cut when the doors are closed, and open when the doors are open.

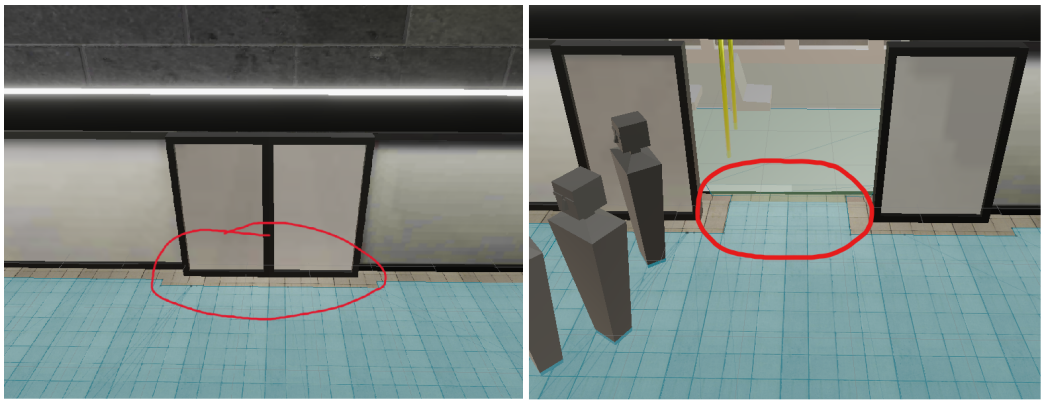


**Figure 2.6:** Environment-agent interaction were the path gets cut when the doors are closed, and vice-versa.

### 2.4.2 Pathing Away from Walls and Escalator Customs

The pathing of agents could be modified to make their navigation seem more realistic. Due to the nature of the A* pathfinding algorithm, agents end up following straight paths very mechanically, with a tendency to stay close to edges of the NavMesh, i.e. "hugging the wall". In the real world, people tend to move further away from the walls, and closer to the center of corridors. There are multiple ways to try and tackle this issue, but at varying costs, which may make it infeasible to implement within the project's time frame. One of the ways this could be tackled would be manually making the areas closer to the walls more costly to traverse.

Within the topic of agent navigation, escalators are a major feature of the Nørreport Metro station, but the prototype only contains static escalators, with agent limitations to directionality enforced through modifying the NavMesh area type, and putting restrictions on whether an agent could traverse it. They are otherwise not treated as special in any way, but in the real world, there are customs to using escalators. As an example, pedestrians stand to the right, leaving space for more busy people to walk through to the left. Enforcing these customs to the agents behavior could improve the realism of the simulation. While this was not explicitly programmed, this seemed to be a characteristic that emerged from the NavMeshAgent obstacle avoidance, and the varying speed at which they move towards the platform.

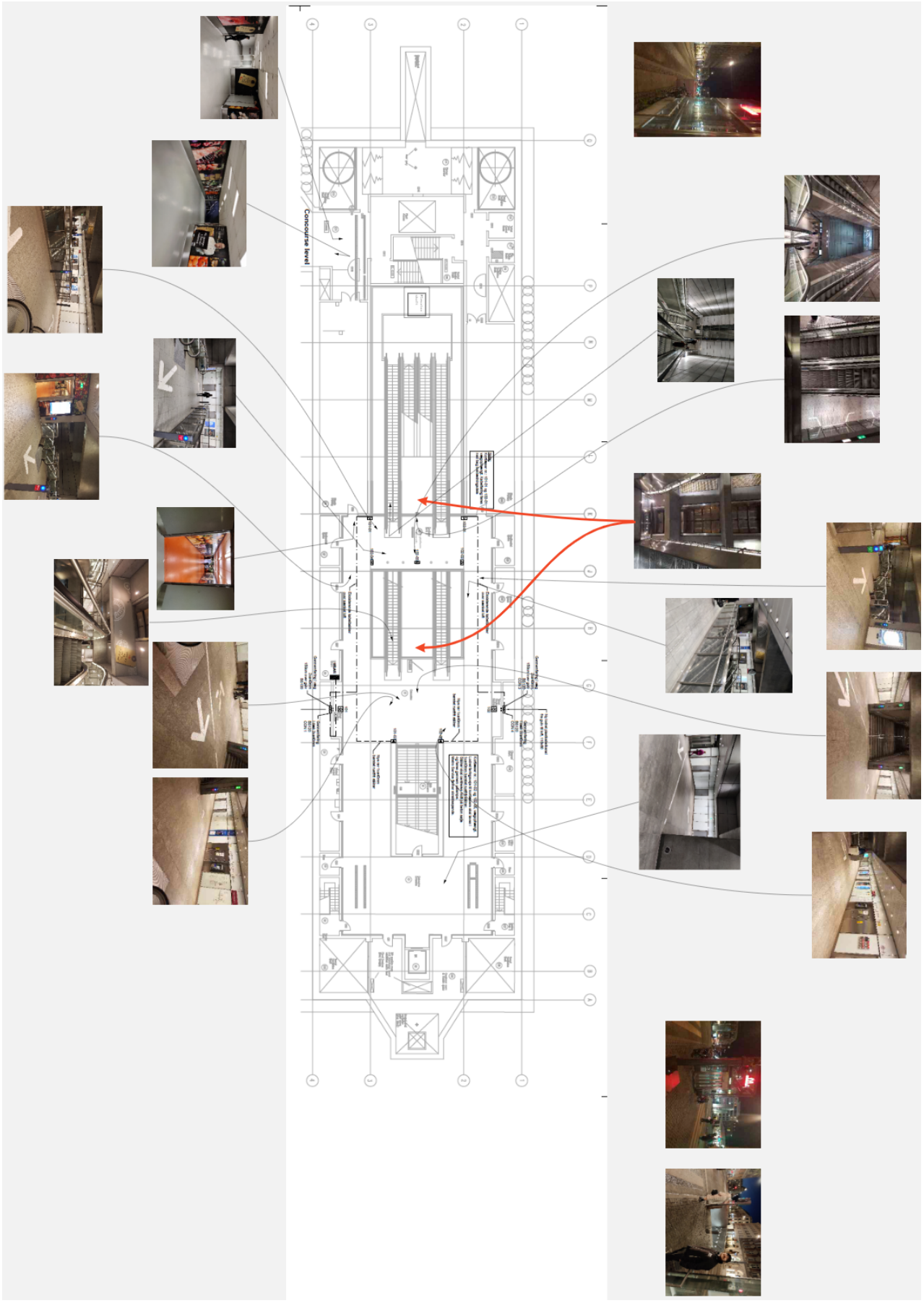



**Figure 2.7:** Concourse level blueprints provided by MS, matched with pictures taken during the stations' tour.

# Chapter 3

# Implementation and Technologies

This chapter presents the technological considerations for implementing the application, as well as a quick overview of the target hardware to run the application. For the implementation, Unity Engine [4] was used as the environment for development, and the target hardware to run the application in VR, was the Oculus Quest 2. Because within Unity, the input system used for setting up interactions was the XR Interaction Toolkit, the application could, theoretically, run in any VR device compatible with the XR Interaction Toolkit.

The considerations presented in this chapter follow the same structure as the high level abstraction reference of the project that was used in the design chapter (figure 2.1), thus, these considerations are divided into the following items:

- VR experience: Including the player avatar, UI, and locomotion.
- Agents: Here we explain why BTs are used over other AI techniques and technologies.
- Navigation: Also under the umbrella of the AI element of this project, but used to enhance the BTs interaction with the agents.
- Environment: Presented as a collection of elements that compose everything with which the agents and the user interact. These include: zones, validators, and other systems.

When referring to the VR experience in this chapter, we refer to the way the user senses the environment. The player avatar serves as a representation of the user's body in the VE, where the virtual hands of the user are where their hands would be in real life, as afforded by the Oculus Quest 2 tracking system. Additionally, the UI affords the user to have more options when interacting with the environment. Making this UI diegetic, was a choice to avoid breaking immersion during runtime.

A noteworthy element, in terms of concrete results that will be discussed in section 4.1.2, was the locomotion system. This was first tested with steering locomotion, and secondly with teleportation locomotion. During the tests with steering locomotion high levels of Cybersickness were reported, contrary to what was reported with teleportation locomotion.

## 3.1 Locomotion Systems in Virtual Reality

Locomotion has been a challenging field of study when dealing with VR as a technology. Many researches have extensively studied the differences between a variety of locomotion techniques, with the focus being on steering locomotion, and teleportation [2, 14, 15, 17, 19, 20, 42, 49]. These

studies provide an insight on the possible causes for Cybersickness, along with (in some cases) introducing newer techniques, such as Podoportation, which is a foot-based locomotion technique based on the principle that if most people move in real life with their feet, so should locomotion in VR be done with the feet [57]. From an interaction point of view, the well being of the user affects their experience in ways that could be experience breaking. Additionally, since locomotion is a part of the VR experience, some research points towards the effects of different techniques affecting presence.

In short, research points towards the following paradigms when studying two of the most prominent locomotion systems for VR experiences (steering and teleportation):

- Steering can increase presence, but it tends to increase Cybersickness.
- Teleportation tends to reduce Cybersickness, but the unnatural nature of its displacement of the user can decrease presence.

Early iterations of the application included a steering-based locomotion system, with the idea of increasing presence. The goal of this project was to simulate crowd flow in an environment, put the user in it, and have them experience it from a first person point of view. However, one of the experts had to go home after having experienced the application once, due to induced Cybersickness. All other experts also pointed out that the steering locomotion created discomfort associated with Cybersickness.

Among the reasons why steering locomotion can create Cybersickness, is the sensory mismatch, which suggest that Cybersickness can come from the disparity between the stationary sensory input versus the visual input of self-motion[20]. Another theory for Cybersickness, is the postural instability theory, which predicts that *"...(1)experiences of motion sickness will always be preceded by increases in postural instability and persist until stability is restored; and (2) people who are more naturally unstable will be more likely to become motion sick during provocative stimulation"* [20, p. 2].

Regardless of the reasons, the reality is that this study's expert participants experienced Cybersickness symptoms after trying the early version of the application, which is why the next iteration's locomotion was designed to be teleportation, instead of steering.

## 3.2 Logging Data and Visualizing the Flow

One of the requirements presented in section 2.2 is the visualization of the data extracted from the simulation. This was particularly asked from MS, and with this purpose in mind, a logging system unity package was added. This software was implemented by Bastian Ilso, Tanguy Blochet and Peter Leth [6]. This system allows to export data from Unity to a comma-separated values (CSV) file. This package was then configured to log data related to the agents simulation sessions. The data being logged during sessions consists of the following:

- Agent ID: Each agent is assigned a unique ID every simulation session.
- Behaviour: Whether the agent is an exit or enter agent.
- Spawn Time: The time since the start of the simulation when the agent spawned.
- Spawn area: The area where the agent "entered" the station.
- Despawn time: The time at which the agent reached its destination.
- Despawn area: The area at which the agent despawned, it can also be understood as the agent's final destination.

- Frame-count: The frame from the start of the session to a particular point in time or frame in the simulation.
- Profile: Whether the agent is a young agent or old agent as mentioned in section 2.3.
- Time-stamp: The system's date and time in real life, independent from the life-cycle of the simulation.
- Agent position: The position of the agent in 3D space in the VE, logged every 2 seconds.
- Number of active agents at that particular time in the simulation.

### 3.2.1 Using R to Visualize the Data from the Simulation

In order to visualize the aforementioned logged data, an R [46] script was created, where the *"plotly"* [51], and *"ggplot2"* [56] libraries are used to create two different examples, that could provide MS an overview of how to visualize the data extracted from a session. First, a 3D scatter plot representing the agents' pathing in the VE is shown in figure 3.1. Here, the structure of the VE can be clearly seen, where most of the agents coincide mainly at the platform level and the tunnels leading to the main train station, following the distribution presented in section 3.4. The dots arranged diagonally represent the agents going down and up the stairs. The code for creating these plots can be seen in appendix C.4.

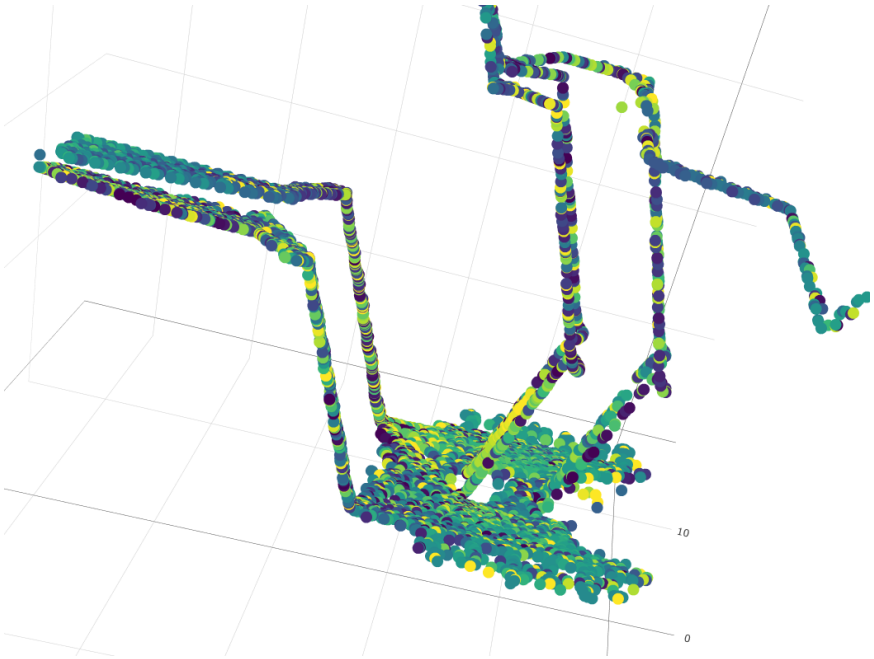
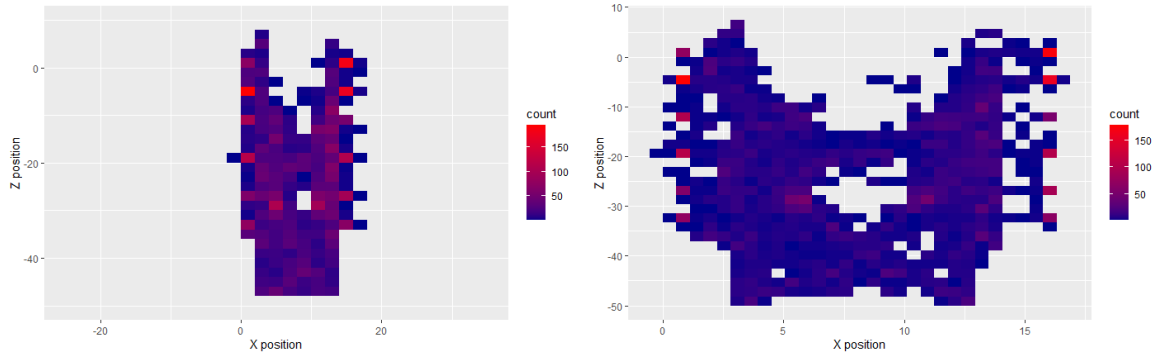



**Figure 3.1:** A 3D scatter plot of the pathing of agents in the VE. The colour of the agents represent their unique ID.

Following the 3D scatter plot, and in order to identify the distribution of the agents, a subset of the data was taken to plot the agents' distribution at the platform level alone. For this, only the agents with the same *Y position = 0 (platform level)* were plotted, and converted into a heatmap/-density plot. Figures 3.2.a and 3.2.b show the areas the agents frequent the most in different levels of detail. Here, it is very clear that the agents at the platform level, frequent areas to the side of the stairs, where validators are.




**((a))** Heatmap of agent distribution logged every 2 seconds. Here, the structure of the platform can be clearly seen.

**((b))** Heatmap of agent distribution logged every 2 seconds. In this close up of the heatmap, the distribution can be seen in more detail.

**Figure 3.2:** A heatmap of the agent's density distribution at the platform level. Blue areas are less frequented than red aread.

## 3.3 Why Behaviour Trees?

This section presents a summary on BTs as the AI technique used for modelling the agents behaviour during the simulation. More detail on BTs, and their comparison to other behaviour modelling AI techniques can be found in appendix C.

### 3.3.1 Behaviour Trees - Definition and Examples

BTs offers a modular approach to creating autonomous control systems, which are stateful, reactive, extensible, and comprehensible to author. BTs have seen great use both within computer games and robotics [21]. Let reactive be understood as *"...the ability to quickly and efficiently react to changes..."*[21, p. 5]. We want our agents to be able to react to the train arriving soon so they walk faster to the platform, and if they are already there, waiting, we want them to board the train as soon as the door opens. For modular, we mean the degree to which these states can be separated and recombined at different stages of the system. For example, both enter and exit agents (as shown in figure 2.1) use the same "move to position" node in several places in their independent BTs, but we created only one Action node "move to position".

Structurally, a BT is a graph structure in the form of a directed rooted tree, where each point in the tree is referred to as a node. BTs are typically illustrated top-down, starting with a parent-less root node with one child node. All other nodes have exactly one parent node connection [21]. BTs are executed through propagating a signal or "tick" from the root throughout the tree. Nodes only execute their functionality when a tick is received, and all nodes must return one of three possible signals back to its parent: *Success*, *Running* or *Failure* [21]. Ultimately, the flow and execution of a BT is guided by these return-signals.

Internal nodes inside the tree are referred to as *"...control flow nodes..."*[21, p. 6]. Classically, most control flow nodes have multiple child node connections which they iterate through in some way. The only exception is a category of control flow nodes referred to as a *Decorator*, which always has a single child node connection, in which they augment the return signal and/or execution process in some way [21]. Leaf nodes are referred to as *"...execution nodes..."*[21, p. 6], which can be further categorized as either *Actions* or *Conditions*. Actions, representing operations that can be performed in the world, return *Success* when an operation is completed, *Running* if it

still in progress, and *Failure* if the operation becomes impossible to complete. Conditions act as a boolean check, verifying some piece of information; if true, it returns *Success*, otherwise *Failure* [21]. Figure 3.3 shows common nomenclature used when referring to BTs.

| Node type | Symbol | Succeeds | Fails | Running |
|---|---|---|---|---|
| Fallback | ? | If one child succeeds | If all children fail | If one child returns Running |
| Sequence | → | If all children succeed | If one child fails | If one child returns Running |
| Parallel | ⇉ | If $\geq M$ children succeed | If $> N - M$ children fail | else |
| Action | text | Upon completion | If impossible to complete | During completion |
| Condition | text | If true | If false | Never |
| Decorator | ◊ | Custom | Custom | Custom |

**Figure 3.3:** Table of common node types, symbols and functionality [21, p. 9].

Due to its graphical structure, BTs are fairly easy to sketch out on paper, and lends itself well to editing and authoring through a visual editor. In fact, entire node substructures can be generalized, leading to high re-usability; A BT can define a complex sequence of actions, and then be used as a node in another BT. This feature can also be used as a design consideration; A high-level BT can be formalized first, then leaf nodes can be broken down further [21]. Extending with custom functionality on a per use-case level is a must, with defining custom action, condition, and occasional decorator nodes forming the bulk of this work. Some level of maintenance of BTs (or pruning) is also to be expected, as the overall system and setting changes [21]. Figure 3.4 shows an example BT, in which a sequence of actions is defined with the goal of highlighting the re-usability of BTs due to its modularity.

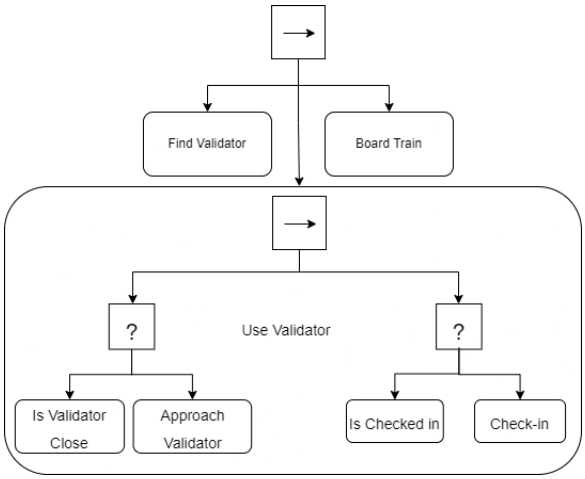



**Figure 3.4:** Example of high-level Behavior Tree, defining a sequence of actions. Here the "Use Validator" action is itself a Behavior Tree that can be reused.

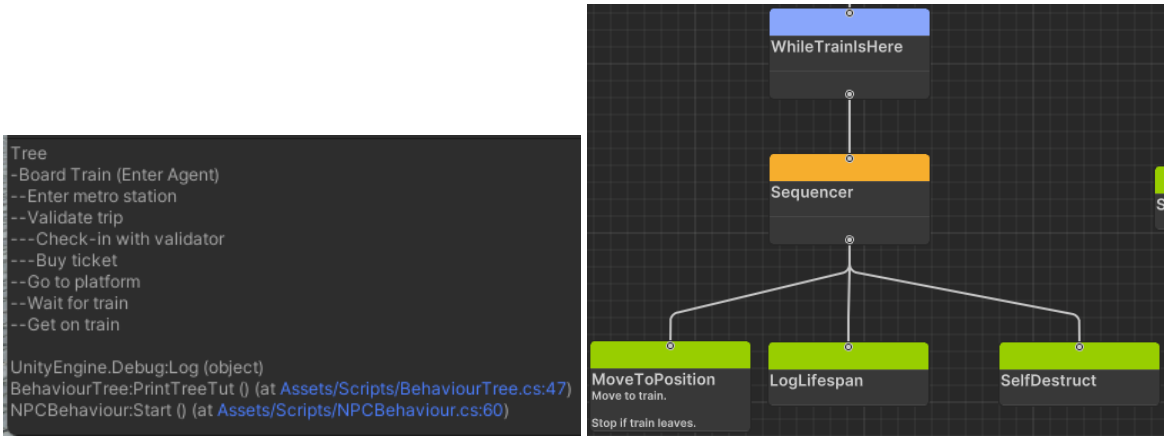
### 3.3.2 Disadvantages of Using Behaviour Trees

Although this section has highlighted some of the advantages of BTs over other AI system architectures, particularly FSMs and HFSMs; it is important to mention that BTs are by no means the magical solution that is the right AI system architecture for everyone to use in every single case. Some BTs disadvantages, and how they were tackled in this project can be defined as follows:

- BTs can be difficult to implement with single threaded sequential programming due to the managing of the ticks and the action execution. Implementing BTs require a different mind-set for designing the system [21].

- Checking all the conditions in a modular system that keeps growing can be very expensive. In this project, we use memory nodes to avoid re-execution of some child nodes that only need to be run once, but are a part of a sequence. This way, the parent node remembers that the child returned success of failure, and do not need to run the node again to check for that condition.
- For some systems that require less complex behaviour, simpler AI architectures can be the best choice. With BTs in this project, we aim to create an unpredictable but rational behaviour, whereas if the goal was just to achieve a rational behaviour, FSM would be just fine.
- *"BTs tools are less mature:"* [21, p. 43] For this purpose, this project uses a free, third party asset that helps visualize, and customize BTs to the user's needs. This asset can be found at `https://thekiwicoder.com/behaviour-tree-editor/` [54].

In relation to the immaturity of BT tools, figure 3.5 shows two different ways of visualizing a BT. In (a) we see a simple printout that requires to manually code a way of visualizing the structure of the BT in a console. In (b) we see the aforementioned BT editor, which helps visualizing the structure and state of BTs. This figure shows how tools for BTs are maturing.




**((a))** Console printout of the structure of a BT

**((b))** The Kiwicoder's [54] BTs editor available at `https://thekiwicoder.com/behaviour-tree-editor/`

**Figure 3.5:** The tools for BTs are maturing.

## 3.4 Agent Distribution

In order to keep the population of the agents realistic in reference to the daily traffic in the real metro, the agent distribution is based on statistics provided by MS. The statistics used are from 2019, and show the amount of people entering and exiting the station from the various entry points, gathered in 20 minute intervals. It was decided that the prototype should show the metro at rush hour, as this would show the most extreme case, and so the statistics used in the prototype are from the busiest 20 minute interval, which according to the statistics were between 7:40 and 8:00 am. In this interval 1825 people entered with 1.52 person pr. second and a distribution of:

- 86.5% entering from the transfer tunnel
- 10% entering from the main staircase

- 2% entering from the back staircase
- 3% entering through the elevator

In the same interval 1964 people exited the station with 1.64 person pr. second and a distribution of:

- 52.8% exiting through the transfer tunnel
- 27.5% exiting through the main staircase
- 13.6% exiting through the back staircase
- 6.1% exiting through the elevator

As these statistics were collected in 2019, it did not include data on the second elevator that was installed later, and so these numbers are not completely comparable to 2022, when this project took place.

MS also provided additional data regarding the passengers boarding and unboarding the train. This was also gathered in 2019, where the passengers of the trains, passing through Nørreport Metro Station, were counted in the time frame between 7:20 and 8:20 am. The data provides information on eight different days, as well as an average for each train going or leaving in that time frame. Furthermore, it shows an average of passengers entering and leaving each gate of each train, thereby providing some very precise data to be implemented.

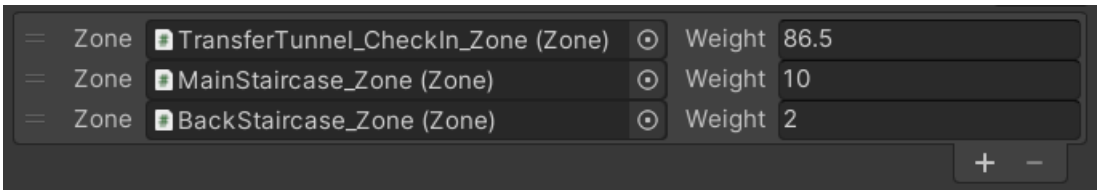



**Figure 3.6:** Weight distribution for all entries in use, shown from inside Unity.

The implementation of the distribution is then done by adding a weight to each exit or entry based on the statistics explained above, which can be seen in figure 3.6. The code will then stack these weights on top of each other and select a random number. This number will be between zero and the sum of all the weights added together. Considering the order of the weights in figure 3.6, if the random number generated should be 8, then it would mean that agent would be spawned at the main staircase. If the number was 16, then that agent would be spawned at the transfer tunnel. As the elevator functionality is not included in this prototype, the weights will not add up to 100. However, this just means that the small percentage missing will be distributed evenly over the rest, as the number generator only picks a number within the given range.
The code picks a number for the enter and exit agents respectively, based on the person pr. second statistic mentioned above. This means that for the enter agents, a number will be generated every 0.66 seconds, corresponding to approximately 1.52 agents pr. second. For the exit agents it will be every 0.61 seconds, corresponding to approximately 1.64 agents pr. second.

## 3.5 Estimating Utility

For most agent interactions, the decision process is straightforward, e.g. travelling agents should only consider boarding a train if it has arrived, and they have valid ticketing. The structure of BTs lends itself well to these kind of decisions, where boolean statements can be evaluated in sequence.

However, some interactions are more situational and dynamic in nature, namely deciding between queuing at occupied validators or choosing to use another, and deciding which area to stand in and wait for the train. For these behaviours, methods for estimating the *utility* of committing to a decision were needed. To solve this, inspiration was drawn from utility theory.

### 3.5.1 Standing in Queue and Patience

As a foundation, an agent sub-behaviour was structured so that, given an unresolved ticketing status, they observe their immediate surroundings, detecting potential validators to use while traversing towards the platform or out of the station. If an unused validator is detected, the agent interrupts its current path, and moves towards the validator in order to use it. But if the target validator has a queue, or becomes occupied before the agent reaches it, the agent's patience is tested using a utility function based on the agents patience-statistic (a normalized value), and the number of other agents currently in queue at the target validator. If the test fails, the agent disregards the validator for a short amount of time, and resumes its originally intended path. If successful, the agent will stand in queue, moving closer whenever the validator is available, until it ultimately is within range to use it, and can complete the interaction.

If an entering agent manages to reach the platform level with unresolved ticketing status, it will inspect validators, starting with the closest, and perform similar patience tests. For exiting agents, if the agent reaches a validator close to an exit point, the test is overruled to ensure that agents are forced to use them, regardless of queue size.

The patience test works by generating a random 0 to 1 float, and using a utility score as a threshold. If the threshold is exceeded, the test fails. Early versions of the utility function used in the patience test were based on dividing the agent's patience-statistic by the current queue size, assuming a queue of at least one. This utility function had the property of mapping the agent's patience directly to queues of size one, but had a dramatic falloff. Even agents with the maximum patience score of 1 would quickly act impatiently with short queues.

A more reasonable utility function was desired, with a more controllable falloff that would maintain higher scores with high patience, using few operations and avoiding divisions if possible to keep computation simple, as well as restricting the range between 0 and 1. Through experimentation, the following equation was formed:

$$Utility = Patience^{Queue \cdot Power}$$

A graph of this equation can be seen in figure 3.7. For simplicity, Power is set globally to 0.5, but could be used as an adjustable parameter alongside the agent's patience to adjust the effect of increasing queue sizes. Additionally, setting patience to 0 or 1 will always return a utility of 0 (never patient enough) or 1 (always patient enough), respectively.

It is worth noting that this behavioural model has limitations. Agents only consider newly encountered validators without comparing between multiple validator queues, nor do they consider leaving a queue prematurely.

### 3.5.2 How Agents Choose Waiting Zones

When referring to zones, we refer to the ones shown in figure 2.5. These areas of interest are predefined as rectangular trigger colliders, which may be referenced as being part of zone collections. All agent navigation tasks are based on picking a zone, usually from a named collection, and then picking a position within that zone as a destination.

Two zone collections are defined as being waiting areas for platform 1 and 2, which an entering agent can pick between once they reach the platform level of the metro station. From observations

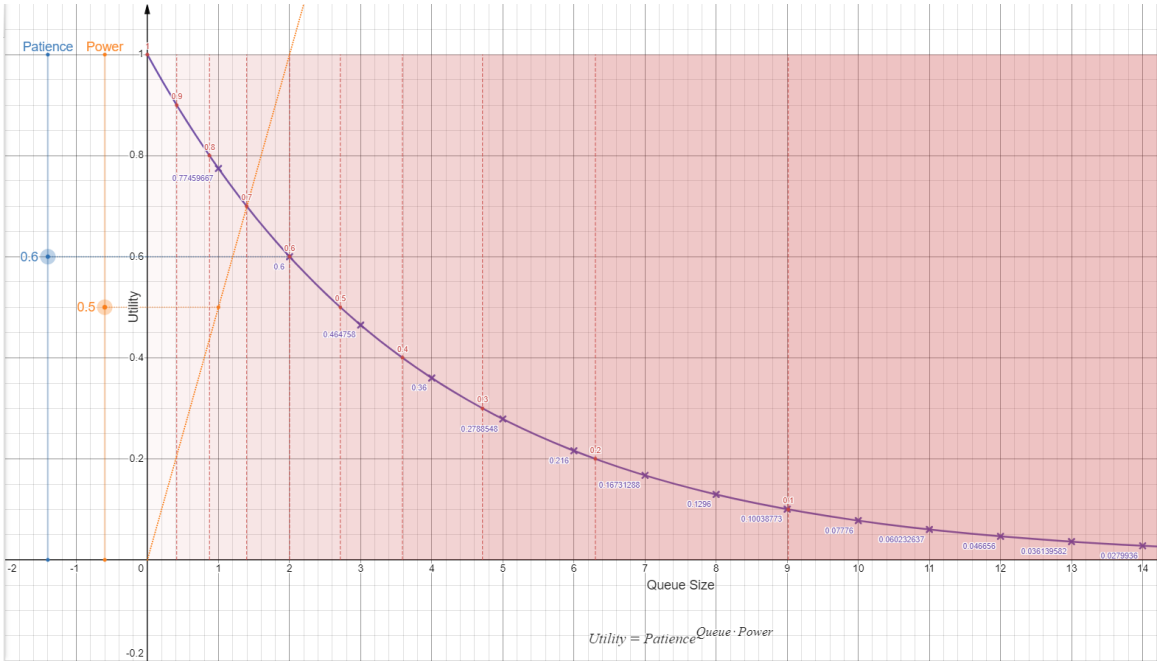



**Figure 3.7:** Plot of utility value based on queue size and agent patience, using a patience of 0.6. The red shading delineates decreasing utility scores in 0.1 steps. The utility is used as a random chance for agents to queue at validators. See `https://www.desmos.com/calculator/bjjxppdhpr` for interactive graph.

of Nørreport Metro, as well as comments from a station steward, it was noted that passengers have a tendency to congest in areas near to their point of entry, especially next to platform gates.

### The Mathematics Behind Choosing Zones Conveniently

Setting up zones next to gates was a trivial matter, but making a satisfying mathematical model for which waiting zone to pick at a given moment proved challenging, as several factors could be considered in such a decision: distance to the area, the current level of occupation, and a persons desired amount of personal space. Some areas could also simply be regarded as more useful due to their relative placement, such as near gates.

Multiple zones would have to be compared during a check, which raised some concerns as to how complex a calculation could be afforded while running the application in real time. Namely distance calculations, which require square roots, are relatively expensive operations. However, the exponential nature of square distances could be utilized to create more non-linear relationships, penalizing larger distances. This does however, lead to a function with an unconstrained upper range.

The distances themselves would also have to be approximated to some degree. The actual traversal distance would follow a path along the NavMesh, but generating a path for each possible destination zone would be very costly. As a compromise, the distances used are the square magnitudes of the direct vector from the agent to the closest point of the zones.

With these considerations in mind, the decision process was modeled so that an agent would calculate a decision weight score for each relevant zone. This weight would be a combination of a distance weight score based on the distance to the zone, and a space weight score based on

the current level of occupation. The agent would then pick a random zone through a weighted randomization process similar to how agent spawn locations are picked, as explained in section 3.4.

For the distance weight, the following equation was formed:

$$DistanceWeight = \frac{1}{Distance^2 + Shift}$$

Where *Shift* is a small value added, as to avoid weights approaching infinity as distances approaches zero. The result is a weight that is high at short distances, and smaller as distance increases.

For the space weight, the following equation was formed:

$$SpaceWeight = \frac{ZoneSpace}{(People + 1) \cdot PersonalSpace}$$

Where *ZoneSpace* is the total square meters that the zone covers, *People* is the number of agents currently present within the zone, and *PersonalSpace* is a statistic of the agent that defines a desired amount of space in square meters within the zone. The resulting weight is essentially a ratio of how big the zone is, to how much space would be used by each agent, including the current agent, assuming they take up a similar amount of personal space. This makes crowded zones more unattractive, and also makes larger zones inherently more attractive than smaller zones.

The two weights can be combined through addition, multiplication, and by averaging, which each lead to behavioural variances, which can be seen in figure 3.8. In the end, averaging the scores was chosen, as to penalize less when one of the two weights were low, and the other was high.

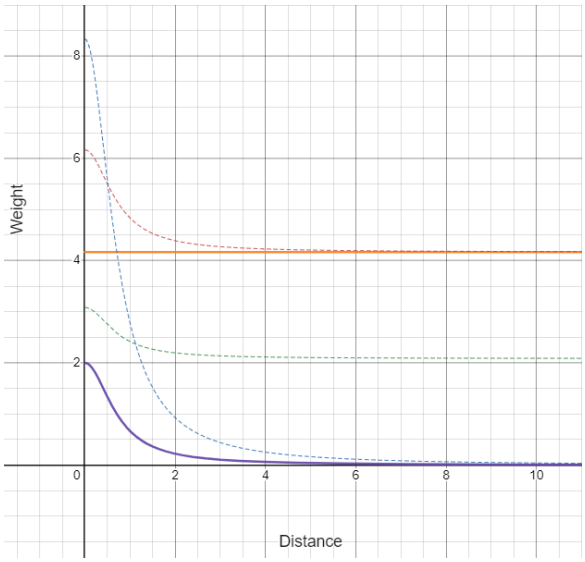



**Figure 3.8:** Plot of utility weight based on number of agents in the zone (5), desired personal space ($2m^2$), size of zone, distance to the zone (x-axis), and corrective distance shift (0.5). Orange line: weight score from available space. Purple line: weight score from distance. Dashed lines: Additive (red), multiplicative (blue), and average (green) weight scores. See `https://www.desmos.com/calculator/finbt2aoou` for interactive graph.

As a final touch, to emulate that some areas are inherently more attractive, the combined weight is multiplied by the weight associated with the zone. Higher zone weights are primarily applied to zones placed close to platform gates.

# Chapter 4

# Evaluation of the Prototype

This chapter presents evaluations of the prototype over two iterations. The initial evaluation consisted of the experts at MS trying the application for the first time. During this time, they were given a presence questionnaire, and a Cybersickness questionnaire. The details of these questionnaires are presented in section 4.1; however it is important to note that the reported Cybersickness was considerably high, and one of the experts had to leave work, and go home after trying the application for a short period of time. It is also important to note that the prototype they tried during the initial evaluation had steering locomotion.

Secondly, this chapter presents the methodology for the final evaluation as described in section 4.2, where a heuristics approach for evaluating usability was also presented to the experts at MS, and one expert at Aalborg University.

## 4.1 The Initial Expert Evaluation

This evaluation was conducted as an initial indication for the validity of the use-case, as well as the overall feeling of presence in the VE, as judged by the experts from MS. This was done in order to make sure that the project was staying on track with the common goal of acting as a potential tool for the experts at MS to use.

### 4.1.1 The Procedure

The evaluation consisted of three participants, which were all from MS, and thereby considered experts in the functionality and daily routine of the metro station. Each participant would answer a Cybersickness questionnaire [33] regarding their current state of well-being. One member from the group was in charge of explaining the questionnaires, and making sure that all questions were answered correctly. After answering the first questionnaire, the experts were guided over to another member of the group, who was in charge of explaining the controller's mapping for using the prototype, and to act as a guide through the experience. The experts were asked to take on the VR headset, and navigate through the VE to get a feel of the space, and to judge the level of resemblance to the real metro station, and how it functions. After the test, the experts were guided back to answer the same Cybersickness questionnaire once again, as well as a presence questionnaire (see section 4.1.3).

### 4.1.2 Induced Cybersickness

When analysing the questionnaires, and the observed behavior during the evaluation, it became apparent that the biggest problem of the current prototype was the locomotion. Each expert experienced discomfort when navigating through the VE, which evolved to a fair amount of Cybersickness, that caused one specific expert to leave work afterwards, due to nausea. It was mentioned by one expert that the nausea became most apparent when turning the camera in VR while staying still in the real world. Other causes of discomfort came from the agents in the VE which, according to all experts, had a complete disregard of personal space, and would in some situations appear to pass right through them.

### 4.1.3 Evaluating Presence

For evaluating the presence, a revised version of the original Witmer and Singer questionnaire [58] was used, revised by UQO Cyberpsychology Lab [34]. This questionnaire holds 24 of the original questions, and divides them into seven sub categories: *realism, possibility to act, quality of interface, self-evaluation of performance, haptic*, and *sound*. Since no haptics or sounds where implemented at this stage, they were left out of the evaluation in order not to affect the relevance of the results.

The prototype scored quite high in regards to realism and possibility to act, as evident from table 4.1.3. During the evaluation, the experts also stated that the VE very much resembled the real metro in size and feel, when navigating around, which could be a factor in the high realism score. Since this was only an initial indication for the prototype, not all of the categories of the table were relevant, since only the basics had been implemented at that current time.
As a closing statement for the evaluation, the experts felt content with the current direction of the project, and was overall positive about the potential validity as a tool for future use.

| Category | P1 | P2 | P3 | Average |
|---|---|---|---|---|
| Realism (49) | 32 | 31 | 40 | 34.3 (70%) |
| Possibility to act (21) | 15 | 15 | 18 | 16 (76.2%) |
| Quality of interface (21) | 16 | 15 | 13 | 14.7 (70%) |
| Possibility to examine (21) | 17 | 12 | 14 | 14.3 (68.1%) |
| Self-evaluation of performance (14) | 11 | 7 | 13 | 10.3 (73.6%) |
| Total (126) | 91 | 80 | 98 | 89.7 (71.2%) |

**Table 4.1:** Results from presence questionnaire. P = participant, number inside category = maximum score, percentage = percentage of maximum score

## 4.2 Final Evaluation

As the final iteration was completed, it was time to go back to MS in Copenhagen, in order to do the final evaluation. This section serves to present the evaluation method and the reasoning behind this choice, as well as the full procedure of the evaluation performed on the experts from MS in Copenhagen. The results and analysis are moreover presented in chapter 5.

Throughout the process of this project, the prototype was iterated and formed in a collaboration with the experts at MS. On the basis of multiple meetings, and one previous initial expert evaluation, a tool was designed with the goal of proofing a concept towards democratizing simulation tools in VR that require no technical knowledge to use and adapt. Ideally, this would be

of value to MS, and could aide them in the creation of initiatives to improve crowd flow in Nørreport metro station. For this reason it felt natural to further use their expertise and knowledge for evaluating the final prototype.

### 4.2.1 Expert Heuristics Evaluation

A Heuristic evaluation of VR applications was chosen as the preferred evaluation method. This method was proposed by Alistair Sutcliffe and Brian Gault [52], and is an extension of Nielsen's usability heuristics from 1994[38], with the purpose of making it applicable to VR applications as well. This method was chosen as it provides great qualitative feedback for the full spectrum of the user experience. This is done by taking advantage of the knowledge and experience of the experts to evaluate the prototype based on 12 heuristics and 6 different aspects of the prototype. The full explanation and breakdown of the chosen heuristic evaluation can be found in Appendix D.

### 4.2.2 Evaluating the Final Prototype

Before venturing back to Copenhagen, some preparations were made to ensure a smooth and clear evaluation. First of all, the heuristics were discussed in order to remove any of the heuristics which were not applicable to the prototype.

First of all the *"clear turn-taking"* was removed, as the current prototype does not have any dialogue or system initiatives. Second of all, the *Clear entry and exit points* was also removed, as no such thing had been implemented into the prototype itself at the current state.

Once the heuristics had been tailored to the prototype, a list of tasks was made to ensure that the experts had an opportunity to thoroughly evaluate all important aspects of the user experience. In order to introduce all features of the prototype, these five tasks were presented to the experts:

- Place three check out validators
- Navigate to the concourse level
- Delete 2 validators
- Place 2 check in validators
- Rotate 2 validators
- Start logging data
- Stop logging data

#### Introducing the Experts

The evaluation consisted of five experts. Four of whom were employed at MS, and one from Aalborg University, working in the field of VR, and thereby had more experience with evaluating VR solutions. Furthermore, the VR specialist had previously worked on a project regarding Nørreport metro station, as a partner to one of the experts from MS, and had therefore also knowledge about that specific station. Three of the five experts had furthermore, been taking part in the production process by providing feedback and information to the team, which meant that they all had prior knowledge about the system and its use-case. This meant that the experts also had a better understanding for what needed to be evaluated.

### Evaluation Procedure

The evaluation started with the expert being asked to fill out a consent form in order to inform them of the extend of the evaluation and to make sure that they understood the type of data we gathered, and what it would be used for. They were additionally asked about their previous experience with VR to get a feeling of how well accustomed they were with using head mounted displays and VR controllers.

As the expert had filled out the consent form, they were asked to answer the same Cybersickness questionnaire as in the previous evaluation, before being sent over to an evaluator who introduced them to the prototype, and the course of this final evaluation. The full setup for the evaluation can be seen in figure 4.1. The experts were then asked to put on the headset, and start the technology audit. Throughout the evaluation one evaluator acted as an informant and guide, if the experts had any questions or problems during the test. Meanwhile another evaluator noted down all comments from the experts. During the technology audit the experts were asked about the topics stated in section 4.2.1. When all topics were covered, the experts were asked to perform the tasks one at a time. No help was given from the evaluators unless asked directly by the experts. After all tasks were completed, the expert was asked to take the Cybersickness questionnaire one last time, as well as the presence questionnaire mentioned in the previous evaluations as well, after which a structured interview was performed. The purpose of this interview was to guide the expert through the comments documented throughout the test, as they were asked to rate them based on the severity of that specific problem. In the end the expert were thanked for participating, and the next was send in.

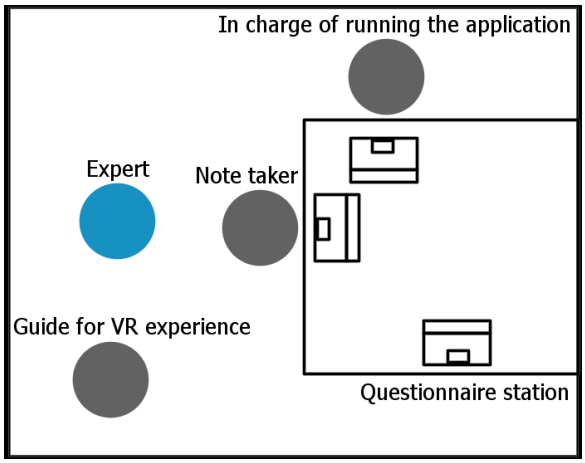



**Figure 4.1:** Overview of the setup for the final evaluation

# Chapter 5

# Results and Analysis

This chapter presents the data collected from the expert heuristics evaluation following the methodology explained in section 4.2.1. The results from the technology audit are presented briefly. A follow up of the Cybersickness questionnaire was also presented to the experts. From the first evaluation at MS, the questionnaire's results suggested that steering locomotion rendered the application unusable, which motivated the locomotion system to be changed to teleportation.

## 5.1 Technology Audit

Before being given tasks, participants had a short period of approximately three minutes to get familiar with the controllers. However, they used most of this time to move around in the environment, and not to interact with the menu. This could be because participants had troubles finding out the watch was an interactable object that would open the menu, possibly due to the lack of signifiers. However, participants commented on the following items:

- Operation of the user's presence: Participants reported a great sense of presence. When giving away the controllers to the facilitators, they expressed *"It is weird to give up your hands"*. However, presence was not necessarily a good thing, since they also mentioned that *"I don't like all these people around me."*, and *"All the moving people makes it difficult to focus (on a task)"* when referring to the agents. Nevertheless, experts expressed that the VE was highly valid for performing these design tasks.
- Haptic feedback: Haptic feedback was considered, but not included in the interaction scheme due to time constraints.
- Interactive techniques: During the technology audit, not many interaction technique problems were reported, other than one participant mentioning that *"It is more intuitive to move (teleport) using the trigger"*. However, many interaction technique problems were reported when interacting with the GUI to complete the tasks. Those are presented in tables D.1 and D.2.
- Realistic graphics: Participants reported perceiving the graphics as very realistic. Some of them expressed feeling that the representation of the metro station was *"Very real"*, or that *"To a very high degree this is very representative of the space"* when referring to the representation of the physical space in the VE. When referring to the simple agents they mentioned that *"The people are obviously boxes, but..." "... it is nice that there is a level of abstraction."*.

## 5.2 Classification and Analysis of the Problems Encountered During the Evaluation

This section presents the problems encountered during the evaluation, which point at some very clear interaction challenges. The proposed design changes to solve these problems are presented in table D.2, and further discussed in chapter 6. These problems were analyzed and generalized from the data presented in table D.1, which contains the problems reported by the participants, and divides them into their corresponding heuristics. For a more detail analysis on the aforementioned tables see appendix D.2.

In short, problems encountered during the evaluation revolve around interaction problems, which made the application difficult to use by the experts. These can be summarized in the following manner:

- A mismatch between the embodiment of the user and their representation in the VE (controllers versus hands).
- Lack of signifiers, feedback, and feed-forward when interacting with the diegetic GUI.
- Misleading perceived affordances when directing the user on how to interact with the validators.
- Mismatch between the control scheme when interacting with the environment versus when interacting with the GUI (controllers versus touch).
- Failure to include multi-modal and multi-sensory interaction considerations in the user experience.

## 5.3 Presence Questionnaire

For the presence, the same revised Witmer and Singer questionnaire, from section 4.1, was used again, and the analysed results can be seen in table 5.1. In this table, the *Sound* category was included, since some sound effects had been implemented at this stage. *Haptics* however, was still of no concern, and was therefor still left out of the evaluation.

| Category | P1 | P2 | P3 | P4 | P5 | Average |
|---|---|---|---|---|---|---|
| Realism (49) | 40 | 40 | 38 | 44 | 38 | 40 (81.6%) |
| Possibility to act (21) | 17 | 18 | 15 | 13 | 16 | 15.8 (75.2%) |
| Quality of interface (21) | 17 | 10 | 13 | 12 | 14 | 13.2 (63.9%) |
| Possibility to examine (21) | 16 | 14 | 15 | 16 | 15 | 15.2 (72.4%) |
| Self-evaluation of performance (14) | 9 | 10 | 11 | 11 | 12 | 10.6 (75.7%) |
| Sounds (21) | 12 | 3 | 13 | 16 | 19 | 12.6 (60%) |
| Total (147) | 119 | 102 | 115 | 122 | 123 | 107.4 (73.1%) |

**Table 5.1:** Results from presence questionnaire. P = participant, number inside category = maximum score, percentage = percentage of maximum score

From the results it can be seen that the presence which the participants experienced within the prototype can be considered to be fairly high, with an overall average of 107.4 out of 147, which gives an average percentage of 73.1% satisfaction. *Quality of interface* had one of the lower scores of only 63.9 percent. When taking into consideration the results from the heuristic evaluation, this makes sense, as many of the problems encountered where related to the UI.

The sound was also scored low in the presence. However, this score may not be completely accurate, as the sound within the prototype was defect for multiple of the participants during the evaluation, causing some participants to possibly misjudge the sound aspects.
This leaves the *quality of interface* to arguably be the most pressing issue in terms of presence, as this can be considered as a focus point for this project.

## 5.4 Cybersickness Questionnaire

When analysing the sickness questionnaires it shows that the participants in general felt no discomfort when using the prototype. In multiple cases the questionnaires actually show an improvement in their mental state across almost all aspects, after using the prototype, and only two out of the five participants noted a slight dizziness and vertigo.

# Chapter 6

# Discussion

This chapter presents the discussion on the final state of the prototype after analyzing the results from the final evaluation. Here, we discuss the viabilty of the application as a tool for mobilities design, as well as arguments for high levels of presence possibly working against the tool's viability. Followingly, we discuss the agents and the modelling of their behaviour, as well as the users themselves as actors that can influence the results of the simulation by just being in the VE, since they can get in the way of the agents. Lastly, we take a deep dive into the design changes that are pertinent to the design of the interaction with the application. Cybersickness and the influence of changing the type of locomotion is also an element of discussion.

## 6.1 Viability of the Application as a Tool for Mobilities Design

With the results from the evaluation, the viability of the application, as a whole, comes into question. In chapter 1 the problem formulation was stated as such:

*"How in-situ VR crowd simulations can support designers in visualizing the flow of a crowd when rearranging the validators of the Nørreport Metro Station in a VE"*

The question then comes down to, if this application can be seen as a viable tool for urban designers. The behavior which was observed during the evaluation, as well as some of the comments from the participants, suggests that certain aspects of the prototype had a more significant impact on the viability, which would have to be discussed further before any final conclusion can be made.

### 6.1.1 Graphical fidelity

In the technology audit in section 5.1 the graphical fidelity is commented as *"Very real"*, and that it represents the real space to a very satisfactory degree. When asking the experts about this graphical fidelity, it became clear that the scale, and graphics made them feel more present in the VE. As all experts had some level of working experience in the specific metro station, this feeling of presence was easy to observe during the evaluation, as most experts easily navigated around the VE, and clearly knew where they were going. Since experts were familiar with the real metro, they intended to place validators in the areas they thought were the most optimal for validator placement. This gives a clear indication that the graphical fidelity of the prototype is sufficient in order to create a representative workspace, in which design tasks can be carried out with a similar mindset as in the real life counter part. In extension to this, multiple experts commented that they could see the VE as highly valid for doing these kinds of design tasks.

### 6.1.2 The User as an Agent in the Virtual Environment

Even though the VE provided the users with a high level of presence, this feeling of presence was, in some cases, also observed to be a hindrance for the users when performing some of the tasks. As the simulated agents are a good way of visualizing flow, and provide immediate real-time feedback, some drawbacks of working in a crowded environment was also observed.

#### Agents as a Disruptive Element

It was made clear, during the evaluation that the existence of the agents in the VE, had some negative effects for some users, as they simply took up too much space. Working in an overcrowded environment can be a factor for a slower workflow, evident by the performance of the experts in the evaluation. Multiple of the experts had trouble carrying out their tasks, as the agents sometimes would block their vision, causing them to lose the overview of the situation. Furthermore, having multiple agents within a close proximity of the user, meant that they had very little personal space, which in some cases caused discomfort for some of the users. This discomfort, and some suggested solutions is discussed further in section 6.3.4.
On the other hand, the simulated behavior of the agents also caused a great deal of excitement from the experts, as they could recognize some of the behavior from their real life experience. The fact that they could feel the busyness from the station also provided an amount of motivation for performing the tasks, as it was commented be an expert that he *"... could feel the response."* from the agents.

#### The User Influencing the Simulation Results

As the agents can be a hindrance to the users, so can the users be a hindrance to the agents. In the current prototype, the user is considered as an obstacle, and therefor has a physical existence in the VE, which the agents will recognise. This was done in order to make the agents try to avoid the user, so that they would not walk directly through them, thereby making the user feel more present in the VE, as they would act as an agent themselves. However, this also means that the user can affect the behavior of the agents, thereby possibly being a liability to the validity of the data collected. By this, it is meant that the user has the chance of blocking the path of the agents, by for example, standing in front of the escalator. This can have a so far unknown effect on the data, and the trade off between the presence and the reliable data, should be taken into careful consideration in a potential continuation of the project.

## 6.2 Discussing User Presence

Going back to the original problem formulation stated in chapter 1, the whole problem revolved around creating a way of supporting designers in visualizing the effects of their work from a more real life perspective, by using VR. In this regard, the presence of the user also has an important role to play, as evident from the evaluation. In the results stated in chapter 5, it suggests that the problems encountered by the user, had a direct impact on the feeling of presence as well. From this it can then be argued, that in order for the user to have the best experience, and thereby using the product in the most optimal way, the user has to be able to make a connection between the virtual space and the real world work space. Thereby, by feeling present in the VE, it would mean that the user can work in a more natural and efficient way, to some degree. This is supported by the comments from the experts, during the evaluation, who stated that they felt a great sense of presence, and thereby saw the VE as a highly valid representation for performing the related design tasks.
The results of the presence questionnaires can be considered to be relatively high, taking into

consideration that this is only a prototype. The realism category shown in table 5.1 was rated especially high, with a 11.6% raise compared to the first evaluation. This is arguably mostly because of the improved behavior of the agents. However, multiple other factors, such as locomotion and interaction design, were also taken into consideration when designing the prototype.

### 6.2.1 Cybersickness and Locomotion

In section 3.1, the different types of locomotion, as well as their pros and cons are discussed. Furthermore, it is stated in that section, that *"Steering can increase presence, but it tends to increase (Cybersickness)."*. This statement, and the research supporting this, let to the initial decision of using steering as the preferred type of locomotion. However, during the first evaluation, it became apparent that this was too much for the participants to handle, as it caused an alarming amount of Cybersickness, as stated in section 4.1.2. This then let to the decision of changing the locomotion type to teleportation instead, as this had a Cybersickness reducing effect, without compromising presence. The Cybersickness questionnaires from the final evaluation suggest this, as almost no Cybersickness was reported, and multiple questionnaires actually showed an improvement in mental state, as described in section 5.4.

It can be argued that the cause of the high amount of Cybersickness observed during the initial evaluation, could be partly due to the low level of experience with VR, shared by the majority of the experts. However, the clear difference in the results indicate that teleportation was the right choice for this prototype.
Interesting to note, is also the minimal difference in the presence results. When comparing the results of the two evaluations again, no significant difference was seen in the overall presence, indicating that the type of locomotion has no real effect on the presence, even though it can cause a significant amount of Cybersickness and discomfort for the user. This however, would require another test with a bigger amount of participants in order to offer any reliable conclusion.

### 6.2.2 Effect of the Interface Quality

Multiple interaction problems were encountered during the heuristic evaluation. These problems were mainly related to the control scheme and the UI, and will be described in more detail in section 6.3. Even so, the nature of these interaction problems can cause an effect on the presence as well. As shown in table 5.1 the category *Quality of interface* received one of the lowest scores when evaluating the presence. Since this category covers over the controls and the quality of visual display, a direct cause of effect can be seen from the problems noted down in the heuristic evaluation, and a decline in the level of presence. This indicates that these problems are to be considered the most critical in terms of presence, as the sound is of no concern at the current state of the project.

## 6.3 Proposing Design Changes Based on Heuristics Evaluation

This section presents a discussion based on the design changes that come directly from the problems encountered during the heuristics evaluation. Section 5.2 presents a summary of the classification of these problems, and this section addresses those problems in detail.

### 6.3.1 Graphical User Interface Display

Here we refer to figure 6.1, which is shown to the user when they open the menu attached to the watch on the left hand of their avatar in the VE. This control overview was both perceived by some users as not clear enough, as well as having too much information. The problems mentioned in

this section were classified as "Annoying", meaning that they were considerably hindering for the participants when carrying out the tasks given during the heuristics evaluation.

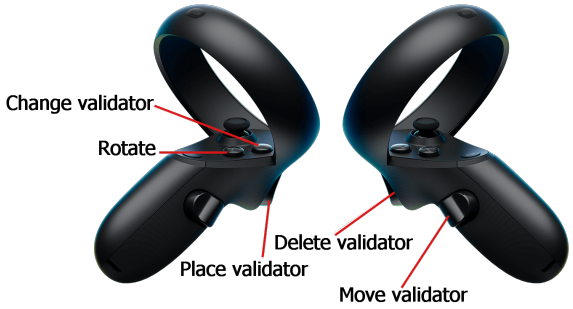



**Figure 6.1:** Controller mapping for the Oculus Quest 2 with regards to validator placement and movement.

Amongst the several reasons for this not being clear enough, we address the following problems:

- The mismatch between the user and their avatar in the VE: In the real world, the user is holding an Oculus Quest 2 controller with their hands, whereas in the VE they only have a representation of their hands, and are not holding the controller. Furthermore, figure 6.1 was made with Oculus Quest "1" controllers, which are different in shape to the Oculus Quest 2 controllers. This mismatch could be causing confusion to the user because in the VE their hands appear to be free and unencumbered, and to read the controllers they have to look at a picture inside a simulation. Depending on the positioning of the user in the VE, the picture could be more difficult to read due to the fact that the background of the picture is transparent. This means that if the user looks at this picture in a dimly lit part of the VE, the black font will be more difficult to read.
- When comparing figure 6.1 to figure A.3, we do not consider the picture has too much information. However, in an attempt to erase the need for a control overview mapping, we propose a control hinting layout that would show up when the user hovers a button with their finger.

The proposed design changes to address the aforementioned problems are presented as follows:

- Change the image to have a view of the Oculus Quest 2 controllers.
- Add a contrasting background to the image, so that its readability is not dependant on the position of the user in the VE.
- Remove the controller overview completely. Instead, change the users' hand in the VE for Oculus Quest 2 controller 3D models that are move as the user moves the controller in real life. Additionally, when a user hovers a button with a finger (before pressing it), a translucent finger appears on top of the button in the VE feed-forwarding what would happen if the user pressed that button.

### 6.3.2 User Interaction with the Virtual Environment

The problems addressed in this section are related to the *"User Presence"* presented in table D.2; however, to avoid confusion with section 6.2, it was named after the common denominators in the problems reported by the expert participants during the final evaluation of the prototype. These problems were then divided into sub-categories that are presented in the following sub-sections. These problems were reported as *"Distracting"*, which means they were less hindering than all the other problems covered in this section.

#### Navigation Features

More than problems with navigation features, one of the expert participants suggested that having to go up the stairs in order to go to a different level/floor in the VE was too much work, and/or used too much time. They attempted to teleport to the floor above by pointing the raycaster on the hands/controllers to the bottom of the floor as shown in figure 6.2.

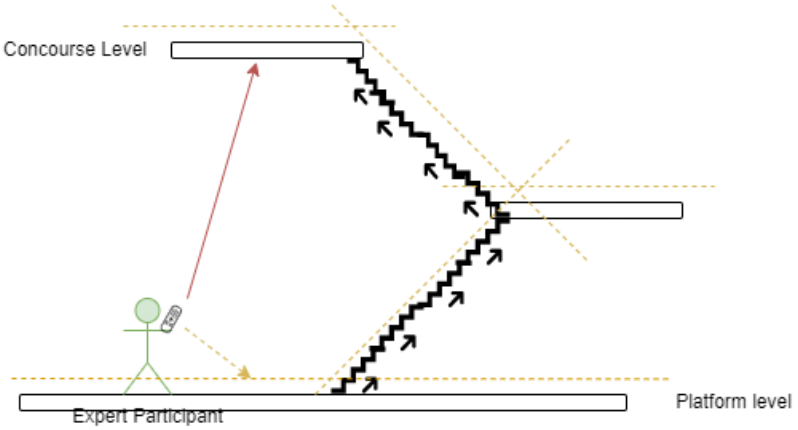



**Figure 6.2:** Representation of the user attempting to teleport to a different floor with the red arrow as the raycast. The yellow dotted arrow and lines represent the path the participant would need to follow, since the teleportation areas are on top of the ground.

The proposed design change for this is to create an *"elevator"* or *"jump"* mode that could be enabled from the GUI, that would allow the user to be re-positioned between floors. The button on the GUI could be elevator doors that would prompt the user to select a different floor by name or level. The elevator doors as buttons would ideally serve as a metaphor for the user. Additionally, a fade to black screen would be added to any form of teleportation, in order to avoid possible Cybersickness and loss of orientation.

### 6.3.3 Interaction with Objects and Tools

This category was were most of the interaction problem were reported, as they hindered the expert participants from completing their task. This was mainly due to the reasons presented in section 5.2. Mainly, related to misleading perceived affordances, which translated to misleading signifiers.

Particular problems reported by the expert participants can be seen in table D.1 under the categories of:

- Natural engagement.
- Compatibility with the user's task and domain.
- Realistic feedback.

- Faithful viewpoints.
- Support for learning.

How to interpret these categories is explained in section 4.2.1. However, the design changes pertinent to this category are described as follows, starting with the changes related to **Interacting with the GUI:**:

- Creating pairing signifiers for objects that afford interaction. An example of this, would be creating a pulsating aura that emanates from the watch that opens the GUI in the application, and at the same time putting the same feature in the finger in the virtual hand that activates it.
- Haptic feedback on UI Interaction: Although this was not implemented due to time constraints, having the user get haptic feedback with interacting with the GUI, could increase immersion. Currently, the user needs to activate the GUI by touching the watch with their virtual hand, but the only form of feedback is visual when successfully opening the GUI.
- Similarly, during the evaluation many users would accidentally press a button in the GUI right after they opened it. This was due to poor positioning of the buttons when designing the GUI, where the "start logging" button would overlap with the watch that activates the GUI as shown in figure 6.3.

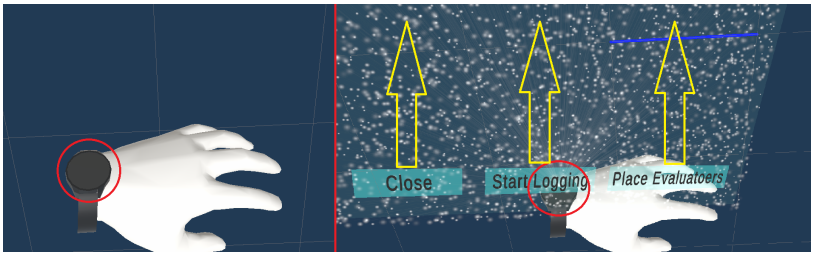



**Figure 6.3:** Overlapping position of the "start logging" button and the activation of the GUI. The yellow arrows indicate that these buttons should be moved upwards to avoid overlapping and accidental presses.

  To solve the problem of accidental button presses, the buttons could be moved to the top of the GUI as indicated by the yellow arrows in figure 6.3.
- When interacting with the menu to place validators, there is no indication of which validator will be placed next, meaining that the user would have to first place a validator, and see whether it is a check-in or check-out validator. To solve this problem, an indication in the GUI would be added. An example of this would be color coding the controller/hand that is used for placing validators to become "red" for check-out, and green for check-in validators. This color coding idea could be applied in many different ways.
- Validators can only be placed after being rotated, which also adds to the user not knowing which validator they would place next. However, changing the sequence for placing a validator to allow users to rotate them before placing them would also allow the users to see which type of validator they are to place next.
- Users were confused by the description of the button for "changing validator" shown in figure 6.1. This was due to misleading design of the signifiers in relation to the perceived affordances. Users would expect to be able to change a validator that was already placed, which makes sense given the "change validator" button. However, this was only possible before placing the validator. In order to solve this problem, the system should afford to change validators that are already placed, from check-in to check-out, and viceversa.

- Since before using the GUI, users would only teleport around in the scene using the controllers and raycaster, they expected to interact in a similar manner with the GUI. However, interaction with buttons in the GUI was only possible by using the index finger in the right hand of the player avatar to "collide" with the button. Although the idea of this, was to give the user the sense that they were actually "pressing" the button physically, this also meant that the interaction was not homogeneous across the application. To solve this problem, the GUI should afford the user to interact using the controllers and raycasting as well.

### 6.3.4 Agents' Proximity to the User

Experts at MS were very perceptive of the agent's proximity to them while the evaluation was carried out. However, we observed that from these four experts, two of them felt the agent's proximity was distracting them from completing the tasks. One of them pointed out that *"I do not like all these people around me!"*, and another one said that *"All the moving people makes it difficult to focus"*.

On the other hand, two other expert from MS did not mentioned any discomfort related to the agent's proximity. Furthermore, a VR expert participant pointed out that *"...the agents were great... I could feel the response"*, when referring to the agents avoiding his avatar in the simulation.

In order to afford the user with a tool that could alleviate the discomfort of having the agents too close to them, a "shield" that can be activated from the GUI could be implemented, so that agents that enter that area around the users would be faded out the closer they get to the user's avatar. Figure 6.4 shows a sketch of how this shield would look like, with the user's avatar in the center.

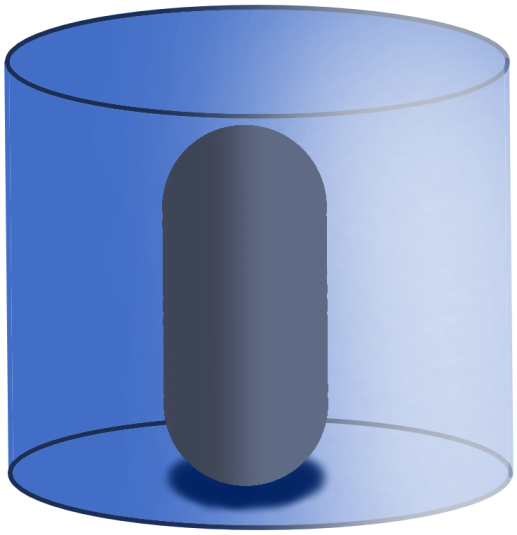


**Figure 6.4:** A sketch for the "shield" functionality that would fade out agents that come inside that area until they leave it.

#### Alternative Virtual Representation of the User

Another interesting approach to avoid the user feeling too close to the agents would be to create alternative points of view for them to oversee the environment. However, this could change the "real world" perspective of the user experiencing the simulation, which in return could affect presence. Different viewpoints could also affect how the space is perceived, and how the space in the VE is associated with the real world space.

After having considered what creating an alternative point of view could do to the experience, a proposed design change could be creating a "security camera" mode, in which the users could interact with cameras installed in the VE, and change their perspective and point of view to observe the VE from there.

## 6.4 The Building Blocks of the Application in Retrospect

This section presents considerations on the technologies and techniques used to build the final prototype, and whether they could be substituted for other tools and techniques that would better the user experience and realism of the simulation. These building blocks refer directly to the way the agents are programmed for behaviour and navigation.

### 6.4.1 Pathfinding versus Behaviour

Reynolds[48] distinguished behaviour into hierarchical layers of Action Selection, Steering, and Locomotion, under the informal notion of steering relating to fast motion that anticipates the future state. Pathfinding was seen as a related but separated topic in regards to steering, as it could provide navigational instructions for steering, but still be separate from the actual steering. Through this lens, one could posit that pathfinding is an element of action selection, as pathfinding creates sequential action plans of target destinations.

Action selection in this application is handled through BTs, which instruct the NavMeshAgent component to pathfind between locations, and then handle steering between path waypoints on its own, until the destination is reached, or it is interrupted by the BT. Nonetheless, the combined steering and pathfinding leads to a seemingly functional navigation behaviour. One MS expert noted how orderly and *"well behaved"* agents walked down corridors. Conversely, most experts commented on the more chaotic embarking and disembarking of trains, as agents did not make space for each other. There is also an issue in that agents tend to follow their paths very strictly, and not leverage the space, leading to what we have referred to as "conga lines".

As such, the pathfinding and steering behaviour of agents should be scrutinized more strongly when modelling behaviour, on top of considering the "when"s of agent decision making.

### 6.4.2 Problems of Pathfinding as a Black Box

Unity Engine's NavMesh system offered an immediate, advanced solution for handling agent navigation and steering, but came with a subset of issues, and some navigational tasks had to be solved in suboptimal ways due to implementation limitations. The core problem is that the system is fairly closed with a limited scripting interface. As an example, while it is possible to retrieve an existing navigation path, and store it for later use, it is not possible to manually modify its internal list of navigation waypoints, nor combine paths together.

Generating paths is only possible through either a non-asynchronous function that can be very costly when generating long paths, or by setting the destination of an agent, which starts a background asynchronous pathfinding process that is globally limited in the number of graph node expansions are allowed across all agents each frame, and has to be checked repeatedly with other functions to determine when pathfinding has completed.

While the navigation mesh areas and costs can be adjusted, and the mesh can be rebaked, the mesh structure itself is not modifiable, nor can it store additional data that could be leveraged for

navigation. Escalator navigation was handled by defining specific areas, and restricting agent accessibility, but being able to define directionality to the mesh data could be an interesting solution.

### 6.4.3 Alternative Tools for Pathfinding

Unity Technologies are working on overhauling their built-in navigation systems, so more robust systems are to be anticipated in the future. Furthermore, through their work on the Data-Oriented Technology Stack (DOTS) framework, Unity Technologies have developed new experimental APIs for NavMesh tasks that are multithreaded. These new features are in active development, and not yet fully mature, but can be leveraged. Beyond significant performance gains, the new interfaces may offer new strategies for tackling some of the encountered issues.

Alternatively, other external navigation packages are worth considering. The A* Pathfinding Project by Aron Granberg offers a feature rich multithreaded toolset, with various methods for defining navigable surfaces and navigating across them [27]. Additionally, exploring the capabilities of Unreal Engine may also be worthwhile, as it also offers a NavMesh system, along with a builtin BT system.

### 6.4.4 The Challenges of Modelling Behaviour

Instructing an agent to follow a path from A to B is a simple task, but modelling decision-making processes quickly grow in complexity, particularly when considering parallel processes, and require understanding of the environment being modelled.

Nørreport Metrostation provides a specific environment with clear objectives and tasks to model - use validators, board a train or leave the station. But considering the behavioral modelling as an iterative process, even such an environment contains many potential sub-behaviors that can be included and refined - deciding which validator to use, when to queue or dequeue, handling how a queue is observed and navigated, gate, escalator, and elevator customs, handling of personal space etc. New interactable objects could be included, such as ticket and vending machines. Even the matter of navigation and steering of agents could be further refined, as to emulate a persons knowledge of the space, as well as make agents traverse corridors more naturally, as opposed to always following the closest path afforded by A* pathfinding through a navigation mesh, which tends to stick close to walls and mesh borders.

Two major challenges arise from considering these potential behaviours. The first is ensuring that the cross-interaction of each sub-behaviour does not cause them to break each other, designing them so that that they work together. The second is determining how much is relevant to include, and to what level of detail, in order to create a convincing representation. With increasing complexity comes increased simulation costs, as well as risk of brittleness in the overall behaviour.

### 6.4.5 Behaviour Trees as a Mature Tool

Contrary to using a black box tool such as the NavMesh system, the kiwicoder's Behaviour Tree Editor [54] allowed us to modify the behaviour of the tool to our needs and the need of the agent's behaviour. Even though literature suggest that the maturity of BTs is a disadvantage, using this editor proved an improvement towards making BTs more accessible for growing projects such as this one. Figure 6.5 shows a simplified version of one of the BTs used by the agents during the final stage of the prototype, here some of the nodes are sub-BTs themselves, which are hidden so that the visualization is not as cluttered. Using sub-BTs as nodes was not an feature of TheKiwiCoder's original tool, but since the source code was made accessible, it allowed us to implement this

improvement. Understanding and developing a BT of this size without this tool, would have much more difficult. Not having this tool would have required us to build a BT Editor tool by ourselves, which could have been a whole semester project by itself.

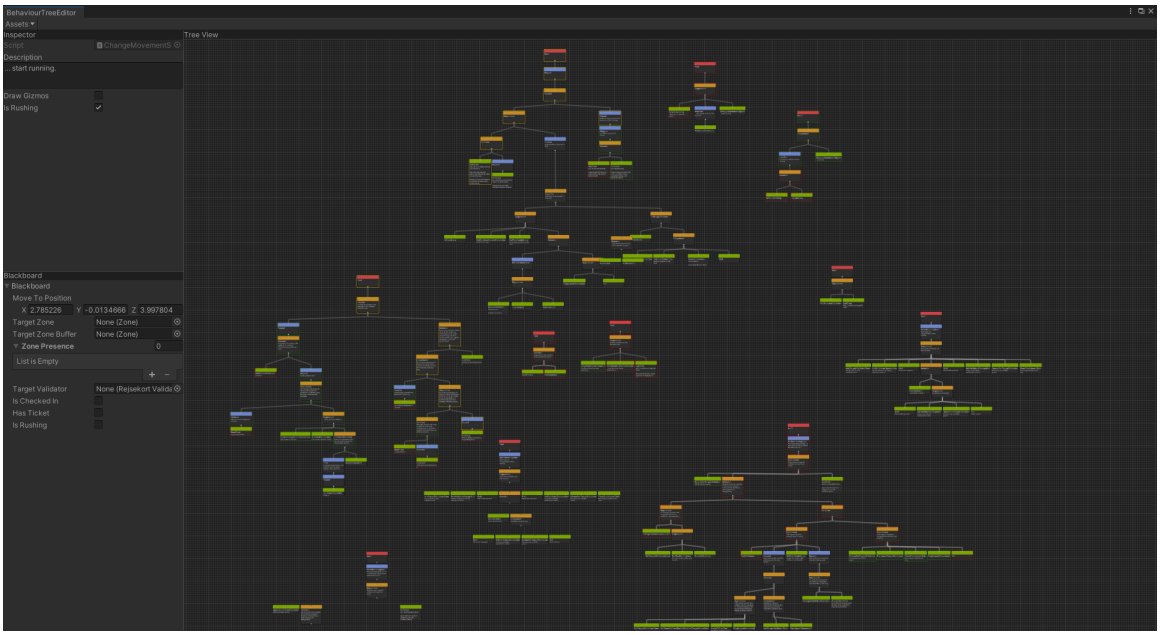



**Figure 6.5:** Sample BT from an agent during the final stage of the prototype.

## 6.5 Future Works - Democratizing the Tool

This section presents the discussion on the possibility for future works, in the terms of democratizing the tool for urban designer experts. Apart from the already discussed design changes that came from the final evaluation, it is important to mention that the tool as is, can be used for a very particular case, and a very particular problem. As presented in section A.1, mobilities have the need for technologies that arise as culture changes to be included in the design process. However, for urban designers and architects this is no easy task, as it requires them to have knowledge in computer programming, apart from the knowledge directly pertinent to their field. Followingly, we propose some changes that would make the tool more customizable and adaptable to other cases, with the challenge of keeping it easy to use for people without deep computer programming knowledge.

### 6.5.1 Importing "Building Information Modeling" Files into the Virtual Environment

A step in the direction of democratizing the tool, is to allow users to import custom VEs by adding support for building information modeling (BIM) files. BIM files are the current industry standard for storing detailed 3D models used by architects, engineers, and contractors. Supporting the implementation of these files in this project would therefor be the best way to make it attractive for the targeted designers in the industry.

So far this project has shown the possibility of visualizing flow in a very narrow and specific use-case. If this project were to implement the support for BIM files, storing different kinds of models in different sizes and forms, it would require a larger improvement in the interaction

capabilities. This would mean the support for a lot of general actions, that would be useful for a more general work flow, and not just replacing validators, as the level of detail stored in the BIM files supposedly surpasses the level of the currently used model. This would also mean that the behavior of the agents would have to be even more modifiable to take into consideration the many possible variations in the environments.

### 6.5.2 Creating a Framework for a Casual User to Modify Behaviour

The modelling of behaviours for the application have been done on a very technical level, where new nodes and functionality have been added as needed. Drafting behaviours can be done by building flowcharts and storyboards, but setting up BTs functionally requires a level of computational thinking that may be difficult for less technical or unaccustomed users. Were a standalone application to be developed, in which the user can set up advanced scenes and behaviours, the usability and understandability of the AI system, along with any AI editor, should be considered strongly.

#### Explainable Artificial Intelligence

The emerging field of explainable AI is typically associated with far more advanced machine learning models, and finding ways to convey how a learning algorithm reached a given answer, as to shed light on what otherwise could be considered a black box. Arguably, being able to interpret both the answer and the reasoning behind any decision process has inherent value, both for the analysis of the system, and for its maintenance.

For the sake of this project, having access to a visual representation of the state of individual nodes in a BT during runtime has been a boon in this regard. However, errors have occasionally arisen both from scripting mistakes, as well as structural issues inside BTs. Determining one from another was not always trivial, and required slowly stepping through stack traces to investigate complicated node sequences. Technical maintenance is given, but adding tools for investigating, testing and evaluating agent behaviours, as well as assisting in crafting desired behaviours, would be great additions in order to support users. Making such advanced tools simple enough for less technical users would be a major challenge from a technical and user experience design standpoint.

#### Different AI structures

Alongside better tooling, changing the foundational system for creating agent behaviours may lead to better abstractions for the user. The following is only a fragment of a vast field of research and opportunities[37].

Standard BTs are far from the only way to structure behaviours, and several variations to BTs and BT nodes have been developed for various purposes. Parametric Behaviour Trees (PBT)[50] offer one way to extend BTs capabilities. Being parameterized, PBTs are capable of receiving data to act upon. The benefit gained is that a PBT can describe an event or activity more generally from an outside perspective (as opposed from the agents perspective), and when agents needs to participate, the relevant elements (agents, objects, values etc.) are passed as parameters, with the agents being driven by the external PBT. This leads both to reusable item interaction behaviours, as well as a structure for describing behaviours involving multiple agents interacting with each other[1, Chapter 7]. Ultimately, PBTs offer a potential way to reduce the complexity of modelling behaviour as a whole.

Other decision making structures are also worth considering, such action-planner style AIs[16, Chapter 9]. Goal Oriented Action Planning (GOAP)[39, 40] is one popular example, which is

intended for making agents automatically finding sequences of actions that will fulfill their desires. Possible actions, including their prerequisites, costs and outcomes, are scripted in advance. Agents, having information of their desires, internal state, and capabilities, will evaluate the currently most pressing desire, and build a graph structure of sequential actions the progressively fulfill the desire, and subsequent action requirements. The lowest cost sequence of actions can then be found through pathfinding. GOAP pairs well with utility-based heuristics, but is a more black box approach. Dynamically constructing action plans also come with inherent processing costs.

# Chapter 7

# Conclusion

This project reports the viability of using VR and simulations as a way to aide mobilities design. This was done through a particular use-case provided by MS that aims to visualize crowd flow at Nørreport metro station in Copenhagen. Simulating crowd flow depending on validator placement in a VE affords the designers three main things:

- Experience the space they are designing for in a 3D manner from a first person point of view.
- Customize the placement of validators throughout the station easier than if they had to do it in real life.
- Extract data from the simulation that could provide them with insights on the effects of these changes.

Other solutions would only allow them to experience this space on a 2D screen, which could yield other results, as it is not as immersive as experiencing it in a 1:1 scale as if they were in the metro station. Additionally, other design solutions that allow for simulating crowd behaviour, require extensive training in order to be used, as they seem to have complicated user interfaces. More importantly, our prototype affords the user to experience the simulation in real-time where the agents react to changes in the environment immediately. Other solutions require the simulation to be set-up, run, and then the user can play it back.

Using heuristics for experts evaluation allowed for an insight on the shortcoming in the design of this prototype. However, very relevant improvements came from having the experts at MS test the application. These shortcomings are however, very easy to fix if we were to continue developing this application. One success case that came out of the initial evaluation, was changing from steering locomotion to teleportation locomotion. To the point were one expert that had tested the prototype with steering locomotion and experienced induced Cybersickness, was afraid of trying the application with teleportation locomotion. They then reported feeling better than before trying the application when filling out their post Cybersickness questionnaire.

Considering some of the experts were architects who worked at the station when it was being built, and they expressed that *"To a very high degree this is very representative of the space"*, the graphical element of the representation of the 3D space was considered a success.

Some shortcomings in the pathing and behaviour of the AI were encountered during the implementation of this prototype; however, fixing these problems would require establishing the navigation of the agents with a different approach. Additionally, more work is required to better the behaviour of the agents, such as avoiding "conga lines", in order to make the simulation closer

to reality. Nevertheless, much of the behaviour that was seen at the train station during the field trip was implemented, an example of this could be agents arriving to the station and favouring waiting zones closer to them.

Finally, we conclude that given the multidisciplinary nature of this project, VR can be used to aide mobilities design. However, for the tool to be less particular to a use-case, it needs more development that would afford the user to manipulate the conditions of the VE and the behaviour of the agents without resorting to technical knowledge about programming or VR. One example could be to afford the user to import a BIM model to act as their VE.

# Bibliography


[1] Saad Ali et al. *Modeling, Simulation and Visual Analysis of Crowds: A Multidisciplinary Perspective*. 2013. ISBN: 9781461484820. DOI: 10.1007/978-1-4614-8483-7_1.

[2] Kurt Andersen et al. "A Comparison Between a Natural and an Inorganic Locomotion Technique". In: *17th International Conference on Information Technology–New Generations (ITNG 2020)*. Ed. by Shahram Latifi. Cham: Springer International Publishing, 2020, pp. 317–323. ISBN: 978-3-030-43020-7.

[3] M. Laura Angelini. *Learning through simulations : ideas for educational practitioners*. eng. 1st ed. 2021. SpringerBriefs in Education. Cham, Switzerland: Springer, 2021. ISBN: 3-030-65540-7.

[4] Unity Technologies ApS. *Unity [Computer Software]*. Version: 2020.3.24f1.1351. URL: https://unity.com/.

[5] Christine M. Bachen, Pedro F. Hernández-Ramos, and Chad Raphael. "Simulating REAL LIVES: Promoting Global Empathy and Interest in Learning Through Simulation Games". In: *Simulation & Gaming* 43.4 (2012), pp. 437–460. DOI: 10.1177/1046878111432108. eprint: https://doi.org/10.1177/1046878111432108. URL: https://doi.org/10.1177/1046878111432108.

[6] Tanguy Blochet Bastian Ilso and Peter Leth. *Logging Manager*. https://github.com/med-material/LoggingManager. 2022.

[7] Bentley Software Inc. *Introduction to LEGION - Bentley's Solution for Pedestrian Simulation - YouTube*. 2021. URL: https://www.youtube.com/watch?v=b5L7IX2au6U (visited on 03/15/2022).

[8] Bentley Software Inc. *Product Data Sheet - LEGION® Modeling and Simulation*. 2021. URL: https://prod-bentleycdn.azureedge.net/-/media/files/documents/product-data-sheet/pds_legion_modeling_simulation_ltr_en_lr.pdf?la=en\&modified=20210615151905 (visited on 03/15/2022).

[9] Bentley Systems Inc. *Buy LEGION Model Builder | Simulation & Modeling Software | Virtuosity*. URL: https://virtuosity.bentley.com/product/legion-model-builder/ (visited on 03/15/2022).

[10] Bentley Systems Inc. *Buy LEGION Simulator | Pedestrian Simulation Software | Virtuosity*. URL: https://virtuosity.bentley.com/product/legion-simulator/ (visited on 03/15/2022).

[11] Bentley Systems Inc. *LEGION Simulator CONNECT Edition Update 4 Help*. URL: https://docs.bentley.com/LiveContent/web/LEGIONSimulatorHelp-v5/en/GUID-3521FF9C-E14F-46D5-A547-F124F3EA1CF4.html (visited on 03/15/2022).

[12] Bentley Systems Inc. *Pedestrian Movement Simulation Software – LEGION Simulator*. URL: https://www.bentley.com/en/products/product-line/mobility-simulation-and-analytics/legion-simulator (visited on 03/15/2022).

[13] Bentley Systems Inc. *Simulation and Modeling Software – LEGION Model Builder*. URL: https://www.bentley.com/en/products/product-line/mobility-simulation-and-analytics/legion-model-builder (visited on 03/15/2022).

[14] Evren Bozgeyikli et al. "Locomotion in virtual reality for room scale tracked areas". eng. In: *International journal of human-computer studies* 122 (2019), pp. 38–49. ISSN: 1071-5819.

[15] Evren Bozgeyikli et al. "Point & Teleport Locomotion Technique for Virtual Reality". In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*. CHI PLAY '16. New York, NY, USA: Association for Computing Machinery, 2016, 205–216. ISBN: 9781450344562. DOI: 10.1145/2967934.2968105. URL: https://doi.org/10.1145/2967934.2968105.

[16] Mat Buckland. *Programming Game AI by Example*. Wordware Publishing, Inc., 2005, p. 521. ISBN: 1-55622-078-2.

[17] Fabio Buttussi and Luca Chittaro. "Locomotion in Place in Virtual Reality: A Comparative Evaluation of Joystick, Teleport, and Leaning". eng. In: *IEEE transactions on visualization and computer graphics* 27.1 (2021), pp. 125–136. ISSN: 1077-2626.

[18] Hiba Chaher, Lilia Rejeb, and Lamjed Ben Said. "A behaviorist agent model for the simulation of the human behavior". In: *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*. 2020, pp. 1–11. DOI: 10.1109/OCTA49274.2020.9151655.

[19] Chris G Christou and Poppy Aristidou. "Steering Versus Teleport Locomotion for Head Mounted Displays". eng. In: *Augmented Reality, Virtual Reality, and Computer Graphics*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 431–446. ISBN: 3319609270.

[20] Jeremy Clifton and Stephen Palmisano. "Effects of steering locomotion and teleporting on cybersickness and presence in HMD-based virtual reality". eng. In: *Virtual reality : the journal of the Virtual Reality Society* 24.3 (2019), pp. 453–468. ISSN: 1359-4338.

[21] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and AI: An Introduction*. Vol. abs/1709.00084. 2017. arXiv: 1709.00084. URL: http://arxiv.org/abs/1709.00084.

[22] Tim Cresswell. "Mobilities II: Still". eng. In: *Progress in human geography* 36.5 (2012), pp. 645–653. ISSN: 0309-1325.

[23] Sharon Decker et al. "The evolution of simulation and its contribution to competency". eng. In: *The Journal of continuing education in nursing* 39.2 (Feb. 2008), pp. 74–80. ISSN: 0022-0124. DOI: 10.3928/00220124-20080201-06.

[24] Patrick Dickinson et al. "Virtual reality crowd simulation: effects of agent density on user experience and behaviour". In: 23 (2019), pp. 19–32. DOI: 10.1007/s10055-018-0365-0. URL: https://doi.org/10.1007/s10055-018-0365-0.

[25] Martin Fellendorf and Peter Vortisch. "Microscopic Traffic Flow Simulator VISSIM". In: *International Series in Operations Research and Management Science* 145 (2010), pp. 63–93. ISSN: 08848289. DOI: 10.1007/978-1-4419-6142-6_2. URL: https://link.springer.com/chapter/10.1007/978-1-4419-6142-6{_}2.

[26] Malene Freudendal-Pedersen and Valentina Cuzzocrea. "Cities and Mobilities". eng. In: *City & society* 27.1 (2015), pp. 4–8. ISSN: 0893-0465.

[27] Aron Granberg. *A* Pathfinding Project*. Stockholm, Sweeden, 2021. URL: https://arongranberg.com/astar/.

[28] PTV Group. "Final Assesment of PTV Viswalk". In: *Case Study - PTV Viswalk* (Nov. 2013). URL: https://www.ptvgroup.com/fileadmin/user_upload/Areas_of_Application/Microscopic_Traffic_Simulation/AoA_Microscopic_SuccessStory_ProRail_Amsterdam_EN.pdf.

[29] Farook R. Hamzeh et al. "Understanding the role of "tasks anticipated" in lookahead planning through simulation". In: *Automation in Construction* 49 (2015), pp. 18–26. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2014.09.005. URL: https://www.sciencedirect.com/science/article/pii/S0926580514002039.

[30] Kevin Hannam, Mimi Sheller, and John Urry. "Editorial: Mobilities, Immobilities and Moorings". eng. In: *Mobilities* 1.1 (2006), pp. 1–22. ISSN: 1745-0101.

[31] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[32] Ole B. Jensen, Claus Lassen, and Ida Sofie Gøtzsche Lange, eds. English. 1st ed. Vol. 1. Changing Mobilities. Publisher Copyright: © 2020 selection and editorial matter, Ole B. Jensen, Claus Lassen and Ida Sofie. United Kingdom: Routledge, 2019. ISBN: 978-0-367-18821-4. DOI: 10.4324/9780429198496.

[33] Robert S. Kennedy et al. "Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness". In: *The International Journal of Aviation Psychology* 3.3 (1993), pp. 203–220. DOI: 10.1207/s15327108ijap0303_3. eprint: https://doi.org/10.1207/s15327108ijap0303_3. URL: https://doi.org/10.1207/s15327108ijap0303_3.

[34] UQO Cyberpsychology Lab. *Presence Questionnaire*. Quebec, Canada: Université du Québec en Outaouais, 2004. URL: https://marketinginvolvement.files.wordpress.com/2013/12/pq-presence-questionnaire.pdf.

[35] Nicolò de Luyk, Benjamin Namdarian, and Benjamin Challacombe. "Touching the future: three-dimensional printing facilitates preoperative planning, realistic simulation and enhanced precision in robot-assisted laparoscopic partial nephrectomy". In: *BJU International* 119.4 (2017), pp. 510–512. DOI: https://doi.org/10.1111/bju.13800. eprint: https://bjui-journals.onlinelibrary.wiley.com/doi/pdf/10.1111/bju.13800. URL: https://bjui-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bju.13800.

[36] Dave Mark and Kevin Dill. *GDC Vault - Improving AI Decision Modeling Through Utility Theory*. San Francisco, California, United States, 2010. URL: https://www.gdcvault.com/play/1012410/Improving-AI-Decision-Modeling-Through.

[37] Soraia Raupp Musse, Vinicius Jurinic Cassol, and Daniel Thalmann. "A history of crowd simulation: the past, evolution, and new perspectives". In: *Visual Computer* 37.12 (2021), pp. 3077–3092. ISSN: 01782789. DOI: 10.1007/s00371-021-02252-w. URL: https://doi.org/10.1007/s00371-021-02252-w.

[38] Jakob. Nielsen. *Usability inspection methods*. eng. Computer science/computers - human interaction. New York: John Wiley & Sons, 1994. ISBN: 0471018775.

[39] Jeff Orkin. *Goal-Oriented Action Planning (GOAP)*. URL: https://alumni.media.mit.edu/\~jorkin/goap.html (visited on 05/22/2022).

[40] Jeff Orkin. "Three states and a plan: the AI of FEAR". In: *Game Developers Conference* 2006.1 (2006), pp. 1–18. ISSN: 00099104. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.8551&rep=rep1&type=pdf.

[41] David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents, 2nd Edition*. 2017. URL: https://artint.info/2e/html/ArtInt2e.html (visited on 03/18/2022).

[42] John Porter III and Andrew Robb. "An Analysis of Longitudinal Trends in Consumer Thoughts on Presence and Simulator Sickness in VR Games". eng. In: *Proceedings of the Annual Symposium on computer-human interaction in play*. CHI PLAY '19. ACM, 2019, pp. 277–285. ISBN: 9781450366885.

[43] PTV Group. *How to become the world's most bike-friendly city - Integration of cycling into the existing Copenhagen transport model*. URL: https://www.ptvgroup.com/fileadmin/user_upload/Products/PTV_Vissim/Documents/Success_Stories/PTV_Vissim_SuccessStory_Copenhagen_EN.pdf (visited on 05/24/2022).

[44] PTV Group. *Success story: Main Roads Western Australia - PTV Vissim & the Eelup Roundabout*. URL: https://www.ptvgroup.com/fileadmin/user_upload/Products/PTV_Vissim/Documents/Success_Stories/PTV_Vissim_SuccessStory_Eelup-Roundabout_EN.pdf (visited on 05/24/2022).

[45] PTV Group. *Traffic Simulation Software | PTV Vissim | PTV Group*. URL: https://www.ptvgroup.com/en/solutions/products/ptv-vissim/ (visited on 03/14/2022).

[46] R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2017. URL: https://www.R-project.org/.

[47] Craig W. Reynolds. "Flocks, herds, and schools: A distributed behavioral model". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987* 21.4 (1987), pp. 25–34. DOI: 10.1145/37401.37406.

[48] Craig W. Reynolds. "Steering behaviors for autonomous characters". In: *Proceedings of Game Developers Conference 1999 held in San Jose, California* (1999), pp. 763–782. URL: https://www.red3d.com/cwr/papers/1999/gdc99steer.html.

[49] Dimitrios Saredakis et al. "Factors Associated With Virtual Reality Sickness in Head-Mounted Displays: A Systematic Review and Meta-Analysis". In: *Frontiers in Human Neuroscience* 14 (2020). ISSN: 1662-5161. DOI: 10.3389/fnhum.2020.00096. URL: https://www.frontiersin.org/article/10.3389/fnhum.2020.00096.

[50] Alexander Shoulson et al. "Parameterizing Behavior Trees". In: *Motion in Games*. Ed. by Jan M. Allbeck and Petros Faloutsos. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 144–155. ISBN: 978-3-642-25090-3.

[51] Carson Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN: 9781138331457. URL: https://plotly-r.com.

[52] Alistair Sutcliffe and Brian Gault. "Heuristic evaluation of virtual reality applications". In: *Interacting with Computers* 16.4 (2004), pp. 831–849. ISSN: 09535438. DOI: 10.1016/j.intcom.2004.05.001.

[53] Elham Tabassi et al. "NIST Interagency or Internal Reports (NISTIR) 8269 (Draft): A Taxonomy and Terminology of Adversarial Machine Learning". In: (2019). DOI: https://doi.org/10.6028/NIST.IR.8269-draft.pdf.

[54] TheKiwiCoder. *Behaviour Tree Editor*. United Kingdom, 2021. URL: https://thekiwicoder.com/behaviour-tree-editor/.

[55] S. Mastrolembo Ventura et al. "Evaluation of building use scenarios by crowd simulations and immersive virtual environments: A case study". In: *ISARC 2018 - 35th International Symposium on Automation and Robotics in Construction and International AEC/FM Hackathon: The Future of Building Things* (2018). DOI: 10.22260/ISARC2018/0076.

[56] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN: 978-0-387-98140-6. URL: http://ggplot2.org.

[57] Julius von Willich et al. "Podoportation: Foot-Based Locomotion in Virtual Reality". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2020, 1–14. ISBN: 9781450367080. URL: https://doi-org.zorac.aub.aau.dk/10.1145/3313831.3376626.

[58] Bob G Witmer and Michael J Singer. “Measuring Presence in Virtual Environments: A Presence Questionnaire”. eng. In: *Presence : teleoperators and virtual environment* 7.3 (1998), pp. 225–240. ISSN: 1054-7460. DOI: 10.1162/105474698565686. eprint: https://direct.mit.edu/pvar/article-pdf/7/3/225/1836425/105474698565686.pdf. URL: https://doi.org/10.1162/105474698565686.

[59] Will Wright and Maxis. *The Sims [Computer Software]*. 2000.

[60] Robert Zubek. “Needs-Based AI”. In: *Game Programming Gems 8*. Ed. by Adam Lake. 1st. Boston, MA, USA: Course Technology Press, 2010. Chap. 3.8, pp. 302–311. ISBN: 1-58450-702-0. URL: http://www.zubek.net/robert/publications/Needs-based-AI-draft.pdf.

# Appendix A

# Appendix A - Background Research

## A.1 Mobilities and Immobilities Design

According to Urry et al. mobilities refers to *"... both the large-scale movements of people, objects, capital and information across the world, as well as the more local processes of daily transportation, movement through public space and the travel of material things within everyday life."*[30, p. 1]. Within the field of mobilities, there is another prominent concept that comes along with it, and it is immobilities. The consideration of this concept within mobilities, according to Tim Cresswell [22], comes from the desire of scholars to explore what seems to be the opposite of a particular field, and for mobilities, it seems pretty obvious to be immobilities, which refers to incorporating and designing the spaces in which there will be stillness and waiting as an element in the practice of moving. Mobilities is a complex field of research, possibly because according to many researchers in the field consider the design of mobilities as more than moving from A to B, and discuss the need to take into account other social aspects such as ethnicity, gender, class, sexual orientation, difference in ability, age, among others [22, 30, 32]. Culture has a huge influence in mobilities design, which means that as culture changes, so does the way people move [26]. As a part of an ever changing culture, mobilities also studies how changes in technology affects the way people move. However, as technology changes the way people move, it would also make sense that mobilities design adapted new technologies in the process of creating the experience of moving. Some technologies (see section 1.4) have been developed to simulate mobilities in large and small scales. However, by the time this project was under development, there were no state of the art applications that implemented VR through this process.

## A.2 Simulations

Throughout the years, simulations have been defined in many ways, and by many people. Maria Laura Angelini tried to conceptualize simulations from what scholars all the way from the 1970's to 2019 have written [3]. In this manner, simulations can be defined, to a degree, as a representation of reality itself, or at least what reality could be. This degree, is then defined by a set of norms upon which the simulation model is separated from reality, so that it becomes a system with desired elements of a real environment, often with the goal of finding possible solutions to a specific problem.

Simulations have been used in social studies, medicine, construction, and many other fields of research. Bachen et al. use simulations to help foster empathy in high school students [5]. They use an educational video game called Real Lives, which *"... allows learners s to live simulated lives*

*in other countries and to develop global empathy, identification, and interest in learning about other countries."* [5, p. 438, 43(4)]. Bachen et al. conclude that in comparison with the control group of their study, high school students who played the game developed more global empathy and were more interested in learning about culture in other countries. Sharon et al. emphasizes on the importance of identifying the right norms and desired elements from reality to frame the simulation, so that it can help foster a new point of view on the approach to problem solving the situation [23]. They use simulations to foster different skills in their medical students, that otherwise would have to practice on real patients. From inserting intravenous catheters, to using haptic birthing simulators, which can measure the force applied to a fetus during delivery, these simulations help test these skills. Luyk et al. use simulations in VR to help enhance the skills needed to operating patients in particular practices (robot-assisted laparoscopic partial nepherectomy), allowing to rehearse the procedure before performing it on patients [35]. Hamzeh et al. mention two other important element of simulations, which include the possibility of recreating a possibly dangerous situation without the risks, as well as reducing the cost of having to experience a situation in real life [29].

Imagine having to operate an expensive piece of equipment that not only is composed of delicate and expensive parts, but is also wears out, and you need to teach technicians how to repair it. This is one of the many scenarios in which simulations in VEs can prove useful. However, simulating real life scenarios in VEs could become a difficult task in the following manners:

- The technical skills needed for creating simulations in a VE is often out of the scope of many of the research fields in which simulations can be used.
- The lack of theoretical design knowledge needed to make a meaningful and representative interaction between the participants and model of reality, can render the ecology of a simulation invalid. For example, failing to identify affordances in the real scenario, could lead to failing to design signifiers for the interaction.
- Theoretical knowledge, related to the real life situation to be simulated, could require extensive research on the particular field. If all these conditions need to be met in order to create a simulation, then all simulations need to be created in a multidisciplinary manner.

However, not all simulations have to take place in VEs. As a matter of fact, low-fidelity simulations have existed since before World War II [3]. This distinction between **Low-fidelity (lo-fi)** and **High-fidelity(hi-fi)** simulations is needed for framing the problem covered in this project. Lo-fi simulations refer to simplified models of reality that only incorporate very few chosen elements, and could be as simple as role-playing a situation. In this paper, we refer to hi-fi simulations, as simply simulations, where as many elements of reality as possible are included in the VE, in order to give the user a life-like experience from which they could reflect on a particular presented case [3]. Figure A.1 shows the distinction between hi-fi and lo-fi simulations according to Maria Laura Angelini [3, p. 5].

This paper focuses particularly on a combination of the **Action Learning** and the **Production** simulation types, which according to Maria Laura Angelini features a realistic strategy sprung from designing and creating different scenarios to test different "products" [3]. Figure A.1 show other commonly used cases for simulating dangerous situations without the risk of involving the participants in this kind of danger, such as disaster simulations.

## A.3 VR

The use of Virtual reality (VR) is showing up in many different fields of research and business. It's immersive capabilities has made it a reliable tool for both practicing any sorts of skills, but also as a way of studying human behavior in different scenarios. For example, in relation to human

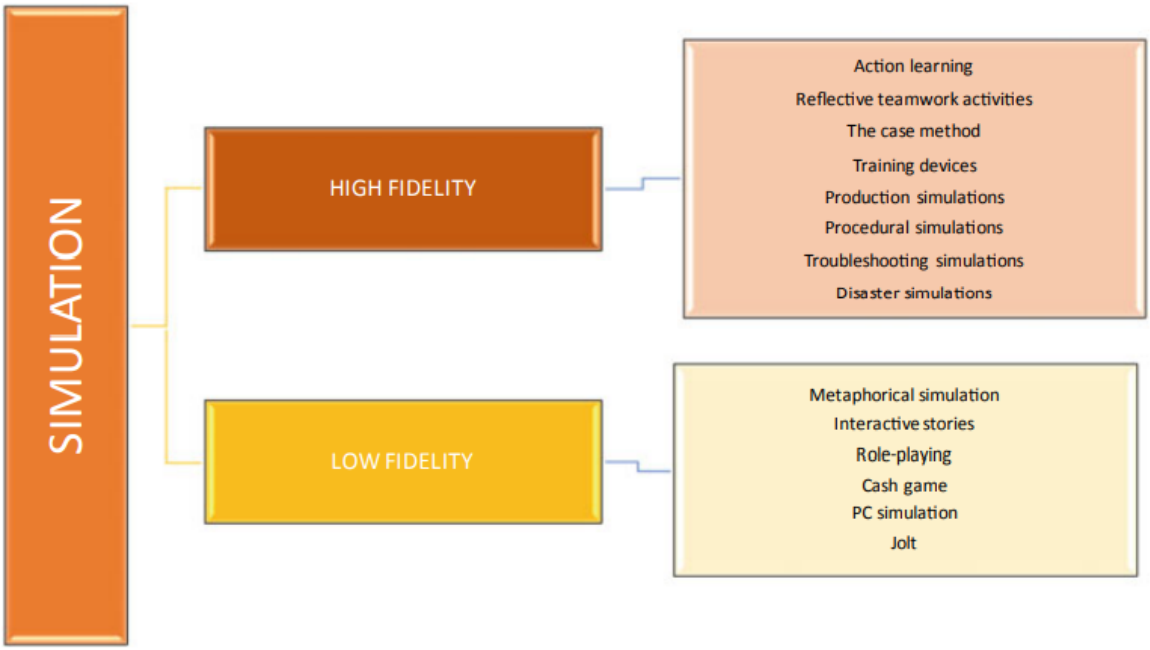



**Figure A.1:** Distinction between simulation types [3]

behavior in dense crowds, a study by Dickinson et al. researched the effects of crowd density on the behavior of human participants in VR [24]. According to their findings, the participants experienced a negative affect and reactive behavior similar to the results of other non-VR human behavior studies, and similar to the participants own prior real-life experiences. According to Dickinson et al. "This opens up new opportunities for the refinement of simulation models along with the application of VR to explore crowd simulation along with human behaviour..." [24, p. 31]. This suggests that the use of VR can be of great beneficial value in the research of crowd simulations, and can realistically show how a crowd can affect the individual human being.

Following up on these "new opportunities" mentioned above, it then becomes interesting to see in what way, these could benefit a real world society. Another paper written by Ventura et al.[55] has an idea for this, as they are investigating how VR can change the architectural design process, specifically the accessibility, spaciness, and usefulness of a room, by simulating the architectural space and its inhabitants in VR. In this study they are acknowledging the capabilities of VR as a training tool for providing realistic feedback for a human user, as they believe the virtual feedback of the crowd sim, and the affect on the user to be applicable in real life. Furthermore, they claim that the use of VR is becoming increasingly common in the industry of engineering, architecture and construction. According to the results, a VR based design tool, combined with a simulated visualized population could be of use in the industry when designing public spaces as a means of simulating and validating the flow of a crowd from both an analytical point of view, but also from the perspective of the end-user who will be populating the building when it's done. This study however, only focuses on the design of the building itself, meaning that the effect of the interior design on the crowd flow, has yet to be revealed.

## A.4 AI

In a broad sense, artificial intelligence (AI) research and development focuses on developing intelligent computational agents that respond to stimuli in a manner that, based on context, can be

considered correct [41, ch. 1.1], e.g. an automated car should be capable of driving from A to B while following traffic regulations and avoiding dangerous situations, a sorting machine using image processing should strive to correctly discern and classify objects, and a virtual opponent in a game should make sound tactical decisions against the player based on some set level of difficulty. It is worth noting, that the concept of Agent encompasses any entity that affects an environment; thermostats, people, countries and weather phenomenon could be considered as agents, intelligence and artificiality notwithstanding [41, ch. 1.1, ch. 2.1]. Various types of AI systems exist, ranging from simple reflexive systems that merely react to immediate stimuli based on some set of rules, expert-defined inference systems that base decisions on some domain-specific knowledge base, and adaptive systems that learn from previous experiences. As such, some AIs may give the impression of behaving intelligently, without necessarily having the capability to reason.

Some AI systems, such as automated cars, need to be aware of other agents in their environment to operate correctly. In one extreme, we may consider competitive AIs that seek to "win" over other agents, or adversarial AIs [53] that intentionally try to sabotage systems or other agents. Conversely, cooperative and collaborative AIs work together with other agents to reach some goal together. Multi-Agent Systems (MAS) [41, ch. 11.1] deal with handling several individual agents that interact within the same world, which also involves interacting with each other, directly as indirectly, and that the total outcome depends on the actions of all agents involved. Many cases can be considered as being a MAS, and crowd simulation is one such domain with more than 30 years behind it [37]. Craig Reynolds innovative, perhaps now classical flocking algorithm for "bird-like objects" can be considered a starting point for crowd simulation, both historically and theoretically [47]. The core behavior of each "boid" is based on three simple rules, used to calculate a motion vector: Avoid collisions with nearby boids (a repulsive force), match velocity with nearby boids (an aligning force), and attempt to stay close to the center mass of nearby boids (an attractive force). The end result is a form of emergent, swarming behavior. Craig Reynolds also proposed dividing the motion behavior of agents into layers, separating action selection, steering, and locomotion of agents into separate parts of a behavioral hierarchy [48]. With steering separated from locomotion, deciding *where* to go now can be defined independently of *how* to go there, and steering behaviors can be reused and recombined across agents with different modes of locomotion. The rules behind boids can be considered as steering behaviors. Craig Reynolds noted that pathfinding algorithms such as A*[31] could form part of a steering behavior specification, but that generated paths would be more analogous to car driving instructions than the actual driving [48].

Numerous refinements and models for MAS have been explored since the 1990's, covering goal-driven decision processes, collision avoidance, adapting psychological models to emulate emotional desires, inter-agent relationships, response to audio-visual ques, synthetic vision, and of course machine learning and neural-network based approaches. These developments have in part been driven by a need to create more responsive, realistic, and valid MAS for simulating and analysing various real-life situations and environments, e.g. crowd flow, emergency egress, traffic systems, and disease transmission. Past events have been used as test cases for the purpose of model validation, and also as input for training new agent models. Digital entertainment has also been a driving factor, seeing a high use for agent-based systems that act and respond to their virtual environment and the players' actions in real-time, and even plan ahead. In the movie industry, MAS has been used for easing the animation of many entities and characters, and for driving visual effects in movies.

## A.5 State of the Art Solutions

### A.5.1 Legion

Developed by Bentley Systems Inc., LEGION® is a pedestrian modeling, simulation and analysis toolkit, centered around two applications: LEGION Model Builder[13], and LEGION Simulator[12]. The model builder is used to generate simulation models and scenarios from architectural CAD drawings. Pedestrian demand, defined areas for queuing and waiting, and travel routes can be planned out and used to generate navigation maps, and characteristics of pedestrians can be defined. Various pedestrian profiles are provided and regularly updated by Bentley. The end output is a model file, containing both 2D models of areas or spaces that can be simulated, and models of pedestrians[8, 9]. The simulator imports these model files to act as foundations for crowd simulations. Based on the models, new simulation runs can be made (on-line analysis), or saved simulations can be played back (off-line analysis). In either case, various metrics can be analysed through numerical and graphical output from the simulation, such as crowd flow, congestion, and journey time. Individual agents can also be inspected mid-simulation[8, 11].

LEGION's agent simulation is described as a microscopic model of individuals, using a learning-adaptive framework[8, 10]. Agents have individual preferences, objectives and interests that they act upon, leading to complex behavior. Navigation of spaces is performed at a macro-level, but agent motion is performed freely by agents on the microscopic level, modelled in a continuous 2D vector space. Motion is driven by what Bentley refer to as the "principle of least effort". Each agent continuously measures their current "dissatisfaction" with their path, based on three metrics: Frustration, their deviation from their preferred speed, Inconvenience, their deviation from their preferred direction, and Discomfort, their deviation from preferred personal space. At each simulation step, in 0.6 second timesteps, agents observes their local environment, and calculates what motion would result in the least amount of dissatisfaction[7]. Additional factors, such as obstacle detection and neighbor awareness (i.e. whether the neighbor is a friend) are also taken into account.

As a whole, LEGION can simulated very advances spaces and scenarios, with agents exhibiting advanced wayfinding behavior, such as multidirectional crossing and navigating through crowds, through large sport events and mass transit stations. Notably, the software has been used for several Olympic Games, the London Underground metro and Beijing Subway, and Dubai Airport[7]. In addition, the LEGION Simulator comes with an API, allowing advanced users to interface the agent simulation engine into their own software. External data can also be supplied from a separate Fire-Dynamics Simulator, so that agents can respond to temperature, visibility and toxicity in a simulation of a fire emergency[7]. LEGION can also connect with Aimsun's transportation simulation software, so that traffic can be included[11]. Simulations can also be rendered in 3D, using the accompanying LEGION 3D application. Alternately, per-agent simulation data can be exported for the purpose of creating 3D visualisations.

### A.5.2 PTV Vissim

PTV Vissim is currently one of the leading tools for traffic simulations, and has been used generously all over the world in order to validate design decisions regarding the infrastructure.[45] Vissim is a microscopic transport model, meaning that it simulates each entity as an individual entity. It mixes multiple detailed simulation models, such as Wiedmanns car following model, a rule-based lane changing model and the social force modeling in order to create a general tool that can simulate a wide variety of scenarios. These can range from a simple pedestrian simulation to something very specific like the bus routes going through your city. The models allow you to

adjust and add specific parameters to the individual entities, that you want to test for, so that you can get the most precise results.

Vissim comes with a wide variety of models to choose from, such as different car models, different kinds of public transport and transport models, different kind of stops, etc. It allows you to quickly and easily set up a virtual representation of your local infrastructure, and allows you to quickly test out different scenarios. Furthermore, it will provide you with valid statistics about the flow of the traffic, such as how likely it is for a car to overturn based on the width of the road, or how much CO2 will be released based on the current layout and traffic density.

As an example, Vissim was used in 2012 in Australia to make the choice of setting up traffic lights in their roundabouts, thereby reducing the yearly amount of traffic accidents on the main roads by 30%.[44] Furthermore, Vissim has also been used in Denmark in order to make the life easier for the bike riders by helping to place new park areas and adjusting the capacity of cycle paths which are heavily used. This has helped Copenhagen to become one of the most bike-friendly cities in the world [43].

Since Vissim is a microscopic model, it means that it requires a lot of detail to set up properly, as the user needs to assign the right properties with the right values to the models, in order to get valid results. According to Martin Fellendorf and Peter Vortisch, Vissim consists of three major building blocks the user needs to implement when setting up the simulation [25]. As shown in figure A.2, without the right information, the results generated by the simulation will not be as valid. The three building blocks require specific information about the infrastructure, such as the exact placement of stop signs, parking lots, traffic lights, as well as an overview of which types of vehicles and routing that exists in the area. Furthermore, even though Vissim is a very detailed tool, it is made for big scale traffic simulations. This also means that the validity of the simulation will decline when focusing on smaller scale simulations such as pedestrian flow in a metro building.

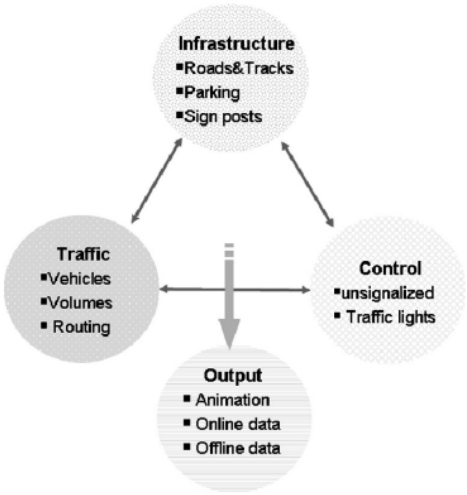



**Figure A.2:** The three building blocks of Vissim. [25]

### A.5.3 PTV Viswalk

This software is a pedestrian simulation software by the company PTV. In this simulations, pedestrians move at particular speeds towards their destination. However, their speed and routes are influenced by other elements present in the VE, such as proximity to other pedestrians, obstacles, buildings, and streets. Additionally, pedestrians can be in a hurry, and their tolerance for proximity depends on the amount of pedestrians present in their surroundings. Their Social Force Model for these pedestrian agents was inspired by Newton Dynamics, and the core principle refers to "... pedestrian movements that are affected by different physical and social forces...", such as the ones aforementioned, as described by the PVT Group in [28, p. 2].

In one of their evaluations of a case study, one of Viswalk's limitations is addressed. Referring to large scale projects, simulations not always run smoothly, and there are many software issues that need the company's intervention to be solved [28]. Nevertheless, the need for pedestrian specific flow simulation solutions seems to be vital in the the field of infrastructure planning. Viswalk can also be used in combination with other PTV Group solutions, such as PTV Vissim in order to unlock features such as interacting with vehicles (buses, cars, bikes) and their flow for crossing streets, and traffic light interaction. One element that might be a consequence of the customizability of Viswalk, is the very complex and overcrowded UI. Figure A.3 shows a snapshot of the many options that are given to the user to stage and run a simulation.

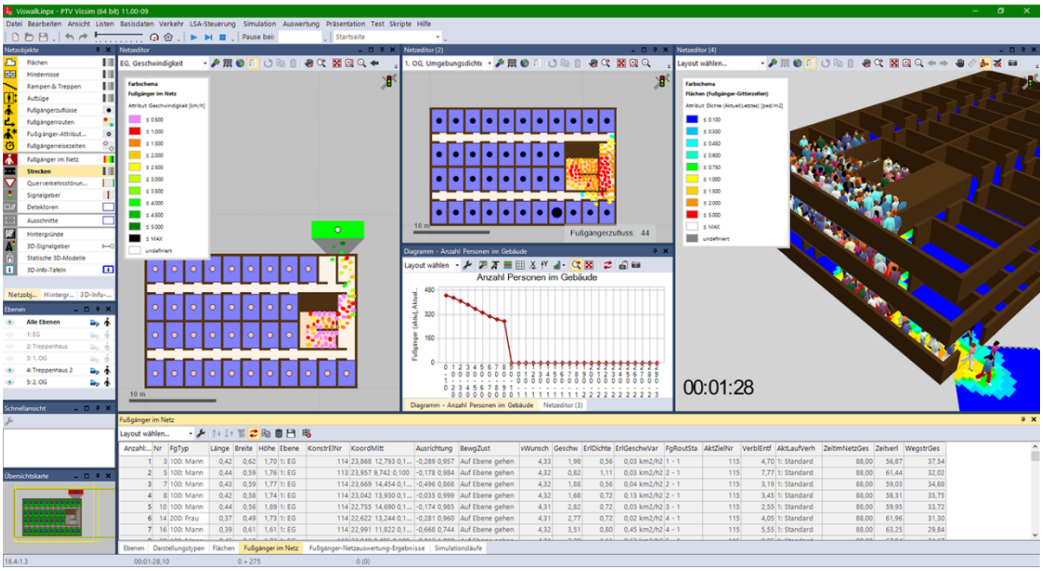



**Figure A.3:** Viswalk UI - `https://www.ptvgroup.com/en/solutions/products/ptv-viswalk/why-ptv-viswalk/`

# Appendix B

# Appendix B - Previous Iterations

## B.1 Stage 0 - A Low-fi Prototype

For the first prototype, focus was on setting the stage for future iterations, getting a sense of the metro station space, and familiarizing with the workings of Unity's NavMesh implementation. An early attempt of an implementation of an agent simply traversing a Navmesh and using stairs was implemented as seen in figure B.1. At this point, the team was working on three different tasks:

- Setting up the navmesh system for the agents to navigate.
- Setting up the virtual environment to make it look like Nørreport Metro Station.
- Making the project ready for VR, so that the user could move around, and interact with the the scene using steering locomotion.

While accurate measurements of Nørreport's metro station were not available at the time, efforts were put into creating a realistic 3D model of the metro station using Blender, and setting up baked static lighting in Unity, as well as proper materials. Unity's Universal Render Pipeline [4] was chosen for the render pipeline. Elevators were not modelled at this point, and escalators only acted as regular staircases. The NavMesh was baked using the standard navigation system offered by Unity. Minor tweaks were performed to the agent baking settings, such as reducing the agent height and radius, and slightly increasing the step-size. A few staircases were fitted with an additional 3D plane to overcome gaps in the generated NavMesh. A simple agent-controller framework was created, capable of sequentially performing simple actions. Only three types of actions were made: navigating towards a defined waypoint using a NavMeshAgent component, randomly selecting between other predefined actions, and self-destructing the agent. Waypoints and Actions had to be defined as independent GameObjects within a scene to be properly referenced. Waypoints were placed at the top and bottom of escalators, and at the top of the entrance staircases. Random selection was used to make agents chose between escalators on two floors. Agents would self-destruct once reaching the exits.

Steering locomotion (continuous movement) was chosen for the VR setup. While the team is aware of the Cybersickness implications that come from using continuous movement, due to the disparity between what is being seen, vs the body not actually moving, this decision was taken to avoid breaking immersion as much as possible, as continuous movement is more realistic than teleporting. The XR action-based control scheme, with Unity's "new input system" was implemented. A body collider capsule was added to the VR rig to avoid the user been able to go through walls.

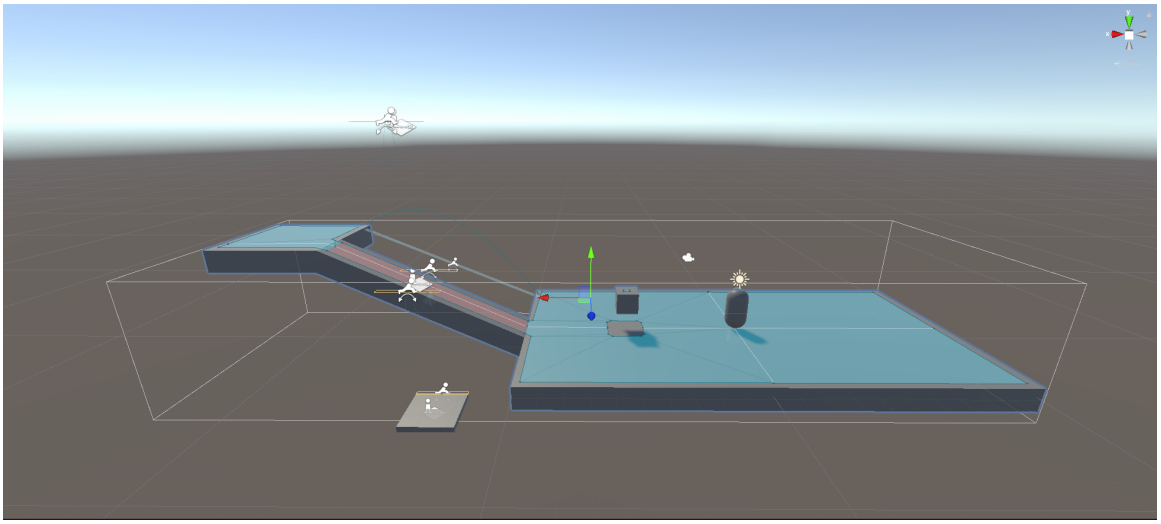


**Figure B.1:** A capsule navigating a Navmesh in Unity

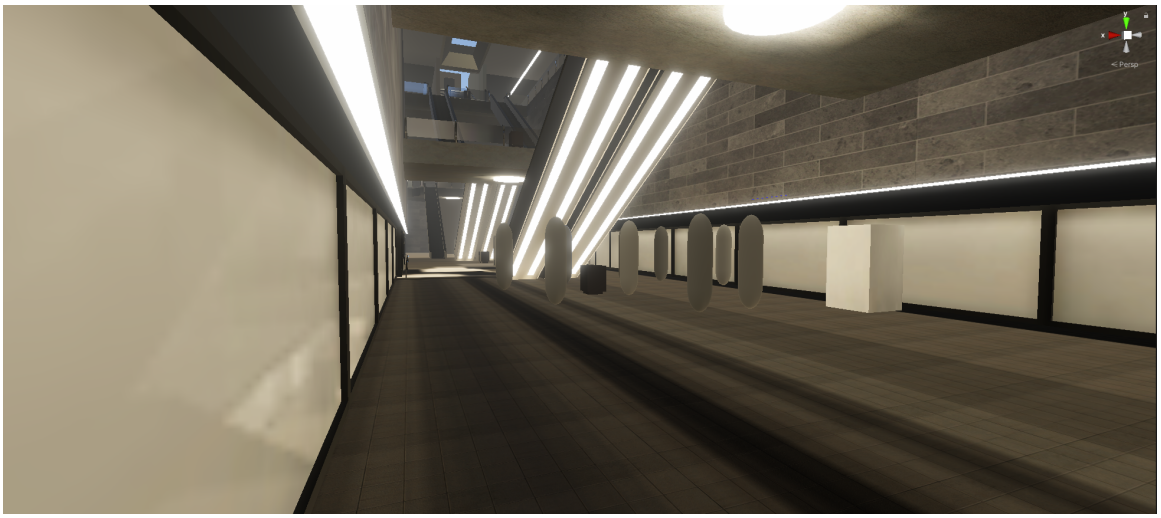


**Figure B.2:** Brainless capsules in Nørreport metro station

A small stress test was run on the first prototype. Starting with 10 agents going from point A to point B using waypoints to get from the platform to one of the exits. Performance-wise, the playthroughs ran smoothly with up to 480 agents. At this point we stopped adding more agents. To see the performance of the CPU-GPU load, we used Unity's Profiler. We saw no significant difference in performance over the five increment of agents going from 10 to 420. However, we saw several problems related to the limitations of the mindless agents, who apart from trying to avoid each other when necessary, would just select waypoints randomly without any regard for the way an average passenger on a metro station would behave, e.g. they would all choose a stairway even though there was another one available, or they would go up the stairway that was for going down, even though they were "not allowed".

This iteration was designed and developed to set the stage to what could be done in a crowd behaviour simulation, and with the idea of presenting the project to MS, so that they could decide if they saw possible use-cases for the product.

After having internally tested stage 0, many limitations were made very clear. From the agents' side, just having to walk from point A to B, with no other functionality than Unity's navmesh

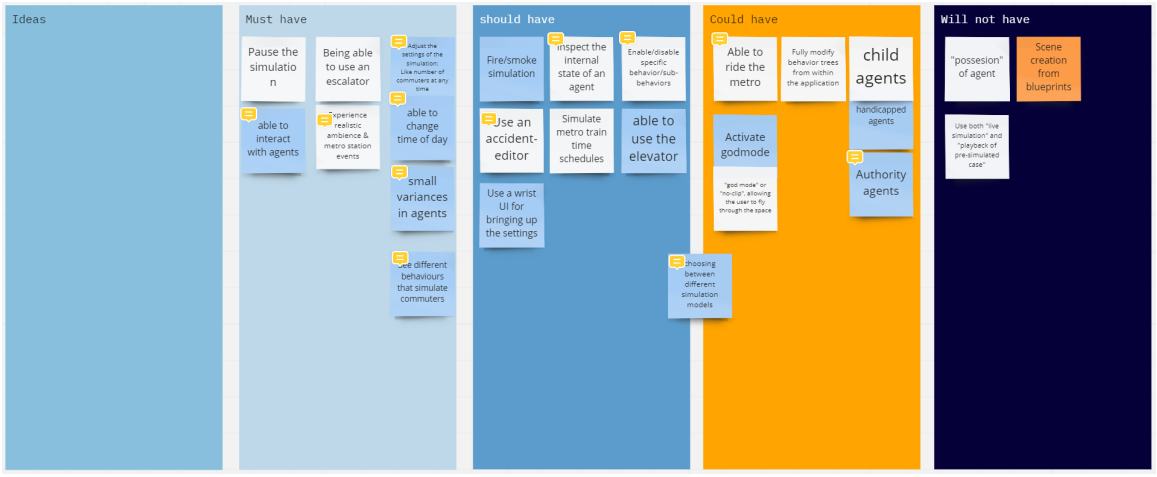



**Figure B.3:** MoSCoW model used for designing the user experience and the technology

agent, was not enough. The randomness of the decision making in this stage, did not fit the realism of a crowd simulation. The agents needed to be more human-like in their decision making, which meant that they needed to be more emotional in making their decision [18]. These considerations carry on towards the design and implementation of Stage-1.

## B.2 Stage 1 - Designing Towards a High-fi Prototype

This section presents the process and considerations for the second iteration of the application. After discussing possible use-cases with MS, the many limitations discovered from testing stage-0 internally, needed to be addressed. Initially, a high level abstraction diagram that was created to guide the design process and the implementation can be seen in figure 2.1. Then, considering figure 2.1 as a starting point, two different MoSCoW boards were developed. The first MoSCoW model to frame the user experience, and the second one to frame the different technologies that were considered when designing the application. Finally, a description of the working elements and what the application could do at this stage. The idea of this iteration, was to make something create something more advanced that what was intended at stage-0, that was presented to MS during the second meeting.

### MoSCoW Model for the User

This section describes in detail the considerations made to frame the user experience, as well as the technology used to achieve that user experience. The process consisted of creating a board full of ideas, and then moving those ideas to either Must, Should, Could, or Won't have(s). Figure B.3 illustrates the process after the ideas had been moved into their respective columns.

For the board containing the considerations that **Must** be present for the user experience, these were described as follows:

- The experience must afford to be paused and resumed at any time of the simulation. This, with the idea of giving the user the opportunity to explore the environment, and see the different paths the agents are taking toward exits, and/or emergency exits, as well as paths from the entrance of the station towards boarding the trains wagons.
- The user must be able to use the escalator. This, because the agents themselves go up and

down the escalators, so for the user to be able to follow the path the agents are taking, they either need to be able to fly within the scene or use the escalator, or both.

- They must be able to interact with the AI Agents. The user's presence must be noticed by the AI agents, and taken into account to avoid collision when possible.
- They must be able to experience a realistic ambience and metro stations like events, i.e. sounds such as announcements, trains arriving, and leaving, etc.
- The user must be able to adjust the settings of the simulation through a diegetic GUI. These options could be to spawn more agents, change the frequency of the trains arriving, making the elevators or escalators non-functional, etc.
- If the simulations varies depending on the time of the day (realistically), then the user must be able to change the time of the day during the simulation. This would allow the user to indirectly affect the settings within the application.

The following were the considerations that the user experience **Should** have:

- The user should be able to simulate an emergency situation in the metro station. These could be a fire, or a smoke alarm that could trigger a state of panic in the AI agents. The implications of having smoke in the stations can affect the spatial perception of the agents.
- The user must be able to inspect the state of the AI agents. Given the agents have properties such as speed, width, height, etc., and possibly emotional states such as fear, calmness, anxiety.
- The user should be able to simulate different models of AI agents behaviour, such as micro and macro models.
- The GUI should be mapped to the user's wrist movement when holding the VR controller.

For the elements that were considered, but were likely to not be included in the implementation of the application, the **Could** have(s) were described as follows:

- The user could be able to "ride" the train. However, this train is meant as a spawn place for the AI agents to exit and path towards the exit, and for the agents coming the exits to enter and despawn.
- The user should be able to experience different types AI agents, with different physiological characteristics. These could be child agents, authority agents (police, fire fighters, staff), differently-abled agents (agents in wheelchairs, blind agents).
- The user could be able to explore the scene without constraints such as gravity, collision, etc.

Some of the considerations that will not be included into this the application at the scope intended for this project includes creating full scenes from blueprints, playing the simulation back, and participating in the simulation as an AI agent.

#### MoSCoW Model for the Technology

Following the pattern presented in section B.2, this section presents the technical considerations for the application. These were different technologies, techniques, and design patterns that could increase performance issues, if there was any.

The technology **Must** make use of the following:

- **Behaviour Trees:** For giving the AI agents intelligent interactions in the form of actions, depending on different states.
- **Navigation Meshes:** The scene must be navigated using Unity's NavMesh system. This implies that the agents will inherit the Navigation Agent behaviour. Additionally, since the "user" was to be taken into consideration for interacting with the agents, the user then must be considered as an agent, or as an obstacle by the AI Agents.
- Navigation logic must be implemented for custom elements of the scene, such as elevators, automatic closing and opening of the doors, mechanical stairs, etc.

The technology **Should** have different ways of dealing with possible bottlenecks in performance using:

- **Smart Objects:** So that the intractability is stored in the objects instead of in a complicated behaviour tree.
- **Draw Call Batching:** To draw all agents at once instead of drawing them one by one, since there could be hundreds at a time.
- Levels of Detail depending on the virtual distance to the objects of the VR camera rig.
- **Fuzzy Logic:** To add "degrees of truth" to the states of the agents.
- The technology should afford the agents to have spatial perception.

More complex concepts and techniques that **Could** allow the application to perform better, as well as create a more realistic behaviour model for the agents :

- **Parametric Behaviour Trees:** These would allow to pass parameters between nodes.
- Setting up the navigation environment to afford the agents jumping over railings, stairs, etc., for when emergency cases are simulated. This could enable the panic behaviours in agents when exiting the station.
- Create a damage system that could enable agents to become injured, or worse. This would impair their movements in cases of emergency, and enable more complex interactions, and use cases to simulate.
- Pre-computed spatial affordances could simulate agents that have prior knowledge of the metro station, which would enable to have agents without prior knowledge that would be more inclined to follow the crowd instead of having a more micro oriented behaviour.
- Using **Vertex Animation Textures** could increase the agent's model animation performance when having hundreds of agents.

Finally, **Data Oriented Technology Stack** is a technology that affords performance by default, due to the way the code, allocation of memory, and distribution of execution is set up. However, it happened to be in early stages of development by the time this study took place, and therefore it will not be used in this application

### B.2.1 Design considerations for the Virtual Environment

This section will go more in depth with the considerations regarding the design of the virtual environment (VE), and how it has progressed throughout the project.

For the first iteration, the VE was designed in Blender from using lots of reference photos, videos and drawings available on the internet. The idea behind this first iteration was to be used as an initial visual representation of the project, which would be send to entice two future external experts, who would provide valuable information and feedback for the project.

From the beginning of the project the question about performance was discussed frequently within the group. As Virtual Reality provides it's own costs and limitations, the use of a multi agent system with hundreds of models running around, could quickly provide a performance problem later on in the process. Therefor, it was important to design with this in mind from the very beginning. For the VE this meant that the polycount would be held to a minimum, and that the fine detailing would be done using textures and normal maps. The biggest focus was put on getting the scale right as well as creating a realistic atmosphere, as this was deemed most important for the purpose of this project, which was to aid the designers in their design process. The atmosphere is created mostly by the light maps which was baked in, in order to save performance, and to use Unity's Global Illumination (GI) system, which provides a more realistic depiction of how light bounces around a scene to create indirect lighting and reflections. Finally, a couple of post processing effects was added to give the last touch of detail, such as the bloom effect, which adds a very aesthetically pleasing glow to the scene.

#### Adapting the VE to the Blueprints

After having send in a short teaser for the project to the external experts, they send back a detailed blueprint of Nørreport Metro, which was then used to update the previous VE in order to make the scale fit as close as possible to the real thing. The blueprint furthermore provided valuable detailed depictions of emergency exits, which was of high importance for the success of the whole project at the time, as a realistic space would be necessary in order to validate the flow of the crowd during emergency situations. Therefore, extra care was put in to make sure that the simulated crowd would have the same prerequisites for maneuvering around the environment, as the people in real life.

At this stage, the trains were also added to the VE in order to further extend the experience of the user, as the crowd now would be able to actually board the train, thereby creating a sense of purpose for the agents. As the metro would have to be somewhat usable by the agents, some functionality also had to be thought into the design of the VE. First of all the space was provided with check-in/check-out terminals, metro rail maps and trashcans, which would naturally be found in a metro station. This adds realism in the sense that the environment looks functional, while also acting as obstacles for the agents to avoid or use to also help heighten the level of realism in the behaviour of the agents as well. Furthermore, for the first iteration the escalators, as well as the elevators, would be static and function only as normal stairs, and mostly as an aesthetic addition to the space in the case of the elevator, in order to quickly implement a simple crowd simulation. However, they where both designed with the functionality in mind, in order to lessen the workload later on in the project.

The VE, as described so far can be seen in figure B.4 and B.5.

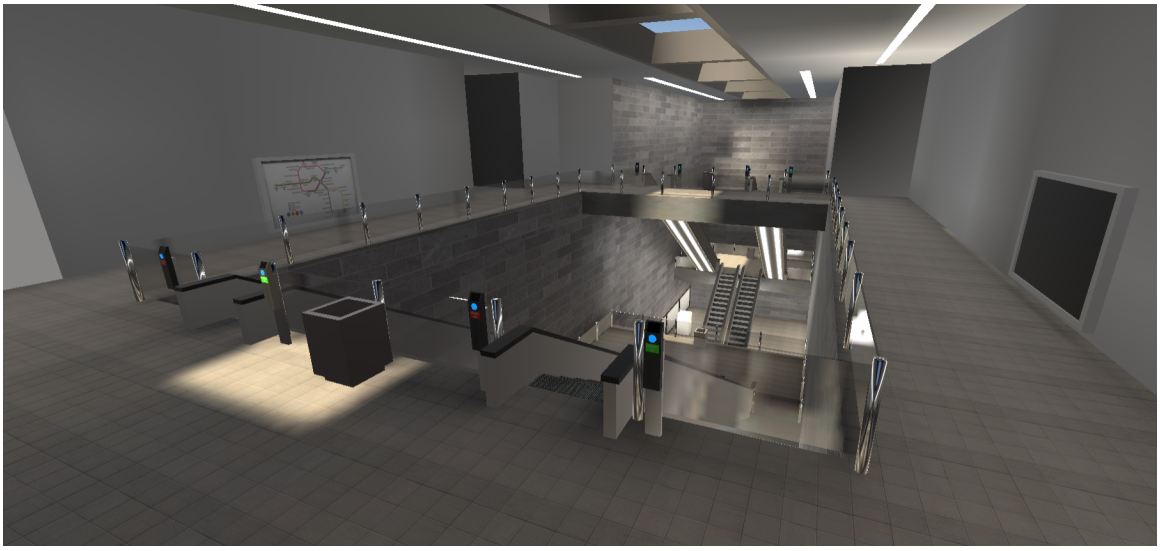


**Figure B.4:** In-game view of the upper level of the metro


**Figure B.5:** In-game view of the lower level of the metro

# Appendix C

# Appendix C - Implementation Considerations

This section presents other considerations researched during the design process for the development of the application.

## C.1 Behaviour Trees versus other AI Techniques

Two major challenges in autonomous system design lie in the creation and design of the underlying system itself, and authoring of behaviors using said system. Previous systems that implemented models such as Finite State Machines (FSM), Hierachical FMS (HFSM), Subsumption Architecture, Teleo-Reactive programs, and Decision Trees (DTs) can be directly compared with BTs [21, ch. 2]. This section presents the rationalization behind using BTs and not some of the other AI techniques, such as the aforementioned.

### C.1.1 Behaviour Trees versus Finite State Machines

FSM are mathematical models of computation that are made of three major building blocks being states, transitions and events [21, ch. 2]. Although FMS are widely used on computer science, and easy to understand (see figure C.1 for an example), they come with major drawbacks as the system grows in complexity.

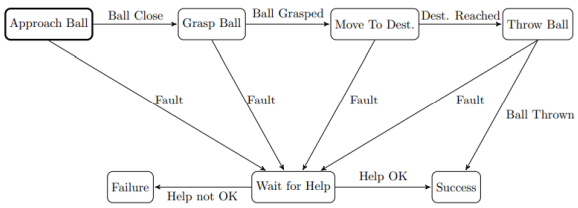



**Figure C.1:** Example of a FSM, defining a sequence of actions to complete a simple task [21, p. 23].

The reactivity/modularity tradeoff stems from the principle that for an FSM to be reactive, it needs many transitions, but each of those transitions are one-way control transfer or *Goto statements*. During these transfers, the execution of the program goes to a different part of the code and continues executing from there; these differ from function calls, where the execution goes to another part of the code, executes it, and then goes back to where the function was called. These

are also known as two-way control transfers [21, ch. 1].

These disadvantages of using FSM instead of BTs can be broken down in terms of:

- Since modifying the states of a FSM requires the system to re-evaluate transitions that increase with the system complexity, using FSM over BTs in our case where behaviour is automated would be impractical, because maintaining state transitions up to date would be too time consuming.
- Managing the transitions of a few of agents at a time might not represent a problem, but when the number of agents is in the hundreds, as it often happens in Nørreport metro station, using FMS over BTs would be too computationally expensive.

### C.1.2 Behaviour Trees versus Hierarchical Finite State Machines

HFSM were developed after identifying the clear shortcoming of FSM in order to reduce the number of transitions needed between states. This is achieved by having *"superstates"* that contain one or more sub-states. Similarly, transition between these superstates are called generalized transitions [21, p. 24]. Figure C.2 shows an example HFSM where modularity can be seen in the "Use weapon" behaviour that separate the task into subtasks.

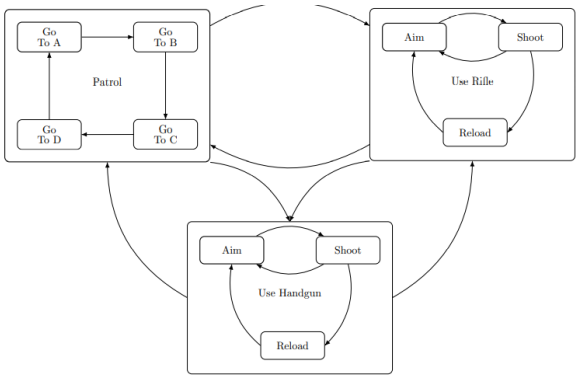
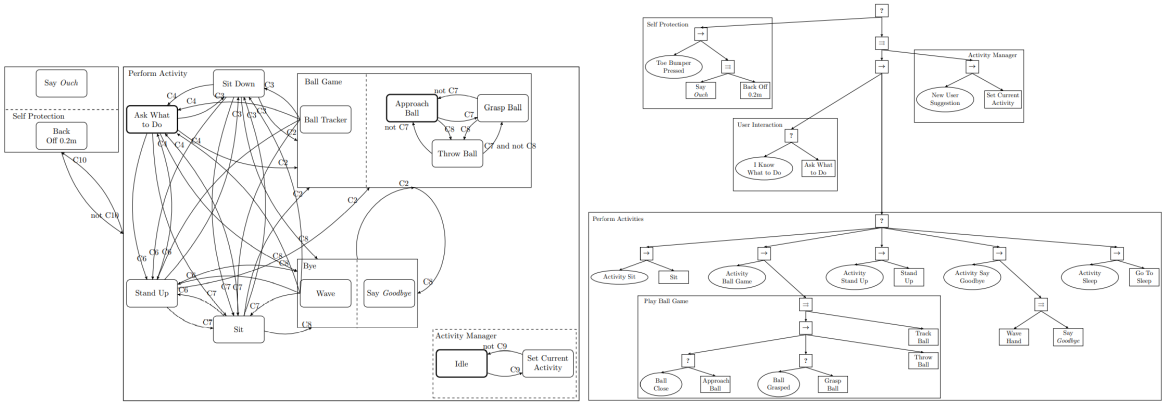



**Figure C.2:** Example of a HFSM, where modularity allows for re-using the "Use gun" behaviour present in two superstates [21, p. 25].

While *"HFSMs is the most similar control architecture to BTs in terms of purpose and use"*[21, p. 26], its maintainability shares disadvantages from FSM. As the HFSM grows, adding or removing states needs to be done explicitly, which means that all nodes need to be connected by a manually created hierarchy. Figure C.3.(a) clearly shows how the conditions to handle transition between states and superstates can get tangled as the system grows. On the other hand, figure C.3.(b) shows how in BTs every subtree represents a module on their own.

## C.2 Fuzzy Logic

The real world is full of things to consider, which are not simply black-and-white matters. In most cases, we can consider things to be some *degree* of true or false, or to partially belong to some categorization. Such categorization is also a matter of context; depending on mode of transportation, the same distance to some destination may be considered as more or less "close by" or "far away". This runs counter to common binary logic, where something either is or is not true or part of a category; that something belongs to a crisp set or not.




**((a))** HFSM where connection between states need to manually handled and grows as the system grows

**((b))** BTs representation of the same system, where each subtree is an independent module

**Figure C.3:** Comparison between HFSM and BTs where BTs in terms of maintainability as the system grows.

As an alternative, fuzzy sets[16, ch. 10] offers a way to define degrees of belonging to a category. Combining fuzzy sets together lead to what is referred as Fuzzy Linguistic Variables (FLVs), which can be used to describe degrees of belonging across multiple categories, as well as a series of operators to find intersections, unions and complements of category memberships. This can be used to set up fuzzy rules as a control system, in which an *antecedent* condition evaluates a fuzzy term or set of terms, leading to a *consequent* response with some degree of activation, which can also be interpreted as the input for evaluating a FLV.

The end result is the ability to create a fuzzy inference system, where given some input values, and expert-defined rule sets working on FLVs that interpret degrees of membership of the input, a conclusion can be inferred and acted upon. This can be a powerful driving element in an agent-based simulation, though with some computational costs. In particular, complex environments lead to a higher number of FLVs, which result in a dramatic increase of rules to create and compute. A strategy known as the Combs Method exists, which reduces the computational growth from exponential to linear, but even then, there are several steps to compute a final inferred response. The above description only covers a small aspect of the theory behind fuzzy sets, fuzzy logic and fuzzy control systems, which is a wide field on its own with many applications.

## C.3 Smart objects

As the amount of possible, complex agent interactions increase in the scene, managing the various assets can become a challenge. Setting up a common framework or interface can aid in making this more manageable, in contrast to defining objects as entirely separate entities with no commonalities, each requiring special handling.

A source of inspiration comes from The Sims[59], which implements a form of needs-based AI[60], using the concept of Smart Objects to manage the implementation of dozens of interactive objects. Here the agents themselves do not contain logic or knowledge on how to use objects; they only posses and process desires and goals. The objects themselves contain the required functionality, and can "advertise" what kind of actions they afford agents, and a promise of a reward from performing said action. The promise does not necessarily match the actual reward; A fridge-object can contain an action that promises reducing hunger, but actually rewards another object (raw food), itself offering a "process food"-action that advertises similar promises, but again rewards another object (cooked food), containing an "eat"-action which finally rewards what is promised. This can lead to an implicit chaining of actions, as if the agent is exhibiting goal-driven

behavior. In addition, actions can list requirements for allowing the agent to perform them; an empty fridge will offer few available actions.

The processing of which actions are most relevant to an agent can be further augmented through Utility Theory[36], where a measure of current usefulness of some action can be based on various parameters; the utility of a hunger-reducing action is higher when the agent has a high hunger score. The relationship between desires and utility also do no need to be linear; the utility of reducing hunger could be modelled as an exponential curve. Utility can also be a function of a combination of other utility scores; buying food could be influence by both hunger and available monetary assets.

Another potential consequence of Smart Objects could be representing agent knowledge. An agent with perfect knowledge of a space would be able to poll every object in the scene, where a more naive agent might require investigating the space in order to consider the actions afforded by before-unknown objects.

Adding such features could afford a increased level of autonomous decision making among agents. It is however quite an undertaking to implement, especially when considering that the simulated setting at its base level would contain rather few intractable objects, primarily elevators and rejsekort-terminals. Some features, such as objects rewarding other objects, could likely be omitted or reduced in scope. The real power of such a system would also be dependant on what kind of parameters (i.e. desires) are used to drive the agents. Some level of custom implementation would also need consideration for one-off goals, such as checking in/out rejsekort. As such, a full implementation of Smart Objects may be overly complicated, but there are aspects of such a system that can prove quite valuable, especially in more complex environments.

## C.4 R Script used for Plotting Data

This section presents the scripts used for plotting the data presented in section 3.2.1 as figure 3.1. Here, *"pos"* is a subset of a dataset containing the agent's ID and position in the 3D VE.

```
1
2  library(plotly)
3
4  #Position on AgentID
5
6  f1 <- list(
7    family = "Arial, sans-serif",
8    size = 12,
9    color = "lightgrey")
10
11 axisz <- list(
12   autotick = FALSE,
13   ticks = "outside",
14   tick0 = 0,
15   ticklen = 5,
16   tickwidth = 5,
17   range = c(-100, 30),
18    titlefont = f1,
19   tickfont = f1)
20
21 axisx <- list(
22   autotick = FALSE,
23   ticks = "outside",
```

```
    tick0 = 0,
    ticklen = 5,
    tickwidth = 5,
    range = c(30, 30),
     titlefont = f1,
    tickfont = f1)

axisy <- list(
   autotick = FALSE,
    ticks = "outside",
    tick0 = 0,
    ticklen = 5,
    tickwidth = 5,
    range = c(-5, 25),
     titlefont = f1,
    tickfont = f1)

scene = list(
   xaxis = axisx,
   yaxis = axisy,
   zaxis = axisz,
   camera = list(eye = list(x = -1.25, y = 1.25, z = 1.25)))

fig <- plot_ly(pos, x = pos$`X position`, y = pos$`Y position`, z = pos$`Z position`, color =
    pos$`Agent ID`, type="scatter3d")
fig <- fig %>% layout(title ="Position on Agents' ID", scene = scene)

fig
```

The following code was used to create the density plot presented in section 3.2.1 as figures 3.2.a and 3.2.b.

```

library(ggplot2)

#Heatmap / Density plot

df2 <- pos_full[pos_full$`Y position` < 0,]
df2 <- df2[c(5, 7)]
df2

#This yeilds the image with the right aspect ratio
p3 <- ggplot(df2, aes(`X position`, `Z position`)) + xlim(-25, 35) + ylim(-50, 10)
p3 + geom_bin_2d() + scale_fill_gradientn(colours = c("darkblue", "red"))

#This yeilds the image with the zoom-in view where more detail can be seen
p2 <- ggplot(df2, aes(`X position`, `Z position`))
p2 + geom_bin_2d() + scale_fill_gradientn(colours = c("darkblue", "red"))
```

# Appendix D

# Appendix D - More on Evaluation

## D.1 Breakdown of Heuristic Evaluation Method

The evaluation is divided into two parts. The first part is a technology audit, where the experts are given permission to familiarize themselves with the system and controls. This audit also serves to establish a baseline for what is reasonable to expect of the system, based on the available interactive aspects of the prototype. This is divided into four categories, which the expert is asked to comment on during the familiarization phase. the four categories are:

- *Operation of the user's presence*: Here the experts comment on the level of presence based on how they are presented in the VE, whether it is a simple cursor, animated hands or a full body avatar, and whether this is sufficient for the purpose of the system.
- *Lack of haptic feedback*: Here it should be discussed how the haptic feedback is handled and if this provides any hindrances, as there are no real sense of touch in a VR experience.
- *Interactive techniques*: Here the expert comments on how the interaction between the user and the world is implemented, and whether these causes any usability problems.
- *Realistic graphics*: Here it should be decided whether the graphics do the application justice, or if they provide any hindrance for the use-case.

After the technology audit, the experts are given a list of tasks to perform during the evaluation. While performing these tasks the experts comment on all major or minor problems encountered, which might hinder them from completing a task or just be a small annoyance.
After the evaluation is completed the problems encountered are assigned to one of these twelve heuristics:

- *Natural engagement*: The degree of realism of the interaction.
- *Compatibility with the user's task and domain*: The expected correspondence of the VE and its behavior to the real world and the affordances of actions.
- *Natural expression of action*: The representation of the user in the VE should not restrict any normal physical actions.
- *Close coordination of action and representation*: The response time of actions should be minimal.
- *Realistic feedback*: Actions performed in the VE should provide realistic feedback which follows the laws of physics.

- *Faithful viewpoints*: The VE should be perceived from the users normal viewpoint without delays.
- *Navigation and orientation support*: The user should not be able to loose orientation.
- *Clear entry and exit points*: It should be clear how to enter and exit the application.
- *Consistent departures*: Design compromises should be consistent and clearly marked.
- *Support for learning*: Active objects should be self-explanatory.
- *Clear turn-taking*: System initiatives should be clearly signalled.
- *Sense of presence*: The user should feel as present in the VE as possible.

Each noted problem will furthermore be assigned to one of six design features, which will show in which part of the system the problem lies:

- Graphics display
- User presence
- Objects and tools
- Environmental features
- Interaction with other controls
- Hardware problems

In the end the experts score all problems on a four point scale based on the severity with *severe*, meaning that the problem caused the user to fail the task, *annoying*, meaning the problem would disrupt the task, but it would be possible to circumvent eventually, *distracting*, meaning the problem could be circumvented relatively easy by providing a small hint and *inconvenient*, meaning that most users would be able to complete the task without any interference. As The final step, each heuristic and design feature is then given an overall score on the same scale based on their related problems.

This provides a great overview of where the problems lie, as well as their severity, which in turn makes it easier to provide solutions and see where the system has the most flaws.

## D.2 Table Showing the Results from The Heuristics Evaluation

This section presents two different tables from the heuristics evaluation of the final prototype.

**Table D.1:** Heuristics and ratings of the problems encountered by the experts during the evaluation

| Heuristic | Rating | Problems encountered |
|---|---|---|
| 1. Natural Engagement | 2 | Participants had troubles opening the menu, and finding the physical button for changing the validators |
| 2. Compatibility with the user's task and domain | 2 | -Participants had difficulties figuring out the sequence to place a validator.<br><br>-They also had problems when having to change a validator, some thought they had to push many buttons at the same time.<br>-Most did not know they had to press the "change validator" button before placing the validator. |
| 3. Natural expression of action | 2 | -Some participants found it very distracting having agents bumping into them because of the overcrowd station, and did not feel comfortable standing on the way of the agents.<br>-They also attempted to teleport to the higher floors/levels wihtout going up the stairs. |
| 4. Realistic feedback | 2 | -Participants found the feedback when placing validators was lacking or confusing.<br>-They also had troubles changing validators, and thought they could change validators that were already placed by pressing the "change validator" button.<br>-The visibility of the validator was obstructed if placed in a crowded area.<br>-Participants could not see what type of validator they had placed without looking at the validator from the front to distinct between check-in and check-out validators.<br>-They also had troubles identifying when the validator could be rotated. |
| 5. Faithful viewpoints | 2 | -The controller overview in the menu was not clear enough, in some places in the VE where the lighting was more dim, it was difficult to read where the buttons where and what they did. |
| 6. Navigation and orientation support | 3 | -Some participants found it difficult to navigate while having the menu open. |
| 7. Support for learning | 2 | -Participants had difficulty finding how to open the menu.<br>-They also had difficulties finding out which button in the controller was for changing the validator. |
| 8. Sense of presence | 4 | -One participant did not know the way up in the physical metro station, so they followed the agents in the VE to get to a higher floor. |

**Table D.2:** Classification, severity, analysis, and proposed design changed for the problems encountered during the expert evaluation

| Feature | Problem description | Rating | Design changes |
|---|---|---|---|
| Graphics display | -The controller overview in the menu is not clear enough.<br>-Too much information in the control scheme picture in the UI. | Annoying | -Change the image to have a clearer view of the Oculus Quest 2 controllers.<br>-Add a background to make the image's readability not dependant on scene lighting.<br>-Control hinting: Hovering the button with the finger would suggest an action afforded by the button - Translucent hand holding a virtual representation of the Quest 2 controller. |
| User presence | -The virtual embodiment of the user do not match their physical affordance to interact with the controllers, and it is thus misleading.<br>-Could not teleport from one floor to another without going up the stairs<br>-Navigation signifiers and affordances are occluded by having the menu open. | Distracting | -Remapping of controls through an iterative co-design approach. Control hinting (see above).<br>-Create an "elevator" or "jump" mode that allows the user to go to the next floor through a fade-out screen.<br>-Hide menu button that keeps the menu and the functionality active, but the menu is not visible or it is minimized. |
| Interaction with objects and tools | -Misleading or lack of feedback and feed-forward when placing validators<br>-Lack of clear signifiers on how and when to interact with validators<br>-Lack of signifiers on how to open the menu<br>-Lack of signifiers and feedback when interacting with the menu | Annoying | -Creating signifiers for objects that afford interaction e.g. pulsating aura from watch and finger that activates it (pairing signifiers).<br>-Change method of interacting with validators to be more contextual: select a validator to enable move/rotate/change of actively selected validator.<br>-Homogenizing the interaction with the buttons in the UI so that they can also be activated by pressing buttons in the controllers.<br>-Feed-forward when hovering UI buttons (they could glow) to signify which button is about to be pressed.<br>-Haptic feedback on UI interaction.<br>-Delay on the interaction with buttons to avoid accidental presses (feedback and feed-forward)<br>-Signify in the UI which validator type is active, and thus will be placed next. e.g. color coding to match green(check-in) red-(check-out) validators.<br>-Making it possible to change a validator that has already been placed.<br>-Redefining the sequence of placing a validator in the VE, so that it can be rotated before being placed. |
| Environmental features | Agent movement and proximity to the user can cause distractions | Annoying | -Make agents fade out when close to user to make them less obtrusive.<br>-Make user not act as an obstacle for agents.<br>-Different method of user embodiment, not on same level as agents, f.x. mid air anchoring. |