Aleksandar Ivanov

EDUCATIONAL AR/VR SYSTEMS FOR MILITARY PROJECTS

Bachelor's thesis
Information Technology

2019

South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Aleksandar Ivanov	Bachelor of Engineering	June 2019

Thesis title

Educational AR/VR Systems for Military Projects — 51 pages

Comissioned by

Observis Oy

Supervisor

Reijo Vuohelainen

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Paragraph 3.

Paragraph 4.

Keywords

CONTENTS

1 INTRODUCTION

Developments and improvements in computing technology have allowed for vastly improved immersion when consuming digital media. The most notable examples of such technologies are augmented reality and virtual reality. The immersion these technologies offer can be used to create educational systems that have more benefits than traditional digital education systems.

This thesis compares the differences between AR (augmented reality) and VR (virtual reality) in the context of educational software. Different implementations and physical devices will be compared and analyzed. A device and a technology will be chosen to create a prototype educational application for Observis Oy related to the company's Situational Awareness System (SAS).

The SAS product has a steep learning curve which raises the need for a more efficient educational tool. A training application that leverages newer technologies is the chosen approach to improve the training process. The primary goal of the training application is to provide a better view on the future prospects of using mixed reality for training. A secondary goal is to offer a training platform for better understanding of the ObSAS (Observis Situational awareness system) software to trainees. Furthermore finding other suitable AR/VR use cases for the current Observis projects can also be a positive by-product of the study.

Advantages and disadvantages of AR/VR need to be considered over more traditional digital educational tools. The future prospects of these technologies are also of interest to the company. One reason for that is the growth of VR and AR popularity and market share over the recent years as the technology has improved (Figure 1).

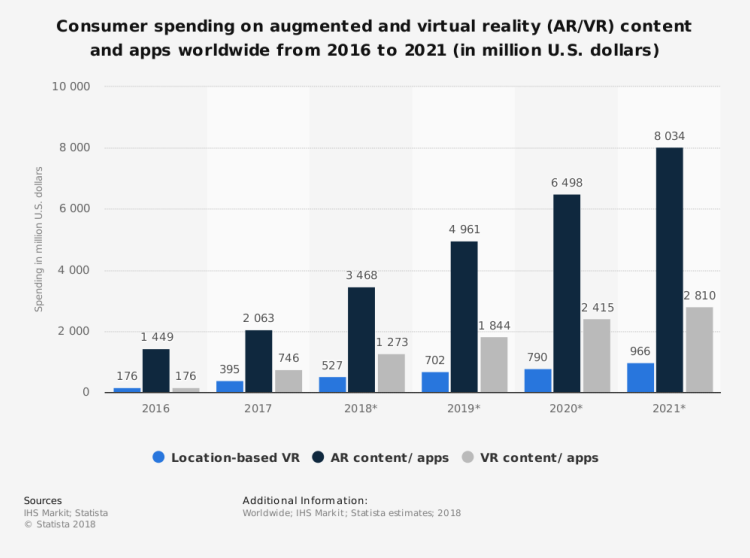

Figure 1. Consumer spending on AR/VR worldwide 2016-2021 (Consumer spending on AR/VR worldwide 2016-2021 2018)

Literature review is used as a research method for the first part of the thesis. The different literature in the fields of AR,VR and CBRN (Chemical, Biological, Radiological, Nuclear) is reviewed and summarized. The second parth of the thesis is a case study of the practical implementation of a training application. The results will be analyzed by debating the success of the final product and its potential use.

2 AUGMENTED REALITY AND VIRTUAL REALITY

Augmented reality and virtual reality are technologies that offer a different view and experience to the physical world. They leverage similar kinds of technology and both aim to provide an enhanced and enriched experience to the user. Both technologies are a part of the general area of mixed reality (Figure 2). However they have different goals and are essentially different in terms of user experience.

Figure 2. Reality-Virtuality Continuum (Paul Milgram et al. 2007)

2.1 Augmented reality

Augmented reality (AR) can be described as the technology that bridges reality with virtual environments. Real life objects are transformed or replaced with virtual equivalents. Information can be added or removed to the real environment. Key aspects of AR are the ability to run in real time, be interactive, three dimensional and combine real with virtual information (Figure 3). AR is most commonly used with the sense of sight, but it can potentially be used with other senses such as hearing, touch, smell, taste, temperature etc. AR can be considered as the next step in graphical user interfaces' (GUI) evolution (Mullen & Mullen 2011). Its current state is comparable to command line interfaces and 2d interfaces in the 1980's and 1990's. It is a vision of future computing and a field that is under research.

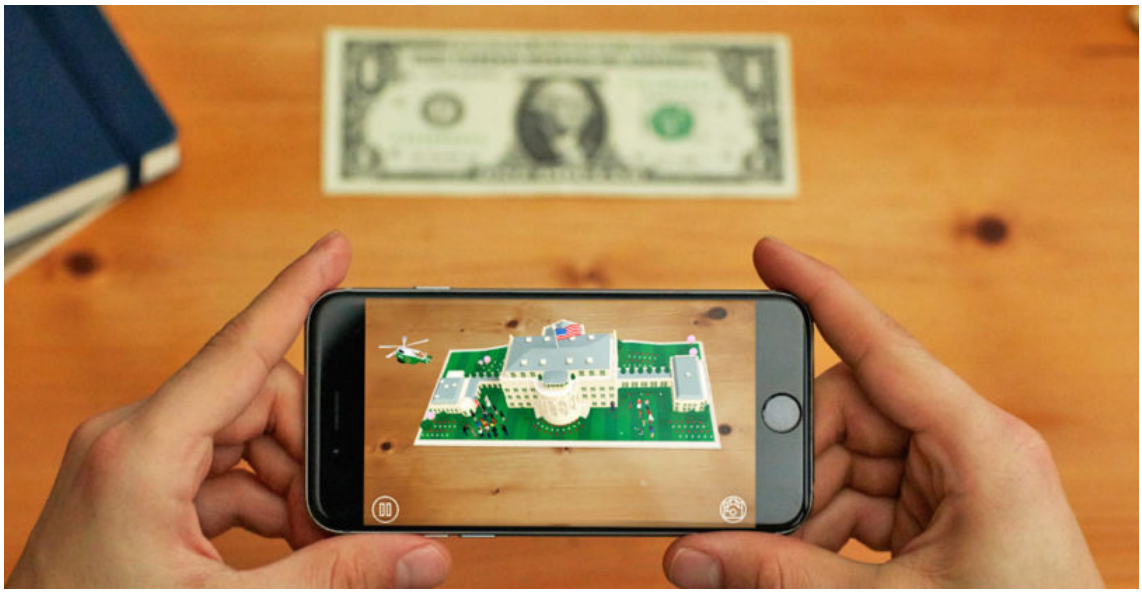

Figure 3. Mobile augmented reality application (Ar experience of the white house 2016)

Augmented reality has higher technological requirements compared to VR which has lead to the slower maturity of AR. AR enabling technologies have been developed throughout the history of which the Optical see-through has become the most popular (Microsoft HoloLens, Google Glass, Intel's Vaunt). Optical see-through is achieved by using opaque displays on which virtual overlays can be rendered. The resolution of the real world is left intact as it passes through the screen. Benefits of this approach include power fail safety, which allows users to still see the real world even during a power outage, cheaper production costs of the used displays, no

parallax effect that irritates the user's eyes. Disadvantages are the reduced visibility and brightness through the opaque lenses, limitation of the field-of-view, requirement of additional tracking sensors such as cameras, gyroscopes and accelerometers. Due to the lack of maturity of other AR enabling technologies only Optical see-through techniques will be considered throughout this work. (van Krevelen & Poelman 2010.)

2.1.1 AR for training and education

Augmented reality provides new paths to conveying information. Learning experiences are more contextual by connecting and embedding information with the real world in real time. These approaches are already utilised by Boeing. Mechanics in the company use AR goggles that aid repairs with embedded textual instructions, illustrate different steps of the repair and help users identify the required tools for a repair. Consequently training resources are reduced and transfer of information between workers is greatly improved. (Johnson et al. 2010.)

Learning through doing is another approach in which AR shines. Mistakes and errors made during the learning experience have no real consequences. This provides for more authentic learning experiences which cannot be achieved easily or cost effectively otherwise. (Kipper et al. 2013.)

2.1.2 AR challenges

An AR framework has basic requirements to accomplish a combination of the real and virtual world. The four main requirements are sensing, tracking, interaction and displaying (Figure 4). Sensing refers to capturing environment events and recognising markers or other objects of interest. Tracking handles updating the viewing direction and position of the user relative to the real world. Tracking is an important component of AR as even a slight tracking error can cause misalignment between the virtual and real world objects. (Wang & Dunston 2007.) Registration refers to how the digital information is delivered to the user. Registration can be achieved through different methods for different senses: videos, audio, haptic

feedback, scent, etc.

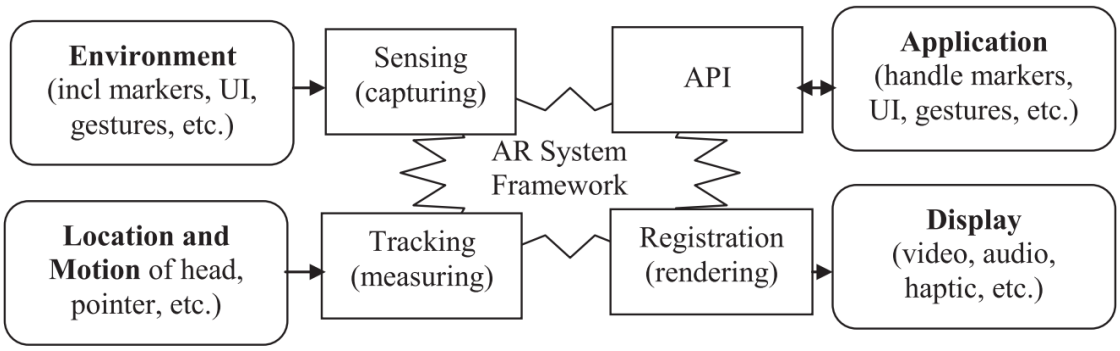

Figure 4. Augmented reality framework (van Krevelen & Poelman 2010.)

As most developing technologies AR, has many challenges that need to be overcome before it can be widely adopted. The challenges of AR arise from the framework requirements, and they can be divided into five groups (Figure 5).

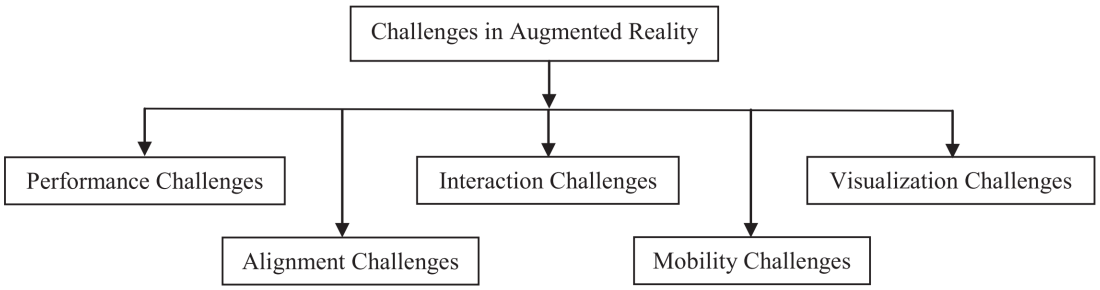

Figure 5. Augmented reality challenges (Rabbi & Ullah 2016.)

Performance challenges arise from the need of real time processing. AR tasks such as marker detection and virtual object rendering are computationally intensive, and this slows down performance. Alignment challenges come from the complexity of tracking the users movement and registration of real life objects. Any errors in these methods can cause misalignment between the rendered objects and real view.

Interaction challenges are concerned with the interaction between users and virtual or real objects. User interfaces need to be intuitive and unobstructive for the best experience. Mobility challenges refer to the need of portability for AR systems.

2.2 Virtual reality

Virtual reality (VR) in the broadest sense is the method or technology of substituting a physical environment with a virtually perceived environment. The Oxford dictionary gives a more detailed definition: ”The computer-generated simulation of a three-dimensional image or environment that can be interacted with in a seemingly real or physical way by a person using special electronic equipment, such as a helmet with a screen inside or gloves fitted with sensors”. This is typically achieved by using an HMD (Head mount display) with integrated motion tracking and a built-in or external rendering unit. The requirements of VR are similar to those of AR with the exclusion of environment sensing (Figure 4). The virtual environment is entirely digitally constructed, thus eliminating the need for sensing as all environmental events are native to the system. This is achieved with a VR headset that has different views for each eye, accomplishing depth perception (Figure 6).

Figure 6. HTC VIVE Pro headset (Htc vive pro review 2018)

Interaction with the environment isn’t necessary for a VR experience, but it increases the possibilities and usability of a VR system. Different approaches to user interaction offer various degrees of immersion and limitations. Motion tracked

controllers, hand motion tracking gloves, pointer centered to the viewport, headset location tracking in 3D space and sekeletal motion tracking are common approaches.

2.2.1 VR challenges

Virtual reality challenges can be categorized in the same way as AR challenges (Figure 5). Performance is a very important factor in keeping the VR experience immersive for a prolonged period of time. High resolution picture rendering at a high framerate in real-time is taxing even for the highest end of current GPUs. Lower framerate or lower resolution picture can cause dizziness, eye tiredeness and overall dissatisfaction. Alignment challenges arise from the difficulty of tracking in real-time. Users' view, location or controllers can become misaligned with the virtual environment, breaking the immersion and interactability. Interaction challenges come from the current hardware limitations on interacting. Tactile feedback, hand interaction and free movement in the 3D space are all possible techniques individually, but combining them at the same time is not yet achievable.

Mobility challenges arise from the need of portability for VR platforms. Common solutions rely on tethering to a machine that handles the rendering and power supply. However, portable VR backpacks and headsets are starting to emerge. A notable example of which is the HP VR backpack (HP Z VR Backpack Overview 2018). Visualization can be difficult due to the complexity of the physical world. Keeping proportions, lighting, textures, view depth, field of vision and other visiual perception aspects can be challenging.

2.2.2 VR for education and training

Virtual reality has many prospects for use in education. Learning can be promoted by interacting with objects and the environment in a virtual world. Similarly self-paced exploration can improve the users' ability to understand a given topic. Learning through doing is even more prominent with VR than AR. Scenarios that are hard or impossible to create with traditional approaches can be executed in a virtual

environment for the fraction of the cost. Users that are physically apart can be in the same virtual environment. Professors or specialists don't need to travel to remote locations to train people.

2.3 Hybrid approaches

Hybrid approaches to VR and AR can be used to create solutions that overcome imposed drawbacks by the techologies. These approaches include using physical objects or combining different sensors. Such a drawback for VR is the lack of tactile feedback or realistic hand interaction with the environment. This becomes a problem when training pilots as they get accustomed to the feel of the virtual environment and are unable to perform tasks as quickly in the real environment. Feedback from the cockpit instruments is essential as well as the ability to use them with peripheral vision. One solution to this problem has been developed by NLR (Royal Netherlands aerospace centre) with the use of a hybrid approach. IR depth sensors and a camera feed are used to map the pilots' hands in the virtual cockpit. Mockups of the flight instruments are 3D printed and are mapped to the virtual environment with tracking markers. This allows for a much more natural feel of the flight instruments and offers better training results. (van Gimst 2018.)

2.4 Simulation training

Simulation training is the process of using a controlled but competitive virtual environment to acquire real world skills. By implication the environment is an imitation of a real environment and the training is an imitiation of a real-life process. The process can also be recorded, analyzed and scored based on the results of the trainee. (Gopinath & Sawyer 1999.)

Simulation training has been used for decades in the fields of healthcare, military and aviation and several benefits have been observed (A brief simulation training history 2019). The simulated environment is safe and secure, which allows trainees to practice and gain experience with the risk of mistakes being eliminated.

Teamwork behaviours such as communication, collaboration, team leading can be improved in practice. Insight into the trainee's behaviour can also be obtained by recording and analyzing training sessions. (Martinali 2018.)

In the military education field simulations are widely adopted. Strategy and tactics games have been used for officer training by the US Army forces. USA Command and General Staff College also uses a turn-based strategy game for their Crops-level operations course. Skill and Team Building has also has benefited from simulation training by using "first person shooter" games. Flight simulation is clearly one of the biggest beneficiary of simulation training according to an extensive study conducted by the US Navy. Higher scores were obtained by students who used the simulation product prior to the early flight training. (Macedonia 2002.)

With so many examples of good simulation training being effective and widely adopted it is possible to conclude that the SAS civil and military defense system can also benefit from such training.

3 PROJECT USE CASE

Observis Oy is developing a product for situational awareness in different environments called ObSAS. This product is a software system running on multiple devices with multiple purposes. The product consists of several software packages with different purposes and providing different features. Depending on the project the required packages are selected and used. One main package is the low level software that communicates with various sensors and devices, such as chemical detectors, radiation sensors, biological detectors. Another package is the server software which summarizes the data and handles automatic actions. User software package allows users to interact with all components of the system. The software packages can be integrated into varying environments and the final complexity of the product installation can also vary. Additionally required hardware and hardware installation support can be provided. Software support and maintenance can also be provided. (ObSAS CBRN system technical description 2019.)

The end use for the product is usually civil defence or shelter management. High

reliability and availability are crucial requirements for the final product as human lives can be put in danger by improper operation of the software or misuse by the users. Training is necessary to ensure proper use of the software and other components of the situational awareness system. So far traditional training approaches have been used, such as powerpoint slides including pictures of the software, textual description of individual components and narration from the training personnel.

3.1 Current training approach

Current training is done on customers' premises in two parts - theoretical and practical. The theoretical part takes place in a classroom with teaching tools such as whiteboard and projector provided by the customer. A group of around 20 people is trained during the theoretical part with the use of powerpoint slides containing screen captures of the software. Additionally, the software enabled in simulation mode is displayed and used in front of the trainees. Devices such as laptops, external hard drives and cameras are not allowed during the training to avoid redistribution of confidential information. (ObSAS Training Material ver 3.0 2019.)

The second part of the training involves practical use of the system. One example project in which practical training was used is the Sassi case. In the Sassi project case the training takes place inside a reconnaissance vehicle with a group of up to four people. Real life devices are used with the software. Test sources are used to trigger devices' alarms and cable disconnection is used to test failure states. Test sources or simulants are compounds or biological samples that are used to verify proper operation of the detection devices. For chemical detection devices such simulants can be samples of acetone, ethanol, MSAL, acetic acid for Toxic Industrial Chemicals(TICs) and DMMP, DIMP for Chemical Warfare Agents(CWAs). The chemical simulants are provided at the air intake of the detection device and the time to alarm from exposure is observed. The time it takes for the device to clear out after the initial exposure is also monitored. These intervals can indicate if a device is operating correctly. The trainees are familiarised with these basic operations of the devices. More advanced operations such as creating reports, marking contamination areas and changing air intake directions are also practiced.

Feedback from the customer in the Sassi project has been positive, although there have been complaints from the trainees about the translation of the materials, but still overall positive feedback has been received. According to Jukka Härkönen(an employee from Observis who conducts the training) the trainees with better English language skills have managed to attain more knowledge from the training. Trainees who were familiar with a previous version of the software have had no problems adjusting to the differences in the new version of the product. Some trainees however have not been able to fully prepare for the use of the software and require further internal training conducted by the customer.

3.2 Future goals

Regardless of the good customer feedback there is room for improvement in the training. Streamlining the training can reduce the time required to create training materials and the amount of staff needed to conduct the training. Another future goal of the training is to be able to experience more lifelike situations and more complex scenarios resulting in better preparedness for critical situations. Misunderstanding caused by a language barrier is something that should be avoided in future training.

Innovation in every aspect of the product is welcomed from the customers. The market has been stagnant and competing companies have been refusing to innovate due to the various risks involved with doing so. Observis is willing to take those risks to set itself apart. A survey and analysis of virtual reality technologies interests the company.

3.3 Defining the application specifications and requirements

A particularly sophisticated requirement of the system is the maintenance procedure. Measurement devices and sensors require periodic maintenance and inspection to remain in good operating condition. Negligence of maintenance can lead to faulty readings over time, device breakdown or even failure to recognise threats. The maintenance involves physical interaction with the measurement device that is

guided by an instructional document. Software use is also required during the procedure to mark used parts and add additional notes. It is not possible to practice all maintenance procedures during training because of the time constraint. Consumables such as filters, batteries, tubings are often needed for the maintenance which can increase the cost of training. For the above-mentioned reasons implementing a training scenario that involves conducting maintenance in a mixed reality environment would be of great benefit.

One particular device that requires maintenance is Environics ChemProDM. The device is a Chemical Warfare Agents (CWA) and Toxic Industrial Chemicals (TIC) detector that is targeted for vehicle use (Figure 7). Typical installation includes two modules: ChemProDM detector and Remote Alarm Unit. A maintenance training scenario for this device is a great candidate for a mixed reality application. Initial installation and testing are also interesting use cases that can benefit from a training application.

Figure 7. ChemProDM and modules (Chemprodm chemical detector for mobile and vehicle applications 2018)

Other devices included in ObSAS are the Thermo SVG2 radiation sensor, Meteo IRDAM MAWS 5060 weather sensor and the Infinicon Hapsite ER chemical

analyzer. Basic knowledge of these devices is a great benefit when using the system. However, it is not required of the users. The user can be familiarized with the devices during training. The knowledge attained can be helpful if on-site repairs are required.

3.4 CBRNe defense

CBRNe is an acronym for Chemical, Biological, Radiological, Nuclear and Explosives. CBRNe defense is the field of detecting, reducing the impact of, or avoiding the threats of CBRNe agents. In recent years the threat of CBRNe attacks has increased due to technological development and increased willingness of terrorists to obtain and use such agents. (Carter 2014.)

A motivation for CBRNe defense is the amount of casualties taken by CBRNe threats all around the world all the time. Such incidents are covered bi-monthly in the CBRNeWORLD magazine and their numbers are not decreasing (Cbrne world 2019). The industry in this field aims to provide solutions that prevent or reduce the impact of CBRN incidents and attacks. Market share of the field has been growing steadily and the forecast is of continuous growth for the future years to come according to Visiongain's market report (Chemical, Biological, Radiological & Nuclear (CBRN) Defence Market Report 2017-2027 cbrn).

ObSAS provides possibilites for early detection and prevention of CBRN attacks. Gathering as much information possible about the attack can help with planning impact reduction and prevention strategies. ObSAS allows for the gathering and analysis of such data: chemical threat identification, air spectrum analysis, radiation measurement, geographical location and meteorological conditions. A vehicle with an ObSAS installation can safely cover and inspect a large area. Communication channels can be used to share and collect data from various CBRN sources. A more clear picture of threats can be established that way.

The CBRN field also encompasses considerations such as decontaminating equipment and clothes, medicine administration, using protective equipment and detection devices require training and practice. Databases providing instructions on

what actions to take depending on the threat are often available in CBRN software. The use of such databases also benefits from prior training.

3.5 Comparing VR and AR in the project context

AR and VR differ not only in hardware but also in software development kits. It is rarely the case that an application written for one of the platforms can be seamlessly ported to the other. Therefore the two technologies need to be compared in regards to the current project. To keep the scope of this thesis limited a single technology will be used to implement a training application.

AR hardware such as glasses and headsets are generally very portable and self-contained. This makes it easy to transport the required hardware during training or installation tirps. The initial setup of the training environment could be more tedious with AR. If real life devices need to be tracked, markers have to be placed and calibrated. As seen from the specification sheet on the Microsoft store (Microsoft HoloLens Device Specifications 2018), the HoloLens have narrow field of view, smaller color space, low luminosity and lower computing power compared to VR headsets. This can result in reduced immersion and engagement with the virtual environment. Another downside is the inability to emulate real life scenarios such as being inside a moving vehicle or simulating an environment that is completely different from the one the user is in. For example, it is impossible to simulate night missions if the user is in a brightly lit room. For these reasons AR is restricted by the environment it is experienced in.

The easibility of the use of AR headsets is one important benefit. While VR headsets require a tether cable, external computer and even base stations in some cases, AR headsets can be used as they are, independently. 3D modelling isn't as big of a requirement compared to VR, as it is enough to create models only of the objects the user interacts with, not the full environment the user is in. This can reduce development time significantly.

VR shines in implementing a completely independant environment. The only limit to what can be recreated is imposed by the hardware performance and development

time and skills. VR allows for spectating and recording users' interaction with the environment. This can be used to analyze their experience and improve the software further. Giving suggestions and guiding during the training is also possible. A multi-user environment is also possible and trainees and teachers can be in the same environment, even if they're physically far apart.

Price difference is very small between VR and AR, if the most advanced frameworks are being considered. An AR headset with the development pack such as Microsoft HoloLens is priced at USD 3000 in the Microsoft store (Buy Microsoft HoloLens 2018). And a VR headset such as HTC Vive Pro is priced at EUR 1500 in the official VIVE store (Buy HTC VIVE Pro 2018) with VR capable laptops priced starting from EUR 1500. Similarly, prices of development kit licenses are not far apart either. This makes the decision between the two based entirely on the features they provide, driven by the requirements of the use case they're aimed for.

3.6 Selection of mixed reality framework

The main benefit of AR over VR in this project is the portability and ease of use it offers. In terms of what training can be implemented VR offers a lot more possibilities. Battery limitations are also reduced or avoided, which enables long training sessions. Good multiplayer support and additional peripherals (such as gloves and controllers) are in favor of VR, too. The portability of AR is not a crucial feature for this project and the usability it offers over VR is not sufficient enough to place AR as a more suitable platform in this project. A conclusion to this chapter is that a VR framework is a more suitable choice in the context of creating a training application for the ObSAS system.

4 PROJECT DEFINITION

This chapter describes the virtual application in more detail and outlines the different technologies used in the implementation. Concepts such as 3D computer graphics, 3D modelling, game engine and tracking will be defined. Source code used in the

development of the application will be provided and explained.

4.1 Researching VR hardware and software

The most notable VR headsets available on the market are Occulus Rift and the successor to HTC VIVE, the HTC VIVE Pro. The widest software and peripheral support is offered to these headsets, therefore other options were not considered. The main difference between the two is that HTC VIVE Pro has higher screen resolution and pixel density (Table 1). Occulust Rift has lighter controllers, but requires at least 3, or even 4 sensors in some cases, for decent room tracking. In contrast HTC VIVE Pro offers a wider tracking area with just 2 tracking sensors as pointed out by WindowsCentral VIVE Pro vs Occulus Rift article (HTC VIVE Pro vs Occulus Rift 2018). Both options provide the required features, but HTC VIVE Pro is more suitable and offers more because of the better graphics.

Table 1. HTC VIVE Pro vs Occulus Rift specs

Category	**HTC VIVE Pro**	**Occulus Rift**
Display	Dual Amoled 3.5	Dual Amoled 3.54"
Resolution	1440x1600(2880x 1600)	1080x1200(2160x1200)
PPI	615	461
FOV	110 degrees	110 degrees
FOV	90Hz	90Hz
Connection	USB-C 3.0 DisplayPort 1.2	HDMI USB-A 2.0 USB-A 3.0

VR hand and finger tracking is a great addition to the immersion of the VR experience. More intuitive interaction with the virtual environment is possible. Many gloves are available as prototypes on the market. Some feature resistive finger feedback and texture and shape simulation like the VRGluv (VRGluv 2018). A large portion of such gloves are in the prototyping stage and can only be preordered. The Noitom hi5 gloves. however, are available as a purchasable product, even though they miss the aforementioned features.

These gloves achieve full finger tracking by using 9 inertial measurement units on each glove. The VR headset controllers are strapped on the gloves to provide

absolute tracking (Figure 8). Vibration feedback is also supported by the hardware and software API, as documented by Noitom on their product website (HI5 VR Glove business edition 2018).

Figure 8. Noitom HI5 VR glove with VIVE trackers (HI5 VR Glove business edition 2018)

The Noitom hi5 gloves are compatible with HTC VIVE Pro and offer SDKs for the most popular game engines (Unreal Engine and Unity). These gloves will be used in the project to achieve better immersion and quality of training.

4.2 Game engines

A game engine is a software framework aimed at simplifying and streamlining the production of games. At the core it provides a 3D or 2D rendering engine, generic physics engine, sound, animation and other features that assist game creation but do not directly specify the game's behavior. If used successfully a game engine can produce a higher quality game with less resources and in less time. An engine isn't required to build applications as all the features can be implemented from the ground up without the need for reusability or modularity, but that takes significantly more resources than using already existing tools. (Lewis & Jacobson 2002.)

Despite having the name game engine, such engines have many other applications besides making games. They can be used for any graphical application. Simulation

applications, training applications, VR and AR applications of any kind, presentations and cinematography can all benefit from using a game engine. Using a game engine is suitable for this project because of the time constraint and limited resources available.

One popular game engine is Unreal Engine, created by Epic Games Inc. It is widely adopted and it offers a large selection of plugins, addons, libraries and assets that aid development. Unreal Engine supports various VR and AR platforms including HTC VIVE Pro and Hi5 vr gloves. This engine will be used to develop the training application.

4.3 Application objectives

The application objectives can be defined as a set of actions that the user needs to complete in a certain order. Using objectives helps quantifying the training progress and give guidance to the user at the same time. Having structured objectives also helps motivate users to continue interacting with the software. (Dondlinger 2007.)

Defining the application objectives also helps the development process. In the context of this 3D application the required assets can be inferred from the application objectives. These assets include 3D models the user interacts with, sound effects, written instructions, 2D graphics and background environment models. Gathering a list of these assets and their description is necessary for outsourcing the 3D modelling work or sound effect composition. Such topics are out of the scope of this thesis, and therefore the production process of the used assets will not be covered.

The training will take place inside a virtual environment. This virtual environment consists of the interior of a vehicle, all detection devices, computational devices, power distrubition and a sampling system. The first objective is to explore the vehicle and become acquainted with the environment. The second objective is to check whether all connectors are plugged at the correct places and inspect all cables for any visual signs of wear. The third objective is to power up the entire system and wait for baseline acquirement of all detection devices. Once that is complete, the next objective is to begin chemical detection tests with simulants. If the simulant

causes a chemical alarm, the user will also be required to acknowledge that alarm and to print a report.

The next objective is to test radio-nuclear detection by using a sample of radioactive rock ore. Again, if an alarm is prompted, the same actions need to be repeated. Afterwards biological detection is tested with simulants. The last objective is to assure that the system has reached baseline operation again and then to shut it down.

With the list of objectives defined in Table 2 it is possible to create a list of the required 3D models. This list can be used to request modelling work, which will be done internally at Observis Oy. Obtaining 3D models of the detection devices proved to be difficult, as companies are not willing to provide these without any ongoing projects related to the devices.

All the objectives take place in a relatively small area, which means they can be a part of a single Level. Unreal Engine Levels are explained in the next section.

Table 2. List of Objectives and required 3D models

#	Description	Models	Compulsory
1	Explore the vehicle	Vehicle	✓
2	Check connections	Wires, Plugs, Sockets, Power/Junction box	✓
3	Power up the system	Power switch, Indication lights, Display and Computer	✓
4	Chemical simulant testing	Chemical simulant, Chemical detector, Air sampling box	✓
5	Chemical Alarm	Keyboard, Mouse, Laptop/Computer	
6	Radiological testing	Radiological detector, radioactive sample	✓
7	Radiological Alarm	Keyboard, Mouse, Laptop/Computer	
8	Biological simulant testing	Biological detector, Biological simulant, Air sampling box	✓
9	Biological Alarm	Keyboard, Mouse, Laptop/Computer	
10	System shutdown	Power switch, Indication lights, Display and Computer	✓

4.4 Unreal Engine level

Levels are defined in Unreal Engine's (UE) documentation as a collection of Static meshes, Volumes, Blueprints and other entities that the user sees and interacts with (Unreal Engine Levels 2019). They are helpful to organize a project into different virtual locations or chronologically. Levels are treated similarly to other assets in UE. They can be created, loaded and modified from the Content Browser.

The full training takes place inside a virtual vehicle, which can be accomplished by using a single level. UE4 comes with starter content when creating a new project. One level that comes with the VR project template is the MotionControllerMap (Figure 9). This level features a VRPawn object which is responsible for the majority of functionalities related to the user. It handles tracking of the Head Mounted Display and syncing camera to it, input from controllers, mapping the controllers' location to static meshes, player positioning in the environment and others.

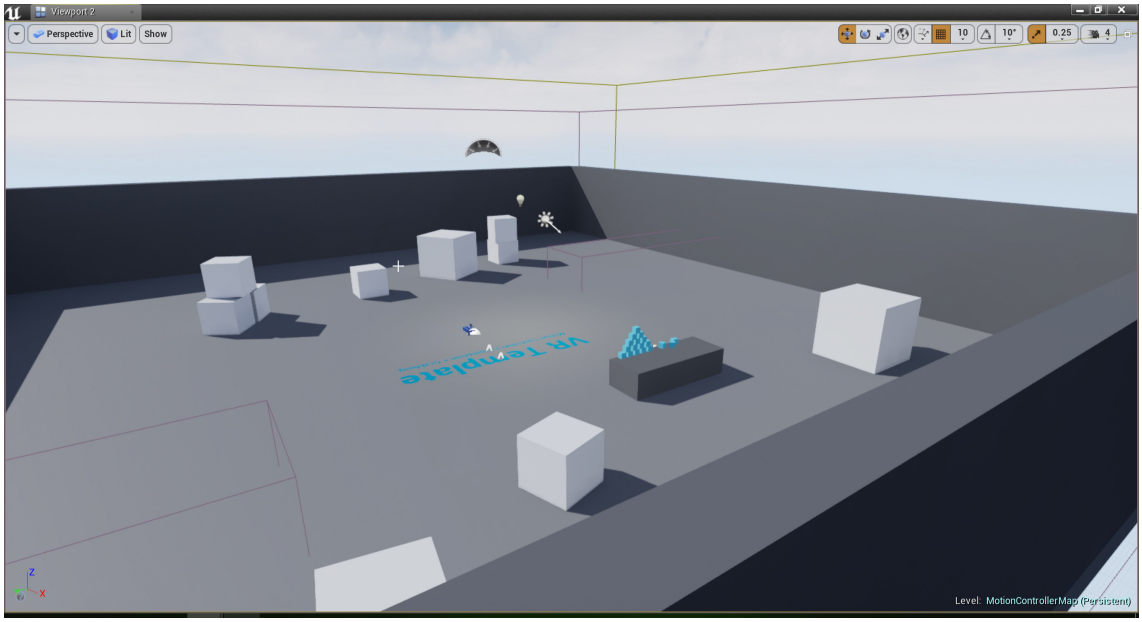

Figure 9. UE4 MotionControllerMap Level Viewport

Other objects in this level are the different light entities which include ambient light, point light, sky light and light interaction and postprocessing. This level also comes with several meshes, some of which have simulated physics interaction. And the last type of components in this level are the NavMeshBoundsVolume and NavModifierVolume entities which define what areas in the level the user can walk

on.

This starter content level provides all the basic functionalities needed for this project. It can be used as a starting point for the implementation and further extended and modified to contain the full training. In the future more levels can be added for different locations and scenarios.

4.4.1 Actors

Actors are generic Classes in UE that support 3D transformations such as transformation, rotation and scaling. These actors can be created (spawned) and destroyed dynamically while the application is running. They can also be updated dynamically by "ticking", which means that they can be updated with every rendered frame(tick). Another main function of Actors is replication and controlling behaviour of objects in network play(multiplayer). Actors hold different types of objects which are called Components, in a sense Actors are containers for Components. The main types of Components Actors can hold are UActorComponent, USceneComponent and UPrimitiveComponent. UActorComponents have conceptual functionality. They don’t exist anywhere in the world. They can be used for example to implement AI functionality, handle user input and saving or loading game progress. USceneComponents are placed in the world and have a location, scale and rotation parameters. UPrimitiveComponents have graphical presentation from a static mesh, particle system, 2D graphics or other primitive assets. Physics, collision and overlap settings are available for these components.

Some examples of actors in this project are StaticMeshActors of the different detection devices, CameraActor responsible for what the player sees, and PlayerStartActor that defines the player starting point. Actors are also hierarchical with one Root Actor having nested actors on several levels. Controlling the actors can be done with gameplay code written in C++ or with blueprints. (Unreal Engine Documentation 2019.)

4.4.2 Blueprints

Blueprint is a visual scripting tool in Unreal Engine that offers the full range of development tools and concepts generally available only to programmers. A node-based visual interface is used in which different Events, Nodes, Functions and Variables are connected with Wires (Figure 10). These connections can define complex gameplay logic by combining and chaning multiple simple operations and creating conditional logic. Objects created with this visual tool are referred to as Blueprints. (Unreal Engine Documentation Introduction to Blueprints 2019.)

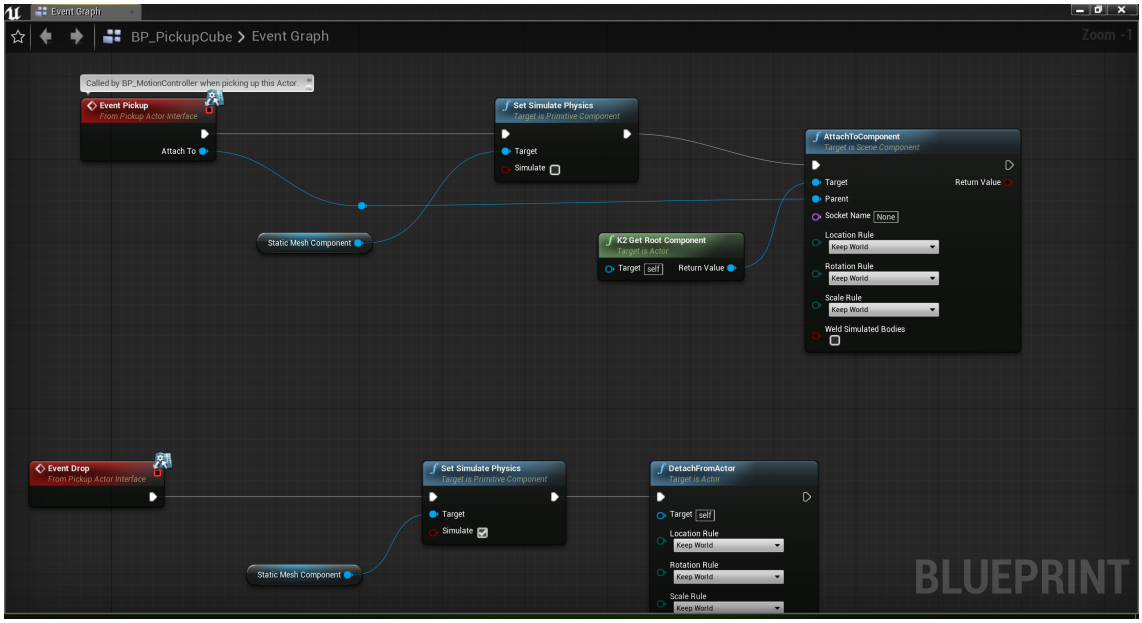

Figure 10. UE4 Blueprint Visual Editor

Blueprints are defined as object-oriented (OO) classes or objects in the engine, which is a well known and used programming paradigm. Internally Blueprints are translated to C++ by a VM which comes with a performance penalty (Unreal Engine Nativizing Blueprints 2019). An alternative to Blueprints is using C++ to define the game logic. However in this project Blueprints are favored due to their ability to visually display logic better than code. There are also more examples and snippets online available for blueprints.

Two common types of Blueprints are Level Blueprints and Blueprint Classes. Every level has a Level Blueprint which can manage things like level streaming, objective tracking (checkpoints), interacting with other blueprints in the level. Blueprint

Classes, on the other hand, are used to create interactive assets such as doors, buttons, levers and destructible objects. The main benefit of these blueprints is their reusability. They can be placed in any level and their behavior will remain the same. Also modifying the base blueprint will update all instances of it in the level.

4.5 Physics simulation

Unreal Engine provides physics simulation to make experiences more realistic and believable. The physics simulation is done through NVIDIA's PhysX 3.3 physics simulation engine. The main physics simulation tools used in this project are Collision simulations and Traces with Raycasts.

Objects in the scene have physics and collision settings. These settings are enabled for individual objects rather than for the scene as a whole. The three possible collision response settings are blocking, overlapping or ignoring (Figure 11). When blocking is enabled on two colliding actors, a hit event will be generated. The two actors are unable to overlap each other and will simply bounce away from each other. If both colliding actors in stead use overlapping, they will go through each other. If overlap events are enabled for both, overlapping events will be generated when the collision begins and ends. If colliding actors use ignoring collision they will go through each other and generate no events.

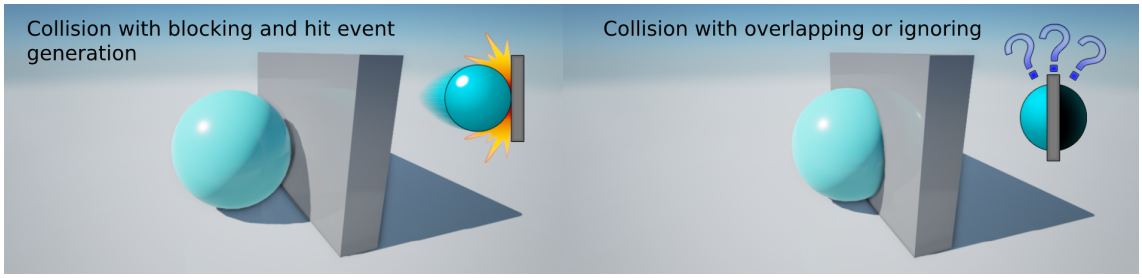

Figure 11. Collision types in Unreal Engine

Traces are straight rays that detect any interescted objects between two points. Intersection of objects generates a trace hit event which supplies a list of the intersected objects. Traces can be ran by channel in which case only objects that are responding to the same channel will be a part of the trace hit event. Additionally,

single hit and multi hit events can be generated depending on how many times the same object is intersected by the ray. The Trace ray can have a shape of a line, box or sphere. (Unreal Engine Traces with Raycasting 2019.)

4.6 Static and Skeletal meshes

A Static mesh is a geometric shape made up of polygons that is cached in video memory and rendered by the graphics card. Being cached in the video memory means that these geometries can be complex and also translated, scaled and rotated in real time. Static meshes are usually 3D models created in external modelling software and imported to an UE4 project. But primitive, meshes such as boxes, cylinders and spheres, can also be made in UE4. Objects in the scene such as doors, sensors, batteries and lamps are examples of Static meshes. (Unreal Engine Static Meshes 2019.)

Skeletal meshes are identical to Static meshes with the addition of a hierarchical set of interconnected bones that can be used to animate the meshes' vertices. The Static meshes, rigging and animations are typically created in external tools like 3DSMax, Maya, Blender (Unreal Engine Skeletal Meshes 2019). These types of meshes are used to create animated objects. In this project the Hi5 VR Gloves are displayed virtually with the use of Skeletal meshes and animations. VIVE controllers' visualization is also achieved by animating Skeletal meshes of hands.

UE Materials define what type the surface of a mesh appears to be. It can be seen as a "paint" applied to a mesh. Materials can define the colour, shininess, roughness, opacity, light emissiveness and many other surface properties. In technical terms a Material calculates how light interacts with a surface. Materials are constructed with a visual scripting tool named Material Editor that is similar to the Blueprint Editor. (Unreal Engine Materials 2019.)

4.7 Widget Components

Unreal Motion Graphics UI Designer (UMG) is a visual editing tool used to create 2D interfaces for the Unreal Engine. This is often used to create widgets such as HUDs (Head-up displays), menus, 2D animations and user input fields. Some examples of UMG elements are checkboxes, text, buttons, progress bars, 2D images and sliders. The designer has two main views: designer view which allows the user to edit widgets and also see how they will look in the application, and a Graph view which is used to assign logic and functionality to the widgets.

Widgets created with UMG can be added to a scene in two different ways. The first method is to create a new widget instance of the widget and add it to Viewport. In this case the widget will be truly 2D, as it will be always facing the camera at the same angle and from the same position. This approach is useful for creating UI interfaces that need to be constantly present on the screen, like game score, healthbars or time counters. The second method for introducing UMG Widgets to the scene is by using a Widget Component. The Widget Component overlays the 2D UMG Widget on a 3D plane. The Widget Component has 3D world transform, meaning it has a set position and orientation in the scene. The 2D interface is projected on the surface of the Widget Component plane similarly to a Material. However, this approach is more suitable for creating assets of TVs, pictures, signs than using Materials, due to the simpler management of Widgets for 2D interfaces. (Unreal Motion Graphics UI Designer 2019.)

5 PROJECT IMPLEMENTATION

This chapter covers the practical implementation of the project application. Level layout and design are created and explained. Blueprints defining the game logic are also created and explained. Other various steps of the implementation process are listed and discussed.

5.1 Project setup

The project application will be developed in the Unreal Studio software. Unreal Studio comes with a new Project wizard and several templates for new projects. The template most suitable to this project is the Virtual reality Blueprint template. It comes with optimized graphics settings for VR and example blueprints for common VR operations (Figure 12).

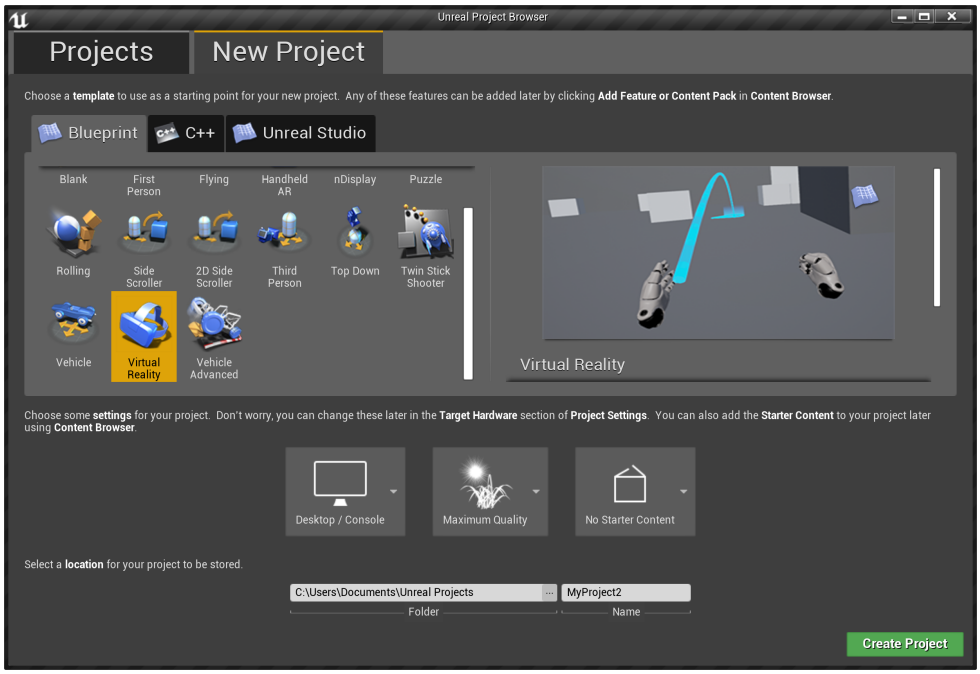

Figure 12. Unreal Studio new project wizard

General project preferences can be selected for the project during the creation process. Target platform for the application and rendering quality options are available. These options pre-configure the project to best suite the needs of the application. Changes can be made to the project settings later on, if needed. It is possible to add starter content to the project which includes several levels, static meshes and blueprints and can be ran straightaway as a VR demo application.

5.2 Level design

Level design begins with the layout and orientation of the different meshes. The visual appearance of the level is made more realistic by using different textures,

materials and lighting. A vehicle 3D model is used as the center point for the level and also for the player starting point.

The player starting point is placed in the interior of a conversion van (Figure 13). If the starting point Actor is colliding with objects in the scene an icon "BADsize" will appear. This means that the starting point is invalid and the engine will try to find the nearest valid starting point. During the development process the camera location in the editor can be used as a strating point which speeds up testing. (Unreal Engine Player Start 2019.)

Figure 13. Starting Level Vehicle Interior

Light sources are required to make meshes in the scene visible to the user. Two primary light sources are used in this level. The first light source is a Directional light and is used to simulate the sun. Directional lights simulate a light source that is infinitely far away, which means all the shadows cast by the light are parallel. This makes Directional lights perfect for simulating sunlight. This light is also moveable, which means its location and orientation can change on runtime. This can be used to cast dynamic shadows, but in this project dynamic shadows will be disabled to improve the rendering performance. The second light source is located inside the vehicle and is used for interior lighting. The type of this light is Point light. Point lights are used to simulate light bulbs, because they emit light in all directions. (Unreal Engine Light Sources 2019.)

A SkySphere (Skydome) is used to create a background for the scene that simulates a sky. While the player is moving in the scene the SkySphere remains stationary, which gives the illusion that the sky is very far away. A 2D texture is wrapped on the inside of a sphere to create a SkySphere. The SkySphere in this project is also dynamic. Clouds are moving and the Directional Light source mentioned in the previous paragraph is moved by the SkySphere to simulate day and night cycles. The intensity and color of the light source is also changed depending on the time of the day.

5.3 Glove calibration and tracking

Hi5 VR Gloves require the user to go through a calibration process before use to ensure better finger tracking. This process needs to be done, if the gloves have been turned on or a new person is going to be using the gloves. Noitom, the creators of the gloves, provide a sample UE4 project that includes the calibration process with instructions for the gloves. The calibration can be done with the stand-alone sample project, but to keep the final project self-contained the calibration will be added to the application.

Inspecting the assets of the sample project reveals that several actors are responsible for the instructions and calibration of the gloves. The UE4 editor comes with an asset migration tool which is used in this case to migrate these actors to the ObSAS VR project. The MainLogicActor contains instructions and the menu with options to calibrate the gloves. It uses WidgetComponent children for the instructions and selection buttons (Figure 14). Glove battery and magnetisation status are also shown. The InteractionPawn Blueprint Class is mainly responsible for the camera projection. The camera view is bound to the tracked real world location of the VR headset. This means that, if the user moves or rotates in the real world, the camera view will update its orientation and position and render different parts of the scene. A WidgetInteractionComponent is a part of this Class. This component contains a Static mesh in the shape of a small square. This mesh is used as a pointer for the user, it is located in the center of the camera projection at all time. If the user positions the camera in such a way that this pointer overlaps a menu option,

a countdown timer will start. If the pointer is not moved outside the menu option, the countdown will end and the option will be selected (Figure 15).

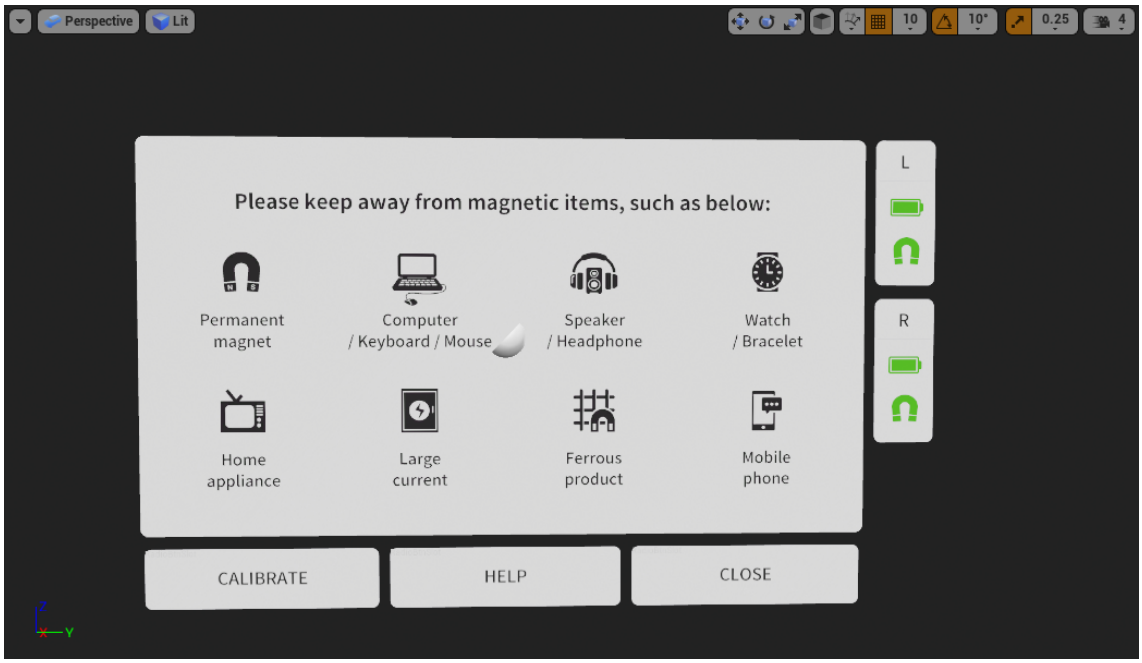

Figure 14. Glove calibration menu and instructions (HI5 VR Glove User Guide 2018)

The calibration process requires the user to perform two hand gestures for a period of time. Calibration data is used and created by the Mocap plugin. GameInstance Blueprint Class is responsible for starting and stopping the Motion Capture service from the Mocap Plugin.

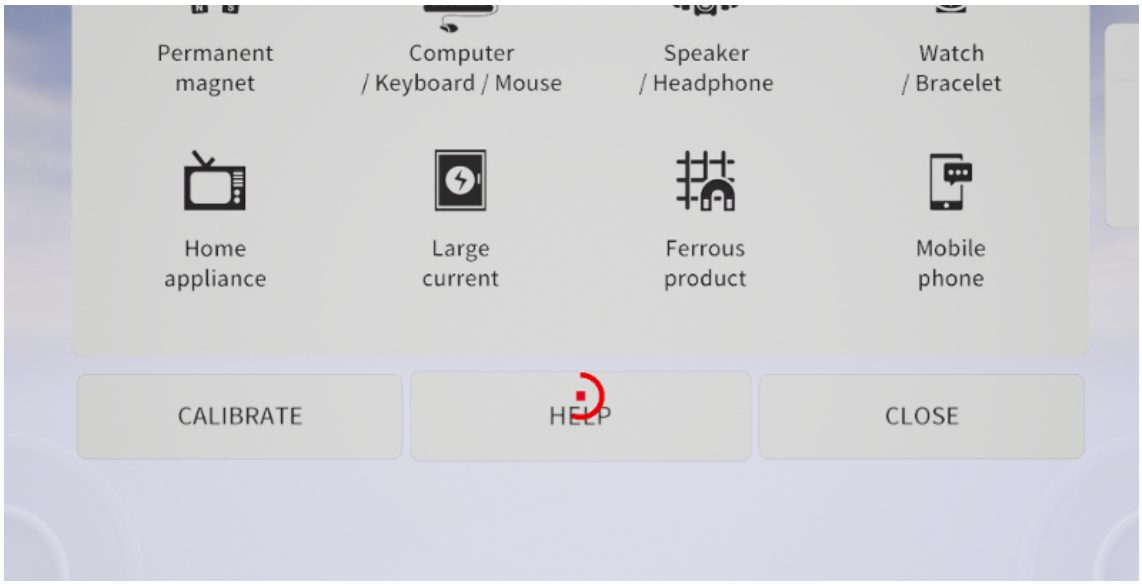

Figure 15. Camera pointer and menu selection (HI5 VR Glove User Guide 2018)

The Hi5 Mocap UE4 plugin is provided by the glove manifacturers. It provides relative tracking of the integrated sensors on the gloves. Combining this relative data with the absolute positioning data from the VIVE Trackers full hand and finger tracking is achieved in VR. Other features, such as acquiring joint data, saving and loading calibration data and getting ID of paired tracker to the glove are available from this plugin.

5.4 Interacting with the virtual environment

Interacting with the virtual environment is accomplished in three different ways. Firstly the camera pointer method described in the previous chapter is used. Secondly the Hi5 VR Glove is used to handle objects. Lastly the VIVE controllers are used to grab objects.

The VIVE controller interaction method is required as a backup to the Hi5 VR Gloves. Magnetic interference or radio frequency interference can make the gloves' tracking inaccurate, unreliable or even completely prevent it. This is not a huge problem during training, as the environment can be controlled and time is not limited. Furthermore, interference can be removed and the gloves can be demagnetized and recalibrated. However, if the application is used in public exhibitions, interference can be a real problem.

5.4.1 Gloves interaction

Using gloves to interact with objects in the virtual environment can be achieved with simulated physics. The gloves are tracked and represented in the scene with a Hands Actor. The hands' meshes in the Hands Actor are configured with a collision preset of Block All Dynamic. As explained in Section 4.5, this means that other meshes with the same collision preset will not be able to go through(overlap) the hands' meshes. Instead, a hit event will be generated, when a collision occurs and the colliding actors will be pushed away with a force. The force depends on the velocity, mass and force modifier property of the colliding meshes.

Holding and moving objects becomes possible with these settings. The interaction is similar to real life hand interaction (Figure 16). Hands and objects block each other on the surface level and do not overlap. One main difference is the haptic feedback real life interaction provides. A person feels the surface of objects, but in VR it is only feasible to have visual feedback. Hi5 VR Gloves have vibration feedback, but this is not isolated to individual fingers, it rather affects the whole glove. This makes vibration feedback unusable for feeling surfaces.

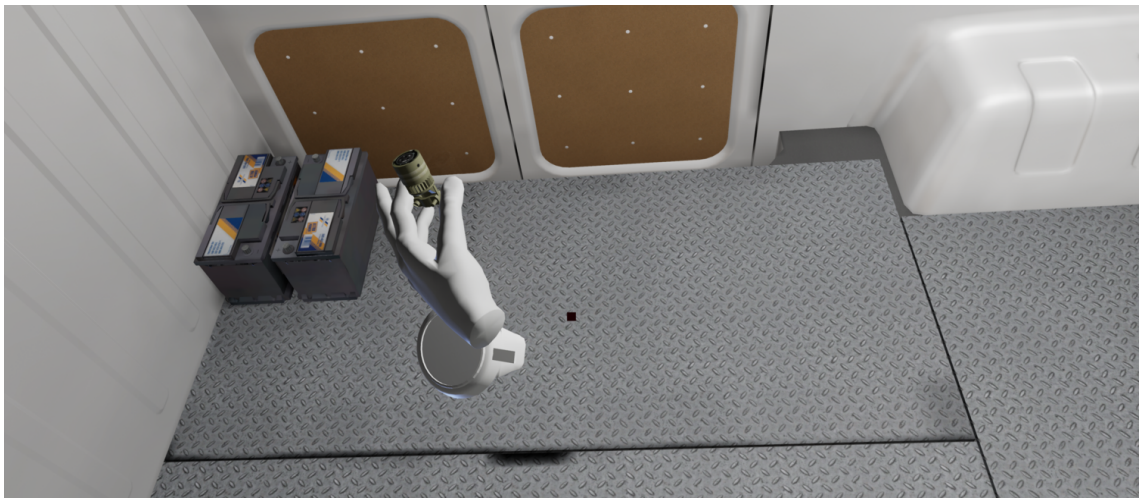

Figure 16. Holding object with gloves

Collision simulation is computationally expensive and also not deterministic. Objects with locked position such as buttons and levers can behave strangly when using collision physics (McMenamin 2018). Because of these reasons a more suitable approach is used to simulate these kind of object interactions. For buttons interaction a trace box surrounding the button is created. This trace box is not rendered, therefore it cannot be seen by the user in the scene. The trace box uses overlapping collision and allows other meshes to pass through it, generating overlap events. When a hand mesh overlaps the trace box an overlap event with a distance parameter is generated. This parameter indicates how far the hand has entered into the trace box and is used to determine, if a button is pressed. Rotary switches and levers use a similar approach, but rather than using a distance parameter, they use the hand mesh location and rotation.

5.4.2 VIVE controllers interaction

HTC VIVE Pro comes with a pair of wireless motion controllers. These controllers are tracked by the base station of VIVE and their primary use is for interaction with the VR world. The controllers have several buttons and a trackpad which provide more interfacing capabilities (Figure 17). Motion controllers will be used as a backup to Hi5 VR Gloves during exhibitions and also during the development process, thanks to the quicker setup for usage.

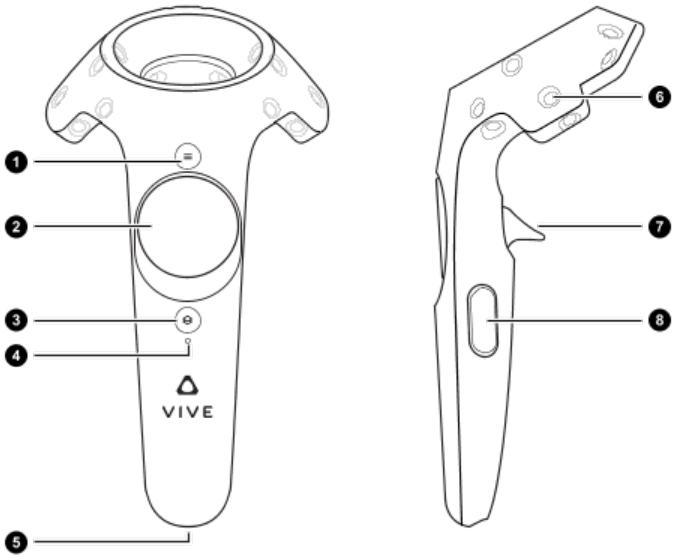

Figure 17. HTC VIVE Pro motion controllers (About the Vive Controllers 2019)

Unreal Engine provides a Motion controller(MC) Component that offers support for commonly used VR MC brands. VIVE Motion controllers are supported by this Component. Two instances of this Component are added to the Camera actor, as the location of the controllers is relative to the Camera location which itself is locked to the location of the VR headset. A static mesh of a mannequin hand is added as a child to each MC Component: one is for the right hand and the other for the left hand. Since only a mesh of the right hand is available, this mesh needs to be mirrored in the Y-coordinate for the left hand. After these steps the hand meshes should be visible in the scene and their position follows the controllers' position.

Interacting with other objects in the scene is done by utilizing overlap events while pressing the Motion Controller trigger button (Figure 17). The trigger button press

and release events are available in the Blueprint Editor. They're used to create grab and release functions that fire grab and release events on objects that are overlapping with MC meshes (Figure 18).

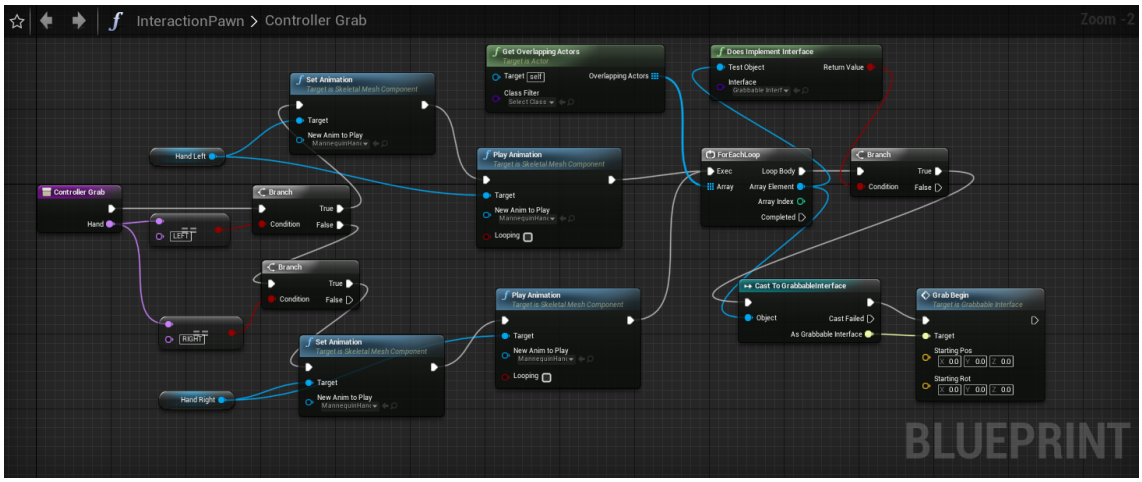

Figure 18. Logic for Motion Controller grabbing

The hands' mesh is animated when the trigger button is pressed. It changes from an open palm hand to closed palm hand, visualizing that grabbing has been initiated. An interface for objects that can be grabbed is created and implemented by every object that needs grabbing functionality. The overlapped actors are iterated after playing the grab animation and checked for implementation of the Grabbable Interface. Actors implementing the interface then have the GrabBegin function called and decide what grabbing behavior they have (Figure 18). If the object is something that can be carried, its position and rotation will follow the MC until released. If the object is a rotary switch, it will follow the MC rotation in one axis. If the object is a lever, only movement on one axis will be used.

5.5 Virtual hardware layout and operation

Hardware layout needs to be planned to be both realistic and simplified for VR use. Creating an absolute clone of the real environment at this point is not only unfeasible due to the time constraint, but also distracting for the user as a majority of the hardware is not of importance during the CBRN training. Additionally having 3D models of all the real hardware in the scene can lead to performance penalties that

would require more work on optimization or higher-end PC hardware to run.

5.5.1 Devices and sensors

The sensors available at this point of the development are ChemProFXi (chemical detection and classification) and IdentiFINDER S900 (radiological detection and identification). In the future IBAC2 will be added to provide biological detection. Weather sensor in the real project is Vaisala WXT536, but a 3D model isn't needed for the current application, because the trainee is not able to leave the vehicle. Instead measurement values are simulated with a random number generator.

Additional devices are present in the virtual environment that are crucial to the operation of the ObSAS installation. Two 12V 95Ah batteries are installed in the vehicle as a power source of the system. A power and data distribution box is used to provide fuse protected power from the batteries to all devices. In the case of a device short circuiting the rest of the system will still receive adequate powering. The box also connects data inputs and outputs between the different devices. A laptop computer and a touchscreen are present for interfacing with the ObSAS Software. A lamp provides light to the interior of the vehicle. A military spec computer TRAX-10 is mounted to the wall. This computer records input from all devices and runs the ObSAS software.

Some considerations have been taken for the realisticity of the device layout:

- Batteries are close to the power box to avoid creating interference and voltage drop.
- Devices are mounted flat to the walls using their mount points.
- Batteries are placed on the floor and held with a bracket to avoid unintentional movement.
- Air outlets on the ChemProFXi go to the outside the vehicle. This avoids pushing contaminated air into the vehicle.
- All devices have cabling to the power box for data and power.

5.5.2 Wiring devices

Devices in the system require DC power to operate. Devices also communicate between each other via Ethernet or RS232/485 protocols. Powering devices requires power cables and data exchange requires data cables. Two types of cable components can be used in the scene. Dynamic cable actors are used for objects the user can interact with. Static cable actors are used only for the visual representation of cables, these actors cannot be modified during runtime.

Dynamic cable components are needed for interactive objects such as connectors with an attached wire. Unreal Engine provides a Cable Component Plugin that is suitable for this use. The Component has start and end attach points with fixed location. Between these points series of free moving particles are located. These points are separated by a distance constraint and use simulated physics to determine their location (Figure 19). The cable simulation is done by attaching the start point to a connector and the end point to the distribution box. When the connector is held and moved around the world the cable will follow it with natural movement by bending and bouncing (Figure 20). Physics simulation is also used to block cable collision with other actors. Without this the cable will go through actors in the scene.

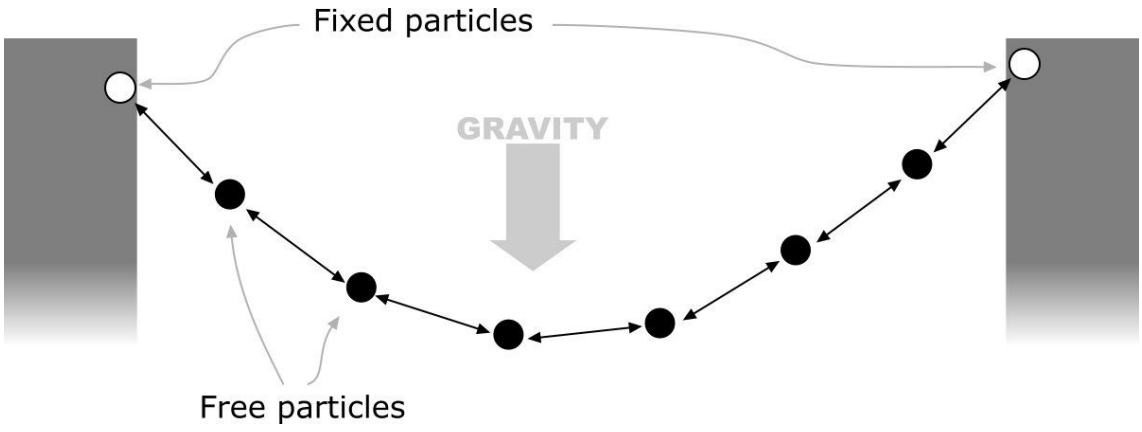

Figure 19. Cable Component Simulation (Unreal Engine Cable Component 2019)

Static cables in the scene do not move during runtime. These cables are used for connecting objects in the scene that also do not change location. One example for such a cable is the connection between the Power Distribution Box and TRAX-10

computer. Neither of these components is designed to be moved by the end user so the cable between them doesn't need to be movable in VR either.

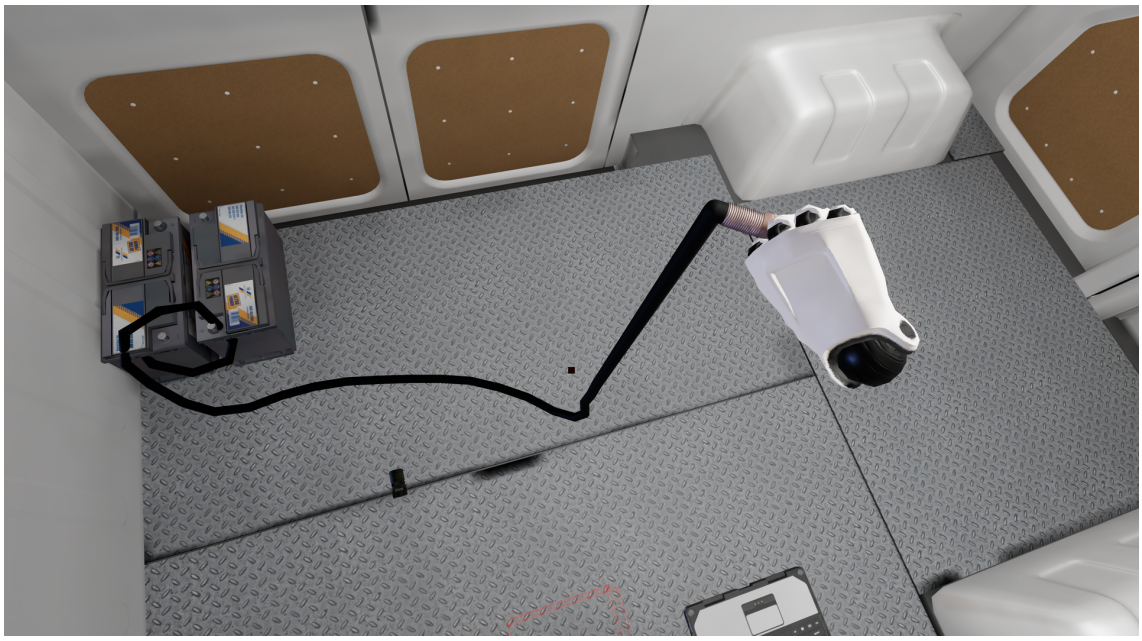

Figure 20. Cable Component attached to objects

Blueprint Spline Component is a component for creating paths. The path is comprised of points and a spline connecting them. This component has two primary uses (Unreal Engine Blueprint Spline Component Overview 2019). The first use is to move an actor across the spline or repeat actors across the spline. In this project the ability of the spline component to repeat actors along a path is utilized. A new Actor Blueprint is created and a Spline Component is added to its root. Logic is added to the construction script function to add a spline mesh of a cylinder for each spline point. The start of the cylinder spline mesh is the current spline point and the end for the spline mesh is the next spline point. The construction script function is called whever the actor is added to the scene or modified (Figure 21).

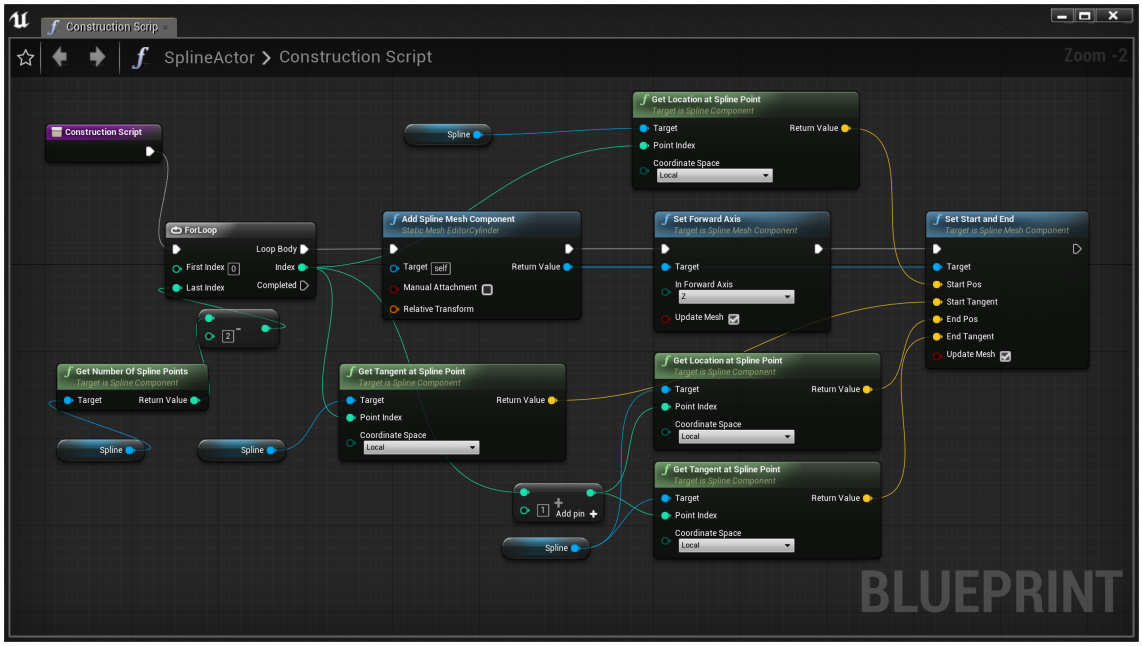

Figure 21. Spline Actor construction script logic

The end result is a Static Cable Actor that can have segments added and modified from the UE4 Editor. This actor is used for the static cabling in the scene and is replicated for every individual cable.

5.5.3 Air inlet and outlet tubing

ChemProFXi chemical detector has an air inlet port through which air to be analyzed comes in and an outlet port for the analyzed air to leave the device from. These ports need to be connected to the air outside the vehicle. This is done by using PTFE tubing which is chemically inert and will not interact with gasses. The PTFE tubing needs to be simulated in the virtual environment. This is done with the use of the Static Cable Actor from Section 5.5.2. A different mesh in the shape of an open cylinder is used and its surface material is changed to resemble PTFE tubing. The outlet is routed to the outside of the vehicle and the inlet is left inside the vehicle for testing purposes.

5.6 Creating a context for training

The virtual scene contains the required devices and interaction capabilities. However, further meaning needs to be added to the application for training purposes. This is done by creating objectives for the user to complete. The goal of the objectives is to make the user familiar with the equipment and prepared for common scenarios encountered in real world use.

Objectives need to be completed in series. The user is given visual objective instructions either with text or by highlighting scene objects of importance. Upon following the instructions the objective is complete and another one will be presented to the user. Logic for displaying instructions and logic for tracking if an objective is completed need to be tightly coupled together to ensure consistency.

5.6.1 Tracking progess

Tracking progress is the proccess of recording the state of objective completion. The application starts with a default first objective assigned. When certain interactions between objects in the scene occur, the objective is changed and objectives interested in that change are notified via an event. The best place for objective tracking is in the Player State Blueprint, since it is designed for holding player state and has access to all other objects in the scene.

Objective tracking begins by creating a new Blueprint from the PlayerState base Blueprint. The PlayerState Blueprint is designed to store the state of each player in a multi-player or single-player environment. A new instance of this Blueprint is created for every player, and only the server or the player itself can modify values of the instance. Other players are able to read values from it, but not modify them.

To track the progress a new variable is added to the new PlayerState Blueprint that holds the current objective. The type of this variable is an Objectives Enum. This enum type contains all the possible objective key names. Two functions are then added to the blueprint, one for getting the current objective enum and another for changing the current objective enum value. Actors in the scene that participate in the

objectives are able to get a reference of the PlayerState and call the SetCurObjective function which changes the value of the current objective. Additionally this function emits an event which is used by actors to update their own state whenever an objective changes (Figure 22).

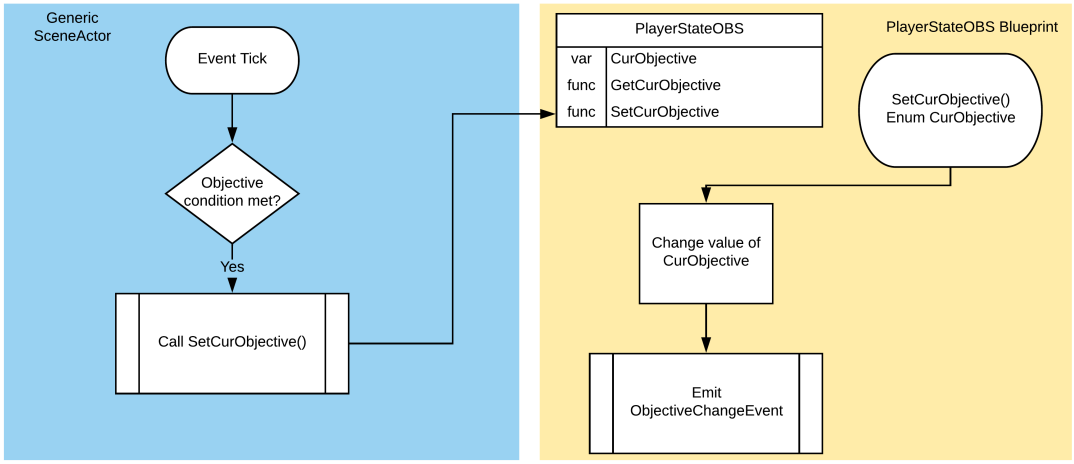

Figure 22. PlayerState Blueprint logic diagram for objective tracking

With the current implementation, whenever the application is restarted, all progress will be reset, because it is kept in the memory. To persist the progress tracking over application restarts the SaveGame Blueprint is used. This blueprint is used to create save slots and store variables in them. These slots are not lost after restarts and can be loaded with each application start. The value stored is the current objective enum, and therefore the application resumes to the last objective the user was on before exiting. (Unreal Engine Save Game Blueprint 2019.)

5.6.2 Objective instructions

Instructions about the different objectives need to be provided to the application users. Without such instructions users can become unaware of what is required of them to do. These instructions should contain the steps required to complete the current objective. Additionally, basic information about the system that is relative to the current objective can be given. This section covers the different methods used to provide instructions and information to the user.

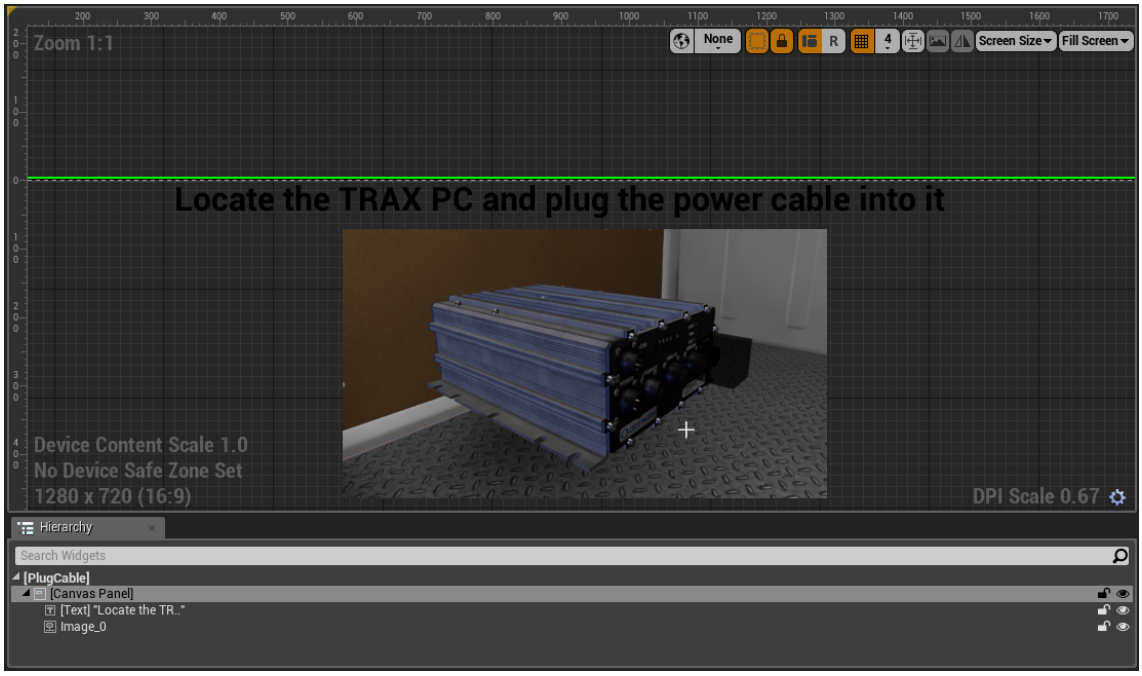

Figure 23. UserWidget containing text and an image element

The primary media types for instructions are text and images. These media types are two dimensional which makes Widgets a good candidate for the delivery of instructions. A virtual widescreen display will be utilised to show these instructions. Firstly, a UserWidget is created for every instruction. Each UserWidget contains text and image elements that guide the user and explain what actions need to be taken. These widgets will not be modified on runtime and will not contain logic. Secondly a new WidgetComponent is created and named ScreenComponent. It contains a white Image element which is used for the text background. Furthermore all the UserWidgets created earlier are added to a WidgetSwitcher component, which itself is added to the ScreenComponent. The WidgetSwitcher component is a simple component which is used to control what is shown from the list of its child components. Lastly the ScreenWidget is listening to objective change events and commands the WidgetSwitcher what instruction widget should be shown. After performing this setup the 2D behavior of the ScreenWidget is complete.

Figure 24. ScreenComponent added to the scene

The ScreenWidget component needs to be added to the scene in order to be visible to the user. This is done by creating a new Actor Component, which contains a 3D model of a touch screen, and a WidgetComponent wrapping the ScreenWidget. The WidgetComponent is aligned with the surface of the 3D screen model, which makes it appear as if the screen is projecting the instructions. Finally the Actor Component is added to the scene in a place, where it can be seen by the user (Figure 24).

5.6.3 Visual and auditory cues for events

User interaction with the virtual environment can lead to events occuring. Visual or auditory cues are needed to signal the user that such events are taking place. These cues can be considered as a feedback to the user when the environment has been changed in an impactful way. A simple example of such feedback is when a device is turned on and shortly after the device status LED starts flashing and it also starts emitting operational noise. This type of feedback is not directly related to the motor input from the user, but more so to the change of state in the environment.

Events on devices can be generated either by a change of the device's own state change or by a change of the global state. An example of a device state change is when a device is turned on. An example of a global state change is when a device

alarms, resulting in alarm sound and visualization coming from the toughbook laptop.

Visualizing of events is accomplished by changing the visual appearance of the scene, by adding, removing or modifying scene actors. One visualization tool used for real life sensor devices is LEDs that indicate status of the device. This is replicated in UE by using a PointLight as an LED light source and setting its visibility on or off, depending on the virtual device status (Figure 25). This method is quite intuitive for the user as it is similar to real life visualization. Another visualization method used in this project is the display of text and images with Widget Components. This method was further discussed in Section 5.6.2.

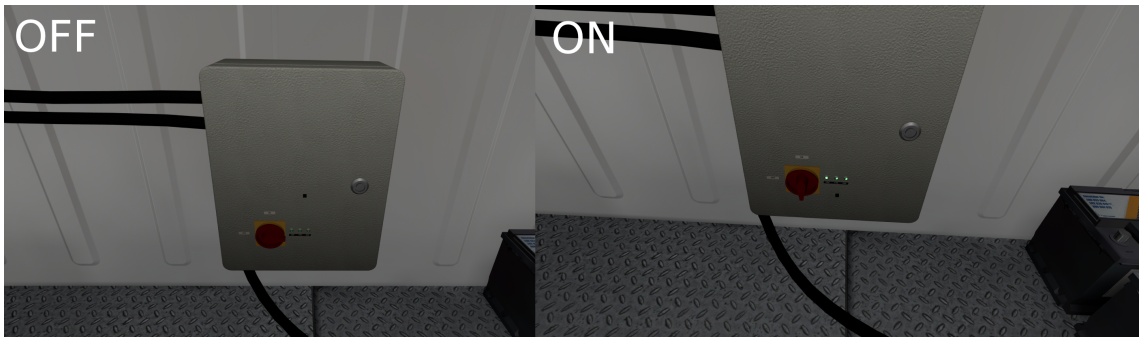

Figure 25. Power status LEDs

Sound effects can offer better immersion by simulating sounds, that are present in the real ObSAS environment. Additional information can be also provided to trainees with the use of sound. Playing sounds in Unreal Engine begins by importing a Waveform Audio File(WAV) of the sound effect as an asset. This asset can be then added to the scene with the use of a Sound Cue Component. This type of component is used to modify audio playback by changing aspects such as playback speed, frequency, volume, pitch and others. The Sound Cue Component also adds spatialization to the sound source. This means, that when a user is further from the sound source the sound will get quieter and vice versa when closer. Additionally, there are sound volume differences between the left and right stereo channels, depending on where the user is looking relative to the sound source. This makes it possible to locate the direction of the source.

5.7 Building and distributing the application

The training application is ran from UE during development and testing. Running the application from UE makes it easy to make changes and see them straightaway in VR. Most changes to the scene, such as moving an object, changing materials, editing blueprints, modifying lights and introducing new actors require a project rebuild. Project rebuilding is modular, meaning it can be done in separate steps in any order. Separate build steps are available for lighting, reflections, visibility, navigation, geometry and others. Because of this modularity it is not necessary to execute all build steps during development, but only the build steps related to what is changed since the last build. (Build configuration reference 2019.)

Running the project from Unreal Studio during development is the preferred approach, thanks to the ability to see changes quickly and debug the application. However the same approach is not optimal for running the application in production environment. The development environment build trades the lack of optimization features for quicker builds. This means that the project will not run with as high framerate or as high visual quality during development. Another downside to running the project in the development environment is that it requires Unreal Studio to be installed and running, which takes around 20GB of disk space and requires extra CPU cycles and RAM, that can be otherwise used by the application itself (Hardware and Software specifications 2019). Because of these limitations another approach is used to running the application in production.

The better approach an application in a production environment is to use a distribution package. This package is self-contained, which means it carries all the necessary assets and executables to run by itself. The package is also in the specific format required by the target platform it will run on, whether that would be Android, Windows or Mac. Building the package is done with the Unreal Studio Package Project wizard. Options are available for target platform, level loading methods, package compression, signing, encryption and others. After selecting the required options the wizard will go through all project build steps and create the distributable package of the application. When applied to the ObSAS training

application the end result is an application package that can be copied to other computers and executed. (Packaging projects 2019.)

6 RESULTS AND CONCLUSION

REFERENCES

A brief simulation training history. 2019. WWW document. Available at: https://www.iti.com/blog/a-brief-history-of-simulation-training- [Accessed 13 May 2019].

Ar experience of the white house. 2016. WWW document. Available at: https://vrscout.com/news/augmented-reality-app-1-dollar-bill-tour-white-house/ [Accessed 06 October 2018].

About the vive controllers. 2019. HTC Corporation. WWW document. Available at: https://www.vive.com/us/support/vive/category_howto/about-the-controllers.html [Accessed 20 June 2019].

Build configuration reference. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Programming/Development/BuildConfigurations/index.html [Accessed 22 June 2019].

Buy htc vive pro. 2018. WWW document. Available at: https://www.vive.com/eu/product/vive-pro/ [Accessed 30 September 2018].

Buy microsoft hololens. 2018. WWW document. Available at: https://www.microsoft.com/en-gb/hololens/buy [Accessed 30 September 2018].

Cbrne world. 2019. Falcon Communications UK.

Carter, H. E. 2014. Crowd behaviour in chemical, biological, radiological and nuclear (cbrn) emergencies : behavioural and psychological responses to incidents involving emergency decontamination. Master's thesis.

Chemprodm chemical detector for mobile and vehicle applications. 2018. WWW document. Available at: https://www.environics.fi/product/chemprodm/ [Accessed 26 October 2018].

Chemical, biological, radiological & nuclear (cbrn) defence market report 2017-2027. 2017. Visiongain ltd.

Consumer spending on ar/vr worldwide 2016-2021. 2018. WWW document. Available at: https://www.statista.com/statistics/828467/world-ar-vr-consumer-spending-content-apps/ [Accessed 06 October 2018].

Dondlinger, M. J. 2007. Educational video game design: A review of the literature.

Gopinath, C. & Sawyer, J. E. 1999. Exploring the learning from an enterprise simulation. WWW document. Available at: https://www.researchgate.net/publication/201381843_Exploring_the_learning_from_an_enterprise_simulation [Accessed 13 May 2019].

Hi5 vr glove user guide. 2018. WWW document. Available at: https://hi5vrglove.com/instructions [Accessed 20 June 2019].

Hi5 vr glove business edition. 2018. WWW document. Available at: https://hi5vrglove.com/store/hi5glove [Accessed 20 November 2018].

Hp z vr backpack overview. 2018. WWW document. Available at: http://www8.hp.com/us/en/campaigns/vrbackpack/overview.html [Accessed 21 May 2018].

Htc vive pro review. 2018. WWW document. Available at: https://www.digitaltrends.com/vr-headset-reviews/htc-vive-pro-review/ [Accessed 06 October 2018].

Htc vive pro vs occulus rift. 2018. WWW document. Available at: https://www.windowscentral.com/htc-vive-pro-vs-oculus-rift [Accessed 29 October 2018].

Hardware and software specifications. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/GettingStarted/RecommendedSpecifications/index.html [Accessed 23 June 2019].

Johnson, L., Levine, A., Smith, R., & Stone, S. 2010. The horizon report. Educause Learning Initiative.

Kipper, G., Rampolla, J., & Kipper, G. 2013. Augmented reality. Syngress.

Lewis, M. & Jacobson, J. 2002. Game engines in scientific research. *Communications of the ACM Vol. 42*. pages 1–5.

Macedonia, M. 2002. Games, simulation, and the military education dilemma. WWW document. Available at: https://www.researchgate.net/publication/260386555_Games_Simulation_and_the_Military_Education_Dilemma [Accessed 13 May 2019].

Martinali, J. 2018. What is simulation training?. WWW document. Available at: https://www.noldus.com/blog/what-is-simulation-training [Accessed 13 May 2019].

McMenamin, M. 2018. Design and development of a collaborative virtual reality environment. Master's thesis.

Microsoft hololens device specifications. 2018. WWW document. Available at: https://www.abcomrents.com/wp-content/uploads/2018/01/HoloLensSpecSheet.pdf [Accessed 1 September 2018].

Mullen, T. & Mullen, T. 2011. Prototyping augmented reality. Wiley.

Obsas cbrn system technical description. 2019. Observis Oy.

Obsas training material ver 3.0. 2019. Observis Oy.

Packaging projects. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/Basics/Projects/Packaging/index.html [Accessed 23 June 2019].

Rabbi, I. & Ullah, S. 2016. A survey on augmented reality challenges and tracking. *Acta Graphica*. 24(1-2):29–46. WWW document. Available at: http://www.actagraphica.hr/index.php/actagraphica/article/view/44.

Unreal engine blueprint spline component overview. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/BlueprintSplines/Overview/index.html [Accessed 27 May 2019].

Unreal engine cable component. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/Components/Rendering/CableComponent/index.html [Accessed 27 May 2019].

Unreal engine documentation. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Programming/UnrealArchitecture/Actors [Accessed 03 April 2019].

Unreal engine documentation introduction to blueprints. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Engine/Blueprints/GettingStarted [Accessed 04 April 2019].

Unreal engine levels. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Engine/Levels [Accessed 01 April 2019].

Unreal engine light sources. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Engine/Rendering/LightingAndShadows/LightTypes [Accessed 21 May 2019].

Unreal engine materials. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Engine/Rendering/Materials [Accessed 24 May 2019].

Unreal engine nativizing blueprints. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/Blueprints/TechnicalGuide/NativizingBlueprints [Accessed 04 April 2019].

Unreal engine player start. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/Actors/PlayerStart [Accessed 20 May 2019].

Unreal engine save game blueprint. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Gameplay/SaveGame/Blueprints/index.html [Accessed 02 June 2019].

Unreal engine skeletal meshes. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Engine/Content/Types/StaticMeshes [Accessed 24 May 2019].

Unreal engine static meshes. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-us/Engine/Content/Types/StaticMeshes [Accessed 24 May 2019].

Unreal engine traces with raycasting. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/Physics/Tracing [Accessed 23 May 2019].

Unreal motion graphics ui designer. 2019. Epic Games. WWW document. Available at: https://docs.unrealengine.com/en-US/Engine/UMG/index.html [Accessed 02 June 2019].

Vrgluv. 2018. WWW document. Available at: https://vrgluv.com/ [Accessed 20 November 2018].

Wang, X. & Dunston, P. S. 2007. Design, strategies, and issues towards an augmented reality-based construction training platform. *ITcon Vol. 12*. pages 1–8.

van Gimst, R. 2018. Creating a high-fidelity and low-cost simulation environment. WWW document. Available at: http://www.nlr.org/article/creating-a-high-fidelity-and-low-cost-helicopter-simulation-environment/ [Accessed 24 June 2018].

van Krevelen, D. & Poelman, R. 2010. A survey of augmented reality technologies, applications and limitations. *The International Journal of Virtual Reality*. pages 1–6.