CHALMERS

Bachelor Thesis - Virtual Reality strategy from a product development perspective

Bachelor thesis within university engineering program - Mechanical engineering

Jonas Abusagr
Mahamed Adan Abdullahi

Supervisor: Silvan Marti
Examiner: Björn Johansson

Chalmers tekniska högskola
Göteborg, Sverige 2023

CHALMERS

Abstract

Virtual Reality (VR) and Augmented Reality (AR) is proving itself to be a powerful tool with immense potential that allows engineers and operators to be immersed into a Virtual environment. This offers the opportunity to inspect, evaluate and interact with 3D designs and provides a sense of scale and spatial understanding that enhances the evaluation process of manufacturing and developing new products.

This bachelor thesis explores the potential of VR and AR technologies that can be beneficial for the product development process of Berg Propulsion. The aim of the thesis is to develop a strategy for effectively utilizing VR and AR, as well as to create an interactive VR-application using Unreal Engine 4 (UE4). The research findings demonstrate that VR and AR can bring significant advantages to Berg Propulsion, including improved design visualization, simulation, and prototyping. The VR-application developed in UE4 showcases the implementation of interactive functionalities, bug fixes, and a reset button for returning objects to their original position. This thesis will show that UE4 offers a user-friendly interface and scripting system for creating interactive experiences. A comparison between Unreal Engine and Unity will be highlighted to show not only their differences in programming languages and visual coding options, but also clarify the best program to use for VR-applications.

Acknowledgements

We would like to extend our gratitude and recognition to several individuals who played a significant role in the success of this bachelor thesis project.

First we would like to express our gratitude to Martin H Persson from Berg Propulsion, for providing the project which our bachelor thesis is based on and his assistance and support for allowing this project to become successful. We are very grateful for the opportunity to work alongside you and the Company.

Second, we would like to express our gratitude to our supervisor, Silvian Marti, for the help and knowledge he has provided about working with VR engines. We would also express our gratitude to our examiner, Björn Johansson for providing valuable feedback on our report.

Finally, we would like to thank Chalmers for giving us an enriching and transformative educational experience, which has broadened our knowledge of mechanical engineering.

Göteborg, June 2023

Content

1. Introduction

In this bachelor thesis, Berg Propulsion has hired us to conduct a project that focuses on exploring the potential of virtual reality (VR) and augmented reality (AR) and how to optimize the product development at Berg propulsion using these technologies.

In recent years, virtual reality and augmented reality technologies have had significant advancement by opening up new opportunities in various industries, including product development. Virtual and Augmented reality are both digital technologies that enable users to interact with a simulated environment in different ways. In virtual reality the user wears a headset that replaces the real world with a computer generated environment, allowing them to feel like they are physically present in a virtual space. Users can interact with the virtual environment using controllers or other input devices, which can simulate physical action like grabbing and manipulating objects. Augmented reality enhances the real world by overlaying digital information on top of it. Using a camera equipped device, such as a smartphone or tablet. AR technology can detect and track real world objects and then superimpose digital content, such as graphics, text, or video, onto the user's view of the real world. This creates a new, enhanced version of reality that can provide additional information or interactive elements to the user's experience.

Berg propulsion, a leading manufacturer of marine propulsion systems since 1912, is interested in exploring the potential of VR and AR in their product development process. By using virtual reality and augmented reality, they hope to be able to future optimize their product development process by improving design visualization, simulation and prototyping.

In VR and AR there are many different tools. UE4 and Unity are the two most popular engines which are used to create 3D objects and simulate the interactive object as close to reality as possible. This project will primarily focus on UE4 as it is used by Berg propulsion, but we will also compare UE(Unreal engine) with Unity to show the respective device's strengths. The methods used for this project will be transparent, which will enable future studies in development and improvement for the necessary requirements needed for VR and AR.

1.1 Aim

The aim of this bachelor thesis is to develop a strategy for Berg propulsion to effectively use VR and AR technology in its product development process. The goal is to get the VR technology to a level where it is easy and natural for designers to use.

The study seeks to answer two main research questions:

1. A strategy for VR and AR within Berg Propulsion, what tool to use and is the current tool we use(UE4) the correct way for Berg?
2. Develop the tool(UE4) used today for visualization to a more interactive level. Focus on making it easy to use.

To further develop the aim of this project further we must define the features of the development environment which are the most important for Berg propulsions product development process. We will measure the available environments and conclude which is the most suitable to use(UE4 or Unity).

1.2 Limitations

This study is limited to exploring the potential of VR and AR technology in the product development process at Berg Propulsion and will not explore other technologies. The project will also be limited to the context of Berg Propulsion's product development process which will not be applicable to other industries or organizations. The project will have a limited time and resources which may impact the feasibility of implementing VR and AR in product development.

1.3 Motivation

This study is motivated by exploring the future potential and the capabilities of virtual reality and augmented reality. Companies today rarely use VR and AR into their product development processes because it is mostly used for visual assistance rather than developing products with it. They also refuse to put a lot of money and time into developing something that is unsure. Today companies use the first best engine that is available to the market that requires the least effort, that is why this thesis will determine which engine is the most optimal to use considering the product development process.

2 Background

When considering the choice between Unreal Engine and Unity for project and game development, several factors should be taken into consideration. This chapter states the current state of the project and explores the comparisons between the engines and their various aspects such as coding, visual coding, interface, documentation and graphics. By examining these areas, developers can make an informed decision based on their specific needs and goals.

2.1 Current state

The project began with a VR-application given from Berg Propulsion with its task to simulate their products in Virtual Reality. The application given had some minor errors, specifically with the rotation of the propeller blades not functioning correctly during interaction. The blades of the propeller did not rotate according to reality.

Another issue encountered in the VR-application was the inability to interact with other objects within the virtual environment. Users were only able to observe the products in the simulation, except for the rotation of one propeller. In addition to these issues, there was a problem with opening one of the files sent by the company. This particular file had an issue with the HP motion controller, resulting in a failure. As a consequence, the VR simulation program in Unreal Engine was unable to read the file properly, which resulted in obstacles to utilize the files for their intended purposes.

Furthermore, the application contained many unnecessary components and features that were not required to achieve the goal. This led to the file size being larger than necessary, which negatively impacted the performance and user experience on Unreal Engines VR simulation program.

2.1.1 Hardware and software

The hardware in use for this project was the HP revere g2, which was provided by Berg Propulsion, as the primary device for simulating and testing the VR-application. The VR-application was simulated using Unreal Engine 4. To be able to perform the simulations in virtual reality, windows own mixed reality portal program must be installed and Steam's VR launcher. This allows the project in Unreal Engine to simulate in VR preview.

2.2 Unreal Engine VS Unity

When it comes to coding, Unreal Engine primarily uses C++ (Epic Games, 2020), which is a powerful and widely-used programming language in the project and game development industry. On the other hand, Unity uses C# (Unity, 2023), which is considered easier to learn, especially for beginners. Although C++ may have a deeper learning curve initially, mastering it can lay a strong foundation for understanding other programming languages. It's worth noting that many large studios use C++ for their custom engines or Unreal Engine itself, making it valuable knowledge for aspiring project and game developers.

Both Unreal Engine and Unity provide visual coding options that can greatly simplify the development process and make it more accessible to developers, especially those who are new to programming(Epic Games, 2020). Unreal Engine's scripting system is called Blueprints. It allows developers to create mechanics, AI behaviors, and other functionalities by visually connecting pre-built nodes that represent code functions. Each node represents a specific action or operation, and by connecting them together, developers can define the flow of logic and behavior in their projects or games. Blueprints provide a visual representation of the underlying code, making it easier to understand and modify without directly writing lines of code(Epic Games, 2020). This makes it particularly beneficial for designers and artists who may not have extensive coding experience but still want to create interactive experiences.

On the other hand, Unity offers its visual coding system called Bolt (Unity, 2023). Bolt is a visual scripting plugin that is integrated into Unity and available for free in all Unity versions. It provides a similar node-based approach to Blueprints in Unreal Engine, allowing developers also to create mechanics, implement logic, and define interactions without writing traditional code. Bolt features a wide range of nodes for various operations and can be used in combination with Unity's existing scripting capabilities. This allows developers to seamlessly switch between visual coding and traditional coding, utilizing the best approach for each specific task(Unity, 2023). It's important to note that while visual coding systems like Blueprints and Bolt can be incredibly powerful and useful, they may not cover every aspect of projects or games development. In many cases, it is still necessary to write traditional code to achieve certain complex functionalities or optimize performance.

By combining visual coding with traditional written code, developers can take advantage of the strengths of each approach. Visual coding provides a more accessible and intuitive way to design and implement mechanics, while traditional coding offers greater control, flexibility, and optimization opportunities. The ability to seamlessly integrate these two approaches is one of the advantages of working with engines like Unreal Engine and Unity, allowing developers to leverage their respective strengths and create robust, high quality projects.

2.3 Comparing Interfaces

Both Unreal Engine and Unity's interfaces share a lot of similarities in terms of their design and functionality. The user interface of both engines is made up of a number of key elements that let developers interact and explore their projects. The viewport, scene, asset browser, inspector, and toolbar are some of these components. The following table, from hackr.io (2023), shows a clear visual understanding on the comparison of both engines.

	Unreal Engine	**Unity**
Engine Type	Cross-platform	Cross-platform
Developed by	Epic Games	Unity Technologies
Programming Languages	C++ for development	C# for development
Usage	Develop games for PCs, mobiles, consoles, and more	Develop games for PCs, mobiles, consoles, and more
Features	A robust multiplayer framework, VFX, and particle simulation	2D improvements, animation, creating snapshots
Source Code	Open-source	Not open-source.
Pricing	Free	Basic version is free
Learning Curve	Difficult to learn	Easy to learn with an intuitive interface
Graphics	Photorealistic graphics used in AAA games	Good overall graphics, but less refined than Unreal

Table 1. Comparison of both engines. Hackr.io, 2023, Unity Vs Unreal Engine

2.3.1 Viewport

In terms of viewing projects and game elements in the scene, both Unreal Engine and Unity offer a similar layout. They provide a viewport where users can see their projects or games from different camera perspectives. This viewport allows users to manipulate and interact with objects in the scene, such as moving, rotating, and scaling them. Users can navigate through the scene using keyboard and mouse controls or specialized tools provided by the engines(Epic Games, 2020). While the core functionality of the interface is similar between Unreal Engine and Unity, there are some differences in terminology and layout. For example, in Unreal Engine, the main workspace is called the "Viewport," whereas in Unity, it is referred to as the "Scene view" (Unity, 2023).

2.3.2 Scene/Inspector panel

Both Unreal Engine and Unity offer a scene outliner that displays the objects and components within the current scene. This hierarchical view allows developers to effectively organize and manage their projects and game entities, showing the relationships between elements. From the hierarchy, objects can be easily selected and manipulated in the viewport(Epic Games, 2020). Another shared feature in both engines is the ability to inspect the details of selected objects. When an object is chosen in the scene, users can access its properties and modify them through an inspector panel. The inspector panel provides a comprehensive view of the selected object's attributes, including its position, rotation, scale, materials, scripts, and other relevant components. Users can conveniently make changes to these properties directly in the inspector panel, allowing for customization of objects' behavior and appearance in their game or application.

2.3.3 Asset browser

The asset browser or project window is where developers can manage their project's assets, such as 3D models, textures, scripts, and audio files. Unreal Engine and Unity support a wide range of file formats for various asset types, allowing developers to bring in assets created in external software or downloaded from online sources. This includes popular formats such as FBX, OBJ, PNG, JPEG, WAV, and MP3, among others(Epic Games, 2020). Once imported, assets can be organized into folders or categories to maintain a structured project hierarchy. Both engines also provide features for asset management, such as the ability to rename, duplicate, delete, or move assets within the project. Allso users can define asset metadata and add tags or labels to assets for better organization and searching. Searching for assets is an important aspect of asset management. Unreal Engine and Unity offer search functionality within their asset browsers, allowing users to quickly locate assets based on various criteria, such as file names, tags, or file types. This allows developers to easily find and identify specific assets when working on their projects.

2.3.4 Toolbar

The toolbar contains various tools and options for working within the engine. It typically includes buttons for common actions like saving, undo/redo, playtesting, and building the project. The toolbar may also provide shortcuts to frequently used features and functionality specific to each engine(Epic Games, 2020).

2.3.5 Documentation

In terms of documentation and educational materials, Unity often offers more tutorials, guidelines, and official documentation than Unreal Engine. Since Unity has such a large community and is used so often, both Unity Technologies and the community have produced a ton of educational resources. Developers may find it simpler to learn more about particular topics or workflows and get solutions to their searches as a result of the variety of tutorials, manuals and forums. Unity's official documentation provides comprehensive explanations, detailed and basic examples for various features and functionalities. It covers a wide range of topics, including scripting, physics, graphics, audio, networking, and more(Unity, 2023). This documentation is a valuable resource for experienced developers who are familiar with the engine and need quick reference material.

For beginners, it is crucial to access a variety of learning resources beyond official documentation. Tutorials, blogs, video courses, and books that comprehensively cover the chosen engine can provide more practical guidance and hands-on experience. These resources often provide step-by-step instructions, practical examples, and real world use cases, which are invaluable for understanding how to apply concepts and techniques effectively.
Unreal Engine may have comparatively fewer learning resources and documentation compared to Unity, it still offers a range of tutorials, documentation, and learning materials to support developers(Epic Games, 2020).
Unreal Engine's official documentation covers various aspects of the engine and provides explanations and examples. Also there are dedicated Unreal Engine learning resources, including video tutorials, YouTube channels, and community forums, which can be valuable for beginners and experienced developers alike.

2.3.6 Graphics

The Unreal Engine has developed a reputation for its industry leading rendering capabilities (Epic Games, 2020), delivering highly realistic visuals that often blur the line between real and virtual scenarios. The engine's advanced rendering capabilities have made it a perfect choice for creating visually stunning and immersive experiences. The introduction of Meta Humans (Meta Humans 2023), which are incredibly lifelike digital characters, demonstrates the Unreal Engine's ability to achieve unprecedented levels of realism.

Unreal Engine's rendering quality is a result of its cutting edge technologies such as physically based materials, dynamic lighting systems, real time global lighting and advanced post processing effects. These features help create lifelike environments with accurate lighting, realistic shadows, and detailed textures. The engine also supports advanced rendering techniques such as ray tracing (Epic Games, 2020), which further improves visual fidelity by simulating realistic light behavior and reflections.

On the other hand, Unity, while historically perceived as lagging behind Unreal Engine in terms of graphics, has made significant progress in its rendering capabilities. Unity's latest demo Environments (Unity, 2023), showcases the engine's potential to achieve highly realistic scenes. It highlights advanced lighting techniques, physically based materials, real time global illumination, and other rendering enhancements (Unity, 2023). Unity's Scriptable Render Pipeline(Unity, 2023) provides developers with greater control over the rendering process and allows for customizations to achieve specific visual styles or optimizations.

3 Methodology

The methods used for the development of this project will include collection of data through research, where we will conduct a thorough review of existing literature, case studies and industry reports to understand the current state of virtual reality(VR) and augmented reality (AR) technology and its application in product development. We will engage with stakeholders at Berg Propulsion, including engineers and designers, to identify their requirements and expectations for VR and AR. This ensures that the project is aligned with the company's goals. We will use case analysis, where we will identify potential use cases for VR and AR in product development at Berg Propulsion. This will involve mapping the product development process and identifying areas where VR and AR can add value to the human machine interface, such as design visualization, simulation and prototyping. Further, we display identified benefits and drawbacks of using VR and AR in product development. A proof of concept will be created to demonstrate the feasibility of using VR and AR, which will involve creating a prototype and by showcasing the benefits of VR and AR in a real-world scenario. Finally an evaluation will be performed to evaluate the results of the project and make recommendations for further implementation and optimization for VR and AR in Berg propulsions product development process. Where improvements can be made and future development.

3.1 Unreal Engine Terminology

This chapter shows the general terminology used in this project to give a fundamental understanding of Unreal Engine. This is necessary as the use of Unreal Engine is the central use for this bachelor thesis, even though the end goal is to give a broader insight on Virtual Reality and Augmented Reality.

3.1.1 Actor

In Unreal Engine, an Actor is a fundamental object that represents anything that can be placed or interacted with in a game or simulation. Actors can be referred to as building blocks of a scene and can represent character, props, lights, camera and much more. Actors in UE have the ability to contain components like Static MEshes and execute actions through scripted functionality. Actors support 3D transformations such as rotation and scaling. Actors form the core of gameplay and interactivity of VR, providing the foundation for creating games and simulations(Epic Games, 2023).

3.1.2 Static Mesh

Static Mesh refers to a type of mesh object that is used to represent detailed and complex geometry within a game or simulation. It is a 3D model that defines the shape and appearances of an object, either pre-made objects in Unreal Engine or imported CAD files using datasmith. According to Epic Games (2023), any level made in Unreal Engine consists of Static Meshes, usually in the form of Static Mesh Actors(All objects). Static Mesh plays a crucial role in constructing the virtual world, providing objects that can be interacted with.

3.1.3 Collision

In UE, Collision refers to the detection and handling of interactions between objects and actors in simulation. It determines how objects in VR respond to each other when they come in contact, whether it is detecting a collision or interactions between objects.The collision behavior of an object in Unreal is determined by its object type and sets a response that defines how it will interact with other types of objects. The collision system enables the developers to create a realistic and interactive virtual world(Epic Games, 2023).

3.1.4 Blueprints Visual Scripting

Blueprints Visual Scripting is a powerful visual scripting system in UE that allows the developers to create gameplay mechanics, interactions and logics without writing traditional code. It provides a node-based interface where users can create and connect nodes representing functions, events, variables and other elements to create complex behaviors and interactions from within Unreal Editor(Epic Games, 2023). Unreal Editor is where the script is made. This method of scripting bridges the gap between programming and game designing, enabling rapid prototyping and iteration while offering a user-friendly approach to developing games and simulations.

3.1.5 Datasmith

Datasmith is a collection of tools and plugins that helps the user bring outside content, such as CAD files, into Unreal Engine. By converting and optimizing the data from a CAD file, making it suitable for real-time rendering, visualization and interactive experiences in Unreal Engine(Epic Games, 2023). Basically converts a CAD file into a Static Mesh. Datasmith supports a wide variety of 3D design applications and native file formats, such as Solidworks and Catia. By utilizing Datasmith, users can integrate their design and CAD files into their Unreal Engine projects, saving time and effort importing while maintaining the accuracy of the original data.

3.2 Interview

In the early stages of the project an interview was conducted, with the goal to get a better understanding of the program that Berg Propulsion uses in their VR applications (Unreal Engine 4.25). The interview gave a better understanding on what is required to perform in the program to achieve the interaction of Actors(Objects) in VR. The interviews were done with a software developer at HERO who had previously been hired for a task at Berg Propulsion developing an VR-application for their product development.

4 Implementation

In this chapter, the implementation of the methods used in the project is described. explaining step by step detail the process of how the interaction between objects(Actors) in Virtual Reality can be created and additional functions.

4.1 Improving current VR application

This chapter describes step by step all the improvements made for the existing VR-application created for Berg Propulsions product development department.

4.1.1 Fixing bug in existing script

The first thing performed in this project was to fix an existing problem that had occurred with the blades on the propeller. As explained in Chapter 2.3, the blades rotation on the axis did not rotate correctly proportional to reality. The figure below shows the script of the current state of the axis and how the blades rotate.

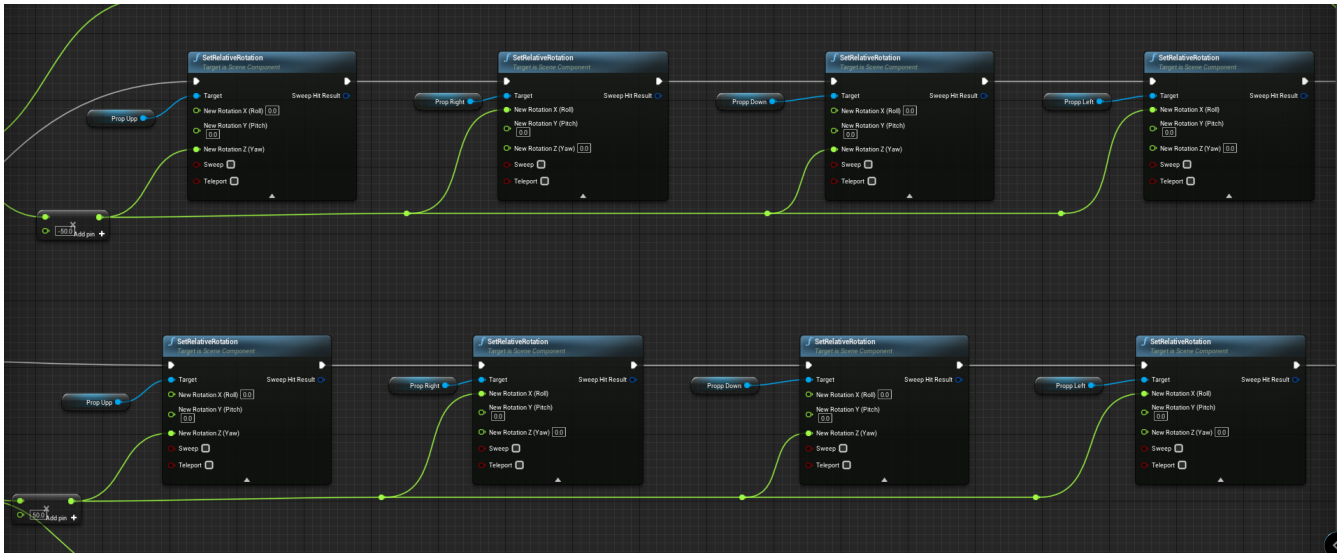

Figure 1. Current state of the script in use for rotating of the blades.

It shows that the values given to the blades are all the same for all four, this gives an inaccurate result on how the blades should rotate in reality. That the blades parallel to each other should rotate in opposite directions when the axis cylinder is active.

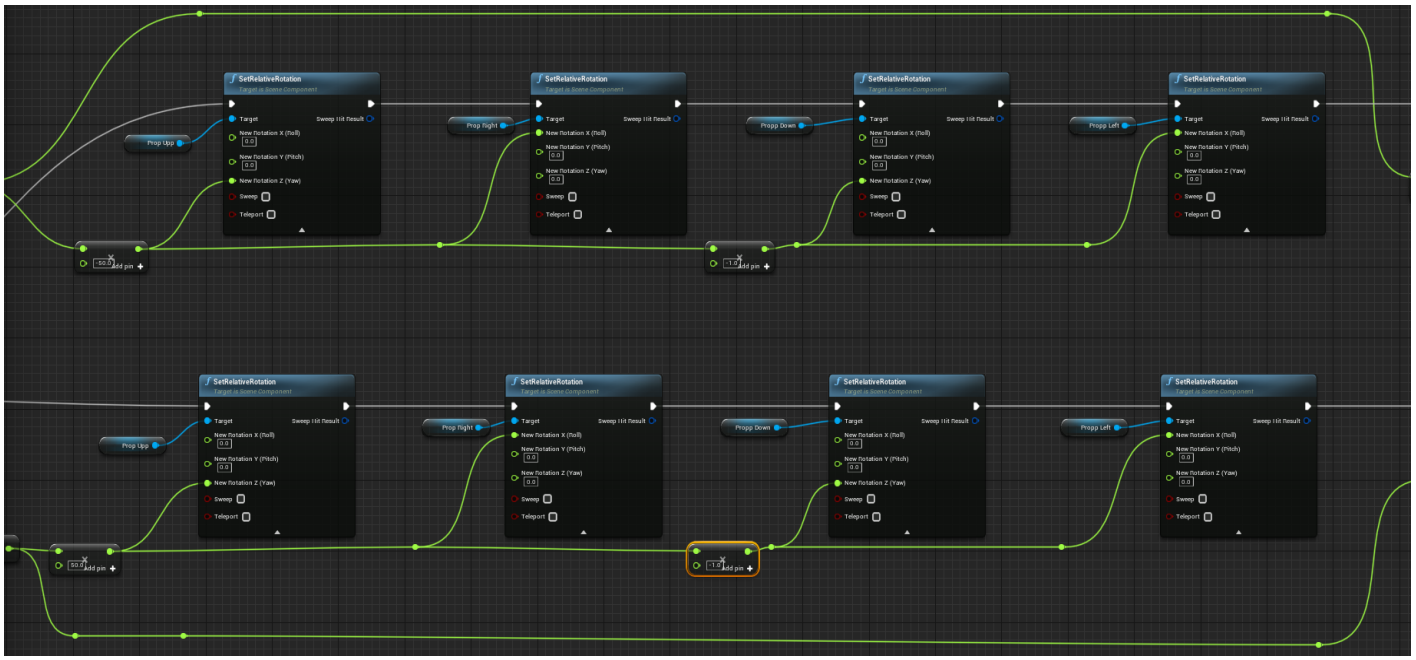

Figure 2. Fixed state of the script in use for rotating the blades.

Figure 2 shows that fixing the current problem could be solved by adding additional values to the blades parallel to each other. By giving the value -1 to the blades parallel to each other, they would rotate in opposite directions in VR. The value must be pinned at -1 in both directions when the blade rotates to ensure that the blades always rotate in opposite directions.

4.1.2 Step by Step interaction Actor

The first step required to perform an interaction on actors in Virtual Reality is to simulate physics of the static mesh(The object). This will allow the static mesh to become interactive in VR. The physics option is available in detail toolbar on the viewport of the static mesh, as shown in figure 3. After simulating the physics of the static mesh one can choose the mobility of the object in transform, to either make it movable(allowing it to move freely in VR) or stationary(allowing it to stay at the position it is placed at). A problem that occurs frequently is that when importing a CAD file using datasmith, is that the simulate physics option on the static mesh does not become available. This problem can be solved by changing the static mesh to a pre-made static mesh created from Unreal engine and changing back to the original static mesh. This allows simulate physics to become available again, as shown in the figure 4.

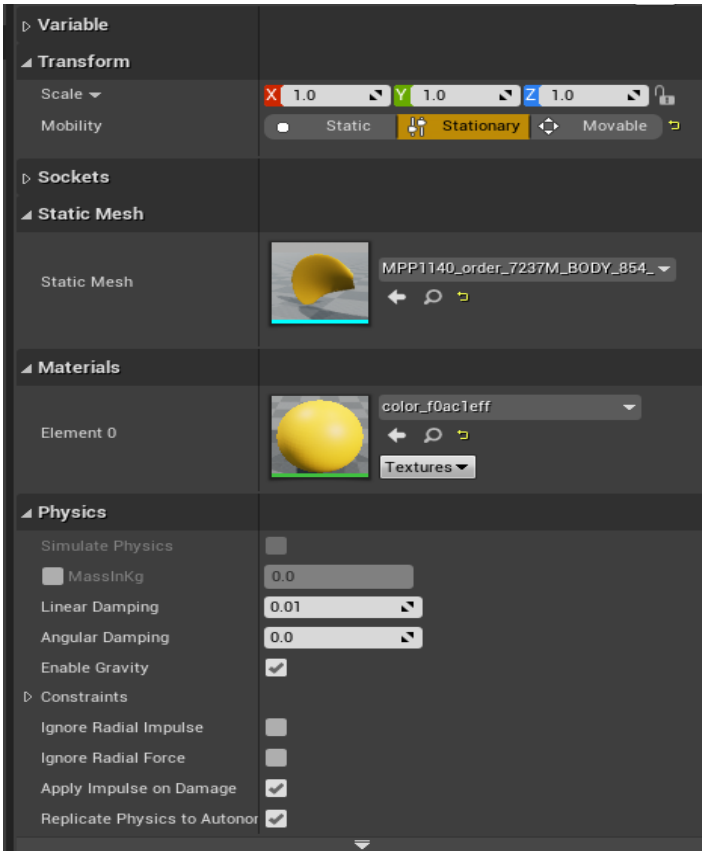

Figure 3. Details toolbar of the object in use.

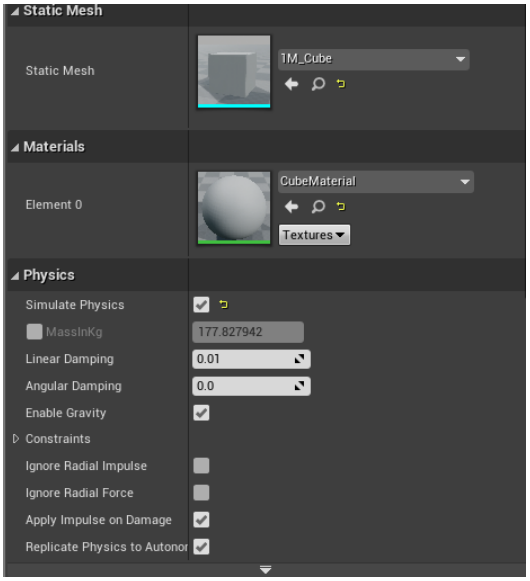

Figure 4. Change in the static mesh.

The next step is to add a box collision in the static mesh to determine where on the static mesh the interaction is going to be in VR. The box is created on the static mesh and can be changed to the developers preferences. Figure 5 shows that a box collision is added under the specific static mesh component. It is important to make sure that the desired collision is under the static mesh to make it intractable, else the object will not be able to move due to the intractable action being in space and not the object itself. Figure 6 shows the box collision of the static mesh and desired parameters can be set.

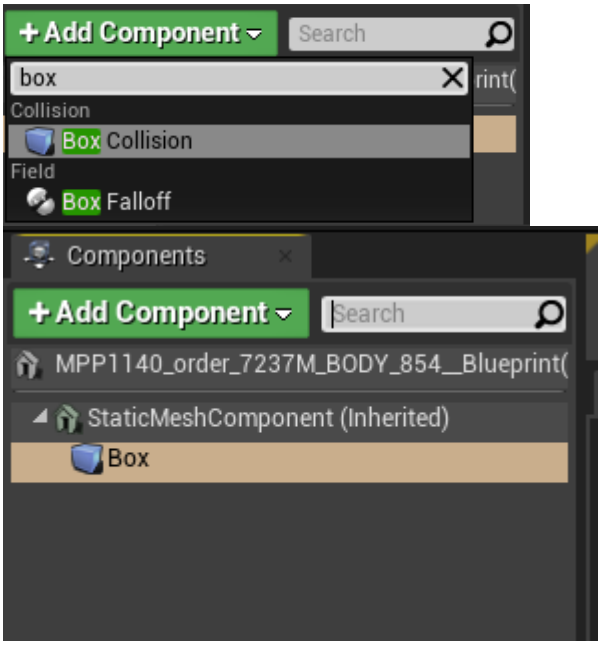

Figure 5. Adding box collision.

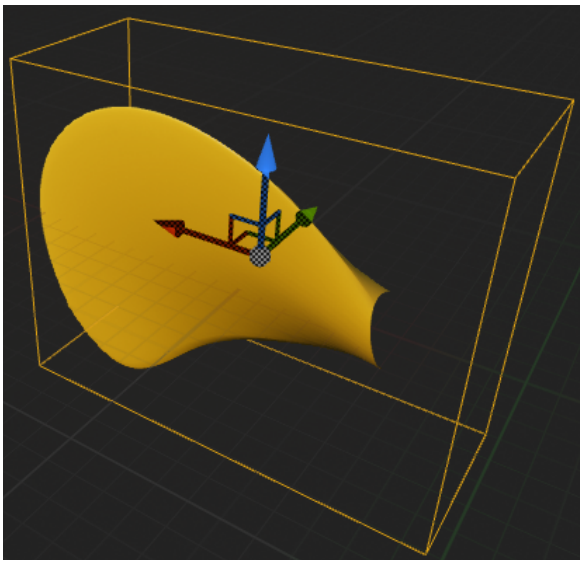

Figure 6. Static mesh with box collision.

After finishing creating the box collision parameters and simulating the physics of the static mesh, a script must be created in the event graph to allow the mechanics and logic of the interactions be set. The figure below shows that a customevent script must be implemented to allow the static mesh to be picked up and dropped.

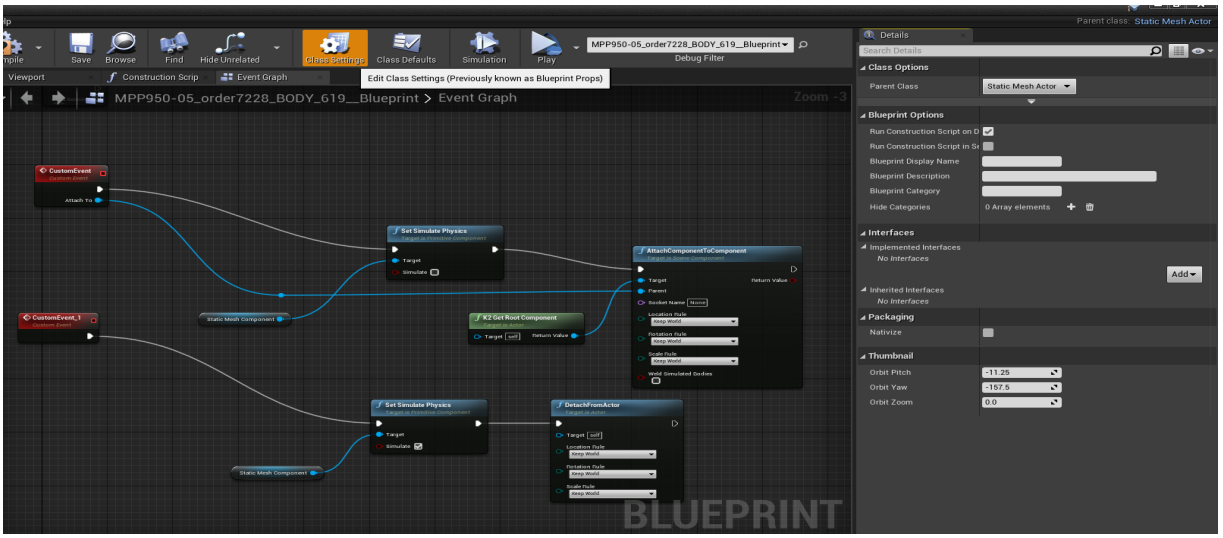

Figure 7. Script using customevent to allow pick and drop of static mesh.

After creating the script, a value interface must be implemented in class settings of the static mesh to allow the actor to pick up the object in Virtual Reality. This can be achieved by adding “PickupActorInterface” in the interfaces toolbar. This will allow the actor(the person interacting in VR) to pick up the static mesh and interact with it in VR. Figure 8 shows the implementation interface of the pick up actor interface and figure 9 shows the result of implementing it.

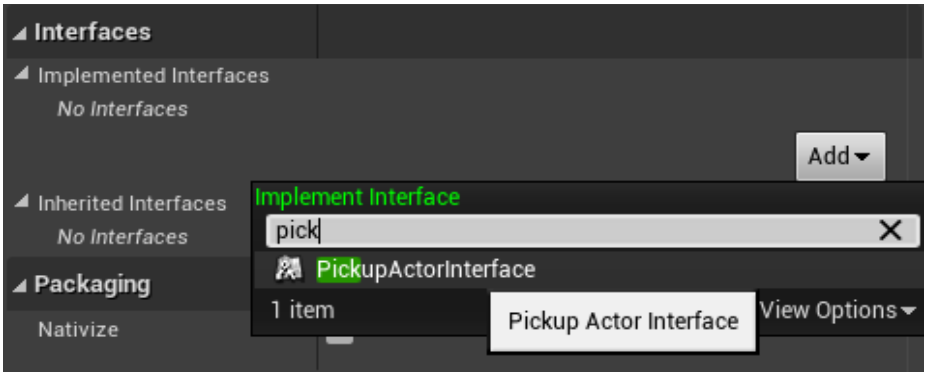

Figure 8. Implementing PckupActorInterface.

Figure 9. Result of implementing PickupActorInterface.

After adding the interface values and creating the script, the script itself will have it´s custom events changed to Pickup and Drop as shown in the figure below. This will allow the static mesh to become interactive in VR. Press on Compile and save the project and now everything should work accordingly.

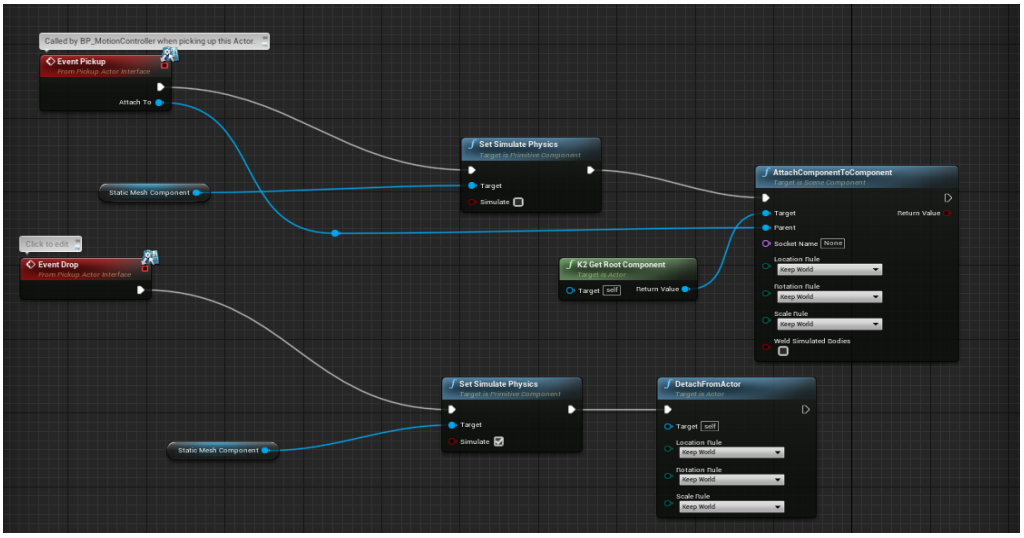

Figure 10. Change in custom events to Event Pickup and Event Drop.

4.1.3 Reset level

The final implementation on the given VR-application was to reset the level to its original position without exiting the virtual simulation. This allows for the operator to interact in VR without the need of restarting the application. Resetting the level in VR can be achieved by creating a button and applying a different kind of script as from the previous chapter(4.2.2). The first step is to create the physical button, by adding a new blueprint class in the content browser. This will allow for a new actor to be created into the VR-application. First step is to import the blueprint class in the content table, as shown in figure 11, to create the object.

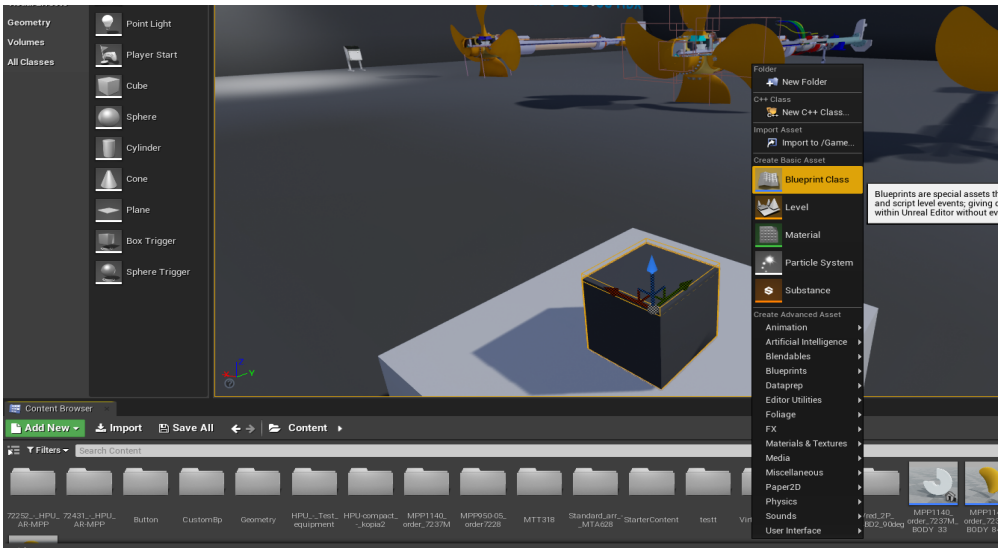

Figure 11. Importing Blueprint Class.

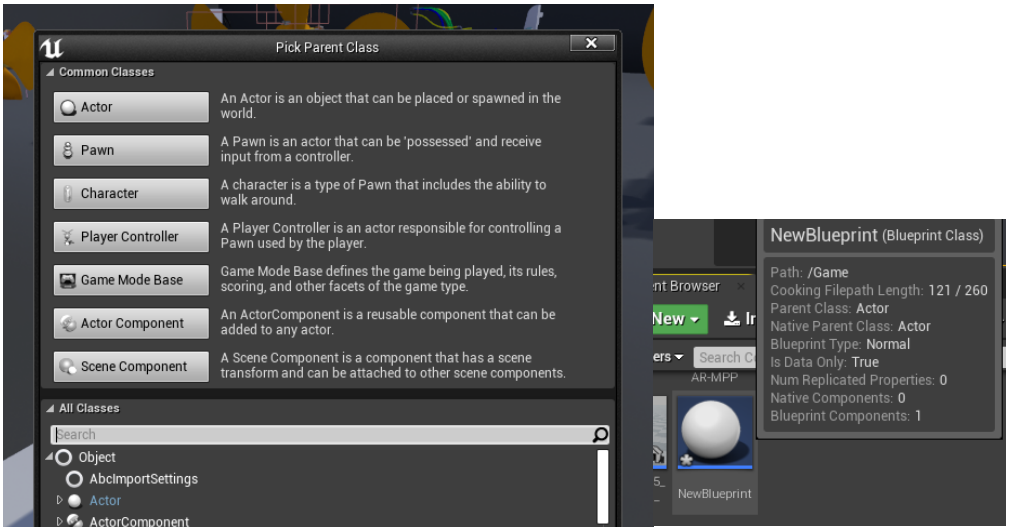

Figure 12. Selecting Actor.

Next is to choose the Actor as the parent class to become a physical object as shown in figure 12. After creating a new blueprint class actor, a static mesh is implemented to create the physical object that one can interact with in VR. After, a box collision is created and set on the side of the static mesh where the interaction of the button is going to be. Figure 13 shows the box collision placed on top of the static mesh to set where on the button it can become interactive as well as the direction when it gets pressed.

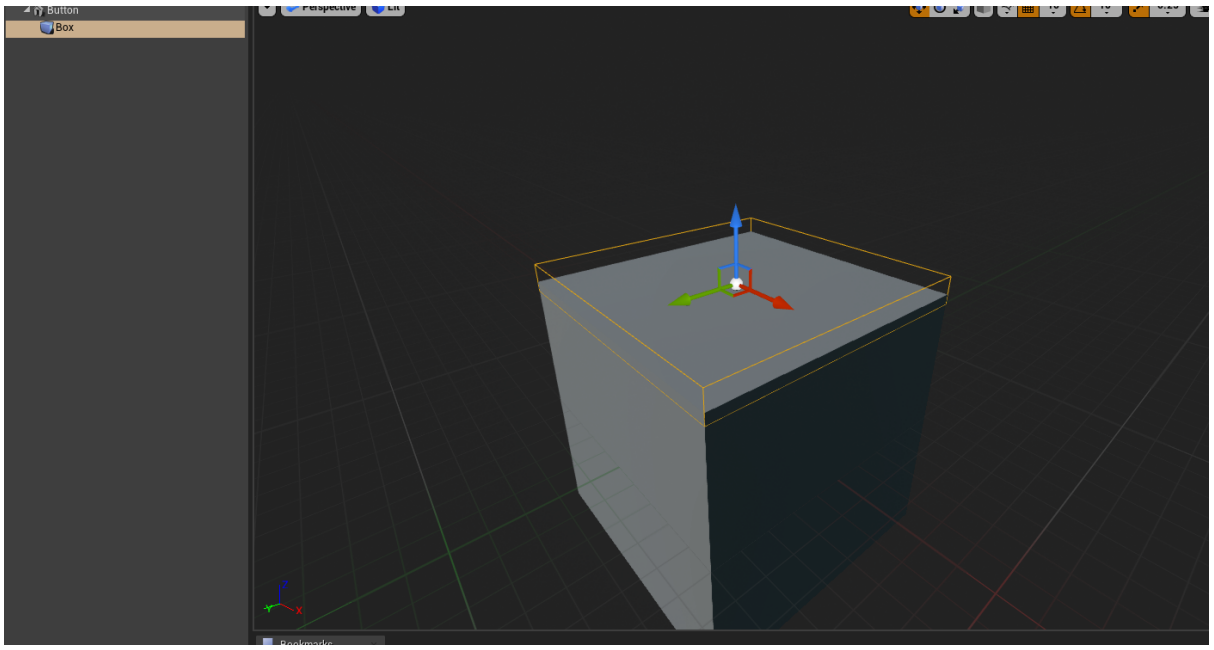

Figure 13. Static mesh with box collision.

Finally the script is created for the button to set its logic. Before creating the script a variable must be set for the button for when it is pressed. The figure below shows that a boolean variable must be set. The boolean variable allows the button to get pressed down and reset to its original position.

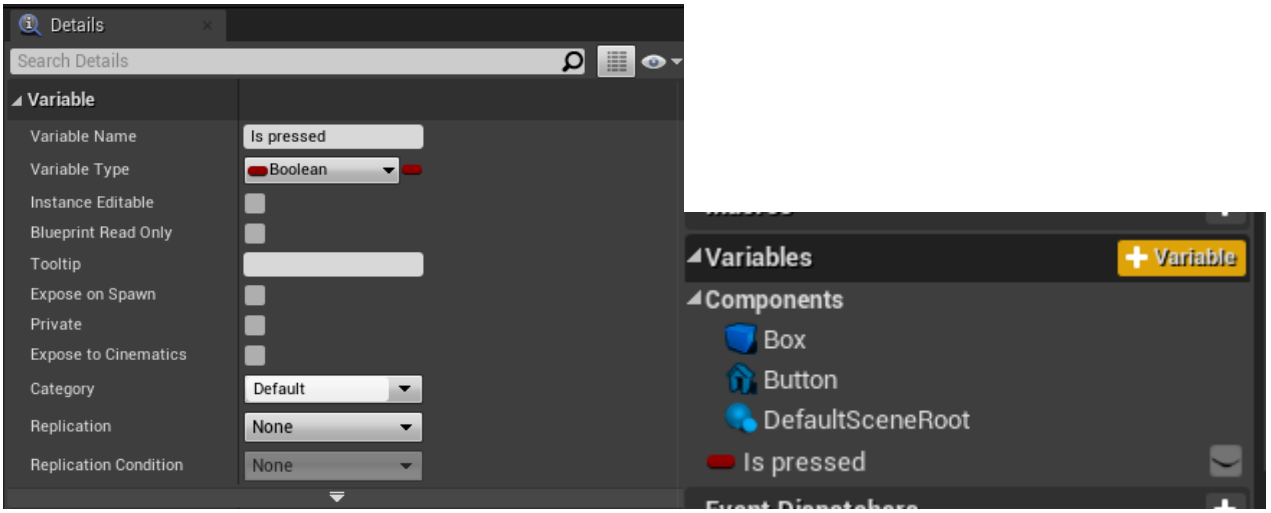

Figure 14. Implementing boolean variable - Is pressed.

The following figures (15-1 and 15-2) show all the necessary modules needed to create the functionality of the button. To allow it to get pressed, return to its original position and resetting the level so all objects return to their original positions as well.
The script for the buttons movement and logic is as the following figures:

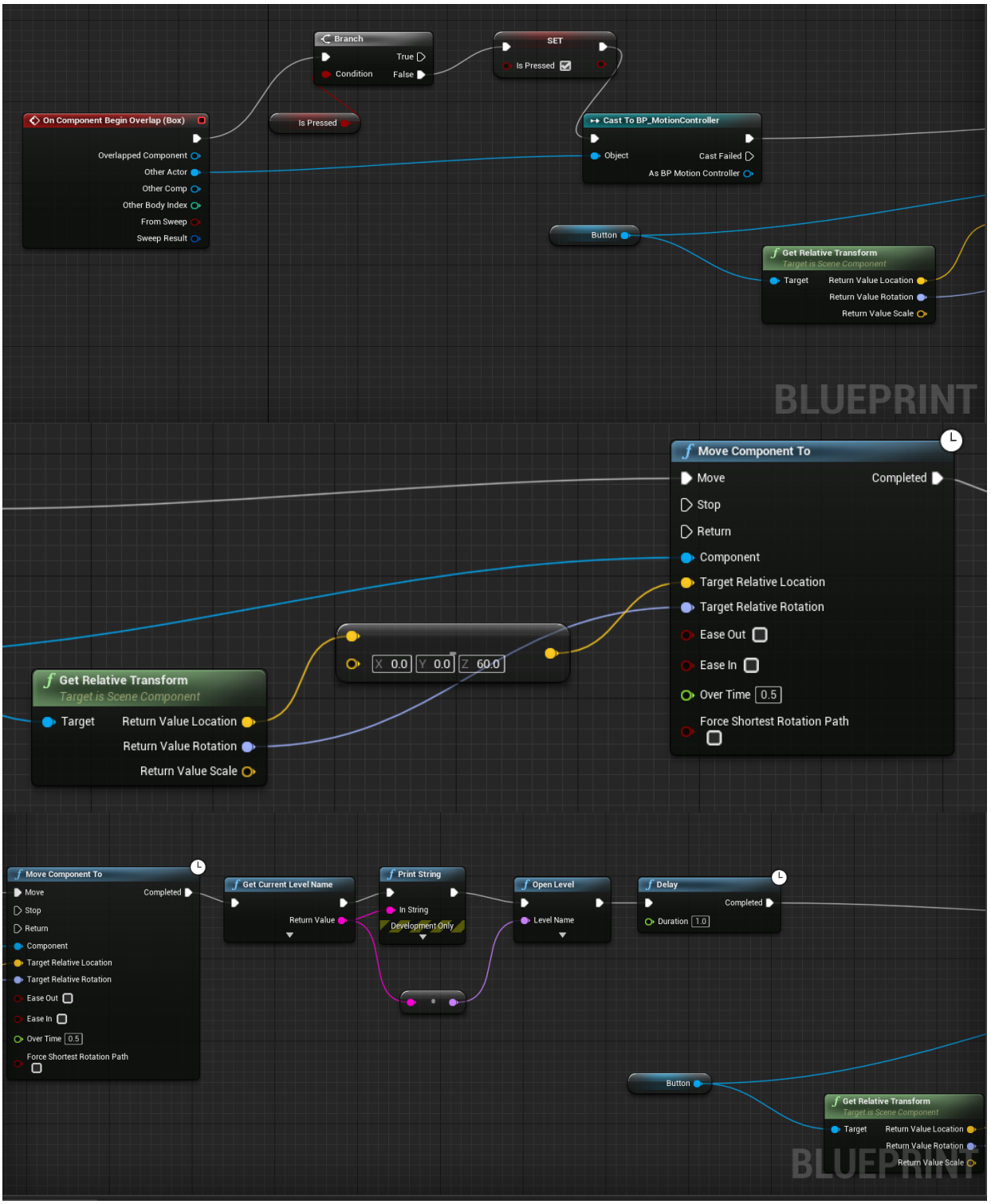

Figure 15-1. Script to reset VR level to original position.

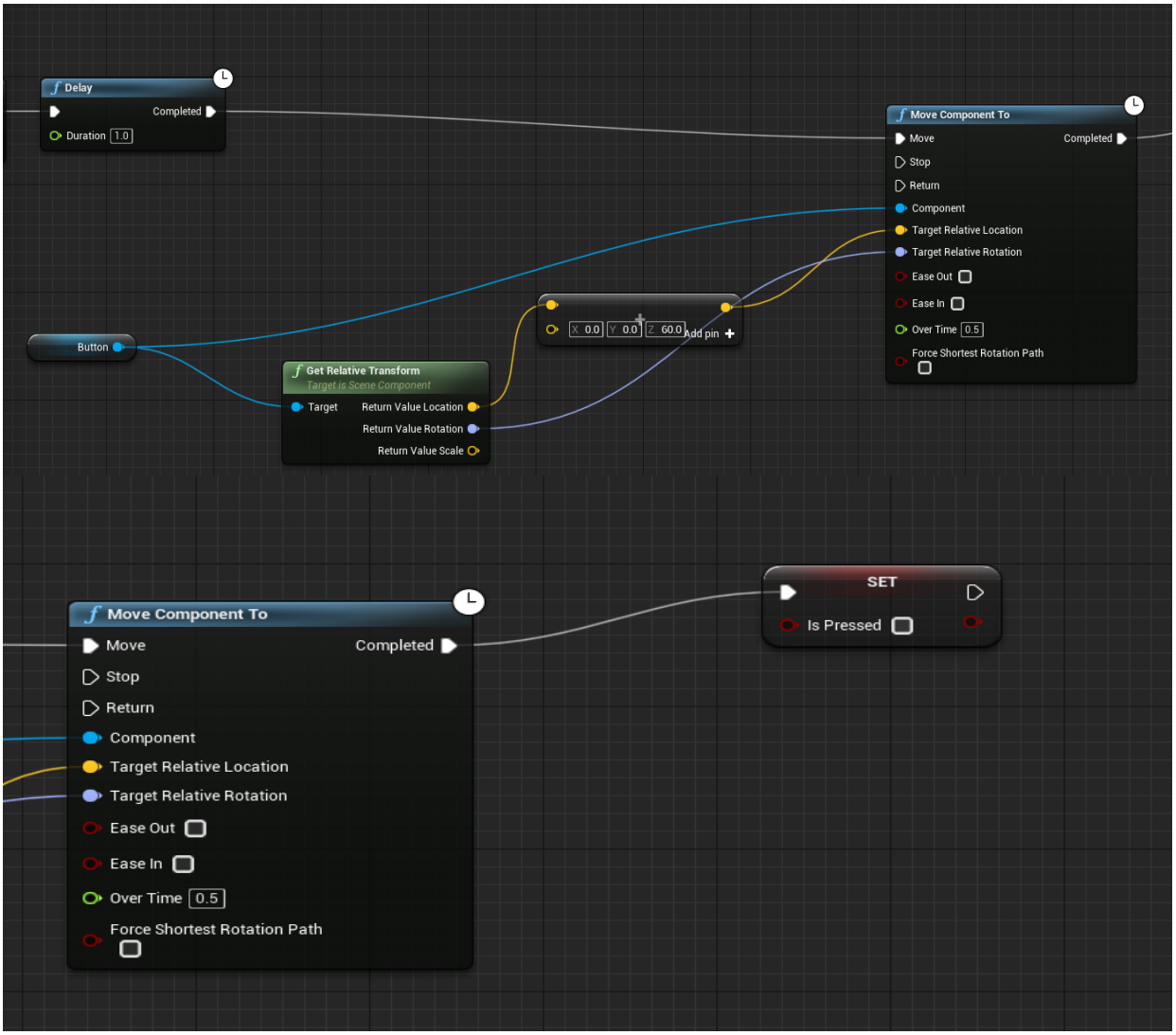

Figure 15-2. Script to reset VR level to original position.

4.2 Results of implementations

In this chapter, the results of all the implementations are presented. The information given from the software developer at HERO(Anton) in the interview resulted in a load of information and functionalities that should exist in the VR-application. Functionalities such as; simulating physics, to allow objects to become interactive. Box collision, to set the parameters on where on the static mesh one can interact with.

4.2.1 Fixing bug

The existing script did not accurately rotate the blades in proportion to reality. By analyzing the script, it was observed that the values assigned to the blades were the same for all blades. The implementation added in the script resulted in the blades to rotate correctly in VR.

4.2.2 Implementing interaction availability

This implementation resulted in objects that could become interactive in Virtual Reality. This will enable the operators to develop a broader understanding and feeling of how things are in reality while performing tasks in VR. The figure below shows the results of the static mesh after implementing interactive availability.

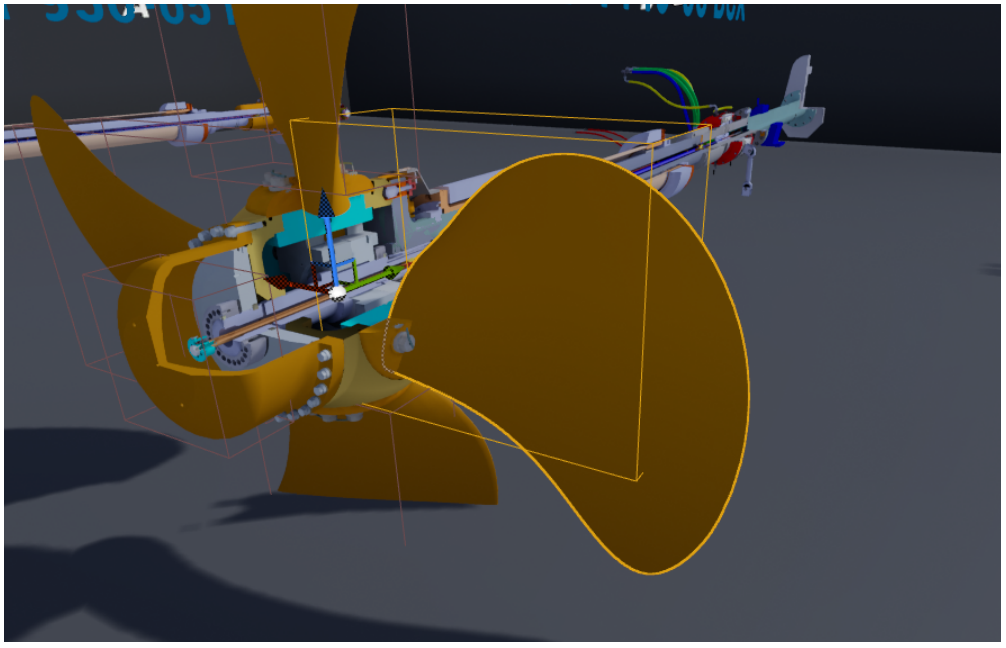

Figure 16. Result of implementing interactive availability.

4.2.3 implementing reset button

The results of implementing a reset button lead to that every object that has been interacted with can be reset to their original position, without the need to leave the program and restart it. This function allows the operator to save a huge amount of time, by allowing the possibility to reset the level in VR if a minor mistake occurs. Figure 17 shows how the objects have been moved around in the VR simulation. This is the result after implementing the interactivity availability, which allows movement for the static meshes.

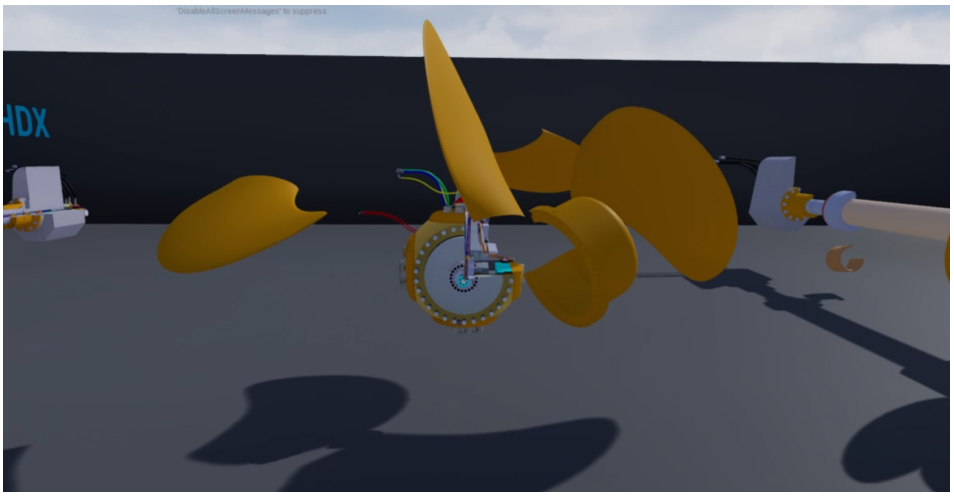

Figure 17. Result of moving the static meshes in VR.

Figure 18 shows that by pressing the button it will reset all the objects to their original position, as shown in Figure 19.

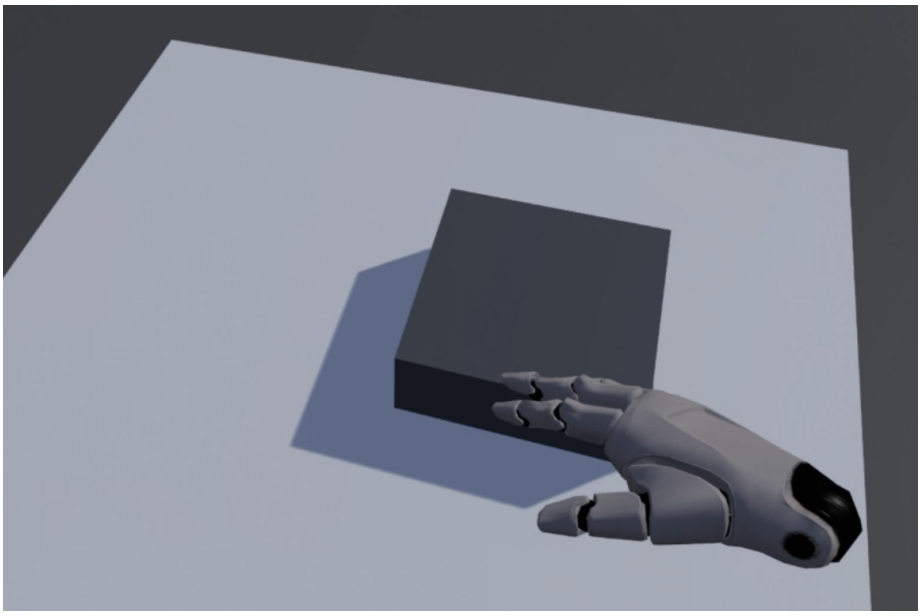

Figure 18. Button press in VR.

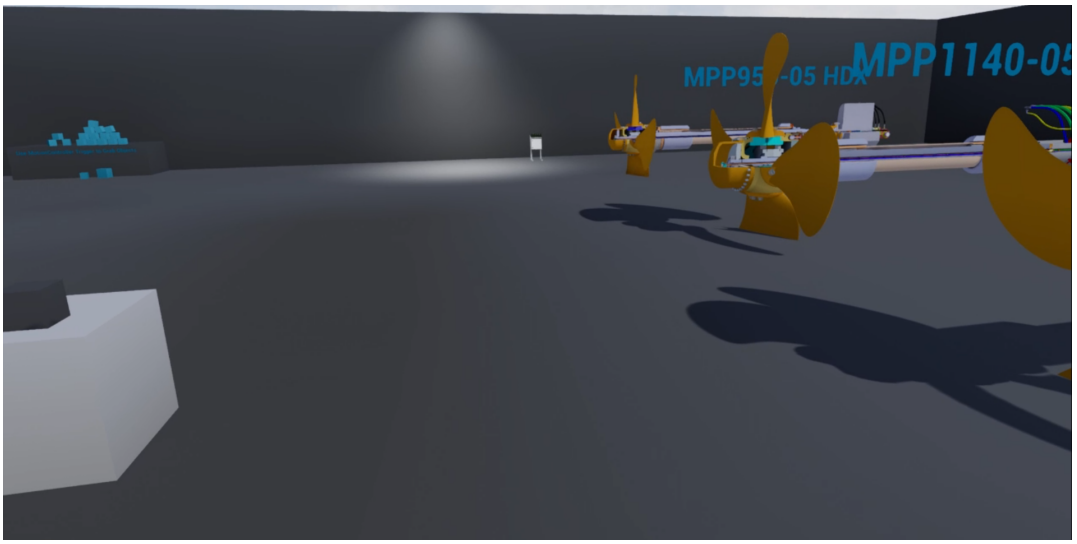

Figure 19. All objects return to their original position in VR.

4.2.4 Final Result

The final result of the given VR-application is shown in figure 20. The project has removed all unnecessary objects which had negative effects on the program which caused occasional lag and overloaded the file. The static meshes(objects) became interactive when simulating the VR-application and the implemented button allowed all the objects to return to their original position when pressed. Finally the rotation on the blades were corrected to rotate accordingly to how they should rotate in reality.

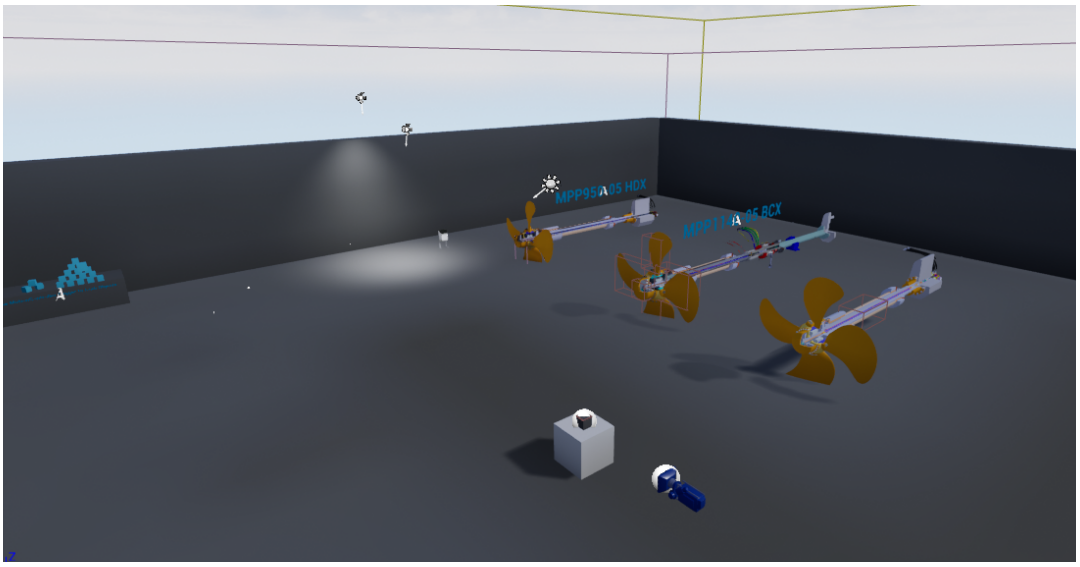

Figure 20. Final result of the project.

5 Discussion

In this chapter the finding and the implications of the research conducted for this bachelor thesis project, focused on exploiting the potential of virtual reality(VR), will be discussed.

5.1 Discussion results

The research findings and results demonstrate that virtual reality (VR) and augmented reality (AR) can bring significant advantages to Berg Propulsion, in terms of simulation and prototyping. By using VR and AR technologies the company can enhance the human-machine interface and allow operators, designers and engineers to reduce time and costs spent on production development and validation.

This bachelor thesis project shows clear results that everything set out to do has been achieved. To develop a strategy to effectively use VR and AR technologies, and to develop the given visualization VR-application to a more interactive level. While developing the VR-application, we noticed that Unreal Engine had a very clear and easy to use interface when going from importing cad files to making the objects interactive. All that was required to import the given CAD files was by using Datasmith, which allowed the objects to have all the necessary settings of Unreal Engine to become interactive instantaneous. After that it was simple to create the logic of the interaction for the objects. Sometimes problems could occur when creating the script as our understanding of the program was not at the highest level, it could easily be resolved by researching in Epic Games website on all available commands for Unreal Engine(Epicgames, 2023).

5.2 Research Questions

The first research question aimed to develop a strategy for Berg Propulsion to effectively use VR and AR technology in its product development process. Through the research and analysis of this thesis, it was found that VR and AR can bring significant benefits to Berg Propulsion, including improved design visualization, simulation and prototyping. The strategy should focus on integrating VR and AR into the existing product development workflow. Ensure that the staff is trained on the effective use of the VR and AR technologies and continuously evaluate and update the tools and techniques to stay up to date with the advancements in VR and AR. This will allow the one operating the VR or AR-application to reduce unnecessary steps when developing new products and reduce time and cost spent. The program that has been used during this research has proven to be very effective and easy to use, considering all the available tools. Unreal Engine has shown, within our research, to be exponentially more simpler to use than Unity from a beginners point of view. It utilizes a visual scripting system called Blueprints, which offers a user-friendly interface for designing game logic and interactions. With Blueprints, game developers can create complex functionalities without requiring extensive coding expertise. Unity relies on coding expertise and the utilization of plugins to use scripts and visual images in order to achieve comparable results as to Unreal Engine.

The second research question aimed to develop the given VR-application in Unreal Engine 4(UE4) used for visualization to create a more interactive level, with a focus on usability friendly purposes. The results show clearly that the given VR-application could become further developed into a more interactive level. In the beginning of the project all that we could interact with was the rotation of the blade which did not even rotate correctly. During the course of this project, we could fix the logic of the propeller so it rotated correctly. We implemented interactivity availability to objects in the VR-application and a step by step guide on how to achieve it in chapter 4.1.2. Finally we implemented a reset button to restore all the moved objects to their original position, without the need to restart the whole simulation program. This is all we could achieve with the knowledge and time given for this bachelor thesis. The research indicated that UE4 offers a robust platform for creating interactive experiences. A simple way to import CAD files using Datasmith plugins and an easy way to create the logic of the simulation using scripts in Blueprint.

5.3 Discussion comparison of Unreal Engine and Unity

Difference between Unreal Engine and Unity is the programming languages they use. Unreal Engine utilizes C++ for coding, while Unity relies on C#. This distinction can influence the preference of developers, as some may feel more comfortable and experienced with one language over the other.

When it comes to visual coding options, it was found that both engines provide alternatives to traditional programming. Unreal Engine offers Blueprints, a powerful visual scripting system that allows developers to create game logic without writing code. Unity has a similar feature called Bolt, which also enables visual scripting. These tools can be beneficial for those who prefer a more visual approach to game development or want to quickly prototype ideas without diving deep into programming. In terms of accessibility, We found that unity tends to be more beginner friendly. It offers a larger number of beginner guides and learning resources, making it easier for newcomers to get started. Also unity boasts a vast library of assets available for use, which can be advantageous for developers who want to leverage pre-made content and save time on asset creation.

Unreal Engine is renowned for its advanced graphical capabilities. It excels in rendering high quality light effects, resulting in visually stunning scenes. On the flip side, Unity is often praised for its superior shadow quality. These differences in graphical capabilities make both engines suitable for different types of projects, depending on the desired visual aesthetics.

The openness of the engines is an important factor to take into account. Unreal Engine stands out as an open source platform, allowing users to freely access and modify its source code according to their specific requirements. This openness can facilitate the development process and allow for more customization. Unity does not offer open source code by default, but it can be obtained through purchasing a license or gaining access to the Unity source code repository.

The problem with this thesis was that we didn't have enough time to conduct a proper experiment in virtual reality (VR)or augmented reality (AR) in Unity, that the discussion is based on limited experience with Unity, as it was primarily utilized for exploratory and research purposes rather than in depth study. Therefore, a more in depth analysis of Unity's capabilities and features would be necessary for a more comprehensive comparison.

5.4 Time Limitations

Although we found ourselves satisfied with the results of this bachelor thesis during these 10 weeks, there were additional things that we wished were implemented if more time was given, as well as how we could further develop this project.

5.4.1 Additional implementations

The additional implementations that should be considered into this project were Visual Dataprep. Dataprep allows the user to efficiently prepare and process large amounts of visual data for use in VR-applications. It provides a visual interface and a set of tools that simplify the process of organizing, filtering and manipulating data assets. In general terms, it prepares all the users desired settings for the imported objects. For example, importing a CAD file using Dataprep will allow the object to become interactive in the simulation without the need of preparing it as shown in chapter 4.1.2. This will reduce a lot of time spent on performing the same task repeatedly.

An additional implementation that would have been implemented is explosion view. This allows all the assembled objects to be separated instantaneously and every single object can be viewed separately. The idea is to implement another button in the VR-application to enable the function of the explosion view.

5.4.2 Further development

To further develop this bachelor thesis, additional experiments in augmented reality(AR) should be conducted. The current thesis has primarily focused on the usability of VR and neglected AR. Even though VR has more convenience and easier accessibility than AR, more experiments should be conducted to ensure VRs validation. Additional testings of the VR-application should be performed in Unity as well to compare Unity and Unreal Engines programming for VR availability. This will give a more clear understanding on which of the programs are the best and more convenient to use for the product development process within VR.

Within this bachelor thesis, if more time was available, we would have liked to test more tools in Unreal Engine. This would have allowed us to understand more functionalities that could be implemented into the VR-application. And that more research on imported files be conducted, if other CAD files other than STL from Solidworks, could be imported into Unreal Engine and if they would provide the same functionalities and VR compatibilities as Unreal Engines static meshes.

Recently Unreal Engine 5 has been released from Epicgames and it would also be very important to conduct further studies on how to transition from Unreal Engine 4 to UE5.

6 Conclusion

To conclude this bachelor thesis, the research findings demonstrate the significant advantages that VR and AR can bring to Berg Propulsion, including improved design visualization, simulation, and prototyping. By utilizing VR and AR, Berg Propulsion can enhance the human-machine interface and reduce time and costs spent on production development and validation. The development of a strategy for effectively using VR and AR technology in the product development process was achieved through the research and analysis conducted in this thesis. The strategy's main objectives are to incorporate VR and AR into the current workflow, teach the workforce, and continually review and update the technologies and methods used. According to the research findings, Unreal Engine 4 (UE4) provides a reliable foundation for building interactive VR experiences. A reset button for item positioning, bug fixes, and interactive functionalities are all demonstrated in the VR application created in UE4.

The comparison between Unity and Unreal Engine reveals how different their programming languages and visual coding possibilities are. A user-friendly interface for creating game logic and interactions is offered by Unreal Engine's use of C++ and Blueprints. In contrast, Unity uses C# and provides a similar visual scripting tool called Bolt that uses plugins. With more starting guidelines and a broader bank of assets, Unity is thought to be more user-friendly for beginners. Due to time constraints, a more thorough examination of Unity's capabilities and features is required for a thorough comparison. Future research in this area will involve doing additional augmented reality (AR) tests to examine its VR-application in the creation of new products. Additional tests in Unity should be run to compare the programming capabilities of Unity and Unreal Engines VR accessibility and compatibility. Additional investigation of imported files and their compatibility with Unreal Engine may shed light on how to enhance the VR application's features.

In the end, this bachelor thesis has successfully achieved its two main objectives, the development of a plan for efficiently employing VR and AR technologies and the creation of an interactive VR application. The research highlights the benefits of incorporating VR and AR into the process of developing new products, as well as the efficiency of Unreal Engine 4 in producing interactive VR experiences. This study adds to our understanding of how VR and AR may improve the design visualization, simulation, and prototyping processes, allowing businesses like Berg Propulsion to build products more quickly and affordably.

References

[1] *Unreal Engine | The most powerful real-time 3D creation tool*. (2023). Unreal Engine. https://www.unrealengine.com/en-US

[2] *Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine*. (2023). https://unity.com/

[3] *Coding Standard*. (2020). https://docs.unrealengine.com/5.1/en-US/epic-cplusplus-coding-standard-for-unreal-engine/

[4] Technologies, U. (2023). *Learning C sharp in Unity for beginners*. Unity. https://unity.com/how-to/learning-c-sharp-unity-beginners

[5] Technologies, U. (2023). *Unity - Manual: Visual Scripting with Bolt*. https://docs.unity3d.com/2019.3/Documentation/Manual/VisualScripting.html

[6] *Unreal Engine 4 For Unity Developers*. (2020). Unreal Engine 4.27 Documentation. https://docs.unrealengine.com/4.27/en-US/Basics/UnrealEngineForUnityDevs/

[7] *Unity Documentation*. (2023). Unity Documentation. https://docs.unity.com/

[8] *MetaHuman | Realistic Person Creator - Unreal Engine*. (2020). Unreal Engine. https://www.unrealengine.com/en-US/metahuman

[9] *Image Settings*. (2023). https://docs.unrealengine.com/5.0/en-US/cinematic-rendering-image-quality-settings-in-unreal-engine/

[10] Technologies, U. (2023). *Real-time rendering in 3D*. Unity. https://unity.com/how-to/real-time-rendering-3d

[11] *Introduction to Lighting and Rendering - Unity Learn*. (2023). Unity Learn. https://learn.unity.com/tutorial/introduction-to-lighting-and-rendering#

[12] Arora, S. K. (n.d.). Unity vs Unreal: Which Game Engine Should You Choose? *Hackr.io*. https://hackr.io/blog/unity-vs-unreal-engine

[13] *Collision Overview*. (2023). https://docs.unrealengine.com/5.2/en-US/collision-in-unreal-engine---overview/

[14] *Blueprints Visual Scripting*. (2023). https://docs.unrealengine.com/5.2/en-US/blueprints-visual-scripting-in-unreal-engine/

[15] *Static Meshes*. (2023). Unreal Engine 5.2 Documentation. https://docs.unrealengine.com/5.2/en-US/static-meshes/

[16] *Actors*. (2023). https://docs.unrealengine.com/5.2/en-US/actors-in-unreal-engine/

[17] *Datasmith Overview*. (2023). https://docs.unrealengine.com/5.2/en-US/datasmith-plugins-overview/

[18] *Editor Viewports*. (2023). https://docs.unrealengine.com/5.2/en-US/editor-viewports-in-unreal-engine/

[19] *Working with Scene Variants*. (2023). https://docs.unrealengine.com/5.2/en-US/working-with-scene-variants-in-unreal-engine/

[20] *Content Browser Interface*. (2023). https://docs.unrealengine.com/5.2/en-US/content-browser-interface-in-unreal-engine/

[22] *Level Editor Toolbar*. (2023). https://docs.unrealengine.com/5.2/en-US/level-editor-toolbar-in-unreal-engine/

[23] *Unreal Engine API Reference*. (n.d.). Unreal Engine Documentation. https://docs.unrealengine.com/5.2/en-US/API/