INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Optimização de uma Plataforma de Treino Militar da Sistrade: Melhoramento da Usabilidade Colaborativa e Visualização

RAFAEL RODRIGUES CARDOSO
Junho de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Sistrade Military Training Platform Optimization: Enhancing Collaborative Usability and Visualization

Rafael Rodrigues Cardoso

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática, Área de Especialização em Jogos, Sistemas Gráficos e Interativos

Supervisor: Paulo Oliveira

External Supervisor: Inês Caetano

Porto, January 2025

Integrity Declaration

I hereby declare that I have conducted this academic work with integrity.

I have not plagiarized or applied any form of undue use of information or falsification of results along the process leading to its elaboration.

Therefore, the work presented in this document is original and authored by me, having not previously been used for any other end. The exceptions are explicitly recognized in the section "Ethical considerations" of the first chapter. This section also states how AI tools were used and for what purpose.

I further declare that I have fully acknowledged and am aware of the Code of Ethical Conduct of P.PORTO.

ISEP, Porto, 28 de June de 2025

Resumo

O treino militar é uma necessidade tanto para os soldados como para aqueles que os comandam, uma vez que estes últimos influenciam mais vidas do que qualquer soldado. Ter comandantes mais disciplinados implica melhores ferramentas e programas de treino. Este trabalho centra-se na melhoria de uma plataforma militar de simulação construtiva desenvolvida pela SISTRADE.

A plataforma em questão já está desenvolvida, mas apresentava problemas de desempenho ao carregar cenários e baixa usabilidade ao editar cenários, executá-los e interagir com eles numa visualização de Realidade Aumentada, levando a uma experiência de simulação insatisfatória. Também foram implementadas anteriormente funcionalidades de colaboração, que não possuem nenhum "feedback" ao utilizador, o que poderia resultar em perdas de dados.

Face a estes problemas, os objetivos principais deste trabalho foram: a melhoria da performance de carregamento de cenários da plataforma; a melhoria da usabilidade da edição e execução de cenários; e, o melhoramento das capacidades de colaboração da aplicação.

O trabalho envolveu a realização de uma revisão sistemática da literatura existente que podia ajudar a resolver estes problemas, utilizando-se o PRISMA. Como resultado, esta dissertação sintetiza e agrega todas as informações encontradas no estado da arte. Estas informações são, então, aplicadas onde necessário, à medida que a plataforma existente é analisada, os seus problemas identificados e propostas as soluções.

A partir destas soluções propostas, é implementada uma seleção delas, com o objetivo de cumprir os objetivos deste trabalho. As implementações surgem sob a forma de melhorias seletivas à plataforma, como por exemplo, implementação de visualização distintiva entre utilizadores aquando da utilização das funcionalidades de colaboração, melhor distinção entre equipas e unidades num cenário de simulação, e melhoramento do carregamento de cenários.

Estas melhorias são depois avaliadas pela sua eficácia na resolução dos problemas apresentados.

A maioria dos objetivos foi cumprida, nomeadamente, a usabilidade da visualização do mapa de cenários foi melhorada e o aspeto da colaboração tornou-se mais intuitivo. O carregamento do mapa de cenários não foi melhorado e necessita de mais desenvolvimento e correções.

Palavras-chave: Realidade Aumentada, Simulação LVC, Simulação Construtiva, Otimização de Software, Renderização Web, WebXR, Angular, Node.js, TypeScript, Babylon.js

Abstract

Military training is a necessity for both soldiers and those who command them, as they influence more lives than any soldier. Having better disciplined commanders comes with better training tools and programs. This work focuses on improving a constructive simulation military platform developed by SISTRADE.

The platform in question is already developed, but it had problems such as performance issues when loading scenarios and poor usability while editing scenarios, executing them, and engaging with them in an Augmented Reality view, leading to a poor simulation experience. Collaboration features that were previously implemented must be improved to give more feedback to the user and prevent data loss from occurring.

Given these problems, the main objectives for this work were: the improvement of the performance of the scenario loading in the platform; the improvement of the usability of the scenario execution and editing components of the application; and, the improvement of collaboration functionalities.

This report carried out a systematic mapping review on existing literature that could help resolve the problems mentioned by utilizing PRISMA. As a result, this work aggregates all information found in the state of the art. This information is then applied where needed as the existing platform is analyzed, its problems discovered, and solutions are proposed.

From these proposed solutions, a selection is implemented, with the goal of fulfilling the objectives of this work. These improvements are then evaluated by their efficacy in solving the problems presented. The implementations come in the form of selective improvements to the platform, such as the implementation of distinctive visualization between users when using the collaboration features, implementation of better GUI, facilitating identification of units between teams and units in a simulation scenario and improving scenario loading.

Most objectives were fulfilled, namely, the usability of the scenario map view was improved, and the collaboration aspect became more user-friendly. The scenario map loading was not improved and needs further development and corrections.

Keywords: Augmented Reality, LVC Simulation, Constructive Simulation, Software Optimization, Web Rendering, WebXR, Angular, Node.js, TypeScript, Babylon.js

Thanks

I'd like to thank my family and friends for always supporting me throughout my formative years, because without them, I couldn't have made it.

I'd also like to thank the organization, especially my supervisor Inês Caetano, for all the support she provided while this work was being developed.

Lastly, I'd like to thank my orienting teacher, Paulo Oliveira, for always being available when I had any questions regarding the work, and for being an excellent orienting teacher.

Table of Contents

Figure List

Table List

Acronyms

Acronym List

API – Application Programming Interface
AR – Augmented Reality
DOM – Document Object Model
FCP – First Contentful Paint
FOV – Field of View
HMD – Head-Mounted Display
HTML – Hyper Text Markup Language
INP – Interaction to Next Paint
ISO – International Organization for Standardization
LCP – Largest Contentful Paint
LOD – Level of Detail
LVC – Live-Virtual-Constructive
MR – Mixed Reality
MVP – Minimum Viable Product
OSM – OpenStreetMap
PICOCS – Population, Intervention, Comparison, Outcome, Context, Study Design
PRISMA – Preferred Reporting Items for Systematic Reviews and Meta-Analyses
RA – Research Area
RQ – Research Question
SME – Small and Medium-sized Enterprise
SUS – System Usability Scale
TBT – Total Blocking Time
UC – Use Case
WBS – Work Breakdown Structure

1 Introduction

The first chapter of this dissertation is an overview of the dissertation project “Sistrade Military Training Platform Optimization: Enhancing Collaborative Usability and Visualization”. This project was carried out to obtain a master’s degree in software engineering at Instituto Superior de Engenharia do Porto (ISEP).

It serves as an introduction to the whole work, contextualizing the reader on the problem and the dissertation’s objectives, as well as the contributions this project has, ethical considerations that have been kept in mind while the dissertation was written, and the document’s structure.

1.1 Context

Training is indispensable for modern military forces, which need to be ready against threats from the land, sea and air, with new technologies being developed every day in each of these core fronts. Arranging training exercises against these threats might have been too difficult in the past, difficult to arrange, or not worth the cost of possible loss of equipment, or even the guarantee that the training outcome would be that personnel were trained appropriately [1].

With the rise of the digital age, more and more armies are integrating specialized simulation platforms to train their soldiers in a more convenient, safe and cost-effective way, that guarantees soldiers can be ready through less dangerous and non-destructive means [2]. These digital platforms support the creation of many scenarios, from localized small exercises to large-scale operations that encompass the three fronts.

SISTRADE’s existing military platform is one of these innovative tools, that allow European soldiers to train without the limits of exorbitant setup costs, questionable safety and doubtful troop skill acquisition.

The platform is a “training simulation creator”, which allows training simulations to be created with various units, such as boats, planes and infantry, allowing this exercise to play out virtually in the simulated world of the platform. It allows the creation and exploration of different scenarios which would be difficult to organize otherwise, or have significant costs associated with running them in the real world.

However, the existing solution presents some problems that must be addressed to allow the solution to be accepted by stakeholders. Addressing these limitations would allow the platform to fulfill the requirements needed for acceptance, enhance operational efficiency, improving military training outcomes and ensuring the relevancy of the platform for years to come, as the field and the strategies employed are constantly changing and evolving with each passing day.

1.2 Problem Description

Military simulation platforms play a crucial role in the defense of a country, allowing soldiers to be ready against many tactics being employed by an enemy. They make training easy for soldiers and allow them to acquire knowledge at a rate much superior to that of live exercise. SISTRADE's platform takes this to another level, by integrating AR (Augmented Reality) into its suite of functionalities. However, the platform has some problems that must be addressed to achieve the intended functionality and provide a better experience for all users.

One of these problems is the usability of the system while the user is in an AR training environment, as it is hard to move around or pivot the user's point of view to look at specific points on the AR map, and it is difficult to move around too. There is also a lack of collaboration in this mode, as users cannot see each other's points of view, or where they are stationed. This creates a degraded viewing experience that is not suited for trainees nor instructors.

An additional problem arises from the fact that the platform requires heavy computational power, as it needs to load mapping data and 3D models, which take quite a long time to load on most devices, and worse on low performance devices.

This hinders the simulation building environment, leading to an unpleasant experience while using the platform. This work intends to correct that.

1.2.1 Objectives

From the problem description, the objectives of this dissertation are clear:

1. Improve the usability of the application's AR component and scenario views to allow it to be a more cohesive and immersive experience than what is currently implemented.
2. Improve the collaborative functions of the application by implementing user awareness.
3. Improve the performance of the map data scenario rendering that is already implemented by SISTRADE.

More specifically, this work intends to document and show the results of researching the current state of the art for the necessary information required for the fulfillment of the above objectives, design and implement these necessary improvements and evaluate the improved application against the previous implementation.

Aligned with these objectives, the improvements made to the existing application have allowed the application to provide a better experience in terms of usability and usefulness to all stakeholders and interested parties.

1.3 Contributions

The contributions from this work are the improvements to the platform that has been developed in conjunction with SISTRADE's team to maximize the value that the platform can bring to its stakeholders.

The project operates within the complex and regulated defense sector, developed as a company internship with Sistrade, an SME, under a European initiative that unites key players like major defense companies, governmental agencies, and industry leaders. Its main goal is to enhance interoperability, efficiency, and resilience in military operations, focusing on the national air force as the end-user.

The intricate nature of the defense ecosystem and the project's European scope require the collaboration of diverse stakeholders, from defense experts to commercial enterprises.

These contributions aim to make training easier and more convenient for military organizations and their personnel, by also providing a heightened collaborative experience.

The platform contributes to the effective training of trainees of commanding positions in the military, who would make large scope decisions, by allowing them to collaborate with their instructors in an AR environment, accelerating learning by being instructed as they practice. The platform allows the creation of different scenarios to suit different trainees or situations they might encounter out in the field.

1.4 Ethical considerations

This work, as written at the beginning of this document, follows the ethical code of conduct of P. PORTO [3]. Some sections of this ethical code of conduct that are of interest to the reader are the duties of the student that wrote this document, as denominated in the article as *Artigo 6º* (6th Article). Namely, point 2.8, which states that no phrases, ideas or paragraphs shall be used without the required citation protocol to correctly identify where the content came from. Citations will be included by recurring to the IEEE standard [4].

Artigo 8º (8th Article) is also respected, as there is the declaration of integrity, earlier in this document. All external code has been checked for licensing and checked with the supervisor if it could be used in the implementation chapter of this dissertation.

Artificial intelligence large language models were not utilized in writing this work.

Besides these ethical research guidelines, the writer also wants to state that, in **no circumstances** should the application that has been developed, be considered "fit for purpose" in its functionality or use cases after the work has been completed. The improvements made to the application **do not** guarantee its usage in real scenarios, where lives are at stake and dependent on efficient and correct training of soldiers.

The improvements made do have a purpose, but these do not prepare the application for usage in real scenarios.

The author would also like to clarify that some parts of this work have been written or produced for another curricular unit, namely, PREPD (Preparação para a dissertação), and have been repurposed for use in this work. These chapters and the relevant sections are listed below:

- Introduction (Sections 1.2.1, 1.4, and 1.5 have been slightly re-written).
- State of The Art (From section 2.3 onwards, the state of the art was re-written for this work. Previous sections were written in PREPD and are largely unchanged).
- Annex I – Planning Annexes were annexes that were utilized in the PREPD report.

Any chapter or section not identified here is original to this work.

1.5 Methodology and Planning

The methodology used in this work is presented in this section to contextualize the work that has been done.

This work uses the **Design and Creation** research strategy, as it aims to improve a software platform by analyzing existing technologies, designing improvements to implement, applying these improvements to an already existing application and finally, evaluating the effectiveness of the implemented improvements and draw conclusions about the success of the work. In accordance with this strategy, each of this work's chapters represent a phase of design and creation.

This strategy was chosen because it fits well with the requirements and objectives of the project. Even if the result is not a completely new application, improvements that were scheduled to be implemented were part of the analysis phase, while the implementation and testing of these improvements were placed in the build and evaluation phases respectively.

The development of the project was carried out by utilizing an agile methodology, where weekly sprints were created for the implementation of relevant features. According to the status and importance of these implementations the features of the next sprint were decided in a meeting held after the sprint's completion.

The development phase of the project began on the 1st of March 2025 and ended on the 31st of May 2025. From this point onwards it was not possible to make any alterations to the project's source code or implement any new features.

1.6 Document Structure

In addition to this introduction chapter, this document has the following chapters:

- A State-of-the-Art chapter, being the collection of research done to achieve this work's goals.
- An Analysis chapter, where the platform is introduced, it's problems shown, and solutions will be drafted in accordance with the State-of-the-Art.
- A Development chapter, disclosing what solutions were implemented to achieve the work's objectives, as well as how these were implemented.
- An evaluation chapter, comparing various functionalities regarding their optimization and usability factors, reaching a conclusion about the effectiveness of the implemented developments.
- A conclusion chapter, presenting conclusions from the work that was developed, how well it aligned with the project's objectives, and future work to be done.

The author hopes that with these chapters, the reader can understand the relevance of the work that has been done, what has been done to make the platform better, and if these developments were effective in attaining the objectives mentioned earlier in this chapter.

2 State of The Art

The aim of this chapter is to perform a literature review by utilizing a systematic mapping approach and then present an analysis of the findings and how they could be used to improve the current application. It will serve as the basis for any improvements made to the existing application, as well as highlighting what can be done to make the application more performant, usable and collaborative.

2.1 Literature Review

For this literature review, a systematic mapping study will be carried out to provide an overview of the research area, and answer the research questions for the study, which will help future improvements to the existing application.

Systematic mapping studies often give a general overview of a research area, with their goal being to scope it out and determine if there are any gaps in the research [5].

The protocol that the author will use to conduct the systematic mapping study will be the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) protocol, and it will have sections for each stage of the protocol, and what has been done for each of the stages [6]. At the end, a diagram will be presented with the process.

As the research area of this work is Augmented Reality (AR), Military training and Software optimization, the systematic mapping study to be done will mainly cover these areas.

2.1.1 Research Questions

For a systematic mapping study, research questions are needed. These questions are directly related to the existing application, as they are targeting the specific objectives laid out in the introduction chapter. The main goal of these research questions is to find a solution to the application's most glaring problems by finding the most relevant studies to analyze. With the main question of this work being "How can an augmented reality platform be improved in terms of performance and usability?", the questions derived from it are as follows:

- **RQ1 -** *How effective are augmented reality simulation platforms in enhancing decision-making skills in military training compared to traditional simulation methods or live training?*
- **RQ2 -** *How can APIs be optimized to improve response times for requests involving large or complex datasets?*
- **RQ3 -** *How can the usability of AR-based military training platforms be improved so that it can be used by trainees and instructors?*

These research questions, as said before, are targeted at finding the best ways to improve the application, especially RQ2 and RQ3. These two questions are targeted at improving the application, while RQ1 is targeted at finding the effectiveness of AR applications while comparing them to traditional soldier training methods. RQ2 directly targets objective 3, which would allow the platform to be improved in terms of performance, while RQ3 directly targets objectives 1 and 2, aiming to find studies where usability improvements make these types of platforms better.

2.1.2 Research Question Treatment (PICOCS)

After defining the research questions, the PICOCS method [7] was used to arrive at the Inclusion and Exclusion criteria for each question, and to each question's research strings.

The following sections aim to show how each research question was treated by using the PICOCS protocol, and how the search substrings were acquired.

2.1.2.1 RQ1

How effective are augmented reality simulation platforms in enhancing decision-making skills in military training compared to traditional simulation methods or live training?

Table 1 – PICOCS application to RQ1

PICOCS	RQ part	I/E	Sub-String
Population	recent studies in military area	I - studies in the military area E - before 2019, general studies about augmented reality	military field; military trainees
Intervention	using Augmented Reality	I - AR application in training	AR, Augmented Reality
Comparison	--	--	--
Outcomes	enhancing decision making	I - measurable outcomes, including decision speed, accuracy, or quality	decision speed; decision accuracy;
Context	military training	I - military context; desktop simulations; live exercises; E - non-military context	live; desktop simulation;
Study Design	empirical studies; comparative studies	E - surveys;	Empirical study; Comparative study;

2.1.2.2 RQ2

How can APIs be optimized to improve response times for requests involving large or complex datasets?

Table 2 – PICOCS application to RQ2

PICOCS	RQ part	I/E	Sub-String
Population	studies about APIs	I - API performance optimization studies;	API; application programming interface;
Intervention	software optimization	I - software optimization techniques;	API optimization; performance optimization; software optimization;
Comparison	--	--	--
Outcomes	improved API response times	I - response times measured in ms/s; latency reduction; benchmarks E – optimizations with no impactful results	response time; latency; performance benchmarks
Context	large datasets	I - studies that handle large amounts of data	Complex data; large datasets;
Study Design	Empirical studies and benchmarks	I - studies that can show the difference between optimization techniques; empirical studies; benchmarks;	empirical study; benchmark;

2.1.2.3 RQ3

How can the usability of AR-based military training platforms be improved so that it can be used by trainees and instructors?

Table 3 – PICOCS application to RQ3

PICOCS	RQ part	I/E	Sub-String
Population	studies about software usability	I - military trainees or instructors using AR platforms	military trainees; military instructors
Intervention	Augmented Reality	I - Augmented Reality	AR; Augmented Reality;
Comparison	--	--	--
Outcomes	usability improvement	I - proven techniques (SUS, success rates, task completion time, feedback) benchmark;	SUS; Success rate; Time measurement
Context	usage by military personnel	I - studies where the platforms are used by military personnel	military; trainee; instructor;
Study Design	Empirical or case studies	I - studies that have measurements that can be	Empirical study; Case study;

		compared, or benchmarks between usability improvements;	

As there is some overlap between RQ1 and RQ3, the query string will be merged into one query string to use in the digital libraries that are to be selected in the next section. RQ2 will have a different query string, as it covers a sufficiently different area, namely, software optimization. They are not merged into one, as introducing a research question from a different area into RQ1 and RQ3's query string might introduce noise into the search.

As such, the final research query string for RQ1 and RQ3 is:

```
("military" OR "military training" OR "military personnel") AND
("augmented reality" OR "AR") AND
("simulation" OR "desktop simulation" OR "live exercises" OR "training
environments") AND
(("decision-making" OR "decision speed" OR "decision accuracy") OR
("usability" OR "SUS" OR "success rate" OR "time measurement" OR "user
satisfaction")) AND
("empirical study" OR "comparative study" OR "case study")
```

The query string for RQ2 is:

```
("API" OR "application programming interface") AND
("response time optimization" OR "software optimization" OR "performance
optimization") AND
("response time" OR "latency") AND
("large datasets" OR "complex data") AND
("empirical study" OR "benchmark")
```

This is a generalized representation of the queries used for searching in the databases and it does not represent the actual queries utilized in the data sources that are presented in the next section, however, the search terms are equal to those used.

2.1.3 Data Sources

The chosen data sources for this study are digital libraries that have a vast number of articles pertaining to software engineering, augmented reality development and military research fields. The main sources can be seen in Table 4, along with their access points and the ID that will be used to refer to each data source later.

Table 4 – Chosen digital libraries.

ID	Name	URL
DL1	ACM Digital Library	https://dl.acm.org
DL2	IEEE Xplore Digital Library	https://ieeexplore.ieee.org
DL3	ScienceDirect (Elsevier)	https://www.sciencedirect.com

DL1 and DL2 were chosen for their track record and quality of computer science articles, while DL3 was chosen as a fallback, as it can help with searching more “hidden” subjects, or subjects that might not have been published to the other two data sources.

2.1.4 Identification Stage

The first stage in the PRISMA protocol is the identification stage. In this stage, it is expected that only numbers are recovered from searching the digital libraries with the query strings that were built earlier.

As this work possesses two distinct areas of research in its research questions, two separate PRISMA processes will be conducted. Each of these processes will be identified along the stages of the PRISMA protocol, starting with the identification stage.

Table 5 below reports the number of records retrieved from each data source for each of the research areas.

Table 5 – Number of records retrieved from data libraries.

ID	Research Area	DL1	DL2	DL3	Total
RA1	Military AR Application (RQ1 & RQ3 Query)	99	92	2	193
RA2	API Software Optimization (RQ2 Query)	44	22	99	165

The research areas were given an ID in the table that will be used throughout the rest of the report.

These results were obtained by utilizing the queries defined above, in each of the defined data sources. These queries were also accompanied by platform specific filters, such as restricting the filtering to only research articles and if possible, only including articles from 2019 onwards.

With these filters, and without duplicate removal (it was not needed, as no duplicates were found), the total number of articles gathered was 358, and in the subsequent stages, this number will be reduced as inclusion and exclusion criteria are applied to these articles.

2.1.5 Screening Phase

In the screening phase, inclusion and exclusion criteria will be grouped from the PICOCS analysis of research questions and will subsequently be applied to the records retrieved from the data libraries.

This will be done by reading the articles' abstracts and titles and filtering them based on the inclusion and exclusion criteria.

Since there are two reviews being conducted in parallel, there will be different inclusion and exclusion criteria for both searches conducted. Table 6 and Table 7 list the criteria to be considered for each of the research areas.

Table 6 – Inclusion and exclusion criteria for RA1.

Inclusion Criteria	
ID	**Criteria**
IC1	Studies that focus on Augmented Reality or similar immersive training tools.
IC2	Studies that analyze Augmented Reality in teaching contexts.
IC3	Studies that include comparisons between desktop simulations and AR simulations or compare these against regular live simulations.
IC4	Studies that measure the usability of military AR platforms.
IC5	Studies where the AR platforms are used by military soldiers or their instructors.
IC6	Studies that have various measurements or benchmarks of usability.
Exclusion Criteria	
ID	**Criteria**
EC1	Studies before 2019.
EC2	Records that are not research papers.

Table 7 - Inclusion and exclusion criteria for RA2.

Inclusion Criteria	
ID	**Criteria**
IC1	Studies that cover API Optimization techniques or efforts.
IC2	Studies that cover software optimization techniques.
IC3	Studies that have recorded results and metrics about the optimizations. (response times, latency reduction)
IC4	Studies that handle large amounts of data.
IC5	Studies that show the difference between many optimization techniques.
IC6	Empirical studies and Benchmarks covering many techniques.
Exclusion Criteria	
ID	**Criteria**
EC1	Studies that provide optimizations, but no real proof of performance improvements.
EC2	Studies about general optimization.
EC3	Records that are not research papers.

Criteria for RA1 were broadened when compared to the PICOCS analysis of the questions, as the author of this report found that the PICOCS analysis criteria was too specific to the military area, restricting the results that could be obtained.

After the application of these I/E criteria to each of the research areas, 174 records were excluded from RA1, and all records from RA2 were excluded.

While screening RA2, the author noticed that none of the articles even came close to fulfilling the inclusion criteria defined above, even when broadened.

While no research articles were found to adhere to these criteria, likely due to the technical nature of the question, there will be an effort to answer this question in the state of the art, by sampling information from other credible sources.

For this literature review, the author will assume that no research has been done in this specific area, and for the remaining stages of the process, will only be performing the PRISMA process on RA1.

2.1.6 Eligibility Phase

In the eligibility phase, the included articles that passed the initial screening phase will be more deeply analyzed beyond their abstracts and titles and compared to the I/E criteria to appraise their eligibility. A classification system was devised, with the main goal being to sort out articles that may help answer each research question in RA1.

Table 8 demonstrates the classification system devised to group the included records.

Table 8 – Classification of retrieved records from the Screening Phase.

Category	Purpose	Number of Records
Usability	Record provides information to answer RQ 3.	3
Viability of AR	Record provides information to answer RQ 1.	9
Irrelevant	Record does not help answer any RQ.	6

The excluded records were placed into this category for not being of any help to answering the research questions that have been posed by this study. Reasons for exclusion range from the following list, with the specific studies being referenced:

- The article was too specific to a program or methodology (case study) and provided no answers or help to any research question ([8], [9], [10]).
- Studies not focused on AR or Mixed Reality ([11]).
- Studies do not fit into the I/E criteria laid out for the Research Area ([12], [13]).

The rest of the inclusions fit into the I/E criteria of RA1.

2.1.7 SMR Conclusions and Report

In this section, the results from the systematic mapping review will be analyzed in a short report, and conclusions will be written on the findings of the literature review. A PRISMA flow diagram will be elaborated to present an easy to digest summary of the literature review that was conducted.

As RA2 did not pass the screening phase, no diagram will be shown here, but conclusions will be taken from the process.

Figure 1 shows the PRISMA flow diagram for the process that was carried out for RA1.

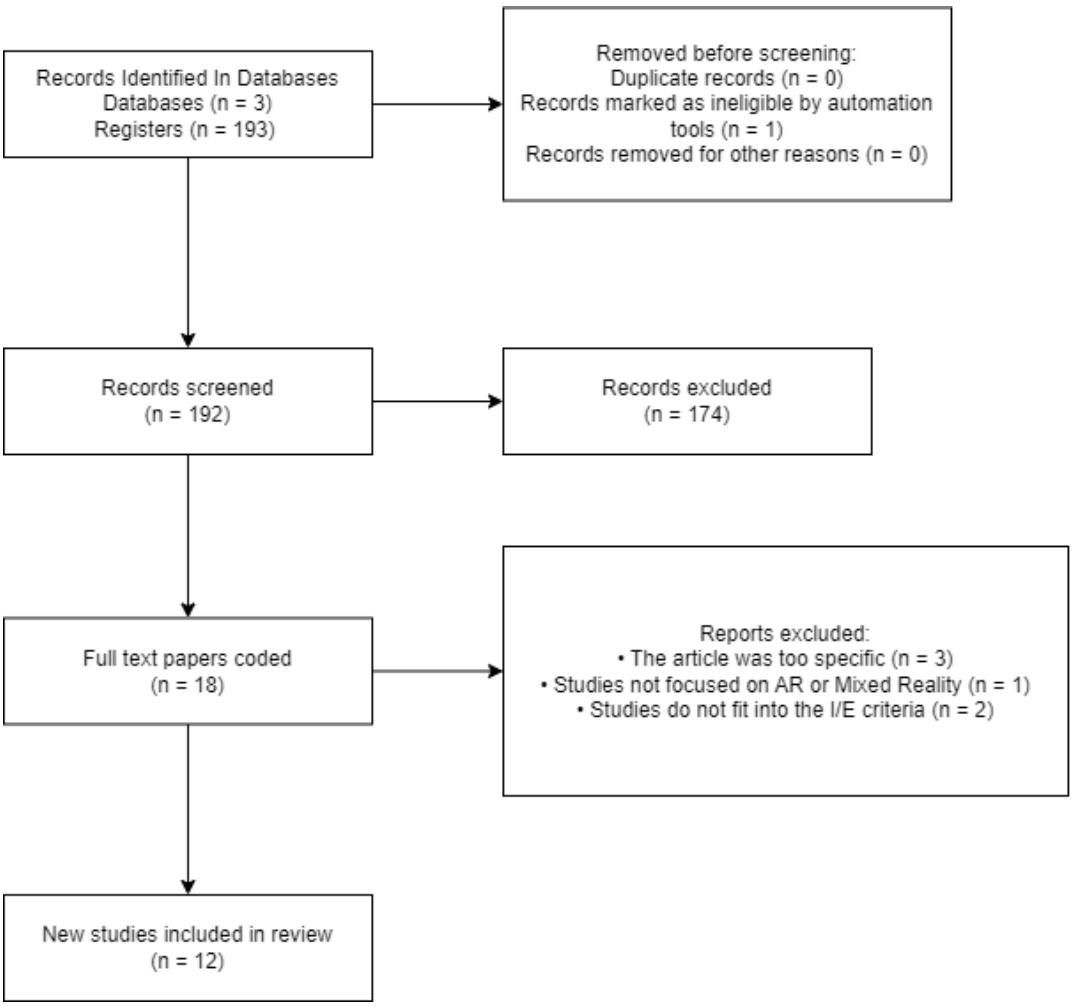


Figure 1 – The PRISMA flow diagram for the SMR of RA1.

For RA1, after analyzing the records retrieved from the databases by using the query, the author noticed that, even after the screening phase, most of these reports did not answer the two research questions that were posed at the beginning of the literature review.

They help paint a picture of the current state of Mixed Reality (MR) technology [14], how the usability can be improved by implementing certain systems [15], and how it can be used as a

tool in certain contexts [16], and also as a learning tool [17], while presenting some of the benefits and risks of using it for training [18].

There is one paper that fit the criteria better for RQ1 as it writes about using an AR head mounted system to enhance an already established live-training exercise by using a baseline (paper map) and then by using an AR application that was designed and developed to aid soldiers in a military context [19].

This study highlights the importance of utilizing AR technologies in the military field, by testing the effects and head mounted display has on the task completion of the assigned mission, and the platform's usability. From the study's results, the author of this work believes that AR platforms do have a place in the military world and can be utilized on and off the field.

While SISTRADE's application is not intended to be used in the same way as the study's application was, as it is an "off the field" application, the study makes a strong case for the continued development of AR prototypes, as they provide a more usable, user friendly experience, while allowing soldiers to do their jobs more effectively [19]. As mentioned before, these platforms can also help with learning new skills [17], so a soldier learning new tactics from a scenario created virtually by an instructor is reasonable to expect given current trends.

As for improving usability, the included papers present ways on how software engineers should approach the development of an AR platform designed for learning [20], analyze different ways of controlling a MR application through touchless means [21] and implement ways of reducing visual UI clutter and only show what is necessary based on context [15].

As these have no military context, it is hard answering RQ3 with only this information. Usability improvements on AR applications will have to be further researched beyond this literature review. More technical sources will be explored, as the question can also be seen from a technical point of view.

For RA2, as mentioned before, the literature review ended in the screening phase, as no articles were kept for further analysis. As mentioned previously, the author of this work assumes that no research has been done in this area by the scientific community, and will further investigate other sources, white papers and articles to answer RQ2 in the state-of-the-art section.

2.2 AR Viability for Military Training

Military training is indispensable for armed forces, as it allows them to act as a coordinated unit towards a common goal, as opposed to a single soldier being lost in the "fog of war". It is focused on discipline and making sure that your forces are better than your enemies' forces, as it makes a significant impact that thwarts even a numbers advantage.

To this end, most military training is done as a collective unit, rather than training soldiers one by one. This singular unit training is seen as an infrastructure cost, together with transport and accommodation and not relevant to the training of the unit.

Military training is done as a just-in-case action, whereas in other fields, such as normal training (gym, hobby) would be seen as a waste of time, or not worth it. Just-in-case training is defined as training for situations that are known about in anticipation of future needs [22]. Soldiers are trained to face conditions which they have never experienced before and hopefully won't ever have to experience. But it remains essential that training is done just-in-case, as it provides assurance to society that the forces that protect them are able to do so better than the ones knocking at their doorstep [23].

These three characteristics make military training quite unique and different from other kinds of personal training.

As such, this training can be conducted in a variety of ways, such as constructive simulations, live simulations and virtual simulations [24]. This trio of simulation that is used in the military, is known as the Live-Virtual-Constructive (LVC) taxonomy, and these types of simulation can be interoperable [25].

2.2.1 Live Simulations

Live simulations are defined as simulations where real operators carry out exercises by utilizing real equipment and simulating real training conditions in a training center or designated locations.

These simulations are often used to train soldier's shooting under pressure, or performing coordinated strikes with their battalion or regiment, but can be expanded to many classifications and groupings of soldiers [25].

The integrity of these simulations can be maintained by using laser systems designed to correctly identify which soldiers have been "downed" or "killed" in action, prompting the rest of the active simulated soldiers to react accordingly [26].

This is an example of how technology has been integrated into even the most basic of training for military operations, and how it has bettered the training of soldiers, by allowing it to be closer to real scenarios they would encounter.

Augmented Reality (AR) can augment these simulations further by providing a constantly updating Heads-up Display, voice commands and a regional map, for example. These implementations use a Head-Mounted Display (HMD) to present these elements into the user's field of view as seen in Figure 2 [19].

Figure 2 – Examples of military augmenting platform by the use of an HMD [19].

These AR platforms, when compared to a baseline status, have been shown to reduce workloads for the soldiers wearing them, a perceived sense of better usability and a perceived sense of better situational awareness [19].

2.2.2 Virtual Simulations

Virtual simulations are defined as real operators, such as tank operators and pilots, that can operate a virtualized version of an otherwise real system or tool. Examples of tools are fighter jets, tanks and other vehicles (Figure 3 depicts an F-35 training module).

Figure 3 – An example of a virtual simulation environment for training F-35 pilots [27].

This type of simulation is mostly used for training pilots, or gunners, ranging from solely training an individual soldier at a specific task (fighter pilot, tank operator), to coordinating maneuvers with others [28] while the simulator keeps track and aware of the simulation's state [25], [26].

These simulations often exist for consumption by civilians [29], although they are very simplified with regards to military procedures, accuracy of in-simulation models or data required for accurate simulation [26], as most recent air frames and flight data are highly classified.

Although these simulations specifically train individuals, the device is fully integrated into itself, making additional AR development into these systems a small extension or a redundancy of their already integrated capabilities.

2.2.3 Constructive Simulations

Constructive simulations are simulations where soldiers do not learn to handle, or handle actual real equipment, but are instead trained on combat tactics and eventualities that are guided by combat rules [24]. They are often utilized by experienced personnel in the military who coordinate and order other smaller collective units, such as division commanders or generals.

If these simulations were all connected to provide an as close to reality as possible training environment, the constructive simulation would be the “server” that coordinates the other simulations. It is possible to issue commands to live or virtual simulations, just like the commanding officer would send commands to the forces under them, however, just like in the real world, the commanding officer has no agency on the result or execution (of tasks) of the simulations or troops they are commanding [25].

If there is no connection with other simulations, these simulations can help a commanding officer become better at decision making by simulating everything that can happen during engagements, such as friendly and enemy units, how these interact with the terrain, and how air units, land units and naval units interact and influence the theatre of the simulation [30].

Known constructive simulation software includes JSAF [31], which is still in use by the united states military, and SISTRADE’s prototype platform.

The utilization of AR platforms in these types of simulations follow a perspective of usability and collaboration purposes, as many people could edit a scenario collaboratively, which can be used by military instructors to explain to trainees why certain decisions are right or wrong, or how forces can be used more effectively [32].

2.3 Platform Optimization

To be successful, a constructive simulation platform must have the software tools necessary to represent simulations digitally.

These simulations are often composed of many software tools, such as a map viewer, to place units in terrain, a visual interface where a user can create units, a scripting engine and language so the user can create complex movements and interactions between the different entities in a single instance and so build different training scenarios [33].

Given these software requirements, these applications must be performant for instructors and trainees to not be drawn away from the simulation, and to provide an immersive experience

for both types of users. Since these applications are mainly software applications, they can be optimized by using proven techniques that apply to most applications.

As such, this section of the State-of-the-Art focuses on ways to generally optimize systems such as APIs, 3D rendering engines, the loading of map tiling data into web rendering engines and the optimization of frontend frameworks. This section will focus on the technologies in use by the already developed application, namely the Node.js runtime and its accompanying DBMS, SQL Server, the Angular web development framework and the Babylon.js web 3D rendering engine.

2.3.1 Node.js

Node.js is defined as a "an asynchronous event-driven JavaScript runtime (…), designed to build scalable network applications"[34]. It has a non-locking design, so no process causes a complete deadlock of the application. It is also possible to leverage multi-threading, even when Node.js itself was designed without them in mind [34].

One of the benefits of working with Node.js is that it can easily be expanded with community libraries that add new functionality to the application without the need to develop it. One of these community libraries is Express.js, which allows developers to quickly create APIs [35].

Before starting to optimize an application built on Node.js, optimization problems must be identified, or else, developers run the risk of optimizing prematurely, without need. This can be done by using Node.js' built in profiler, which polls the stack of the runtime in intervals, allowing for the identification of bottlenecks anywhere in the system [36]. After profiling an application, and detecting a problem, some common fixes can be applied to the application in the key areas or known hot spots that are causing performance problems.

The root causes of performance issues in Node.js (and JavaScript applications in general) stem from inefficient API or language usage, where built-in JavaScript APIs implement the functionality to achieve the developer's goal, but the client (the developer in most cases) does not implement this straightforward functionality [37], because JavaScript is extremely flexible and allows the client to implement any functionality without any suggestions of a better solution. This is usually seen in string operations (seen in Figure 4), runtime checks of object properties or function invocations.

```
1  //Ineficcient API usage.
2  let str = 'Hello World!'
3
4  str.split("l").join("o");
5
6  str.replace("l", "o");
```

Figure 4 – An example of Inefficient API usage (line 4 can be replaced by line 6, code snippet adapted from [37]).

Two less notable but still prevalent problems are the inefficient reimplementation of something JavaScript has already implemented, and the recurrence on generic APIs, which are inefficient for the requirement being considered as seen in Figure 5.

```
8  //Generic API Inefficiency.
9  let arr = [1,2,3,4,5,6,7,8,9];
10 let n = 3;
11
12 arr.slice(n)[0]
13
14 arr[arr.length + n]
15
```

Figure 5 – In this case, the use of the API causes inefficiency, as slice does more operations than a simple access (code snippet adapted from [37]).

Besides the problems of API and language features usage by developers, there are other causes of inefficiency that can serve as hotspots for inefficient implementations, such as inefficient iterations of collections of data [38], the repeated execution of a particular operation, such as events creating an (expensive) regular expression each time they are called, the unnecessary copying of data into different objects, and repeated checks of the same conditions that can be avoided [37].

By looking out for these common issues with JavaScript applications, it is possible to identify the most likely culprits of inefficiency in a program and apply the necessary fixes to each case, with the accompanying use of the Node.js profiler.

2.3.2 Angular

Angular is a framework for developing Single-Page applications built on TypeScript. It allows developers to build websites with reusable components, and it is centered on creating applications that follow an SPA pattern [39]. This means that it is the best choice for highly interactive applications, such as the SISTRADE platform.

Most of the optimizations which apply to Node.js, can be applied to Angular, as it is based on TypeScript, a superset of JavaScript, so most optimization that can be implemented in terms of application performance, can be implemented in Angular components as well.

The user's experience is a priority while developing an Angular application as it is "client-facing". As such, optimizations to the rendering of the web page should be performed if needed (by the use of profiling tools such as the Angular Profiler [40], Lighthouse [41] and Chrome DevTools) to improve user experience while using the web application.

The most important metrics to pay attention to while profiling an Angular application are First Contentful Paint (FCP), Total Blocking Time (TBT), Interaction to Next Paint (INP), Largest Contentful Paint (LCP), and the page's framerate and smoothness [42], [43]. These metrics are explained in the next paragraphs.

FCP measures the time it takes for the web site to render a new page of content. If users see content appearing quickly, they are more engaged. This metric can be improved by ensuring initial content is rendered quickly, by optimizing the loading of critical data, such as user data from an API, reducing script instructions that block website rendering, and by optimizing CSS files [42].

Time to Interactive (TTI) has been retired as a metric by Google, as it proved to be too sensitive of a metric to track [44]. It has been replaced by TBT, INP, and LCP metrics.

TBT is a metric that "measures the amount of time after the FCP where the main thread of the application was blocked long enough to prevent input responsiveness"[45].

INP is a metric that measures a website's responsiveness to user interactions by measuring the time it takes for new content to appear after a user has interacted with a certain prompt [46]. As such, this metric is measured in milliseconds. The scale can be seen in Figure 6.

Figure 6 – A scale of the INP metric [46].

Optimizing TBT is better done by optimizing INP as a whole by avoiding unnecessary JavaScript, or unused code, breaking up long tasks (an operation that takes longer than 50ms) and avoiding large rendering updates, by keeping template documents small and reorganizing the template's reads and writes [47].

LCP is a metric that measures the load time of the biggest element on a web page at the time of the user's visit. A good score for this metric is 2.5 seconds or less [48].

Figure 7 – A scale of the LCP metric [48].

Optimization of the LCP metric requires the profiling and optimization of the entire load process of the website, to identify improvement areas and gradually make improvements to the identified LCP element to improve the element's loading time and by consequence improve LCP [49].

A consistent framerate of 60 frames per second is crucial for the perceived smoothness of the application. This can be achieved by the use of smooth CSS animations, and limiting expensive operations by the use of debouncing [42], [50].

These metrics can all be tracked from the profiling tools mentioned previously, and while performing a benchmark, it should be ensured that the environment is the same so results stay accurate.

To further provide optimization solutions, some of the several core parts to an Angular application will be explained. An Angular application consists of components, modules and services.

Components are the building blocks of any Angular application. They consist of a TypeScript file, which coordinates the HTML template which will render visually in the browser with the help of a CSS style sheet. This allows the division of the application into many components, creating a tree [51].

Modules are blocks of code which are dedicated to a certain way of working with Angular or implementing certain functionality. This is used for the entry point of the application, and many other modules provide developers with essential functionality, such as the *HttpClientModule* and the *FormsModule* [52]. These modules provide developers with http client functionality and form functionality, respectively.

Services are the last piece of the puzzle for an angular application as they implement functionality that must be shared across components, or that communicates with external entities, such as APIs [53].

Another core component of an Angular web application, which may cause performance bottlenecks is called "change detection", which is the process by which Angular detects changes in the DOM (which has specific data) that is being rendered and shown to the user. Change detection allows Angular to seamlessly change and update DOMs while they are in the User's view, leading to a more interactive experience [54].

Performance of change detection can be improved by recurring to the change of its strategy on a specific component to one of two values: “Default” and “OnPush” [55].

The default value triggers Angular to check all components in the application when the change detection mechanism detects a change. This is inefficient for large applications with many components, as it forces Angular to reload every component of the application.

Setting this the value to “OnPush” (see Figure 8) for a component ensures that change detection only executes when an application event is triggered, or the component’s input properties change. It is also possible to trigger change detection manually by recurring to the “ChangeDetectorRef” service present in Angular and injecting it into the desired component.

```
@Component({
  selector: 'app-root',
  template: `Number of ticks: {{ numberOfTicks }}`,
  changeDetection: ChangeDetectionStrategy.OnPush,
  standalone: false,
})
```

Figure 8 – Change detection setting in an Angular component definition [56].

These techniques allow developers to optimize the change detection mechanism of Angular to avoid re-rendering components that are expensive to render [42].

Memory leaks are defined as a program error that causes the program to not free utilized memory after it is no longer relevant to that program. This can cause performance issues in the long term, as other applications are not able to make use of the memory being “held hostage” by the memory leaking application [57]. They are a common issue in Angular, often slowing down a website overtime.

“Observables” are used by Angular for the tracking of asynchronous operations, usually involving services and an external API. They work as a subscription-based service, which, if not unsubscribed from in a component, can cause memory leaks.

Another cause of memory leaks is detached template elements. When an Angular component is destroyed, its template and associated elements should also be destroyed. However, these associated elements can stay in memory if references to them exist, with no way of removing them from memory when appropriate.

The excessive use of Angular template binding can also be the cause of performance issues. One way binding is the most efficient, and two-way binding component elements and HTML properties should be done only when necessary.

These Angular specific optimization techniques, in conjunction with the optimization techniques for rendering that were presented above would allow the development team to properly optimize the simulation platform.

2.3.3 Babylon.js

Babylon.js is an open-source web rendering engine built to allow web applications to render and display 3D models, animations and environments [58]. This could be used to provide a 3D representation of the simulation, providing the user with a 3D visualization of the simulation in space, while rendering the map in a 2D plane and models that represent units on the map.

A quick way of optimizing Babylon for VR or AR applications comes in the form of Multiview rendering, which allows the Babylon renderer to render multiple views (for example, both eyes when the user is utilizing VR or AR glasses) which makes rendering "1.5 to 2.0 times faster" [59].

3D models and their complexity can bring a host of performance problems in computing. As such, there is a need for premature optimization in the inclusion of models in a web application, as they are not traditionally designed with 3D rendering in mind. In the next paragraphs, a summary of model optimization in Babylon.js is presented.

The first optimization to do when working with 3D models is to "decimate" the model. This can be done with blender [60], and allows the model to maintain its shape while reducing its total vertex count. This is useful because when the model is loaded, less vertex data will be loaded, improving first load performance, as Babylon.js uses less computational power to load the model.

If the model loses too much quality and is then unfit for purpose after one or many decimations, these additional decimated meshes can be used to build a Level of Detail (LOD) scale, as seen in Figure 9, which Babylon.js natively supports [61], improving performance and allow more models to be loaded into a single scene, without losing detail [62].

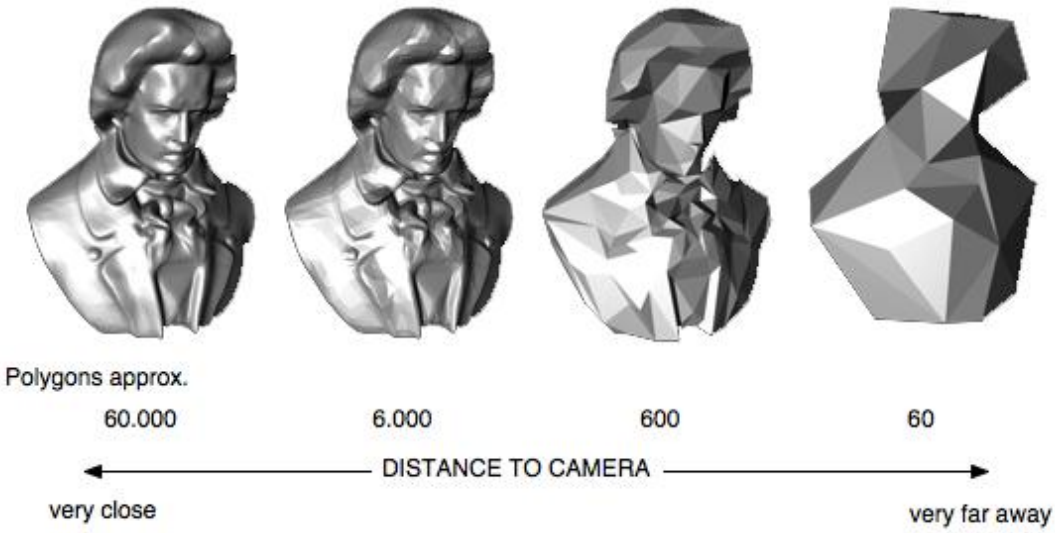


Figure 9 – LOD demonstration [63].

Depending on the starting position of the camera and its distance to the loaded models, this can also be a factor in improving the initial load times of the models, as the camera is not always close enough to render a model at maximum detail [64].

To have a successful constructive simulation, a map needs to be rendered in the web environment, that is synchronous with the real world outside of the simulation. This is so that instructors can more accurately and effectively create scenarios that a trainee would encounter in their day-to-day operations, which follow the rules of real terrain.

Map rendering in the form of a web application can be done by utilizing a map tiling server, which takes geo spatial data, such as coordinates and zoom level, as input, and returns map images that can be shown in the browser. The diagram shown in Figure 10 exemplifies this process.

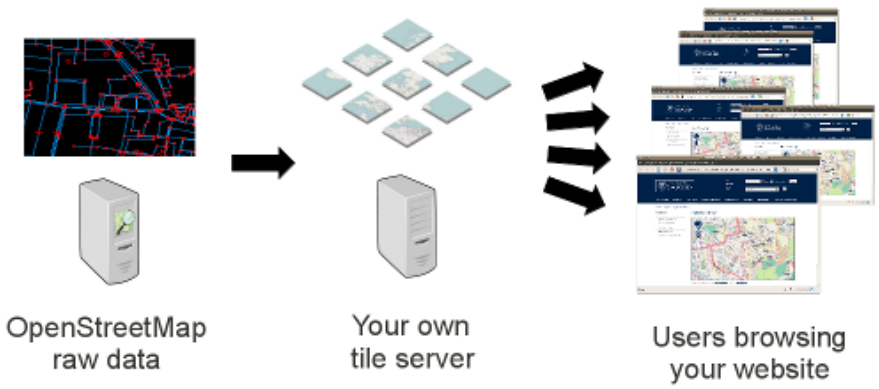


Figure 10 – Simplified map tiling architecture [65].

OpenStreetMap (OSM) is an open-source project which distributes geographical data for free, including images, and all the underlying map data for each map tile [66].

OSM can be used for free, including map tile generation and fetching via a client that wants to display the image. Although the free usage of this tiling service is very limited by OSM themselves [67], there are alternatives to using the tile server provided by OSM, and these are revisited later (in section **3.4.4 Miscellaneous Improvements**).

Rendering map data and displaying it over a flat surface in Babylon.js can prove to be slow, especially due to the delay in fetching tiles from the server, depending on the size of the area the user chose to render. If the area is too big, it must be fitted into the screen, which can cause a massive wait for the user as the map slowly loads in. This can be reduced by implementing a form of "object culling" into the map rendering algorithm.

In general, Culling techniques are defined in 3D rendering as algorithms that can detect if a certain object or mesh is not visible by a certain camera, and simply not render it, saving resources for things the camera can actively see [68]. Babylon.js allows the use of culling techniques for more efficient rendering [69].

Culling algorithms can be applied to map rendering by conditioning the rasterization of the map tiles to the camera's view frustum which would allow the map to render procedurally, as the user pans and views more tiles of the map. To reduce initial load times, the zoom can be set to a reasonable level and, when the user adjusts it, the rest of the map is loaded procedurally.

2.4 Usability Improvements

Usability in software development is defined as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use", by the International Organization for Standardization (ISO) 9241-11 standard [70], [71]. This definition can be applied to Augmented Reality (AR) by considering the ways in which the user interacts with the AR system to accomplish their goals of using the software.

This can be enhanced in a variety of ways, for example, by providing certain controllers or other human interface devices which augment the software's usability [21], or implementing gesture based navigation, which in certain circumstances can be more usable than other ways of interacting with the AR system [72]. Other ways include utilizing a contextual awareness system, which contextually provides information to the user, and positions this information in a non-obstructive way in the user's field of vision, eliminating the need for elaborate gestures for navigation [15].

In an application intended to train other people, it is often nice to have interactions with the trainees, especially those who learn by example. Collaboration is a crucial part of training, and as such, makes a good constructive simulation better, when feedback is always visible on the screen. It also allows long-distance training, reducing the costs of travelling and getting everyone in the same room, which in turn makes the application more accessible.

2.4.1 Y.js for collaboration

Y.js is an open-source library that allows developers to implement applications such as text-editors with the functionality of real-time editing [73]. The awareness system allows developers to implement functionality that helps users to effectively work together by allowing users to see each other's cursors, additions, and deletions, and whether a certain document is being edited by another user [74].

The library does this by implementing the conflict-free replicated datatype (CRDT) algorithm to provide editing capabilities that are synchronized between all clients without any merge conflicts [75].

For this implementation, the selection of a network or connection provider is necessary. Y.js offers many implementations, but one that allows us to implement a client-server architecture (to keep the truth of the document central) and the sharing of awareness data (so that users can see each other) is "y-WebSocket". The creation and set up of this server can be done on the runtime such as Node.js, allowing in-editing documents to synchronize to clients by the use of the WebSocket server[76].

Developers can emulate a user entering a shared editing instance using the "Y.doc ()" function [77]. This instantiation also allows developers to utilize shared datatypes, such as arrays and maps , as seen in Figure 11 [78], [79].

```
import * as Y from 'yjs'

// Yjs documents are collections of
// shared objects that sync automatically.
const ydoc = new Y.Doc()
// Define a shared Y.Map instance
const ymap = ydoc.getMap()
ymap.set('keyA', 'valueA')

// Create another Yjs document (simulating a remote user)
// and create some conflicting changes
const ydocRemote = new Y.Doc()
const ymapRemote = ydocRemote.getMap()
ymapRemote.set('keyB', 'valueB')

// Merge changes from remote
const update = Y.encodeStateAsUpdate(ydocRemote)
Y.applyUpdate(ydoc, update)

// Observe that the changes have merged
console.log(ymap.toJSON()) // => { keyA: 'valueA', keyB: 'valueB' }
```

Figure 11 – Basic usage of the Y.js library [78].

Awareness information sharing is the hallmark feature of collaborative applications. While sharing no awareness information is also possible, awareness allows the users of an application to collaborate through real-time editing, by helping each other [80].

Awareness information is shared through Y.js through the network provider as a JSON object that can have any fields the developer wants. This can be used to track any variable about the shared user in a session: mouse cursor position, caret position, username, and user color [80].

When any variable in the user's JSON object changes, an event is "fired" to the Y.js awareness CRDT implemented by the network provider to synchronize these changes with other connected users and apply the changes on their end [81].

2.4.2 Software Usability

Software usability follows certain principles that have stood the test of time throughout UI/UX generations. This allows software to feel familiar and makes it easy to use for any potential users of said software [82]. But augmented reality software suffers from the problem of not having concrete guidelines for developers to follow, as current drafts only cover limited aspects of UI design for these areas [83]. As such, this work focuses on the exploration and research of proven usability standards and guidelines for software, and what their inclusion would do for the user.

Many researchers and design experts have tried to standardize the process of usability design in software and physical products, and because of this, there are many standards, and principles that can be applied at various points of the design process, when it comes to developing a usable program.

2.4.3 Ben Shneiderman's Golden Rules

One of the most notable set of rules for usability design are the eight golden rules proposed by Ben Shneiderman [84], which can be applied in the design phase of a certain product. The next paragraphs go into detail for each rule.

Strive for consistency. Sequences of actions should be consistent when required by similar situations, and prompts, menus and other screens should use consistent text, colour and capitalization. Exceptions should be understood organically by the user and limited.

Seek universal usability. You should design your application with the needs of diverse users in mind. Facilitate the transformation of content for experienced and novice users, as adding features for each type of user enriches the perceived quality of software.

Offer useful feedback. Every action should have a reaction. This reaction should be feedback, tuned to the frequency and severity of the action.

Design dialogs to yield closure. Actions should follow a sequence of "beginning, middle and end." There should be a logical conclusion and feedback for finishing the sequence.

Prevent errors. Design interfaces that do not allow the user to make major mistakes, conscious or unconsciously. Errors should be met with feedback for state or information recovery, focusing only on the erroneous parts.

Permit the easy reversal of actions. Actions should be reversible, as this improves the user's willingness to experiment.

Keep users in control. Experienced users want control, not frequent changes to actions they are already familiar with. Data should be easy to obtain and transform into something useful.

Reduce short-term memory load. Reduce instances where a user must remember information from other displays to use it effectively.

These rules can be further refined and interpreted for different environments and applications, and should be utilized primarily at the designing phase of the user interface [84], [85], [86].

2.4.4 Jakob Nielsen Heuristics

In line with the work developed by Ben Shneiderman, in 1994, Jakob Nielsen has proposed a set of principles that should be utilized for the heuristic evaluation of user interfaces [88]. The rules are listed in Figure 12.

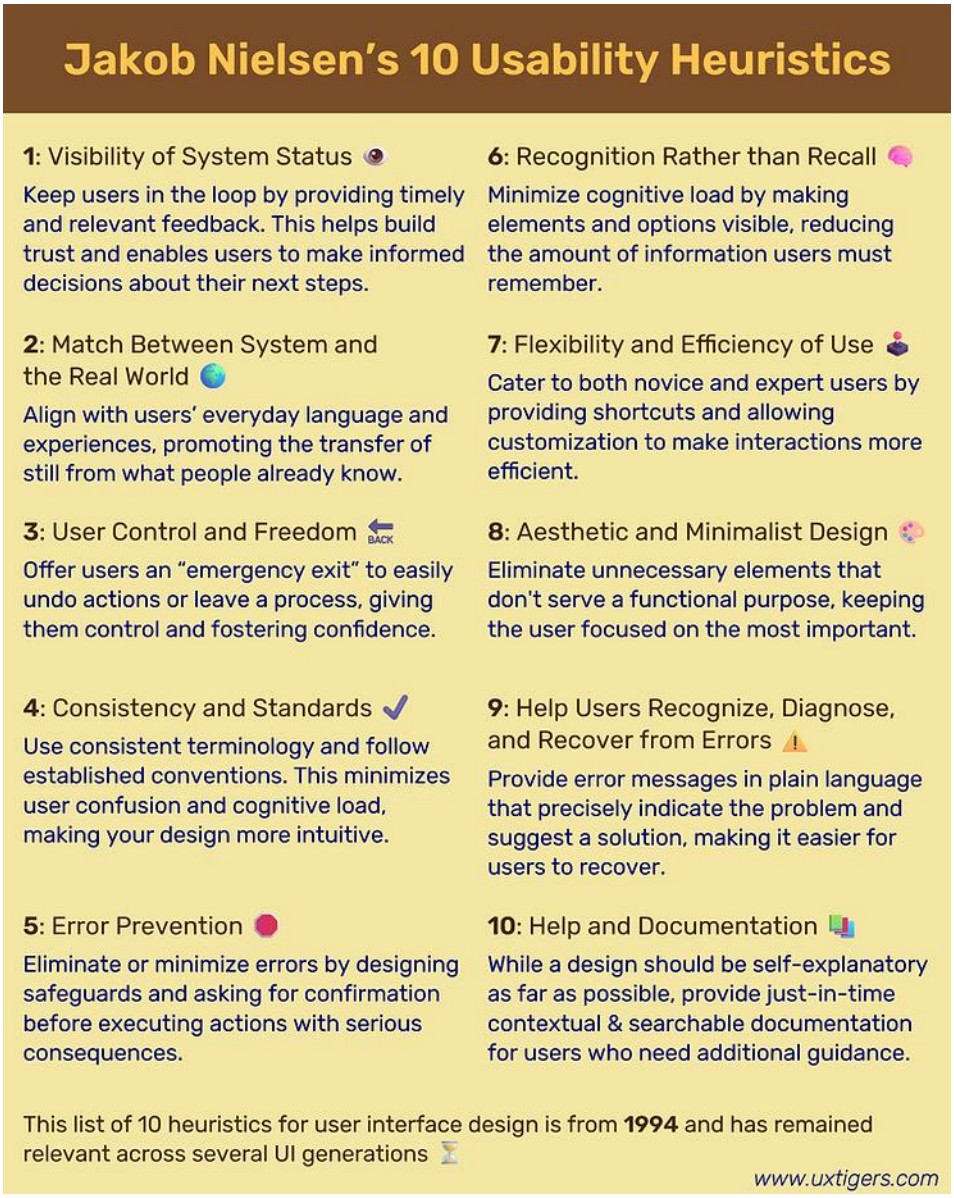


Figure 12 – Jakob Nielsen's usability heuristics [88].

In fact, most of the usability heuristics overlap with the rules proposed by Ben Shneiderman, except heuristic 2, 8 and 10. These exceptional heuristics which can prove useful for further improving an application's usability by providing users with more documentation about the application, streamlining the application for maximum functionality, and matching the system to the real world to increase familiarity [89].

The overlapping heuristics suggest that these are axioms that should be followed when implementing anything that is user-facing, for maximum familiarity and in turn, usability.

3 Analysis and Design

This chapter aims to present the application that has been previously developed, contextualizing the reader of the baseline used for the elicitation of requirements and improvements conducted in this dissertation.

To achieve this, this chapter is composed of four sections, listed in the paragraph below.

The first section touches upon the application's structure, domain and architecture, as analyzed by the author of this document. The second section lists the already implemented requirements by the application, and which of these requirements have been successfully implemented. The third section identifies and presents the problems with the application. Lastly, some suggestions for improvement on the identified problems of the application are proposed on the fourth section, as a transition point to the Chapter 4 Build and Development.

The application that has been developed by SISTRADE was implemented in the already mentioned technologies, Node.js and Microsoft SQL Server for the backend of the application, and Angular and Babylon.js for the frontend. Currently, the application has implemented most of its functionality, however, the developed prototype still needs to be polished and refined to be brought to an acceptable level of quality.

3.1 Problem Domain and Design

In this section, the architectural design and domain model are demonstrated by an analysis of the current application.

3.1.1 Domain Model

The system's domain is quite complex, being the prototype of a military simulation platform, and as such, the domain analysis by the author of this dissertation tries to explain each domain concept as clearly as possible, by analyzing its purpose in the application.

There are five groups in the application domain:

The Assets Group. The assets group represents every emplacement, unit, vehicle or operational equipment that can be present in a certain scenario. An entity can have various types, ranging from Human troops to Aerial vehicles. This group is simplified for the reader's convenience, as the relations are much more complex than what can be represented in the Domain Model. The assets group can be seen in Figure 13.

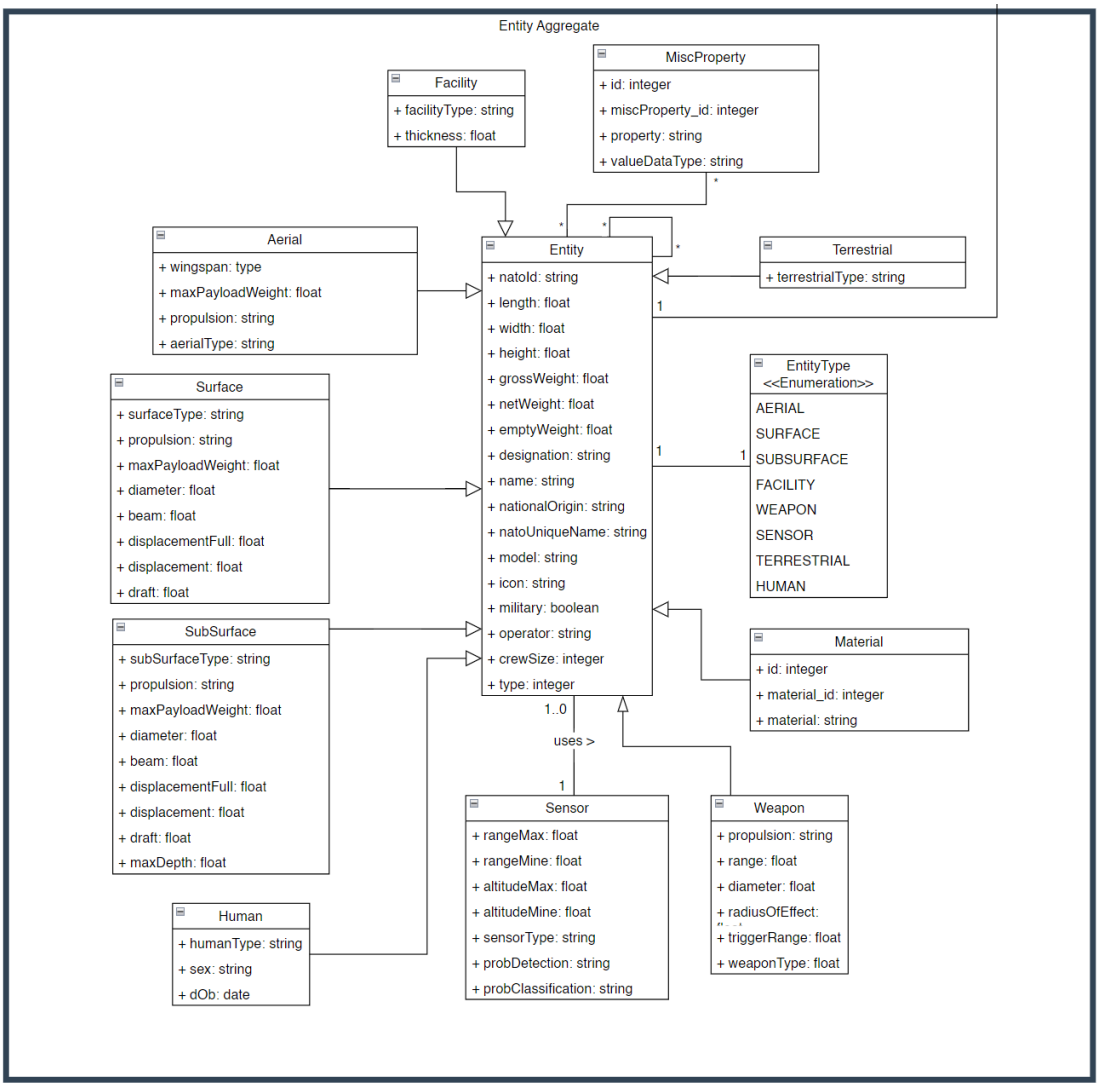


Figure 13 – Simplified entity group.

The Map Group. This group contains every entity that represents a unit on a map. The MapDomain entity defines the boundaries of the scenario map, while the Tile and MediumType entities define the location and properties of the individual map tiles. These entities can be seen in Figure 14.

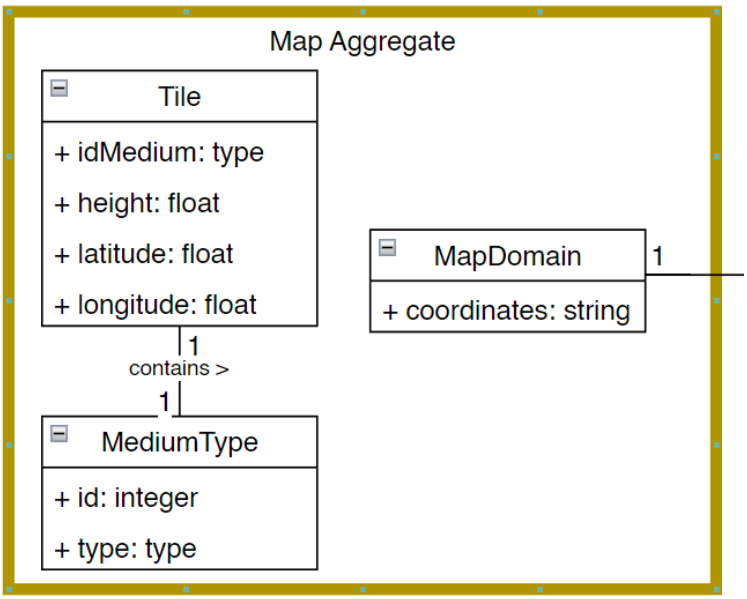


Figure 14 – The map group.

The Scenario Group. The scenario group defines a scenario in which teams can interact and place entities. The entities placed can be scripted to make certain actions or movements with or against teams, shown in the model depicted in Figure 15.

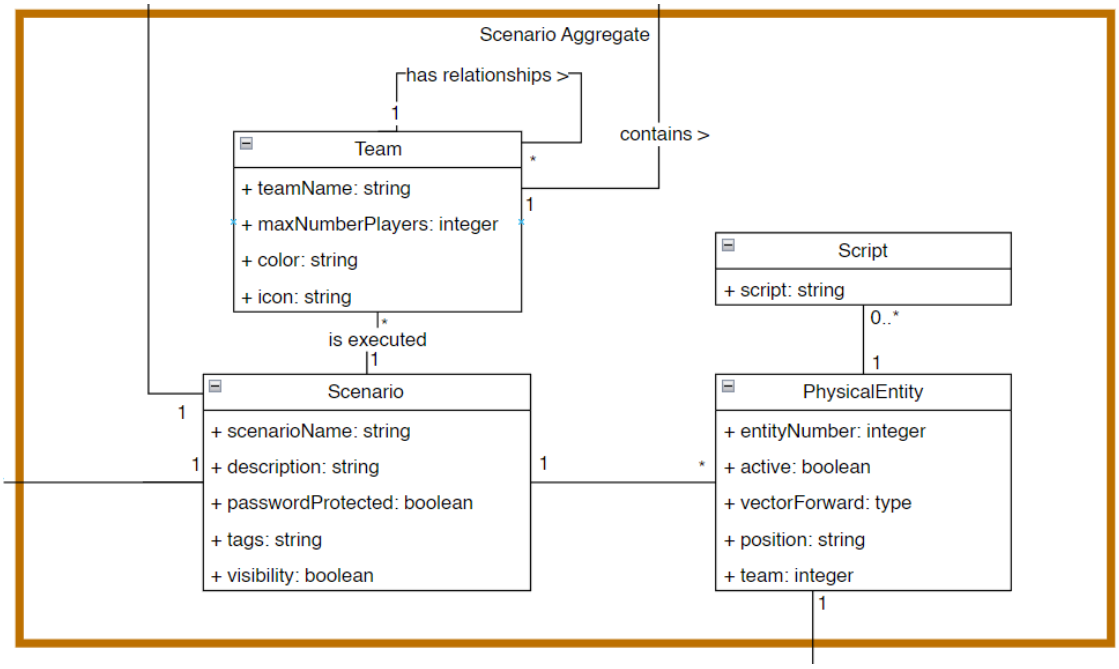


Figure 15 – The scenario group.

The Session Group. Sessions are carried out by players in a team and support collaborative messaging between players. They also hold information about what roles the players are to have during the simulation's execution. These attributes can be seen in Figure 16.

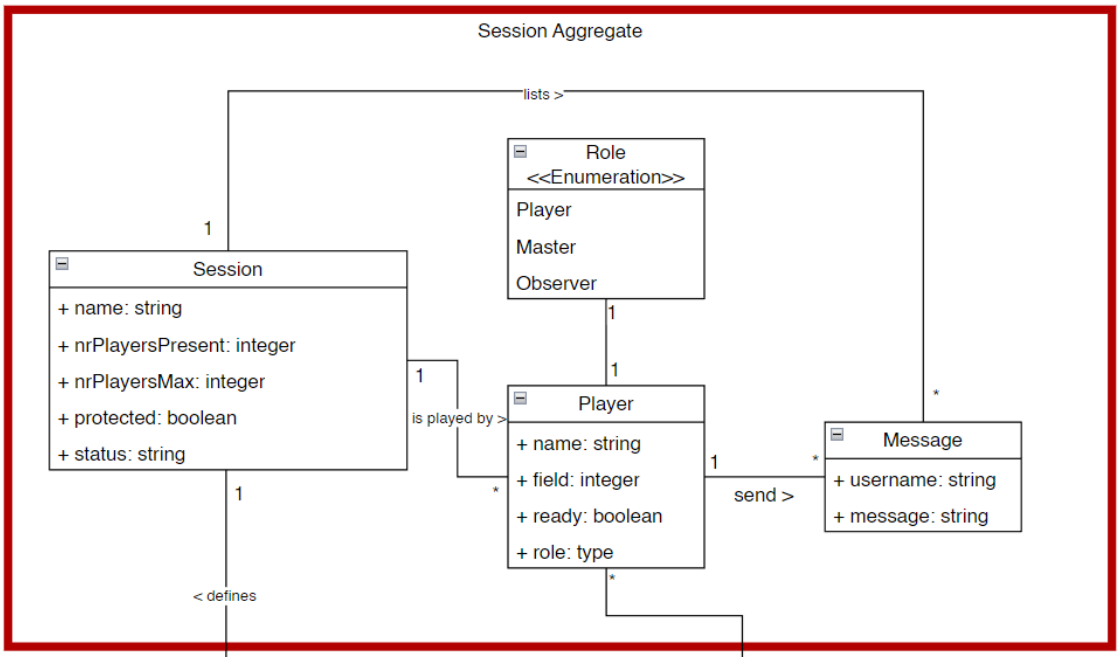


Figure 16 – The session group.

The Users Group. The final group, users, defines the users of the application, along with their institutions, which represent real organizations. This group has the purpose of tracking which user from which organization did any changes to the application. Figure 17 depicts this domain.

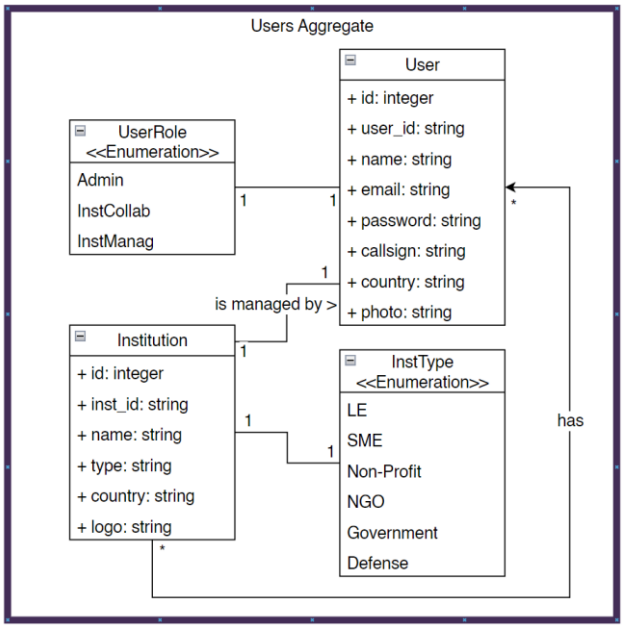


Figure 17 – The users group.

The full Domain Model can be consulted in Annex II – Analysis Annexes.

3.1.2 Architecture

The current system is composed by two separate runtimes (containers) that each have their responsibilities.

One of these containers is the simulation game engine, which has the responsibility of creating various game objects and maintaining the simulation state across players, as well as allowing clients to create new assets, scenarios and teams of players. The created objects are then stored in the database for persistence across sessions. The game engine also manages the collaboration feature, acting as a WebSocket server for collaborative communication requirements.

The simulation visual interface is the second container, and its job is to allow the user to utilize the game engine's functions, especially when it comes to the creation of Scenarios and Assets, while allowing the users to collaborate in the insertion of assets and the creation of scenarios. This container makes heavy use of the simulation game engine API to create and persist various objects and provide the various functionalities of the system to the user interacting with the application.

The visual interface container also utilizes the OSM (OpenStreetMap) tiling API, for the rendering of the world map, when building, editing or playing a scenario.

A diagram of the current system architecture is shown in Figure 18.

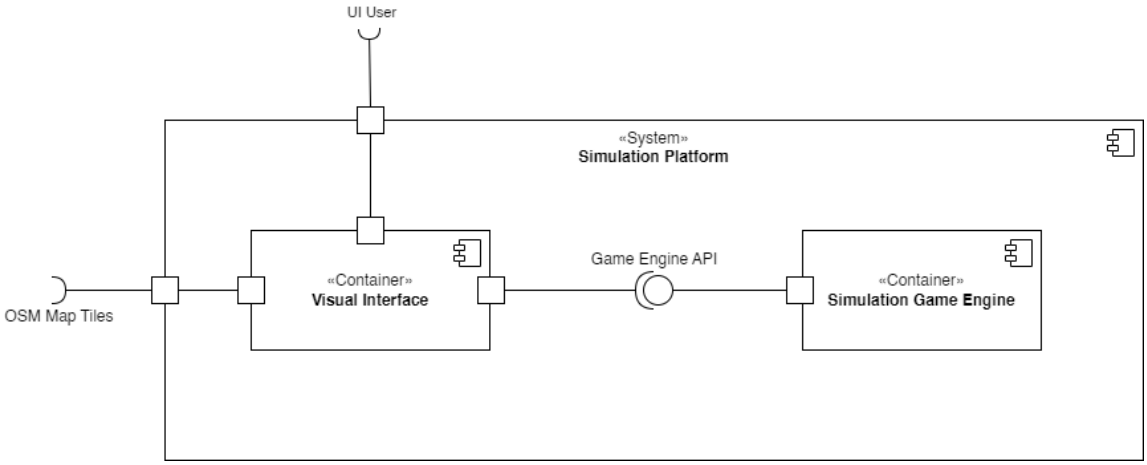


Figure 18 – The current system architecture.

3.2 Implemented Requirements

As mentioned in the previous chapter, the visual interface container was developed by utilizing the Angular framework, along with the use of Babylon.js for augmented reality and scenario building, while the simulation game engine was developed in Node.js, supported by a MSSQL database. With these technologies, the main requirements for the application were implemented by previous developments carried out by SISTRADE. The objectives of previous developments are listed, along with a visual demonstration of how they were implemented in the platform before the developments carried out in this dissertation.

The objectives for previous development cycles that were successfully implemented into the application were as follows:

- Allow a User to create and edit simulation entities, which would represent real life units.
- Implement scripting of entities.
- Allow the creation, editing and execution of scenarios.
- Develop a user interface that allows the user to carry out all functions listed above.
- Allow collaborative editing in the UI between users that are logged into the application.
- Implementation of the AR component for better visualization of a scenario and their execution.

These objectives are further represented by a simplified use case diagram in Figure 19.

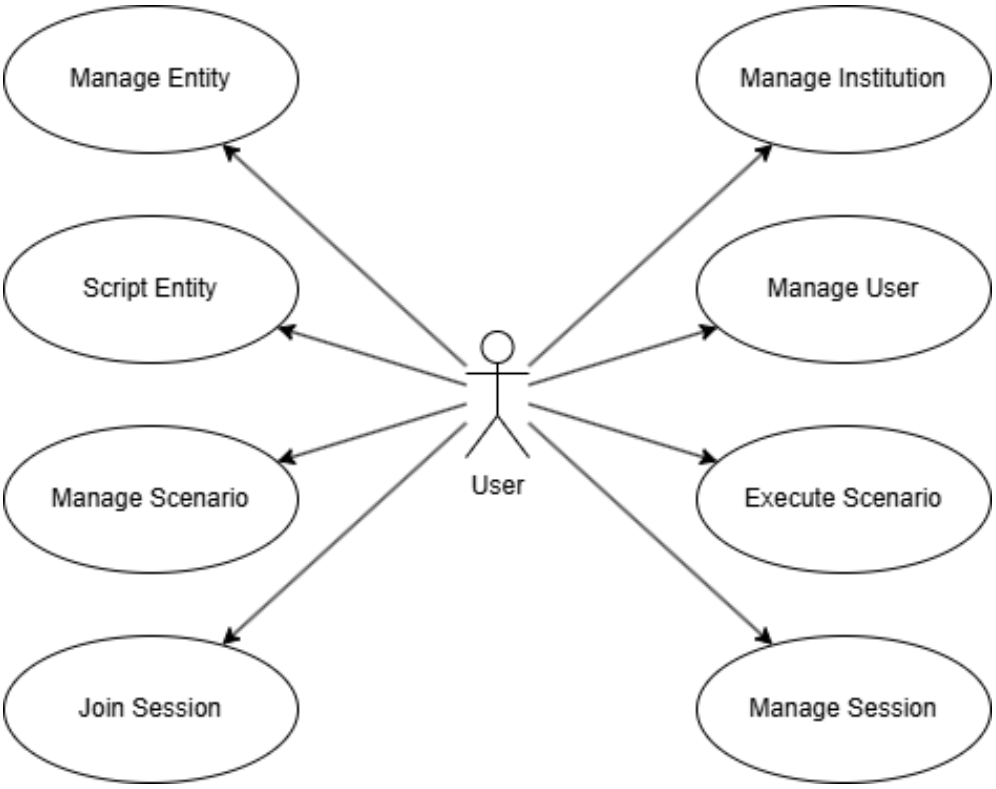


Figure 19 – Previous use cases of the application (simplified).

The implementation of these objectives is demonstrated from various functionalities of the application. Figure 20 demonstrates the first page a user sees after logging into the application.

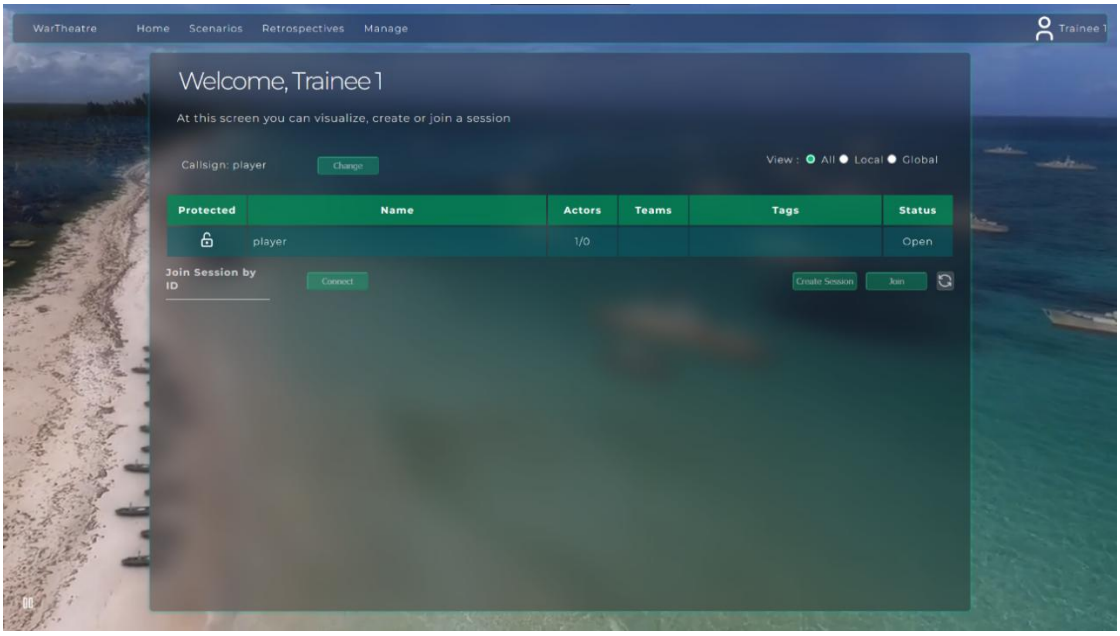


Figure 20 – The session creation screen, where a user can create or join sessions.

The first functionality to cover is the most important one, the creation of sessions and the ability to play out scenarios with different teams and factions. As seen in Figure 20, the user can create a session or join by clicking on any sessions opened by other players and utilizing the “Join” button.

For the sessions to be playable, there needs to be scenarios created in the application. The next relevant feature is the scenario creation feature, which allows users to map out a section of the world in which they wish to carry out their scenario, followed by filling out the form for the scenario details and the creation of the teams that would operate in that scenario, along with the relationships between the different teams. Part of this functionality can be seen in Figure 21.

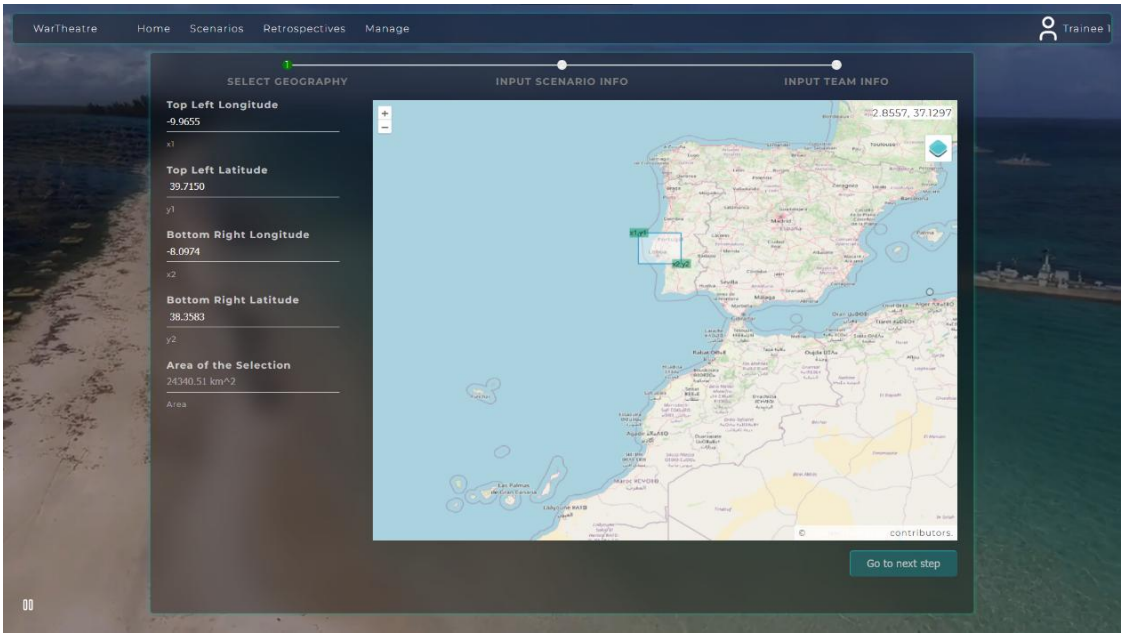


Figure 21 – The selection of an operations area as part of Scenario creation.

After the scenario is created, a user can edit it by placing entities down on the map. Entity creation is done by accessing the “Manage” tab of the application, where the user can manage Users, Institutions or Assets (Entities). In asset management, a user can create and edit already existing entities that can be used in scenarios. It is also possible to upload a 3D model to be used in the simulation execution as a rendering object. This feature can be seen in Figure 22.

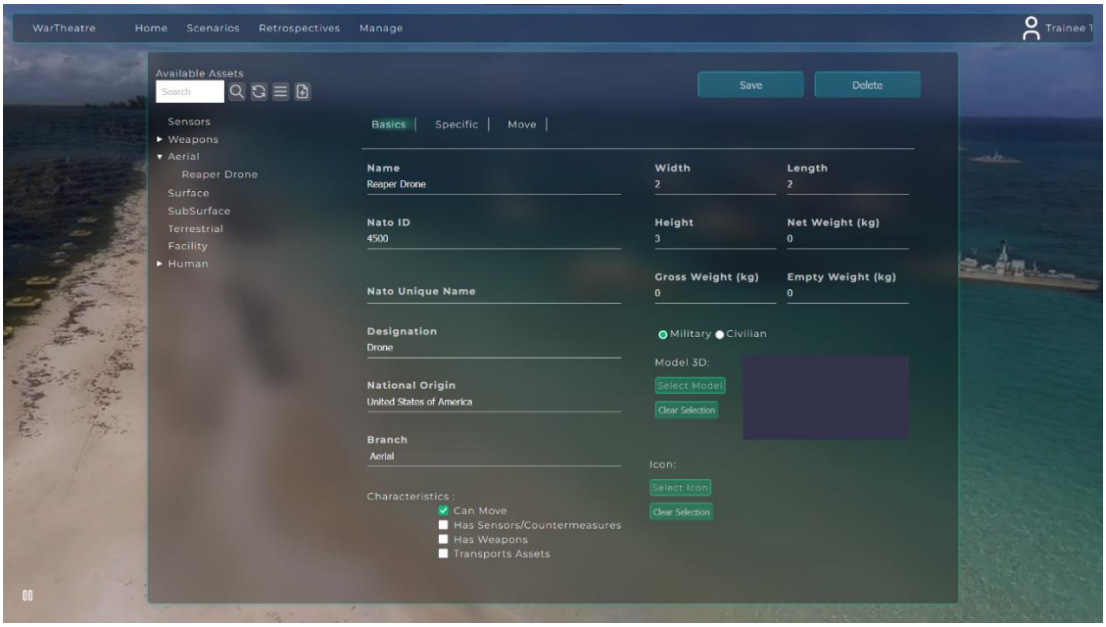


Figure 22 – Asset management page.

With the assets created, the user can now go edit the created scenario and place units by using the edit scenario option, seen in Figure 23. This option can also be utilized in AR, provided the user has the hardware to utilize the feature, seen in Figure 24.

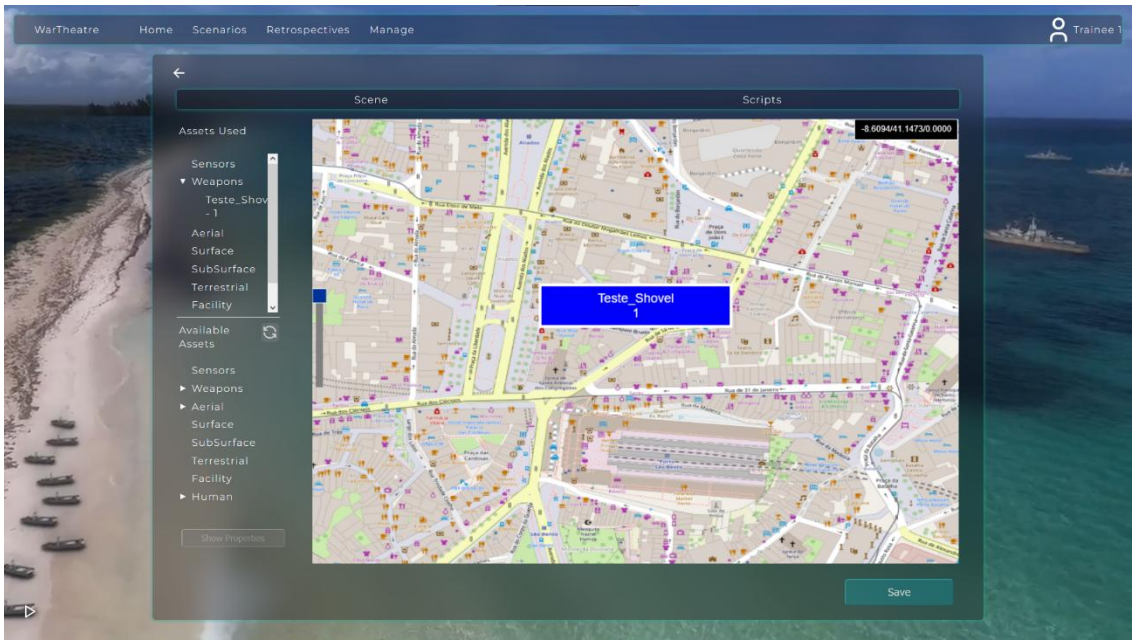


Figure 23 – The edit scenario page, where a user can place down entities after creating them.

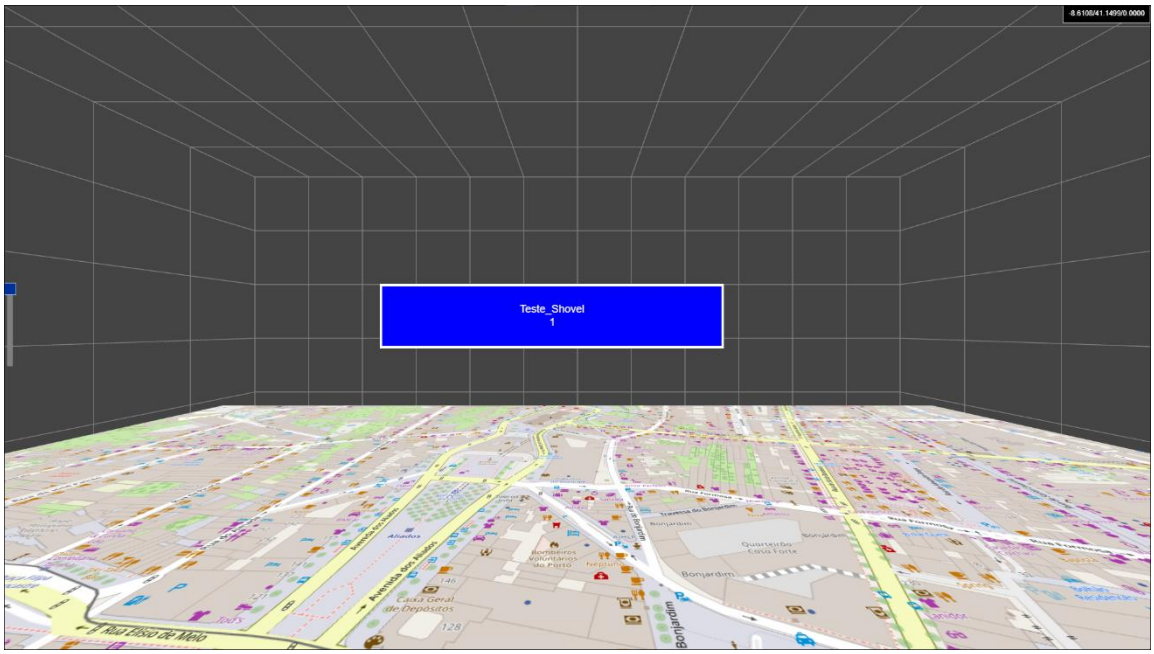


Figure 24 – An AR implementation of the edit scenario page.

3.3 Problems

As exposed in the introductory chapter of this dissertation, the main problems with the application lie in the usability of the platform, especially its AR component, and the usability of the collaborative features implemented in previous versions of the software.

The performance problems of a very specific part of the application must also be addressed to make the application perform to what is expected by the user. This section places a pin on these exact problems: where they are in the current implementation of the application, why they are problems, and what their cause is.

3.3.1 Usability Problems

The most glaring usability problems come in the form of the collaboration feature that was implemented, and the visualization of units in the edition of scenarios in AR or in the regular application.

The collaboration feature was implemented in screens such as the Asset Management screen, and while the feature is functional, there is no warning to any user of the existence of a third-party editor, editing the same asset as the user. This highly impacts the usability of the system, because without the awareness of another user, they cannot collaborate in the editing process, and, at worst, can cause the loss of data by mistake. In Figure 25, we can see that a user (on the right) is editing a specific field in the form, which in a collaborative application should trigger feedback to the user in the same form. However, no feedback is provided.

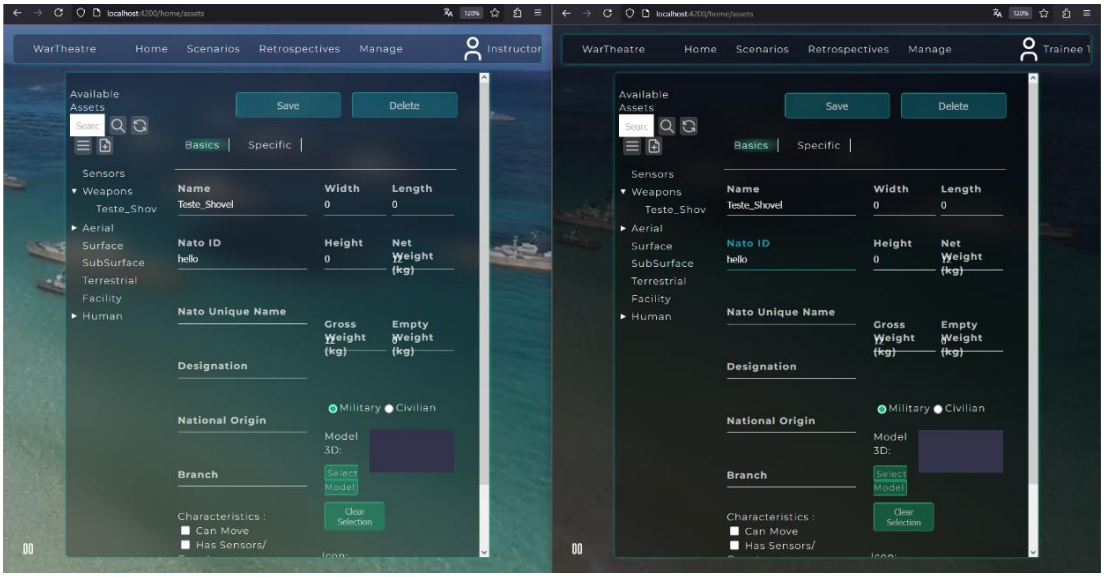


Figure 25 –Users editing a collaborative form.

This implementation also has some caveats, such as not prioritizing the user that first made edits to the form:

1. User A enters the form page, pulls data from the database and makes edits.
2. User B enters the form page, connects to the collaborative WebSocket and pulls data from the database.
3. The collaborative WebSocket synchronizes the data to the most recent (User B's database data), which erases all the data inserted by User A.

This can cause the form to reset back to its already persisted (in database) values, erasing the data the first user was inserting.

Other usability problems stem from the edition of scenarios and the view utilized for this purpose.

Users add entities to a scenario by double clicking on them from the available assets list on the left of the screen. Doing this allows users to then move entities on the map, as shown previously in Figure 23.

The entity box identifying the entity only appears on hover, and if the entity has no 3D model set, no model is used (the unit is completely invisible to the user), leading to confusion about where the entity was placed before. This forces the user to move their mouse around, waiting for the box to reappear. The user can upload 3D models to be used by the entity in asset management (Figure 22), but it is unrealistic to assume every entity can be represented by a custom-made 3D model.

Another source of confusion is that the preview box does not consider the team it belongs to. This can cause confusion about which entities belong to which teams while editing or viewing the scenario.

This is especially troublesome in the AR implementation, where the user is reliant on the models to see the units, but the big text boxes can also obscure the 3D models of the units, leading to an unpleasant and frustrating viewing experience.

To add to this unpleasant viewing experience, the user is unable to move in the AR environment, unless they physically move the headset, as the Babylon camera is attached to the viewpoint of the headset.

Furthermore, interacting with the entity can cause weird graphical glitches in Babylon, making the entity's box permanent on the screen while the user is dragging it, which is depicted in Figure 26. This slowly makes the map view unusable.

Figure 26 – A demonstration of a problem stemming from utilizing “on hover” events in Babylon.

In the scenario’s scripting, the user cannot pick a coordinate from the map to give a move command to an entity, the user must manually write the coordinates, as can observed in Figure 27.

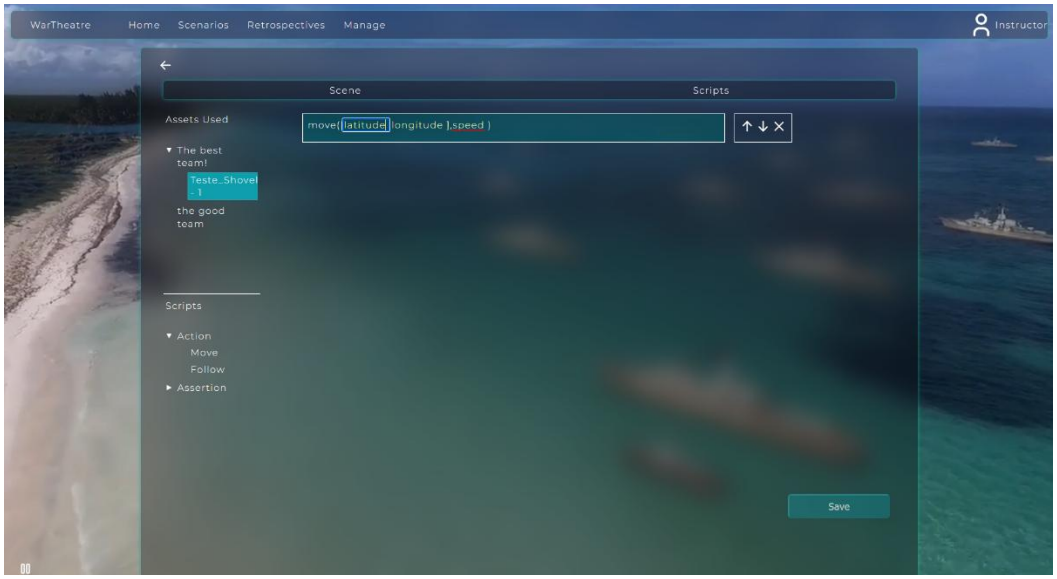


Figure 27 – Manual coordinate insertion in the scenario scripting.

3.3.2 Performance Problems

The major performance problem with the application can be found while editing or viewing a scenario. As could be seen in Figure 21, the first step of the creation of a scenario is to define an area in the world in which the scenario will be taking place.

If this area’s dimensions are large (for example a whole country), this will heavily degrade the performance of the browser while loading the map in the use cases of playing the scenario and the editing of the scenario. An example of this can be found in Figure 28.

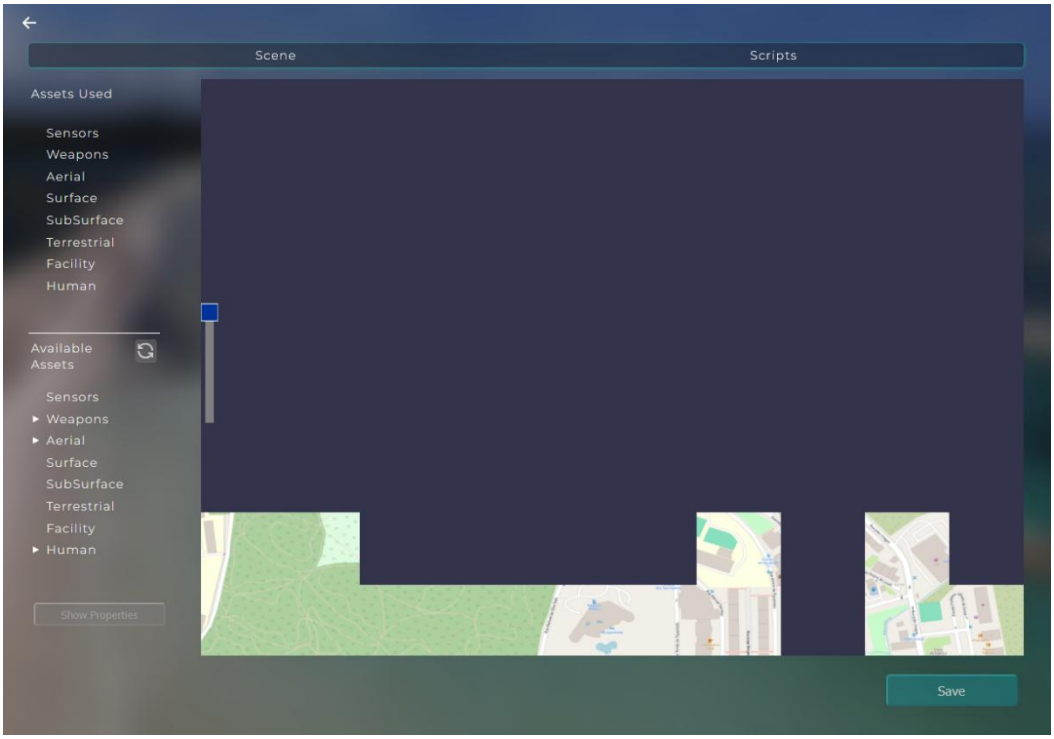


Figure 28 – An example of a slow loading map.

In other cases, the performance degradation is so severe that it causes the client browser to crash every time a specific scenario is loaded which ends up heavily impacting usability as well. This occurs because for the loading to be completed, the browser is waiting for every tile that can be loaded to finish loading by making an HTTP GET request to OSM's (OpenStreetMap) tiling server. This causes heavy performance degradation for the application and the client's browser itself.

3.3.3 Miscellaneous Problems

The user is incapable of logging out of the application, which comes with added security issues.

As researched in the previous chapter, the use of OSM's API comes with the risk of being blocked for excessive utilization of the tiling server. This can mean the platform becomes unable to support the creation, execution and editing of scenarios, as rendering the map view will become impossible due to the block placed by OSM.

3.4 Improvements and New Requirements

Considering the problems highlighted in the previous section, this section aims to present the solutions brainstormed by the author with SISTRADE, while keeping in mind the technologies that were used to build the application, and what was researched in previous sections of this work.

New requirements will also be proposed, to enhance the usefulness and security of the platform.

Table 9 summarizes the problems that were identified by their category.

Table 9 – List of problems grouped by category.

Type	Category	Problem
Usability	**Collaboration**	Collaboration is “blind”.
		Collaboration can cause the loss of data.
	Scenario	Recognizing the entities placed on the scenario map is difficult.
		Finding the entities on the map is difficult.
		Recognizing which team the entity belongs to is difficult.
		The AR visualization version also suffers from these three problems, and the user cannot move easily.
		Entity identification boxes can get stuck on the map view and in AR.
Performance	**Map View**	The performance of the application can be degraded significantly when loading the map view for big locations.
Miscellaneous	**Security**	There is no “logout” button.
	Map tiling	The use of OSM’s API can incur penalties if not properly rate limited.

The next sections will focus on elaborating the solutions devised for each of the points presented.

3.4.1 Collaboration Usability

Maintaining the use of the Y.js library already present in the application, the awareness system for the WebSocket connection option would be used to display the status of other users to each client that is interacting with the same page of the application.

Since the Y.js system is very flexible in which attributes can be sent through a WebSocket connection, details on what HTML element the current user is changing would be sent, and the client(s) receiving these changes would then be tasked with rendering the appropriate visual element that indicates a user is editing a specific HTML element. The rendering of these visual elements would be done by recurring to programmatic HTML DOM (Document Object Model) manipulation.

While the author recognizes that this practice is not intended by Angular conventions of performance, as researched in the previous chapter, it is a better approach if a centralized angular service is created that handles all the DOM manipulation, instead of being implemented in a component-by-component basis, utilizing Angular’s built-in tools. This is further expanded upon in the future Build chapter.

As the awareness system propagates changes to every user connected to the WebSocket, this would allow every user to see every other user if they are editing a form element in the specific webpage, allowing users to easily collaborate, while keeping track of other users.

Furthermore, elements being edited by other users would be disabled, to prevent situations where data can be tampered with accidentally.

To prevent the loss of data, priority can be given to the data that is in the shared editing data structure and the data that comes from a normal database read operation should be completely ignored if a field is present in the collaborative editing structure.

3.4.2 Scenario Usability

The first thing to improve is the visualization of the entity in the Babylon.js generated view, whether a user is editing or executing the scenario.

Three improvements can be made in this regard:

- A default model can be supplied for each entity. This default model would be based on the unit type.
- The unit's tooltip would have to be improved to always appear, and to include more information about the unit, positioning and team it belongs to.
- In AR, the unit should cast a line down onto the map to help the user better understand where the specific unit is on the map.

Providing default models would entail finding a generic model for every type of unit. The units can be further simplified to an arrow and a cube (aerial/terrestrial), if no suitable models are found or can be made. This change would lessen the burden that is placed on the user of supplying an accurate 3D model for a specific unit and allow the rendering of units without models on the scenario view.

The unit's tooltip can be changed into a square box that surrounds the unit, while keeping the entity model in the center. This box would be colored in the unit's team's color. Besides this box, there would also be unit information, such as the unit's name, what world position it is on, and what team it belongs to.

To further improve visibility in AR, the unit would be casting a line down onto the world plane so the user could more accurately know where this unit was placed.

The diagram in Figure 29 demonstrates the changes in a visual way for the convenience of the reader.

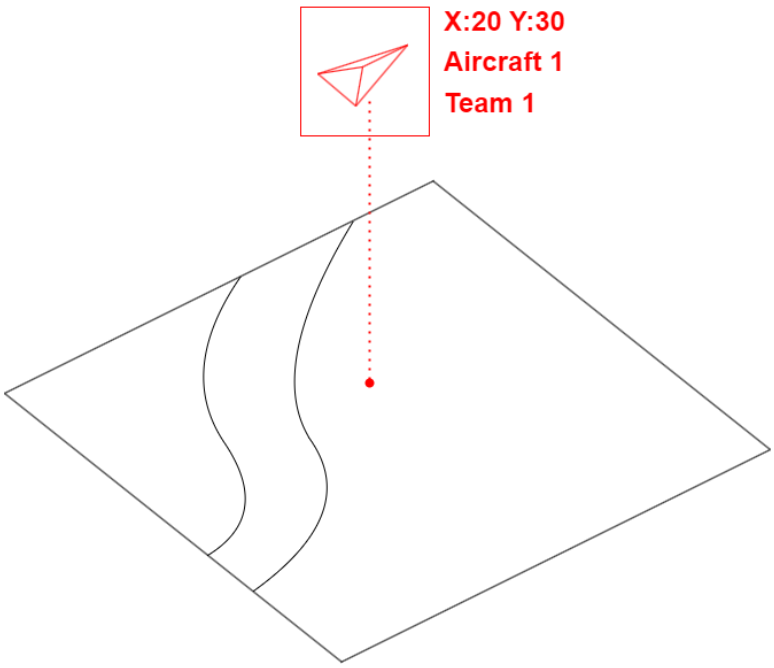


Figure 29 – Diagram envisioning the changes to be made, from the perspective of an AR user.

AR usability can be improved further by allowing the user to move with many input devices. For example, the user could utilize an on-screen UI to move the camera (supporting AR pointing devices) or make use of a keyboard to achieve the same functionality with the WASD keys on a keyboard to move forward, backwards, left and right respectively. Some keys can also be bound to control the camera's pitch and yaw.

The author recognizes that the implementation seen in Figure 29 may become cluttered over time, as the user keeps adding units to the scenario. To alleviate this, an on-screen control (and a keyboard key bind) can be implemented that hides the UI from the scene, allowing the user to focus on the 3D models in the scene, or a specific unit.

By implementing these changes, the usability of the scenario view will be increased drastically, both in 2D and AR modes.

Entity identification boxes becoming stuck is most likely a technical bug in the current implementation and this was fixed by implementing this system in the Build chapter.

3.4.3 Map View Performance

The performance degradation can be attributed to the requests made to OSM during the first loading of the scenario map that was defined upon the creation of the scenario. Regardless of camera positioning, each map tile from the whole area that was chosen upon scenario creation (previously depicted in Figure 21) generates a request for OSM's tiling map server.

As discussed before, the processes brainstormed by the author try to avoid practices that go against the responsible usage policy of OSM's tiling server, such as bulk requests and tile image caching.

3.4.3.1 FOV Culling Method

The exaggerated loading caused by the requests can be reduced by manually implementing techniques that mimic occlusion in 3D rendering engines, but for tile map rendering and the logic of sending requests.

Considering the FOV (Field of View) of the camera, the tiles that are only inside this FOV (and tiles immediately around the FOV of the camera) should send a request to the tiling server, reducing the initial load on the rendering engine, and the webpage itself, improving its LCP (Largest Contentful Paint) metric. The maximum and minimum zoom of the camera should also be changed so that the camera only captures some tiles for this solution to work at page load.

However, this comes at a cost, as the map tiles are based on a web request, this could cause the map tiles to slowly load in as the user moves the camera, due to the inherent delay of a web request. Rapid camera movements and zooming out a considerable amount in 2D mode can also cause the application to send requests for every tile that the camera saw in that movement, worsening performance while the user is utilizing the platform. Figure 30 depicts a diagram demonstrating this technique.

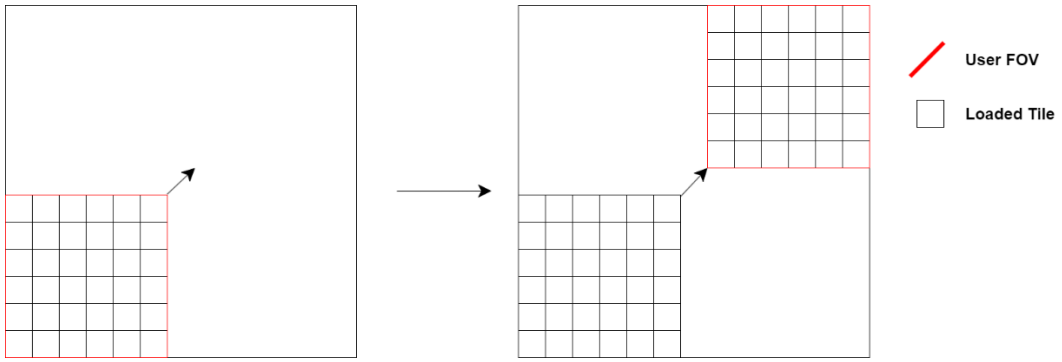


Figure 30 – Visual diagram of the FOV culling technique.

3.4.3.2 Tile Priority Method

The tile priority technique would entail attributing a loading priority (0-10) to each tile that must be loaded in the map. Tiles inside the camera's FOV would be attributed the priority of 10, while tiles outside the FOV would be given a lower priority, decreasing from 10 in function of the tile's distance to the camera's FOV border. Tiles would then be placed into data structures that would allow them to be iterated upon for sending the request and rendering.

This would allow a staggering of the rendering process, by first loading the tiles with the highest priority, and moving down the priority ladder, this ensures that the initial performance load on

the system is adequate and short for the performance the user expects, while making sure that the **whole map would eventually be loaded**, according to the priorities.

If the user moves while the map is not fully loaded, the priority rendering will have to stop momentarily to rebuild the rendering "stack", as the tile priority has completely changed. Tiles not loaded are not unloaded, as that is not necessary, only the priority is changed by the movement of the user's camera and rendering can start again. Figure 31 is a diagram demonstrating this technique.

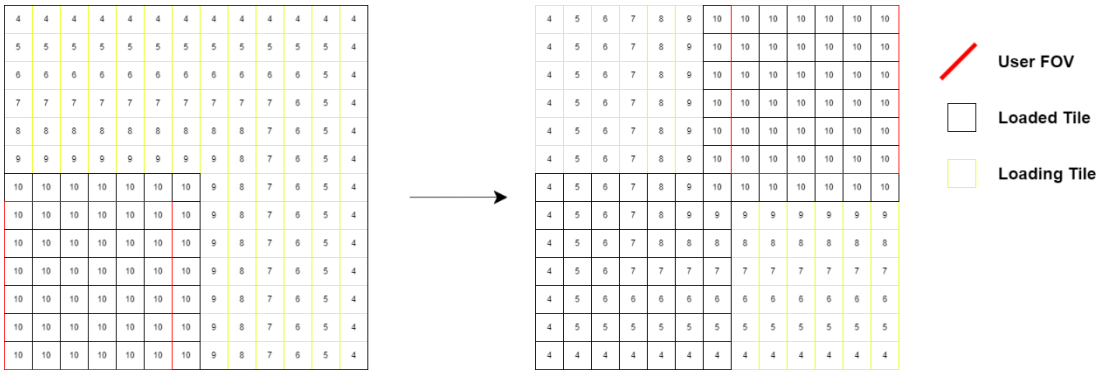


Figure 31 – Visual representation of the tile priority method.

3.4.3.3 Chunk Method

The chunk method is based on the division of the map in units called "chunks", which have a defined size. The size of a chunk represents how many tiles there are in a chunk. A size 9 chunk represents a 9 square tile chunk. Chunks would also have states, which tell the rendering engine if they are loaded or unloaded.

The starting FOV of the camera would need to be reduced to the chunk's size – 1, to account for small movements of the camera and corrections. The camera would be given a square shaped bounding range surrounding its FOV that would trigger the loading of chunks by touching an unloaded chunk. Loaded chunks are ignored.

This method would be faster than loading each individual tile separately, and it segments the rendering into faster loading parts.

The downsides are that the user must manually explore the whole map to load it completely, and that depending on the movement of the user, the bounding range of the camera and the chunk size, this could trigger loading spikes, where many requests for many chunks are sent out at the same time.

This would severely impact the performance while the user utilizes the application, reducing the initial loading times quite significantly, without a limit on chunk loading. However, if such a limit is imposed, tile loading performance could be impacted while unrelated, invisible chunks are loading. Figure 32 illustrates this technique.

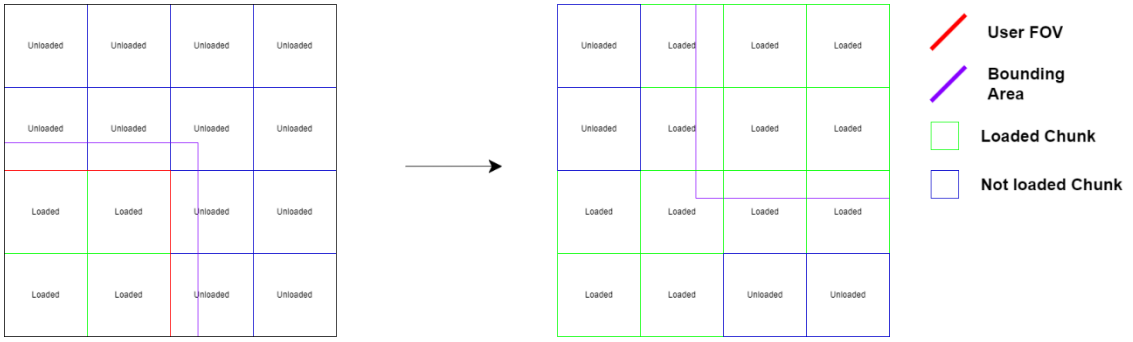


Figure 32 – Visual representation of the chunks method.

Table 10 summarises the merits and demerits of each approach.

Table 10 – Comparison of the different tile loading methods.

Method	FOV Culling	Tile Priority	Chunk
Merits	• Reduces initial loading times significantly.	• Reduces initial loading times significantly. • Allows the full map to eventually load. • Allows the whole map to start rendering immediately.	• Reduces initial loading times significantly. • More responsive loading of camera sized tiles and the surrounding area.
Demerits	• User camera movement can cause loading spikes and degrade performance. • Some parts of the map are not fully loaded.	• Rendering cannot start or continue if the camera is not “still”.	• User camera movement can cause loading spikes and degrade performance. • Some parts of the map are not fully loaded. • Forces Camera to have Chunk Size – 1 Tile FOV

From the merits and demerits listed, the priority technique seems the most favourable to use given the restrictions and conditions of the map loading process.

3.4.4 Miscellaneous Improvements

A logout button should be added to the navbar to allow currently logged in users to log out. This is an essential feature for security reasons. By clicking the log out button the user will be redirected to the login page.

The reliance on OSM's (OpenStreetMap) tile server can be detrimental to the application as written before in the state-of-the-art chapter. There are various alternatives to consider, such as hosting a dedicated tile mapping server for the application, running in a docker container, or the selection of another tile mapping server available on the internet.

Setting up a dedicated server is explained by OSM, and can be done by utilizing docker and a dedicated image provided by them [90]. This docker image is quite intensive to render the entire world, needing at least "1TB of fast storage, a quad-core processor and 24GB of memory" [65].

Another alternative to hosting a tile map server is to use another tiling map server that is active on the internet, and using OSM's map data [91]. Although many of these impose the same restrictions as OSM, if not even stricter demands on following each server's policies.

3.4.5 Retrospectives

A new requirement to be implemented comes in the form of debriefings, or retrospectives, in which the user can browse through the history of executed scenarios, see which teams and players were present at the scenario's execution and what time the scenario started and was concluded. It is also expected that the user can edit a shared document or leave collaborative messages in each debrief, with a record of edits made with a timestamp to track who wrote the messages.

From this short description, these are the functional requirements:

- The system should store executed scenario sessions.
    - When? At the start of a Session? Or when all the players leave?
- The system should allow the user to consult which teams and players were in the session.
- The system should allow the user to list every session history.
- The system should allow the user to leave notes on one session history.

From these requirements, two user stories can be drafted:

US1: As a user, I want to consult the session history of a scenario, so I can debrief it.

US2: As a user, I want to leave messages in a session's history, so I can leave notes for other users.

The SSD diagrams for both of these use stories can be seen in Figure 33 and Figure 34.

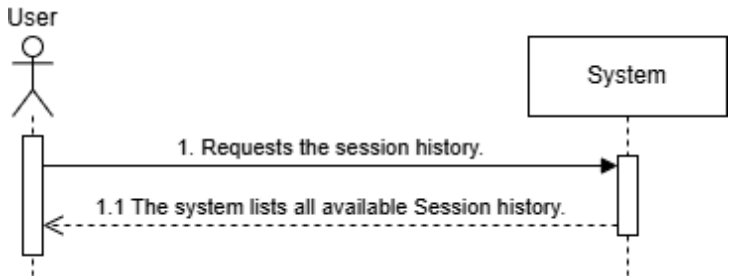


Figure 33 – US1 System Sequence Diagram.

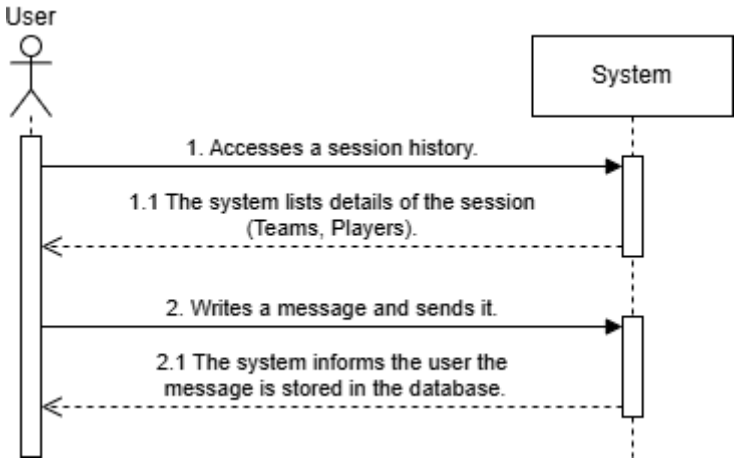


Figure 34 – US2 System Sequence Diagram.

4 Build and Development

In this chapter, aspects of the development of the application are demonstrated, building upon the analysis done in the previous chapter. Implementations are shown, explaining why it was decided to choose specific approaches over others, and what was done to implement those into the application.

To this end, some code snippets and screenshots of the application are shown, to elucidate the reader on the work that has been carried out and implemented into the application. The order of the chapter's sections follows the order of the problems to solve, exposed in Table 9 available in the previous chapter (**Section 3.4 - Improvements and New Requirements**).

4.1 Collaboration Improvements

For collaboration, the focus was to implement a solution which would be applicable to any component and page of the application that already had the collaboration functionality.

This would ensure that the usability improvements made to the collaboration functionality would be able to be added into pages which do not yet have this functionality, or new pages where this functionality is needed, making it extensible beyond a singular implementation.

It is important to accurately distinguish the feature implemented here, as it is only a way to see who is editing the page, and what these external users are editing, not the collaboration feature itself. This feature was already implemented in previous versions, but it was completely anonymous, users could edit without any indication other users were editing the same form.

The feature that was implemented was this awareness, that allows users to see who is editing and what is being edited.

The implementation (from here on called the "awareness service") was done by leveraging the Angular toolset and the Y.js library which was already in use by the application. An Angular service was created, which handles all requests for awareness that can be performed through a Y.js WebSocket. These requests include the beginning of a session; the highlighting of elements being edited by other users and the deactivation of these elements. As was alluded to in the previous chapter (**Section 3.4 Improvements and New Requirements**), this is done by manipulating the DOM (Document Object Model) object by utilizing JavaScript.

Firstly, the component must set up an awareness session by utilizing the awareness service that was implemented. This first call initializes the necessary components required by Y.js to track awareness, such as setting up the JSON object (seen in Figure 35) that is used to transmit information and assign colors to the users. Every time the local state object suffers a property change, an event is triggered that sends this JSON object to other connected clients.

```
setUpAwareness(provider: WebsocketProvider) {      You, a month ago • Reafctored code to be in a r
    this.wsUserAwareness = provider.awareness;
    this.currentUserColor = selectColor();
    this.wsUserAwareness.setLocalState({
        name: this.currentUser!.name, // Name of the user.
        color: this.currentUserColor, // Random user color.
        locatedOnTab: null, // For pages with tabs.
        previousElement: null, // Used for removal of previous awareness elements.
        currentInputElement: null // Used for new additions of awareness elements to HTMLElements.
    });
```

Figure 35 – Preparing the local state object.

For the collaborative forms, this implementation tracks the user's name, color (which is assigned at a login of a user, as seen in Figure 36), and which field the user is editing, along with the previous element that was edited, so that previous HTML elements can be removed. It also supports pages with tabs.

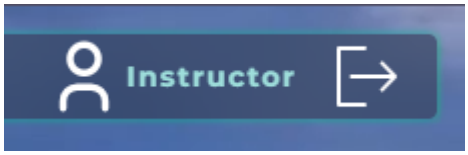


Figure 36 – User color in application navbar.

After this setup, when elements that have collaboration are focused, an *onfocus* event is triggered in the component, which then calls a function (addEditingAwarenessToElement, shown in Figure 37) from the service which highlights this element and deactivates it for the users which can see this highlighting. When the elements are unfocused, the *blur* event is fired, which removes the highlighting from the element and reactivates it (removeEditingAwarenessToElement, Figure 37).

```
private addEditingAwarenessToElement(element: HTMLElement, userName: string, userColor: string) {
    element.classList.add('rca-user-editing-input');
    element.setAttribute('style', 'border-color:' + userColor);
    element.setAttribute('disabled', 'true');


    let editCard = document.createElement("div");

    editCard.classList.add('rca-user-editing-card');
    editCard.id = "currentEditingElementCard-" + userName;
    editCard.setAttribute('style', 'background-color:' + userColor);
    editCard.textContent = userName;

    element.parentElement?.append(editCard);
}

private removeEditingAwarenessToElement(element: HTMLElement, userName: string) {
    element.classList.remove('rca-user-editing-input');
    element.setAttribute('style', 'border-color:unset');
    element.removeAttribute('disabled');

    let elementCard = document.getElementById('currentEditingElementCard-' + userName);

    if (elementCard !== null) elementCard.remove();
}
```

Figure 37 – Functions that are used to manipulate the DOM of collaborative elements.

A user list was also implemented using an Angular component (html and typescript implementation shown in Figure 38), allowing it to be inserted as needed into collaborative pages that already exist, or possible future ones. This implementation always uses the user list from the awareness service, so it is synchronized with the number of users in each awareness session.

colab-user-list.component.html

src > app > colab-user-list > colab-user-list.component.html > div.rca-collab-users-content.editingUsersFlex

Go to component | You, a month ago | 1 author (You)

```
1  <div class="rca-collab-users-content editingUsersFlex">
2      <div *ngFor="let user of userList; index as i;" class="rca-user-presence-card"
3          [ngStyle]="{'background-color': user.color}">
4          {{user.name}}
5      </div>
6  </div>
```

colab-user-list.component.ts

src > app > colab-user-list > colab-user-list.component.ts > ...

You, a month ago | 1 author (You)

```
1  import { Component, Input } from '@angular/core';
2
```

You, a month ago | 1 author (You)

```
3  @Component({
4      selector: 'app-colab-user-list',
5      templateUrl: './colab-user-list.component.html',
6      styleUrls: ['./colab-user-list.component.css']
7  })
8  export class ColabUserListComponent {
9      @Input() userList: Array<any> = new Array();
10 }
```

Figure 38 – The user list component (HTML on top and Typescript on bottom).

These implementations allow users to edit collaborative forms in the application with instant feedback about other users (i.e., awareness), as can be seen in Figure 39. This figure shows the Trainee 1 user (left) editing the “Branch” element of an asset. While the instructor (right) cannot edit the element, as it is disabled, he can watch and get feedback that another user is editing this element.

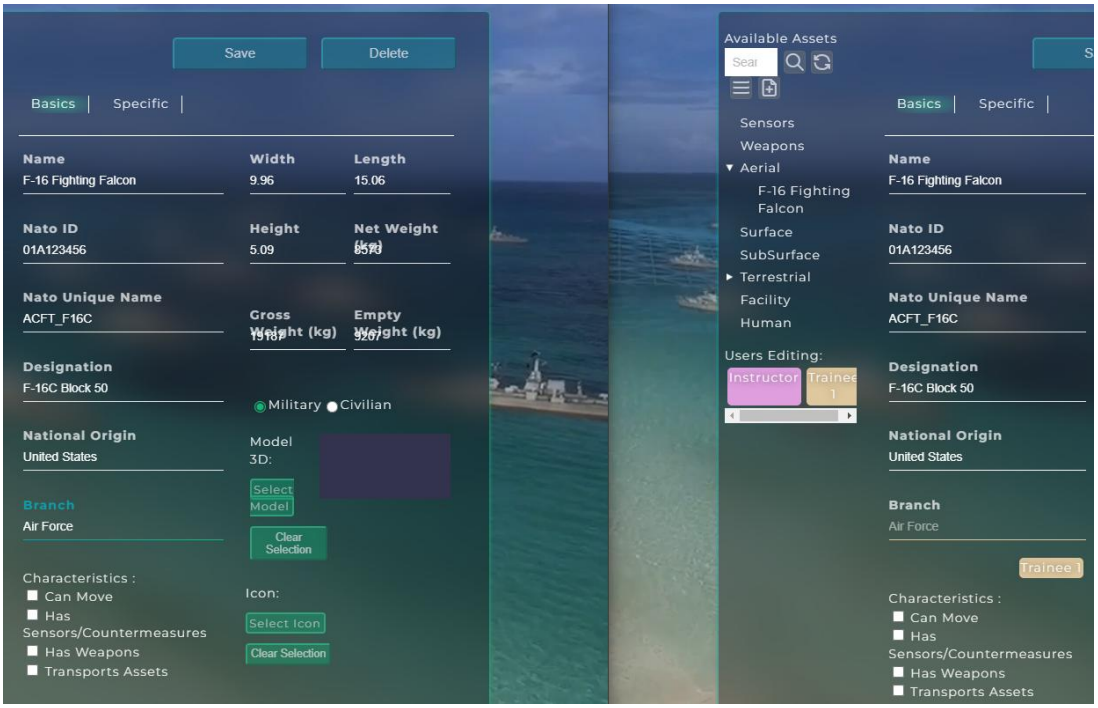


Figure 39 – An example of collaborative awareness.

When the elements are unfocused, the *blur* event is fired, which removes the highlighting from the element and reactivates it.

Whenever two users enter a page, a user list appears, to indicate that more users are on this page and can edit other fields. This was implemented as a separate Angular component, so that it can be reutilized and repurposed for different pages and future implementations of collaboration.

These implementations result in the collaboration awareness that can be seen in Figure 40, where a user is editing the “Branch” field, and while the Instructor user cannot make changes to the field, he can see that Trainee 1 is editing it. We can also see the user list below the asset list, which highlights the users present on this page. Some other alterations were made to the navbar to facilitate awareness integration, notably, the user login color (see right top conner, also demonstrated in Figure 36) is matched with awareness service allowing an easy identification of the fields being edited by the user.

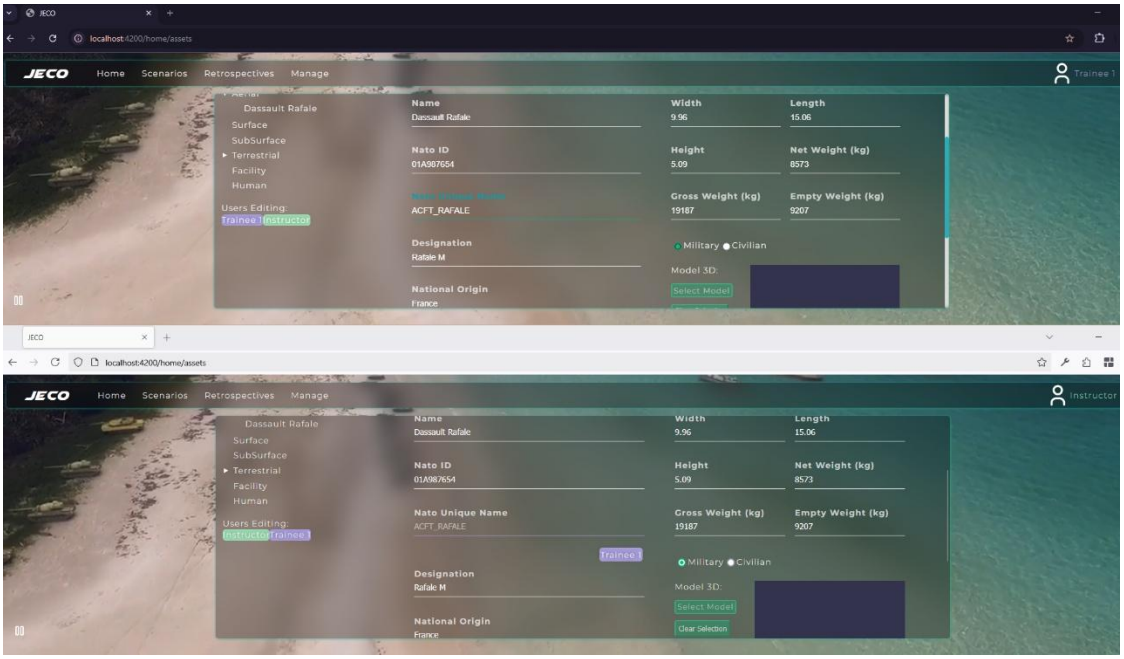


Figure 40 – The collaboration awareness present in the asset management page.

Since collaboration is also an element in Babylon.js scenes, the implementation follows the same philosophy but adjusted to a Babylon.js scene's needs. In the scene, the cursors of the users need to be highlighted in 2D space. In accordance with these objectives, the local state object tracks the position of the user's cursor in the Babylon scene and shares it with the other users. This setup can be seen in Figure 41.

```
public setupBabylonCursorAwareness(provider: WebsocketProvider) {
    this.wsUserAwareness = provider.awareness;
    this.currentUserColor = selectColor();
    this.wsUserAwareness.setLocalState({
        name: this.currentUser!.name, // Name of the user.
        color: this.currentUserColor, // Random user color.
        babylonUI: null, // The UI object of babylon.
        babylonSceneVector: null, // The position of the user's cursor raycast into babylon's scene.
        renderCursor: false // If the client should render this cursor.
    });
```

Figure 41 – Set up of the Babylon.js specific local state.

The mouse tracking is done by creating GUI elements which the awareness service can reposition in the scene as users are moving, with the positions of the cursors in the scene being transmitted through the WebSocket that was set up for collaboration.

The position of the projected cursor is sent to the local state each time the cursor moves, which causes the event to fire each frame that the cursor is in a different position. This update function can be seen in Figure 42.

```
// Add a pointer move event listener to the canvas
canvas.addEventListener("pointermove", (event) => {
    // Get the mouse position on the canvas
    let pointerX = event.offsetX;
    let pointerY = event.offsetY;


    // Get the intersection between a ray from the mouse position and the ground plane
    let pickResult = scene.pick(pointerX, pointerY);
    if (pickResult.hit) {
        // Display the resulting coordinates on the screen
        let x = pickResult.pickedPoint!.x.toFixed(4);
        let z = pickResult.pickedPoint!.z.toFixed(4);


        this.awareness.updateCursorAwarenessPosition(pickResult.pickedPoint!);
```

```
ervice.ts C:\Users\rafca\Mah Stuff\SISTRADE WORK\Repositories\EDOCC\WarTheater\war_theater_frontend\src\app - Implementations (1)
xport class AwarenessService {
  public updateCursorAwarenessPosition(vector: BABYLON.Vector3) {
    let localWSState = this.wsUserAwareness.getLocalState();
    if (localWSState === null) return;
    localWSState.babylonSceneVector = vector;
    localWSState.renderCursor = true;
    this.wsUserAwareness.setLocalState(localWSState);
  }
```

Figure 42 – The event in the Babylon.js scene and refresh of the local state.

These implementations enable the real-time visualization of collaborative users' cursors, both while editing scenarios and executing them, as shown in Figure 43.

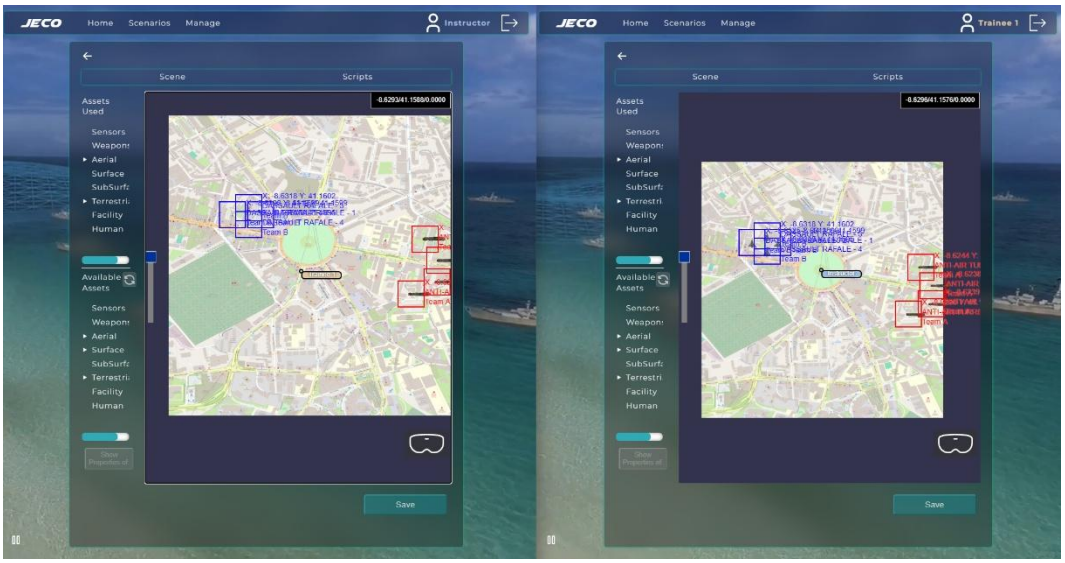


Figure 43 – Awareness in Babylon.js by utilizing cursor positions in the same scenario.

Finally, as previously described in the previous chapter (**Section 3.4 Improvements and New Requirements, Table 9**), the primary bug in the asset management page, in which the simultaneous editing of a field would cause data loss, was also fixed. The fix consisted in, if the shared collaboration structure already had data in it, then, apply the changes the first user had made, that were automatically synchronized in the Y.js WebSocket, to the second user's

instance of the collaborative data structure. Figure 44 shows this fix implemented in the code, making use of Y.js.

```
this.wsProvider.on('sync', (isSynced: boolean) => {
    console.log('Yjs sync status:', isSynced);
    if (isSynced) {
        if (this.ymap.get(0) === undefined) {
            this.ymap.insert(0, [temp]);
        } else {
            this.ymap.insert(0, this.ymap.get(0));
        }
    }
});
```

Figure 44 – Fix for collaboration data loss.

4.2 Map View Improvements

The improvements made to the scenario editing come in the form of usability improvements that touch upon the identified problems, such as identifying entities in the map view, and correctly dragging them around the view area, identifying which team they belong to, and giving the entities default models to facilitate the dragging of the entity (see **Section 3.4.2** for more details).

4.2.1 Models and GUI Improvements

Simple models were implemented by recurring to default polygonal 3D shapes built into Babylon to ease the load on the system. As can be seen in Figure 45 these shapes are added to the scene when the entity does not possess a valid 3D model, and when and Aerial entity, the polygonal shape is different to distinguish it.

```
addEntity(modelChosen: PhysicalEntity, idNode: number, positionExisting: string) {      You, 2 months ago • Added 2D cu

    let entity: BABYLON.Mesh;
    let aerial: boolean = false;
    if (modelChosen.asset instanceof Aerial) {
        entity = BABYLON.MeshBuilder.CreatePolyhedron(modelChosen.asset!.entity.name + "-" + modelChosen.entityNumber,
            { type: 0, sizeX: 0.06, sizeY: 0.04, sizeZ: 0.02, updatable: true },
            this.scene);
        aerial = true;
    } else {
        entity = BABYLON.MeshBuilder.CreateBox(modelChosen.asset!.entity.name + "-" + modelChosen.entityNumber,
            { width: 0.1, height: 0.1, depth: 0.1, updatable: true },
            this.scene);
    }

    BABYLON.FilesInputStore.FilesToLoad = {};

    let imageInput: { name: string, content: string | ArrayBuffer | null; }[] = modelChosen.asset!.entity.model;
    // console.log('before the prepare');
    this.prepareEntity(entity, modelChosen, idNode, positionExisting, aerial);
```

Figure 45 – Adding default shapes to the entities in the Babylon scene.

In addition to the default models, changes were made to allow users to more easily parse the information presented to them, via new GUI elements that show information about the entities in each scenario. These elements show the entities position, name and affiliation, while being the same colour as the team they are affiliated with, as can be seen in Figure 46. Lines were also added to aerial units to make them stand out more when compared to other types of units in the scenario.

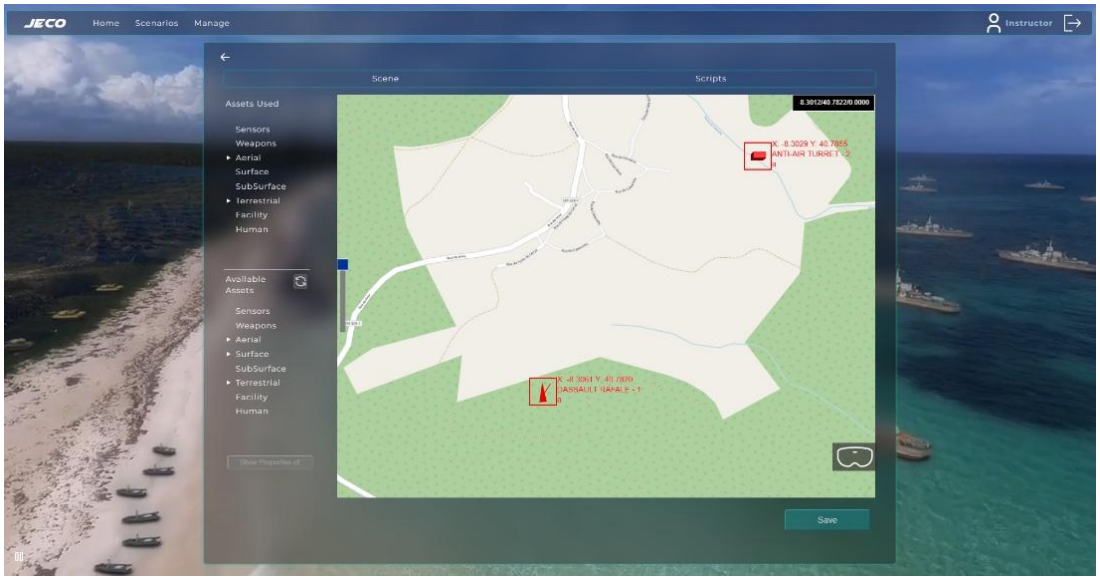


Figure 46 – An example of a scenario with the new polygonal models.

As mentioned before, the user can also import his own custom models, which would overwrite these polygonal shapes. Figure 47 shows an example of custom models in the edit scenario view.

Figure 47 – An example of a custom model imported in the scenario view. (Model from [92])

4.3 Map loading

As analyzed before (**3.3.2 Performance Problems**), the map loading done in the edit scenarios screen contributes to a major loss in performance, especially when being loaded for the first time. This causes a major usability issue for the user, which must wait until the loading is finished to effectively edit the scenario.

To resolve this issue, the priority strategy presented in the analysis chapter (see **Section 3.4.3.2**) was chosen, for its guarantee that the map would be loaded and the apparent reduction in instant loading times.

Firstly, the loading of the scene utilizes the Babylon CreateTiledGround function to create a tiled plane surface in which the map's textures can be loaded and units placed. Herein, a MultiMaterial, with the same subdivisions (same number of tiles) as the tiled ground, is created, and filled with placeholder material textures (implementation shown in Figure 48).

```
var multimat = new BABYLON.MultiMaterial("multi", scene);


console.log(this.sizeMap);
var xTileBase = Math.floor(lo2t(+this.scenario.map.coordinates[0][0], zoom));
var yTileBase = Math.floor(la2t(+this.scenario.map.coordinates[1][1], zoom));
const placeholderMaterial = new BABYLON.StandardMaterial("placeholder", scene);
placeholderMaterial.diffuseColor = new BABYLON.Color3(0, 0, 0);

for (let row = 0; row < subdivisions.h; row++) {
    for (let col = 0; col < subdivisions.w; col++) {

        multimat.subMaterials.push(placeholderMaterial);
    }
}


this.tiledGround.material = multimat;
```

Figure 48 – The creation of the multimaterial.

Secondly, sub meshes are also added to the main tiled ground object. In this process, a new class containing all relevant metadata for specific tiles is added into an array data structure, so that it can be iterated through later. This metadata includes the positioning of the tile, if the tile is loaded, the tile's material reference, what coordinates to send to the OSM server for texture rendering and the tile's sub mesh reference. This new class and its insertion in the array is shown in Figure 49.

```
for (var row = 0; row < subdivisions.h; row++) {
    for (var col = 0; col < subdivisions.w; col++) {
        var submesh = new BABYLON.SubMesh(
            index, 0, verticesCount, base, tileIndicesLength, this.tiledGround
        );
        this.tileData.push(new TileMetadata(
            row,
            col,
            index,
            false,
            xTileBase + col,
            yTileBase - row,
            undefined,
            submesh
        ));

        this.tiledGround.subMeshes.push(submesh);
        base += tileIndicesLength;
        index++;
    }
}
```

Figure 49 – The instantiation of new tile objects into the tile data Array.

Finally, with the previous developments set up, the priority map loading algorithm was implemented. This was done by recurring to Observables found in Babylon.js, namely, the *onBeforeRenderObservable.* This observable allows Babylon to run instructions before each frame is rendered, allowing the application to scan where the camera is currently looking, and assign priorities to the tiles based on the center tile.

The first operation of the priority map loading algorithm is finding the center tile. To do so, different approaches were tested, such as ray casting from the camera directly to the tiled ground. This approach impacted performance severely, as with every frame, the camera was casting a ray down onto the tiled ground and reporting a collision.

Thus, instead of utilizing ray casting, it was decided to combine the use of variables that already existed, such as map size, subdivisions and the tile's width and height values to try and find the closest tile to the camera's current position and use this as the starting point (implementation seen in Figure 50).

```
scene.onBeforeRenderObservable.add(() => {

    let x = this.camera.position.x;
    let z = this.camera.position.z;      You, 2 weeks ago • Implemented the map loading prioritization techni…


    const [minX, minZ] = this.sizeMap;
    const cols = subdivisions.w;

    const col = Math.floor((x - minX) / tileWidth);
    const row = Math.floor((z - minZ) / tileHeight);

    const validCol = Math.max(0, Math.min(col, subdivisions.w - 1));
    const validRow = Math.max(0, Math.min(row, subdivisions.h - 1));


    const index = validRow * cols + validCol;


    const submesh = this.tiledGround!.subMeshes[index * 2];

    for (let tile of this.tileData) {

        if (submesh?._id == tile.subMesh?._id) {
            newCenterTile = tile;
        }
        if ((centerTile !== undefined && newCenterTile !== undefined) && (newCenterTile.index !== centerTile.index)) {
            for (let i = 15; i ≥ 0; i--) {
                this.priorityMap.set(i, new Queue<TileMetadata>());
            }
            currentPriority = subdivisions.w;
        }

        centerTile = newCenterTile;

    }
```

Figure 50 – Finding the center tile in relation to the camera's position.

Using the center tile as a reference, the loading priorities of the other tiles can be calculated based on their distance to this center tile as seen in Figure 52. The priority of a tile can be defined by a numeric identifying value that is assigned to a queue which contains a list of tiles of the same priority in a Map created before the rendering Observable loop (creation can be seen in Figure 51). A queue is utilized to enable movement of the camera, which can add more unloaded tiles to the queue without stopping the loading of other tiles.

```
for (let i = subdivisions.w; i ≥ 0; i--) {
    this.priorityMap.set(i, new Queue<TileMetadata>());
}
```

Figure 51 – Declaration of the map that holds every tile queue.

```
for (let tile of this.tileData) {

    if (centerTile !== undefined && !tile.loaded) {
        const distCol = Math.abs(tile.col - centerTile.col);
        const distRow = Math.abs(tile.row - centerTile.row);

        const dist = distCol + distRow;
        const maxDistance = subdivisions.w + 1 / 2;
        const priority = Math.max(1, maxDistance - dist); //

        this.priorityMap.get(priority)?.enqueue(tile);
    }

}
```

Figure 52 – Queueing of tiles based on the distance to center tile.

After all tiles are queued, and guaranteed to have priorities, the next operation is executed by utilizing the respective observable (onAfterRenderObservable) which kickstarts the process of fetching textures from the OSM tile server. This observable is similar to beforeRender, with the main difference being that this runs after a frame is rendered (implementation seen in Figure 53). The addOnce() function is used to attach a continuously running function to the DOM, instead of running this function every frame, which would be expensive.

```
scene.onAfterRenderObservable.addOnce(() => {
    // Run after current frame to ensure priorities are ready
    setInterval(() => {
        let finished: boolean = true;
        if (currentPriority >= 0 && finished === true) {
            finished = this.dequeueAndLoadTexture(multimat, subdivisions, currentPriority);
            currentPriority--;
        }
    }, 250);
});
```

Figure 53 – The code that begins the rendering process, by priority.

The goal with the use of the *setInterval* function, is to have the tiles be loaded independently of the internal rendering process of Babylon, staggering the requests to OSM for each tile, improving performance. The dequeue and load texture function will then sequentially load textures for tiles of a certain priority.

As a way of avoiding overloading the browser with requests, an adequate delay for the function had to be decided.

If the delay was too big (> 750ms), the browser would crash with a stack overflow error, due to the high volume of images that are kept in Babylon.js' buffer before they are loaded into the meshes in the scene.

If the delay was too small (< 100ms), the tiles would load fast, overloading the browser and severely impacting performance.

The chosen delay was 250ms as it still allows small scenes to load at a rapid pace, while flushing Babylon's buffer quickly enough.

4.4 Improved AR mode mobility

A problem that was detected in previous iterations of the software was that the user cannot move while in AR space (see **Section 3.4.2 Scenario Usability**), which means that the only way for the user to move while in AR mode is by using the bundled controllers that would come with their device.

After further clarification with SISTRADE, this mode is intended to only immerse the user in the simulation, which means that AR movement by using the included controllers was considered non-necessary, and, as such, some implementations were made to facilitate the movement of

the camera with the use of key bindings and mouse movement. This allows the user to move their point of view while running or editing simulations or scenarios, while at the same time being immersed in the simulation through his device of choice.

The HoloLens 1 (2019) device available at SISTRADE turned out to be incompatible with the WebXR version used by the Babylon implementation, due to the in-built browser of the device being the old Edge browser (not using chromium) which does not support WebXR and is outdated in relation to the newest chromium version of Edge. Therefore, it was not possible to enter WebXR sessions using the HoloLens 1 device. Furthermore, it was discovered that the device itself has lost all ongoing support for WebXR [93].

As a way of overcoming the previous limitations, all developments in this section were implemented and tested by utilizing the WebXR emulator available for chromium browsers [94]. While this emulator allows for some simple debugging and development it is cumbersome for developing gesture-based navigation and to use similar AR features.

The developed solution is based on the implementation of two camera systems: the Locked Camera, which can be controlled by rotation motions, provided keybinds and by walking with the AR device, and the Free Camera mode, which can be controlled by the user by utilizing his keyboard and mouse.

The Locked Camera mode is the predefined mode when entering the AR view. The starting position is in the middle of the scene, facing north, as seen in Figure 54. From this position, the user can utilize the provided key bindings (Table 11) to move around the scene by teleporting.

Table 11 – The key bindings available to the user in the “Locked Camera” mode.

Key	Effect
Q	Cycle viewpoints to the left
E	Cycle viewpoints to the right
R	Reset to starting position
F	Toggle between the two camera modes

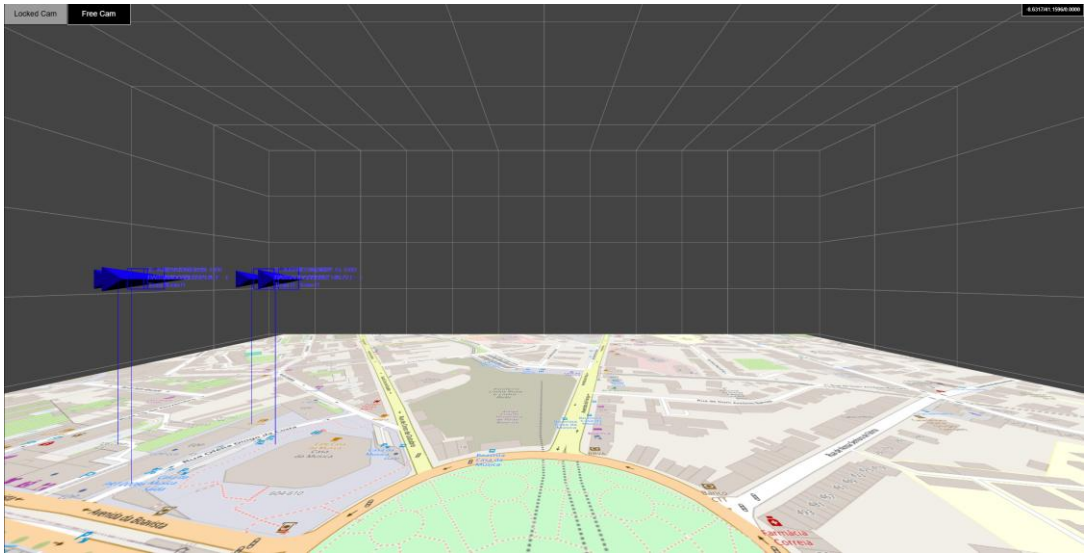


Figure 54 – A demonstration of the starting position of the user when entering the AR session.

By utilizing the key binds “Q” and “E” from this point, the user can teleport into a corner of the scene and get different perspectives on the units in the simulation, as can be seen in Figure 55. While these perspectives are pre-defined positions based on the map’s dimensions, the camera's point-of-view and rotation changes as the user utilizes the key binds.

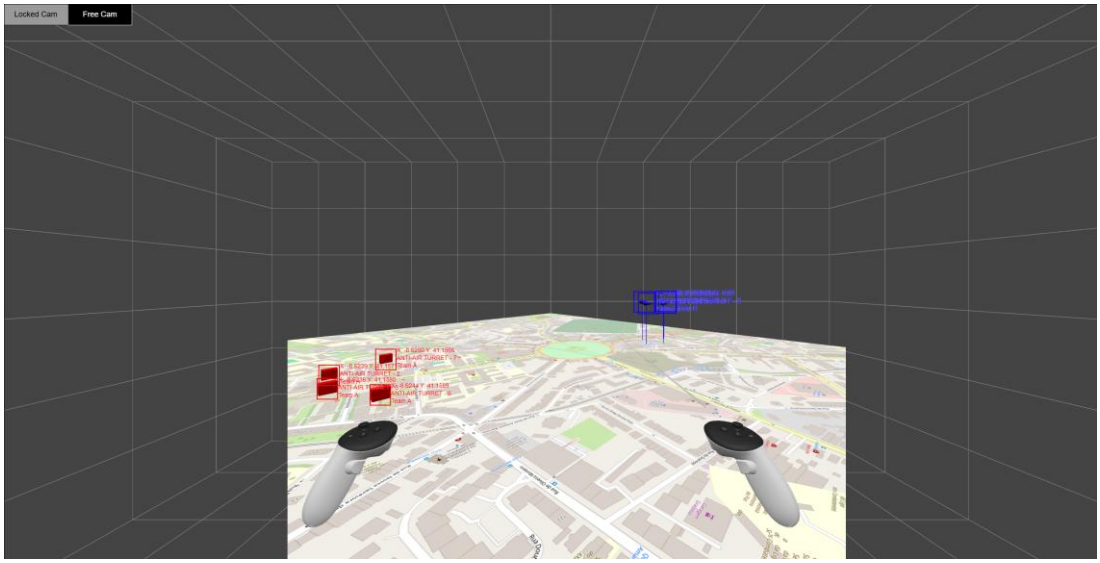


Figure 55 – A demonstration of a different perspective the user can cycle to.

The user can also get a birds-eye view of the simulation by cycling through the available positions until the last one by using the “E” key bind. This is a top-down view (seen in Figure 56) which covers the whole scene by utilizing an Orthographic camera with the dimensions of the map. This has the caveat of making the implemented graphical user interface hard to read, as it will appear farther away if the map is too big.

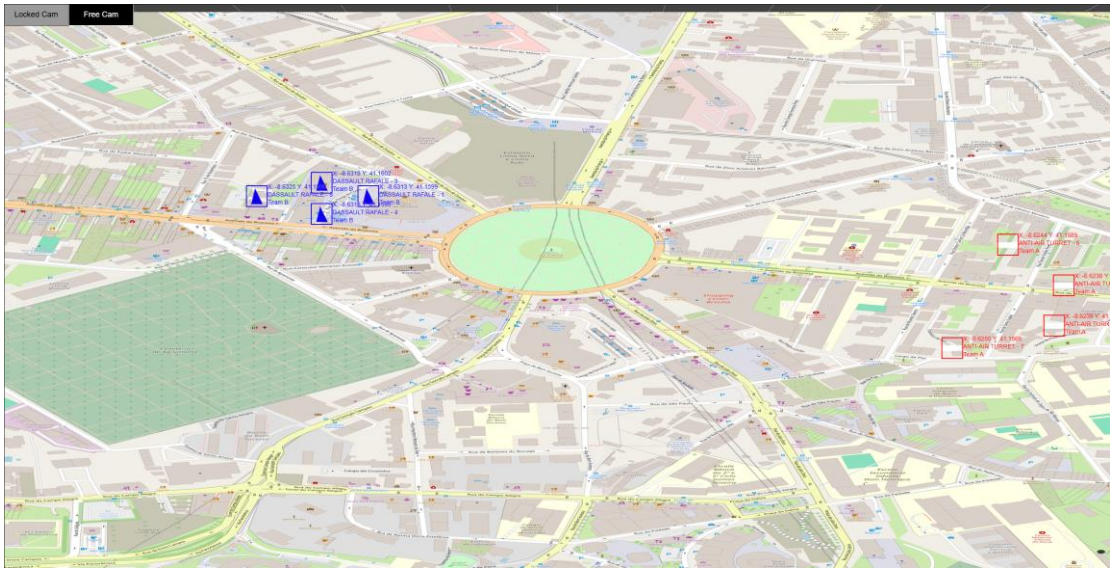


Figure 56 – The top-down view.

When in the Locked Camera mode, the user can at any time switch to the Free Camera mode by pressing the “F” key. This will take the current position of the camera (excluding the orthographic top view) and allow the user to control the direction and movement of the camera. The key binds allowed in this mode are listed in Table 12.

Table 12 – The key bindings available to the user in the “Free Camera” mode.

Key	Effect
W	Move forwards
A	Strafe left
S	Move backwards
D	Strafe right
Q	Rotate left
E	Rotate right
R	Reset to starting position
F	Toggle between the two camera modes

This mode allows the user to freely move the camera around the scene by utilizing the “W”, “A”, “S”, “D” key binds, as well as his mouse input. The mouse input will be locked into the Babylon scene upon clicking the canvas window, and the user will be able to move his camera around, making shots like the one in Figure 57 possible.

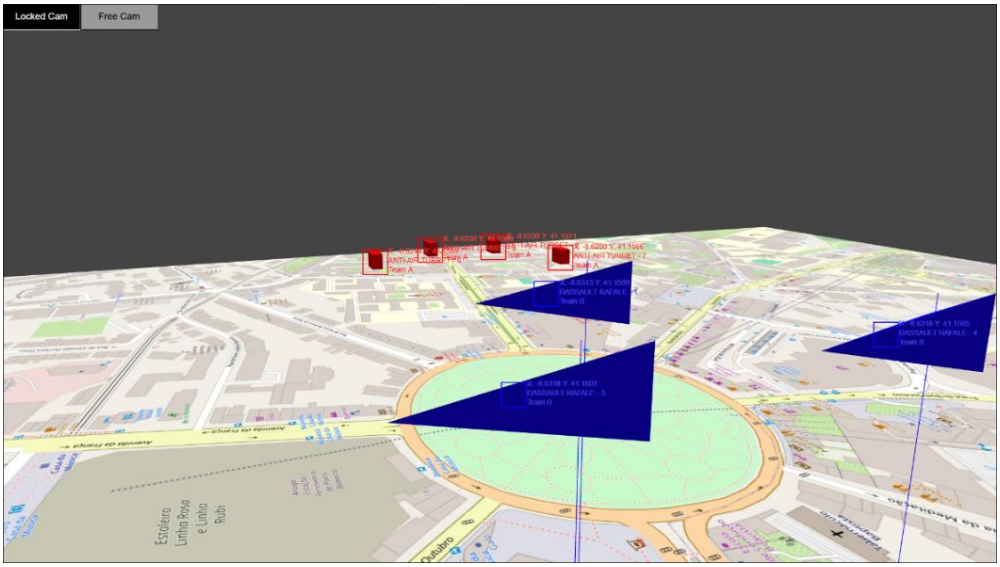


Figure 57 – An example of a camera angle only achievable through the Free Camera.

If no mouse input is available, key bind rotation (Q, E) was also implemented to help cover the missing mouse input. However, if the user is missing a mouse input before entering the AR mode, there is no key bind to enter the AR mode or dedicated button. As explained before, this is all done through the WebXR emulator, which requires either the extension to be active in the browser, or an MR device.

These developments, as well as the optimized map loading, were also implemented in the AR view of the application's simulation execution screen, allowing the user to move his perspective around while the simulation itself is playing.

4.5 Additional Developments

This section lists the miscellaneous developments carried out on the application that are not related to the previously exposed developments but were required by the project where the application is being developed. As such, these are enumerated into a small list, presented below:

- **A log out button was implemented besides the username in the navbar (**seen in Figure 36**)**. As discussed in the analysis chapter (**Section 3.4 Improvements and New Requirements**), this improves the application's security.
- **Implemented the deletion operation of scenarios (**seen in Figure 59**).** This development came as a spur of the moment implementation, to make the application easier to use and manage.
- **Added a restart button to the simulation execution (**seen in Figure 58**).** The restart is synchronized between collaboration users.

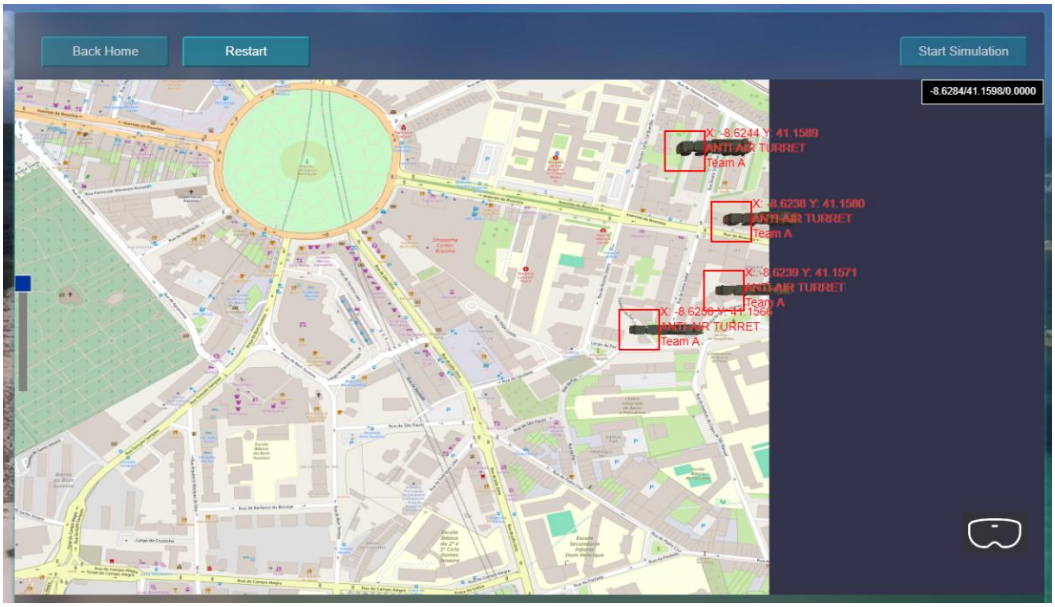


Figure 58 – The restart button implemented in the Simulation execution page.

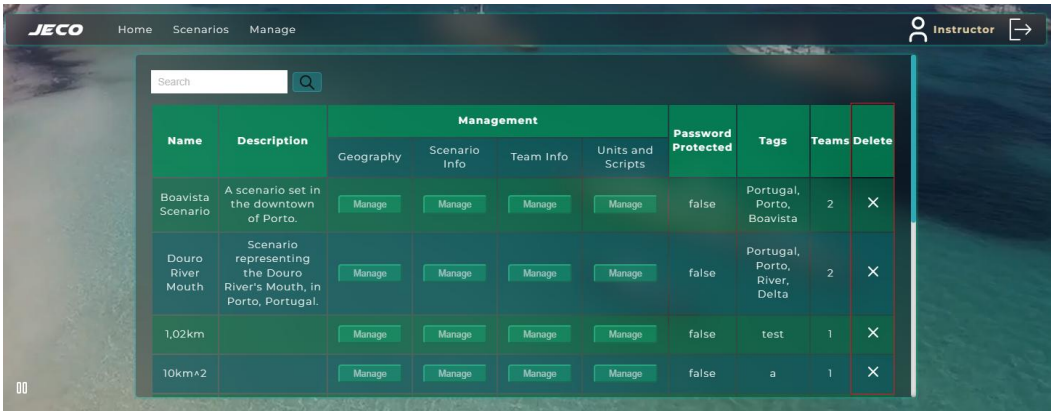


Figure 59 – The delete operation in Edit Scenario (highlighted in red).

The retrospectives requirement was a good requirement to enhance the capabilities of the platform, however, it was overshadowed by the developments described in this chapter, as they took center stage in this phase of the project due to time constraints. These developments were considered integral to the development of the minimum viable product.

5 Evaluation

In this chapter, the implementations carried out in the previous chapter are evaluated to determine if they were successful in achieving the desired outcomes and fulfilling the objectives proposed in the first chapter of this document.

While the chapter is focused on the most “testable” development implemented, the map loading optimizations, some other evaluation metrics are also be explored, such as the usability of the AR component and the scenario view.

5.1 Definition of Evaluation areas

The progress of the project during the development phase caused the techniques studied in the State-of-the-Art chapter (**Section 2.3 Platform Optimization**) to become inadequate to the developments that were carried out. This section explains and gives justification as to why some techniques in the mentioned chapter were not utilized before, during, or after development.

Before development started, and without access to the baseline platform, the performance problems were not known in full. Due to this, the author carried out a cursory analysis of techniques that could be utilized to improve the performance of the technologies that were already in use by the platform. As the project started, these performance problems became clearer, eventually culminating in the scenario map loading being the only major performance problem present in the application.

With this revelation, some of the metrics and techniques no longer applied to this project, such as the use of the Node.js profiler (there were no identifiable performance problems with the backend of the application), some metrics from Lighthouse reports and angular optimizations become unusable, as this is uniquely a Babylon.js issue which is not tracked by the browser metrics. It is also not possible to easily evaluate the usability developments done with the research rules and heuristics.

Some performance metrics can still be utilized, such as INP, as this technique records how much time it takes for a click to take effect in loading a full page of HTML (in this case), but this does not account for the Babylon.js scene and what happens inside of it.

To account for these developments, the loading of the scenario map is still evaluated, by recurring to the use of the built-in Node.js function *performance.now()* to create accurate timestamps and a time difference. For the usability improvements, the implemented developments were evaluated by comparing screenshots from the previous version of the application with the newest developments. This comparison was carried out in the areas most affected by these optimizations, such as the map view, collaboration and AR view.

5.2 Map Loading Performance Evaluation

As mentioned in the previous section (**5.1 Definition of Evaluation areas**) the evaluation of this development was conducted through the "built-in function *performance.now()* to create accurate timestamps and a time difference". The intent for the use of this function (and this evaluation) was to quantify how much faster (or slower) the new map loading implementation is when compared to the older version of the application.

From a more technical point-of-view, this function was utilized to make a timestamp when the Babylon.js scene started and when the map is fully loaded, producing a load time from the timestamp differences. The load times were then compared to arrive at a conclusion.

5.2.1 Testing Environments

As a way of exploring the interference of computation effort versus machine performance, the map loading tests were run on two different machines. While the first machine represents a common laptop, with low-tier specifications, Machine B possesses a dedicated GPU which can help with rendering tasks such as the one being tested.

The general specifications of both machines are listed in Table 13.

Table 13 – Both of the machine's specifications utilized in testing.

Part	Machine A Specifications	Machine B Specifications
CPU	11th Gen Intel Core i5-1135G7 @ 2.40GHz (4.20GHz Turbo)	AMD Ryzen 5 7600 @ 3,80GHz (4,90Ghz > Turbo)
RAM	8GB	32GB
Storage	239GB	1,8TB
GPU	Integrated CPU graphics. (Iris Xe Graphics)	AMD Radeon RX 7800 XT
OS	Windows 10	Windows 11
Power Plan	Balanced (Slider on Best Performance)	Balanced (Best Performance option in settings)
Browser	Chrome (Anonymous)	Chrome (Anonymous)

5.2.2 Testing Scenarios

Testing was done by utilizing nine predefined scenarios which grow by 5km² (excluding the first scenario, starting at 1km²). The scenarios start at 1km² and grow to 40km². Counting by utilizing the specified function only begins when the Babylon.js scene has started rendering, the HTML page load is ignored. The test ends when all textures are loaded in all tiles, at which point a browser console message is printed with the time achieved during the test.

Tests that fail for any reason were identified as such as DNFs and a rerun of the test is then executed. If this run succeeds, the time is registered. If the test DNFs (did not finish) more than three times, it is registered as a DNF on the table.

5.2.3 Testing Results

The conducted testing strategy focused on the evaluation of how fast the map is loaded, and the usability of the edit scenario function while it is loading.

Results obtained for both machines are shown in Table 14.

Table 14 – Test results by machine and loading method.

Method	Size	Avg. (Tiles/km²)	Machine A	Machine B
Regular	1km² (88 Tiles)	88	2222ms	296ms
	5km² (399 Tiles)	79,8	6230ms	1162ms
	10km² (756 Tiles)	75,6	12078ms	1546ms
	15km² (1248 Tiles)	83,2	16425ms	3778ms
	20km² (1560 Tiles)	78	18653ms	4344ms
	25km² (1855 Tiles)	74,2	18053ms	4992ms
	30km² (2340 Tiles)	78	36929ms	5533ms
	35km² (2784 Tiles)	79,54	51548ms	20574ms
	40km² (3087 Tiles)	77,17	80977ms	20886ms
Priority-Based	1km² (88 Tiles)	88	1798ms	690ms
	5km² (399 Tiles)	79,8	7787ms	2550ms
	10km² (756 Tiles)	75,6	31029ms	6503ms
	15km² (1248 Tiles)	83,2	33106ms	12645ms
	20km² (1560 Tiles)	78	64529ms[1]	19561ms
	25km² (1855 Tiles)	74,2	88887ms	25734ms
	30km² (2340 Tiles)	78	264914ms[2]	40080ms
	35km² (2784 Tiles)	79,54	456949ms[1]	61360ms[3]
	40km² (3087 Tiles)	77,17	463087ms[2]	86090ms

[1]Had a memory error, finished after clicking continue in browser dev tools.) [2](2x Memory error DNF) [3](1x Memory error DNF)

As can be seen from the results obtained in both machines, the regular method (send all requests at page load) provides faster map loading which is prone to less memory errors in the browser. On the other hand, the developed priority-based method lags significantly, especially in weaker machines which do not use dedicated graphics processing hardware. Even with capable hardware, the assigning of priorities, and sending requests with queue-based approach (loading each queue at once) were found to cause too much overhead for the browser, which results in degraded performance. As for the Tiles/km² variations, these can be attributed to the way the map is created based on real world geographical coordinates. Tiles that are cut in half when the user selects the area of operations are not considered and included, which makes the tile/ km² progression not linear and not proportional to the area.

Considering the previous results, additional implementation, and testing, would be required to optimize the loading of the map. For instance, one option would be to use what has been implemented and find a way to fix the memory errors and the low usability; while another option would be to revert to the previous implementation. While due to timing constraints and project priorities, the author was advised to not pursue with these developments, the following strategies were discussed:

- Instead of waiting for a queue to load, there should be a small limit (5) on how many tiles can load at once to not overload the browser with requests. This would cause a slowdown in the map loading, but more important areas would still be loaded first, and memory errors should be prevented.
- Reverting to the regular method is also possible, but it comes with the decision of prohibiting the user from interacting with the scene until the map is loaded, for example, with a loading screen. The loading time is dependent on the size of the scene.

5.3 Usability Improvements

This section compares the usability of the application before and after the development phase of the project. The focus is on the areas which were worked, and each development, to better highlight the improvements.

5.3.1 Collaboration

The biggest improvement in the collaboration feature is that it now provides instant feedback about which field is being edited and by which user. As the field is disabled, the users cannot edit a conflicting field, and accidental data loss is prevented. This behavior, combined with the fix of the bug which would overwrite data when a user joined a form makes this implementation of collaboration a much more robust system. Figure 60 shows the difference between versions of this feature.

Figure 60 – A showcase of the form collaboration improvements.

Another collaboration improvements that were implemented were the added visualization of users in every map view (simulation execution and editing), and the collaborative restart of the scenario execution in the scenario execution page. Both features can be seen in Figure 43 and Figure 58 respectively, previously shown in **Chapter 4 Build and Development.**

5.3.2 Scenario View

In the scenario view, many changes were implemented:

- The use of cheap to render polygons to substitute custom 3D models when they are not available, which improves initial loading times (excluding map loading).
- The highlighting of units on the map by utilizing the Babylon GUI, which allows users to quickly identify which team a unit belongs to by color coding.
- Displaying relevant information about the entity by utilizing the GUI, which allows the user to quickly identify where the entity is, and what they are.
- Addition of more movement modes in the scenario execution and editing views, which can be accessed through an AR headset, or the XR browser emulator.
    - Each user possesses their own local Babylon scene and camera, allowing both to view different perspectives.
    - These views also synchronize the movement of the units in execution mode, and the collaborative dragging and dropping in editing mode, allowing two users to view these actions from different perspectives.

The developments implemented in scenario views allow the user to optionally upload 3D models, removing the requirement to do so. With the implementation of GUI and default models, the user can more accurately and easily move entities, even if the 3D models are not available just yet for them to upload, allowing them more flexibility and choice on how to use the platform.

Figure 61 shows a comparison between a 3D model representation of an entity and its representation in the platform after all changes were implemented, to evidence the impact of these developments. As can be seen in the figure, if the model was not uploaded previously, nothing would appear on the map (not even GUI), difficulting the user's perception of the entity and where it was placed.

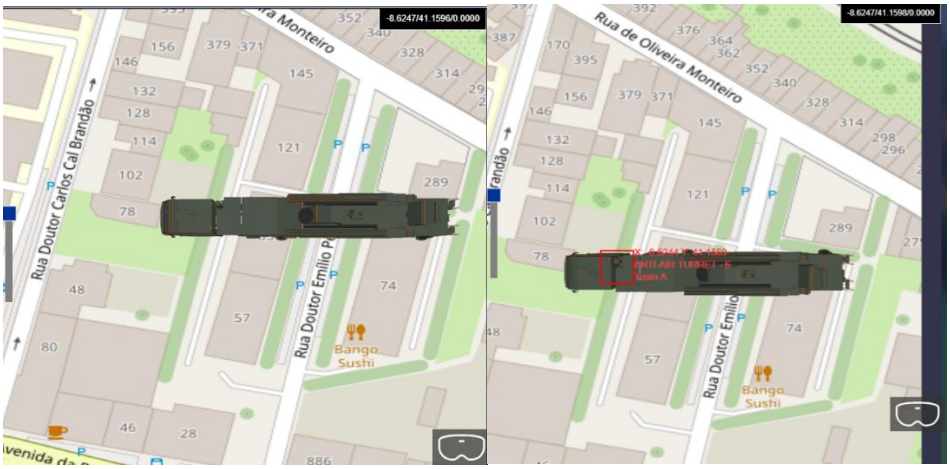


Figure 61 – A comparison between two models and the implemented GUI.

There are still some problems with this implementation, namely, with too many entities on the map, and in proximity to one another, the zoomed-out view is obstructed by the GUIs of the units in the map. This is due to the use of the Babylon Fullscreen GUI for these GUI elements, which scales with the distance of the camera to the objects. Figure 62 shows an example of this.

Figure 62 – The issue with utilizing the Fullscreen Babylon GUI.

This issue can be resolved by utilizing GUI on the entities meshes instead, which would require a reimplementation of the current GUI code.

Another change made to the scenario view is in AR, where all the previous changes are applied (the Babylon scene is the same), is the addition of two modes where the user can move and reposition the camera with the aid of readily available tools like mouse and keyboard.

In the previous implementation, the user would be stuck with the default controls that come with their AR device (if applicable) or be restricted to head movements and walking, which would make it impractical to use the rest of the application. From the point of view of allowing the user to move and explore the scenario, this implementation is evaluated as a success.

The users can also execute a simulation in real time, and through their individual local cameras can see alterations and movement in the Babylon scene, synchronously. This allows users to experience the simulation's execution by taking different perspectives as shown in Figure 63 (Aerial unit model from [95]).

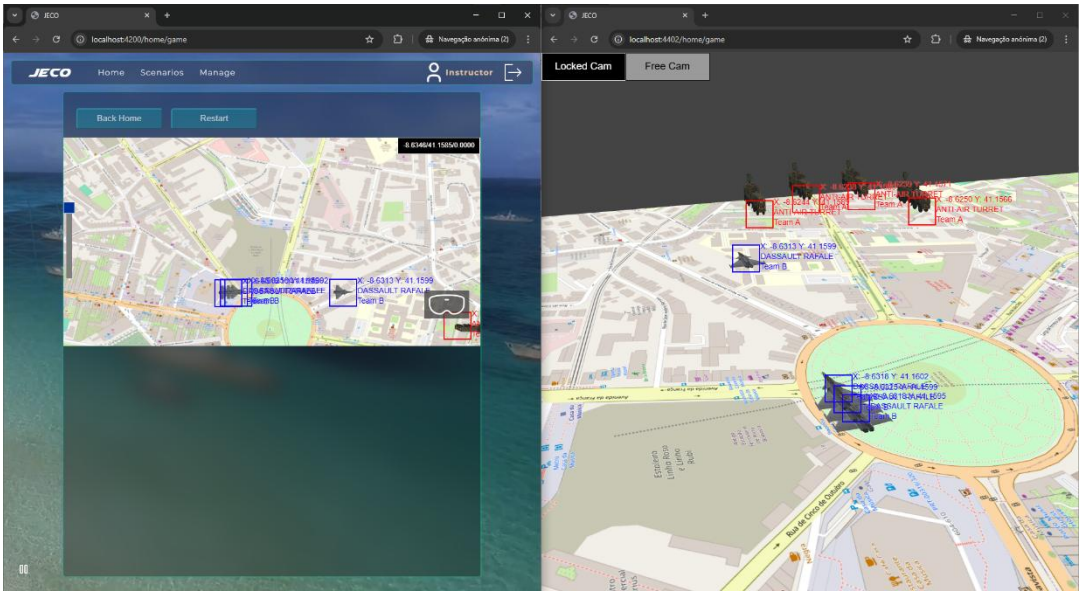


Figure 63 – An example of different perspectives from two users during simulation execution.

This synchronization allows users to correct and point out relevant information in real time, improving the pedagogic value of the platform and its functionalities.

6 Conclusion

This is the last chapter, which concludes the work. It details if the objectives of the dissertation were completed, what limitations the platform still exhibits, even after the developments, and finally, future work that can be done to mitigate these limitations.

This work had set out to optimize and improve SISTRADE's constructive military simulation platform. The main problems were identified, and research was done to try and find solutions to these problems, namely, the lack of usability in the scenario views, the blindness of the collaboration feature, and AR (Augmented Reality) component of the platform, and its lackluster performance when loading big and intensive scenarios with 3D models and big maps. While the research that was carried out could not be utilized in its entirety, it helped the author analyze adequate solutions to the platform's problems and implement them.

The problems with the map loading could not be fixed, but the scenario view and AR usability were improved drastically, as well as the collaboration component of the application. Some additional developments were also carried out at SISTRADE's request.

All these developments allowed the platform to be approved internally by SISTRADE after their subsequent evaluation.

6.1 Fulfilled Objectives

This section describes each objective and provides reasoning for its completion status. Table 15 shows the objectives of the work (previously defined in **Section 1.2.1 Objectives**), and their completion status.

Table 15 – An analysis of fulfilled objectives.

#	Objective	Completion
1	Improve the usability of the application's AR component and scenario views to allow it to be a more cohesive and immersive experience than what is currently implemented.	Completed
2	Improve the collaborative functions of the application by implementing user awareness.	Completed
3	Improve the performance of the application scenario rendering that is already implemented by SISTRADE.	Partially Completed

Objective 1 was marked as completed due to the implementation of a movement system that allows the user to move without necessitating the user to walk with an AR equipped device. This movement system allows the user to easily experience the AR component of the application and achieve a much more immersive experience that was not possible to be derived

from the previous views. It also reduces the limitations of the AR component, making it easily accessible even when the AR equipment is not calibrated to the environment or installed. An emulator can also be utilized to access this mode without the need for the headset.

Objective 2 was marked as implemented, as the collaboration awareness was implemented and most major collaboration problems and bugs corrected. Now the application shows and warns users that a third party is editing the form collaboratively, listing the users. The loss of data was also completely fixed, and fields now disable when a user is utilizing them, preventing problems with data being erased accidentally. Furthermore, the implemented solution is extensible and can be added to any pages that need it in the future, provided these future pages also have the collaboration feature implemented. This solution also implements cursor awareness in the Babylon.js scenes where synchronization is supported, allowing users to be aware of each other while editing units in a scenario or executing scenarios.

Objective 3 was marked as partially completed due to the poor performance observed in the previous chapter. This poor performance hinders the application's main functionality, severely prejudicing the application performance and usability. For these reasons this objective cannot be marked as fully completed. It was marked as partially completed due to the map now loading form the most optimal starting position (the center point of the user's view), and for the implementations done that allow the loading to be further refined. But the main objective of improving the performance could not be achieved.

6.2 Limitations

A limitation of the platform itself is the reliance on OSM for mapping tiles. The performance solution did not work as intended, and other solutions like it might never work if the OSM tiling server rate limits requests done too rapidly. Slower techniques might yield better results, improving usability at the cost of loading speed, as explained in the evaluation chapter.

Another limitation is that this application is web based. Even if the map could be loaded in correctly, for scenarios with a lot of surface area, the browser would have trouble rendering such a complex number of 3D elements (even if most of it is a plane). This does not account for other elements that might need to be loaded in the future, such as more complex models, or simulation-based calculations that must be done, for example, entity on entity interactions.

If the browser struggles with loading the 3D elements for the scenario, then these calculations cannot be done in the same runtime, or the browser will have trouble keeping up with the demand, even on machines with good processing power. This limitation currently restricts the platform to small scenarios, which might not be enough for today's conflicts or training.

A project limitation came from the limited time and tight deadline of the project's developments, as this resulted in the non-implementation of the Retrospectives requirement. This implementation would have given some more merit to the work, but it could not be started on time, as other developments took precedence over this new functionality.

6.3 Future Work

As a way of tackling the previous limitations, some suggestions covering different paths of future development were defined. As a major limitation stems from the application being built as a web application, two routes of development can be taken: the application remains a web app; or the application is transferred to a more robust platform, capable of implementing the required functionality in a better and more efficient way.

The main advantages of following the web application route are that most of the work is already done, and optimizations can be implemented to cover the rest of the implementations. A big disadvantage is that developers will always be fighting browser limitations, potentially resulting in implementations that cannot be finished, or cannot perform at the desired level.

Leaving web applications and focusing on a desktop application developed, for example, in a game engine such as unity, would allow developers to implement much of the required functionality such as entity interaction and map loading in a more compatible environment, which would fully utilize the computer's processing power. The disadvantage of this approach is that all development made on this platform will not be able to be reutilized, especially not the frontend developments.

If it is decided to stay on the web development route, future possible developments include:

- Re-implementation of map loading to a different technique or to utilize a loading screen.
- Implementation of the Retrospectives functionality.
- Implementation of an alternative to OSM map tiling, including local tiling servers or utilizing sources that allow the client to use tile caching.
- Implementation of further simulations (attacks, defensive structure, entity interactions).

If choosing to leave the web development route:

- Reimplementation of simulation functionality by utilizing a game engine (for example Unity) for simulation rendering.
- A full port of the application management interface (creation of users, scenarios, etc.) to the engine.
- Implementation of simulation game logic in the game engine (entity interactions).
    - Further implementation can include a user-facing scripting language for entities' actions in the field (instead of pre-defined actions).
- VR/AR support in the game engine (generally more robust).

References

[1] F. V. Mjelde, “Performance assessment of military teams in simulator and live exercises”, Accessed: Dec. 27, 2024. [Online]. Available: https://core.ac.uk/reader/225934740

[2] “Mixed Reality, the disruptive technology to increase exercise realism, improve live training efficiency and shorten exercise preparation,” *Mix. Real.*.

[3] “Despacho n.º 11171_2020 - Código de boas práticas e de conduta P.PORTO.”

[4] “IEEE_Reference_Guide.pdf.” Accessed: Dec. 27, 2024. [Online]. Available: http://journals.ieeeauthorcenter.ieee.org/wp-content/uploads/sites/7/IEEE_Reference_Guide.pdf

[5] K. Petersen, S. Vakkalanka, and L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015, doi: 10.1016/j.infsof.2015.03.007.

[6] M. J. Page *et al.*, “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews,” *BMJ*, vol. 372, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.

[7] “What is a PICOC? » CEBMa.” Accessed: Dec. 28, 2024. [Online]. Available: https://cebma.org/resources/frequently-asked-questions/what-is-a-picoc/

[8] B. Mills *et al.*, “ParaVerse: co-design of a parachute rehearsal and training virtual-reality enhanced simulator for the Australian Defence Force: combining a generative co-design framework and an agile approach to development,” *Virtual Real*, vol. 28, no. 4, Oct. 2024, doi: 10.1007/s10055-024-01053-5.

[9] K. O. Koumou and O. E. Isafiade, “Asset Management Trends in Diverse Settings Involving Immersive Technology: A Systematic Literature Review,” *IEEE Access*, vol. 12, pp. 141785–141813, 2024, doi: 10.1109/ACCESS.2024.3461548.

[10] K. A. Pollard *et al.*, “Level of immersion affects spatial learning in virtual environments: results of a three-condition within-subjects study with long intersession intervals,” *Virtual Real*, vol. 24, no. 4, pp. 783–796, Dec. 2020, doi: 10.1007/s10055-019-00411-y.

[11] H. Stefan, M. Mortimer, and B. Horan, “Evaluating the effectiveness of virtual reality for safety-relevant training: a systematic review,” *Virtual Real*, vol. 27, no. 4, pp. 2839–2869, Aug. 2023, doi: 10.1007/s10055-023-00843-7.

[12] H. A. Al-Jundi and E. Y. Tanbour, “A framework for fidelity evaluation of immersive virtual reality systems,” *Virtual Real*, vol. 26, no. 3, pp. 1103–1122, Sep. 2022, doi: 10.1007/s10055-021-00618-y.

[13] V. Mittal and A. Davidson, “Combining Wargaming With Modeling and Simulation to Project Future Military Technology Requirements,” *IEEE Trans. Eng. Manag.*, vol. 68, no. 4, pp. 1195–1207, Aug. 2021, doi: 10.1109/TEM.2020.3017459.

[14] J. Motejlek and E. Alpay, “Taxonomy of Virtual and Augmented Reality Applications in Education,” *IEEE Trans. Learn. Technol.*, vol. 14, no. 3, pp. 415–429, Jun. 2021, doi: 10.1109/TLT.2021.3092964.

[15] Z. Stefanidi, G. Margetis, S. Ntoa, and G. Papagiannakis, “Real-Time Adaptation of Context-Aware Intelligent User Interfaces, for Enhanced Situational Awareness,” *IEEE Access*, vol. 10, pp. 23367–23393, 2022, doi: 10.1109/ACCESS.2022.3152743.

[16] M. Shakeri, A. Sadeghi-Niaraki, and S.-M. Choi, “Augmented reality-based border management,” *Virtual Real*, vol. 26, no. 3, pp. 1123–1143, Sep. 2022, doi: 10.1007/s10055-021-00611-5.

[17] I. Virca, G. Bârsan, R. Oancea, and C. Vesa, “Applications of Augmented Reality Technology in the Military Educational Field,” *Land Forces Acad. Rev.*, vol. 26, no. 4, pp. 337–347, Dec. 2021, doi: 10.2478/raft-2021-0044.
[18] L. Podoletz, M. McGill, D. McIlhatton, J. Marshall, N. Healy, and L. M. Tanczer, “A Critical Review of Virtual and Extended Reality Immersive Police Training: Application Areas, Benefits & Vulnerabilities,” in *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology*, in VRST ’24. New York, NY, USA: Association for Computing Machinery, 2024. doi: 10.1145/3641825.3687707.
[19] C. Gomes, T. Guedes, and A. Esteves, “A First Exploration on the Use of Head-Mounted Augmented Reality in the Context of the Portuguese Military,” *Proc ACM Hum-Comput Interact*, vol. 7, no. MHCI, Sep. 2023, doi: 10.1145/3604239.
[20] N. Pellas, S. Mystakidis, and I. Kazanidis, “Immersive Virtual Reality in K-12 and Higher Education: A systematic review of the last decade scientific literature,” *Virtual Real*, vol. 25, no. 3, pp. 835–861, Sep. 2021, doi: 10.1007/s10055-020-00489-9.
[21] L. T. De Paolis and V. De Luca, “The effects of touchless interaction on usability and sense of presence in a virtual environment,” *Virtual Real*, vol. 26, no. 4, pp. 1551–1571, Dec. 2022, doi: 10.1007/s10055-022-00647-1.
[22] “Just in Time Training: Implementing Strategies for Your Team,” 360Learning. Accessed: Dec. 30, 2024. [Online]. Available: https://360learning.com/blog/just-in-time-training/
[23] J. D. Fletcher and P. R. Chatelier, “An Overview of Military Training:,” Defense Technical Information Center, Fort Belvoir, VA, Aug. 2000. doi: 10.21236/ADA408439.
[24] D. Dutta, “Simulation in Military Training: Recent Developments,” *Def. Sci. J.*, vol. 49, no. 3, pp. 275–285, Jan. 1999, doi: 10.14429/dsj.49.3840.
[25] W. Choi, K. Yu, B. J. Park, S. Kang, and J. Lee, “Study on L-V-C (Live-Virtual-Constructive) Interoperation for the National Defense M&S (Modeling & Simulation),” in *2008 International Conference on Information Science and Security (ICISS 2008)*, Jan. 2008, pp. 128–133. doi: 10.1109/ICISS.2008.39.
[26] M. Čolić, “LVC Concept of the Military Simulation Systems in Support of the Education and Training of the Armed Forces,” in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, May 2024, pp. 1283–1288. doi: 10.1109/MIPRO60963.2024.10569642.
[27] L. Martin, “Advancing F-35 Pilot Training.” Accessed: Jan. 01, 2025. [Online]. Available: https://www.f35.com/f35/news-and-features/Advancing-F-35-Pilot-Training.html
[28] “F-35 Lightning II Training,” Lockheed Martin. Accessed: Jan. 01, 2025. [Online]. Available: https://www.lockheedmartin.com/en-us/products/f-35-lightning-ii-training-systems.html
[29] “Digital Combat Simulator | DCS World | Combat Simulator.” Accessed: Jan. 01, 2025. [Online]. Available: https://www.digitalcombatsimulator.com/en/
[30] N. K. Thapliyal and D. J. Chouhan, “MILITARY SIMULATION TECHNOLOGIES,” *J. Southwest Jiaotong Univ.*, vol. 59, no. 2, Art. no. 2, Oct. 2024.
[31] K. C. Trott, “Simulation Development for Dynamic Situation Awareness and Prediction:,” Defense Technical Information Center, Fort Belvoir, VA, Sep. 2005. doi: 10.21236/ADA440082.
[32] G. Šimić, “Constructive Simulation as a Collaborative Learning Tool in Education and Training of Crisis Staff”.
[33] “VBS4 | BISim,” Bohemia Interactive Simulations. Accessed: Mar. 03, 2025. [Online]. Available: https://bisimulations.com/products/vbs4
[34] “Node.js — About Node.js®.” Accessed: Jan. 02, 2025. [Online]. Available: https://nodejs.org/en/about
[35] “Express - Node.js web application framework.” Accessed: Jan. 02, 2025. [Online]. Available: https://expressjs.com/

[36] “Node.js — Profiling Node.js Applications.” Accessed: Mar. 05, 2025. [Online]. Available: https://nodejs.org/en/learn/getting-started/profiling
[37] M. Selakovic and M. Pradel, “Performance issues and optimizations in JavaScript: an empirical study,” in *Proceedings of the 38th International Conference on Software Engineering*, Austin Texas: ACM, May 2016, pp. 61–72. doi: 10.1145/2884781.2884829.
[38] L. Song and S. Lu, “Performance Diagnosis for Inefficient Loops,” in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, May 2017, pp. 370–380. doi: 10.1109/ICSE.2017.41.
[39] “Angular - What is Angular?” Accessed: Jan. 02, 2025. [Online]. Available: https://v17.angular.io/guide/what-is-angular
[40] “Angular Dev Tools.” Accessed: Mar. 06, 2025. [Online]. Available: https://angular.dev/tools/devtools
[41] “Introduction to Lighthouse,” Chrome for Developers. Accessed: Mar. 06, 2025. [Online]. Available: https://developer.chrome.com/docs/lighthouse/overview
[42] A. Ramos, “Advanced Techniques for Angular Performance Enhancement: Strategies for Optimizing Rendering, Reducing Latency, and Improving User Experience in Modern Web Applications,” *Int. Humanit. Appl. Sci. J.*, vol. 4, Apr. 2024.
[43] “Lighthouse Scoring calculator.” Accessed: Mar. 08, 2025. [Online]. Available: https://googlechrome.github.io/lighthouse/scorecalc/#FCP=0&SI=0&FMP=4000&TTI=7300&FCI=6500&LCP=0&TBT=0&CLS=0&device=desktop&version=10
[44] “What’s new in Lighthouse 10 | Blog,” Chrome for Developers. Accessed: Mar. 08, 2025. [Online]. Available: https://developer.chrome.com/blog/lighthouse-10-0
[45] “Total Blocking Time (TBT) | Articles,” web.dev. Accessed: Mar. 08, 2025. [Online]. Available: https://web.dev/articles/tbt
[46] “Interaction to Next Paint (INP) | Articles,” web.dev. Accessed: Mar. 08, 2025. [Online]. Available: https://web.dev/articles/inp
[47] “The most effective ways to improve Core Web Vitals | Articles | web.dev.” Accessed: Mar. 08, 2025. [Online]. Available: https://web.dev/articles/top-cwv
[48] “Largest Contentful Paint (LCP) | Articles | web.dev.” Accessed: Mar. 08, 2025. [Online]. Available: https://web.dev/articles/lcp
[49] “Optimize Largest Contentful Paint | Articles | web.dev.” Accessed: Mar. 08, 2025. [Online]. Available: https://web.dev/articles/optimize-lcp
[50] “Debouncing and Throttling in JavaScript,” DEV Community. Accessed: Mar. 08, 2025. [Online]. Available: https://dev.to/vedanthb/debouncing-and-throttling-in-javascript-10oo
[51] “Angular Components.” Accessed: Mar. 06, 2025. [Online]. Available: https://angular.dev/guide/components
[52] “Angular NgModules.” Accessed: Mar. 06, 2025. [Online]. Available: https://angular.dev/guide/ngmodules/overview
[53] “Angular Services.” Accessed: Mar. 06, 2025. [Online]. Available: https://angular.dev/tutorials/first-app/09-services
[54] “Angular Runtime Performance.” Accessed: Mar. 06, 2025. [Online]. Available: https://angular.dev/best-practices/runtime-performance
[55] “Angular Change Detection.” Accessed: Mar. 08, 2025. [Online]. Available: https://angular.dev/api/core/ChangeDetectionStrategy
[56] “Angular Change Detection Usage.” Accessed: Mar. 08, 2025. [Online]. Available: https://angular.dev/api/core/ChangeDetectorRef?tab=usage-notes
[57] “AIX 7.3.” Accessed: Mar. 08, 2025. [Online]. Available: https://www.ibm.com/docs/en/aix/7.3?topic=performance-memory-leaking-programs

[58] “Babylon.js: Powerful, Beautiful, Simple, Open - Web-Based 3D At Its Best,” Babylon.js. Accessed: Jan. 02, 2025. [Online]. Available: https://www.babylonjs.com
[59] “MultiViews Part 1 | Babylon.js Documentation.” Accessed: Mar. 08, 2025. [Online]. Available: https://doc.babylonjs.com/features/featuresDeepDive/cameras/multiViewsPart1
[60] “Decimate Modifier - Blender 4.3 Manual.” Accessed: Jan. 02, 2025. [Online]. Available: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html
[61] “Levels of Detail (LOD) | Babylon.js Documentation.” Accessed: Jan. 02, 2025. [Online]. Available: https://doc.babylonjs.com/features/featuresDeepDive/mesh/LOD
[62] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*. Elsevier, 2002.
[63] “LOD Document,” S.Thomas Creations. Accessed: Jan. 02, 2025. [Online]. Available: https://moderndynamics.wordpress.com/year-2/object-orientated-design/lod-document/
[64] “Progressively Load .glTF Files | Babylon.js Documentation.” Accessed: Jan. 02, 2025. [Online]. Available: https://doc.babylonjs.com/features/featuresDeepDive/importers/glTF/progressiveglTFLoad
[65] “Serving Tiles,” Switch2OSM. Accessed: Mar. 09, 2025. [Online]. Available: https://switch2osm.org/serving-tiles/
[66] “About OpenStreetMap - OpenStreetMap Wiki.” Accessed: Jan. 02, 2025. [Online]. Available: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap
[67] “Tile Usage Policy.” Accessed: Mar. 09, 2025. [Online]. Available: https://operations.osmfoundation.org/policies/tiles/
[68] I. Pantazopoulos and S. Tzafestas, “Occlusion Culling Algorithms: A Comprehensive Survey,” *J. Intell. Robot. Syst.*, vol. 35, pp. 123–156, Oct. 2002, doi: 10.1023/A:1021175220384.
[69] “Occlusion Queries | Babylon.js Documentation.” Accessed: Mar. 09, 2025. [Online]. Available: https://doc.babylonjs.com/features/featuresDeepDive/occlusionQueries
[70] O. Sohaib and K. Khan, “Integrating usability engineering and agile software development: A literature review,” in *2010 International Conference On Computer Design and Applications*, Jun. 2010, pp. V2-32-V2-38. doi: 10.1109/ICCDA.2010.5540916.
[71] “ISO 9241-11:2018(en), Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts.” Accessed: Jan. 03, 2025. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en
[72] D. Gavgiotaki, S. Ntoa, G. Margetis, K. C. Apostolakis, and C. Stephanidis, “Gesture-based Interaction for AR Systems: A Short Review,” in *Proceedings of the 16th International Conference on PErvasive Technologies Related to Assistive Environments*, Corfu Greece: ACM, Jul. 2023, pp. 284–292. doi: 10.1145/3594806.3594815.
[73] “Yjs Shared Editing.” Accessed: Jan. 03, 2025. [Online]. Available: https://yjs.dev/
[74] “Awareness & Presence | Yjs Docs.” Accessed: Jan. 03, 2025. [Online]. Available: https://docs.yjs.dev/getting-started/adding-awareness
[75] “yjs/README.md at master · yjs/yjs,” GitHub. Accessed: Mar. 11, 2025. [Online]. Available: https://github.com/yjs/yjs/blob/master/README.md
[76] “y-websocket | Yjs Docs.” Accessed: Mar. 11, 2025. [Online]. Available: https://docs.yjs.dev/ecosystem/connection-provider/y-websocket
[77] “Y.Doc | Yjs Docs.” Accessed: Mar. 11, 2025. [Online]. Available: https://docs.yjs.dev/api/y.doc

[78] “Introduction | Yjs Docs.” Accessed: Mar. 11, 2025. [Online]. Available: https://docs.yjs.dev
[79] “Shared Types | Yjs Docs.” Accessed: Mar. 11, 2025. [Online]. Available: https://docs.yjs.dev/api/shared-types
[80] “Awareness & Presence | Yjs Docs.” Accessed: Mar. 11, 2025. [Online]. Available: https://docs.yjs.dev/getting-started/adding-awareness
[81] “Awareness | Yjs Docs.” Accessed: Mar. 11, 2025. [Online]. Available: https://docs.yjs.dev/api/about-awareness
[82] J. Nielsen, “How I Developed the 10 Usability Heuristics,” UX Tigers. Accessed: Mar. 13, 2025. [Online]. Available: https://www.uxtigers.com/post/usability-heuristics-history
[83] I. Börsting, M. Heikamp, M. Hesenius, W. Koop, and V. Gruhn, “Software Engineering for Augmented Reality - A Research Agenda,” *Proc. ACM Hum.-Comput. Interact.*, vol. 6, no. EICS, pp. 1–34, Jun. 2022, doi: 10.1145/3532205.
[84] B. Shneiderman, C. Plaisant, M. Cohen, S. M. Jacobs, and N. Elmqvist, *Designing the User Interface: Strategies for Effective Human-computer Interaction*. in Global edition. Pearson, 2017. [Online]. Available: https://books.google.pt/books?id=nhDYtQEACAAJ
[85] A. Dix, *Human-computer Interaction*. Pearson Education, 2004.
[86] “Ben Shneiderman.” Accessed: Mar. 15, 2025. [Online]. Available: https://www.cs.umd.edu/users/ben/goldenrules.html
[87] F. K. Mazumder and U. K. Das, “USABILITY GUIDELINES FOR USABLE USER INTERFACE”.
[88] J. Nielsen, “How I Developed the 10 Usability Heuristics,” UX Tigers. Accessed: Mar. 15, 2025. [Online]. Available: https://www.uxtigers.com/post/usability-heuristics-history
[89] “User interface guidelines: 10 essential rules to follow - UX Design Institute.” Accessed: Mar. 15, 2025. [Online]. Available: https://www.uxdesigninstitute.com/blog/10-user-interface-guidelines/
[90] “Using a Docker container,” Switch2OSM. Accessed: Mar. 29, 2025. [Online]. Available: https://switch2osm.org/serving-tiles/using-a-docker-container/
[91] “Raster tile providers - OpenStreetMap Wiki.” Accessed: Mar. 29, 2025. [Online]. Available: https://wiki.openstreetmap.org/wiki/Raster_tile_providers
[92] *Pantsir-S1 - 3D model by weap22*, (Mar. 15, 2022). Accessed: May 21, 2025. [Online Video]. Available: https://sketchfab.com/models/acf0e04474f44b5bba1226dee4087a66/embed?autostart=1
[93] vtieto, “Using WebXR with Windows Mixed Reality - Mixed Reality.” Accessed: May 21, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/javascript/webxr-overview
[94] “Revolutionizing WebXR Development with the Immersive Web Emulator.” Accessed: May 21, 2025. [Online]. Available: https://developers.meta.com/horizon/blog/webxr-development-immersive-web-emulator/
[95] *Dassault Rafale - Download Free 3D model by andertan*, (Aug. 30, 2024). Accessed: Jun. 21, 2025. [Online Video]. Available: https://sketchfab.com/models/fabb8472bc2e413282c80406b13ff1a7/embed?autostart=1

Annex I – Planning Annexes

In this annex, the WBS dictionary provides explanations for all the project phases, deliverables and work units, along with the WBS diagram for easier viewing, and finally the planning Gantt chart. These planning annexes were not followed in the project but were built before the project as a guide, mainly for the PREPD curricular unit.

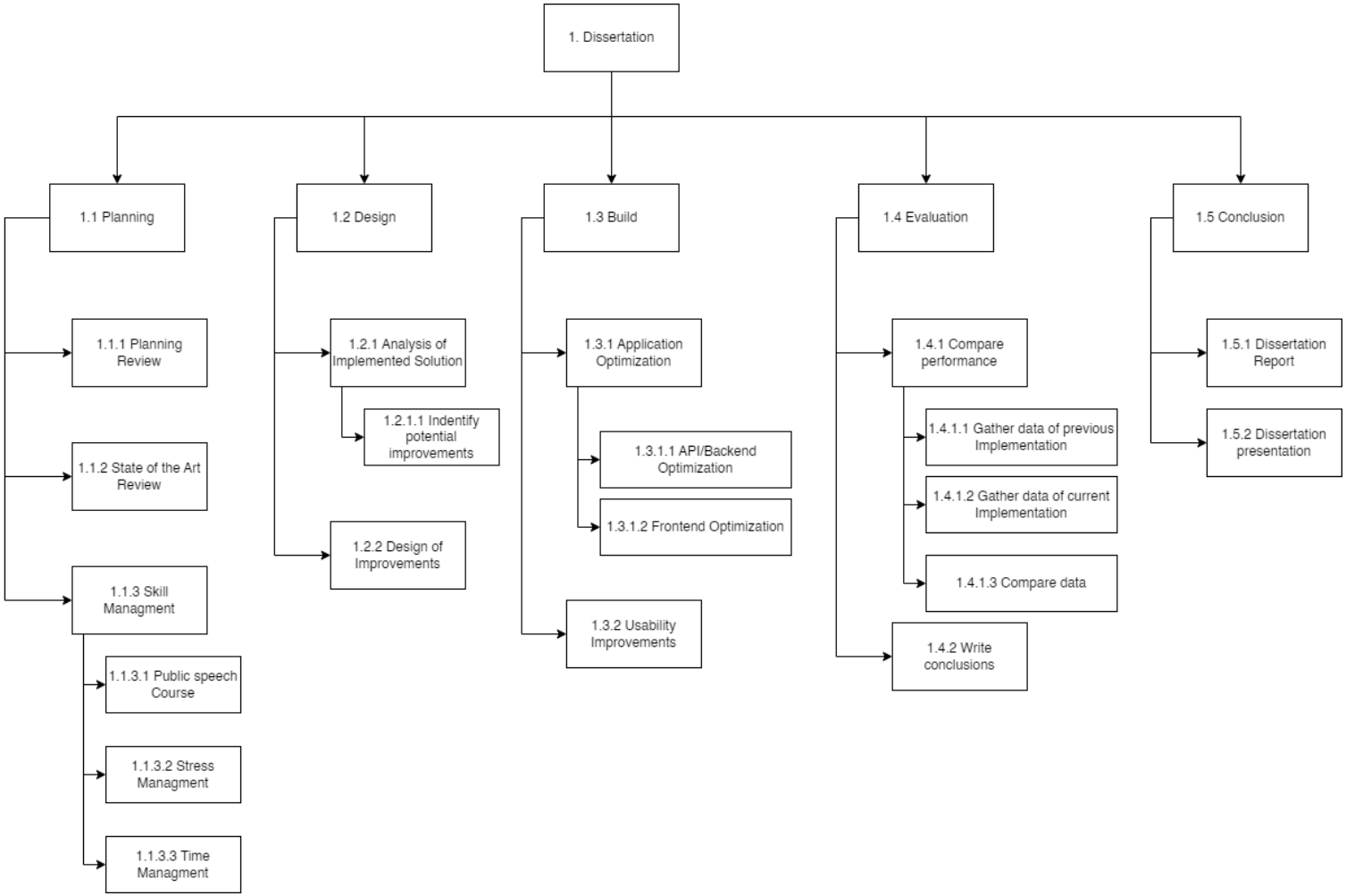
WBS Dictionary

TASK	TYPE	COMPLEMENTARY DESCRIPTION
1.1 - Planning	Phase	The state-of-the-art and project planning will be reviewed ahead of time to ensure the most up to date planning and state-of-the-art. This phase will be concluded after both reviews have been approved by the Supervisor and Advisor.
1.1.1 – Planning Review	Deliverable	The project's planning will be revised to include the dates mentioned in the Monitoring section of the Project Planning chapter.
1.1.2 – State of the Art Review	Deliverable	The state of the art will be reviewed lightly to ensure it is of a good quality to aid further sections of the project.
1.1.3 – Skill Management	Deliverable	This deliverable will not be "delivered" in a sense, but it is included for the project's planning chapter.
1.1.3.1 – Public Speech Course	Work unit	Explained in the Skill Management section.
1.1.3.2 – Stress Management	Work unit	Explained in the Skill Management section.
1.1.3.3 – Time Management	Work unit	Explained in the Skill Management section.
1.2 - Design	Phase	This phase will comprise all the design aspects of the project, from the analysis of the already implemented solution to the identification of potential optimization and usability improvements, and how they'll be implemented into the system's architecture.
1.2.1 – Analysis of Implemented Solution	Deliverable	The existing solution's design will be analyzed and from the methods for optimization found in the State of the Art, will be optimized with various changes in architecture, API interfaces. These will be reflected in the revised solution design.

1.2.1.1 – Identify potential improvements	Work unit	Analyze and identify what parts of the application's interfaces and processes can be optimized.
1.2.2 – Design of Improvements	Deliverable	Design of the improved solution.
1.3 - Build	Phase	This Phase aims to improve the already existing solution with the design that was developed in the previous phase.
1.3.1 – Application Optimization	Deliverable	This deliverable contains all the tasks pertaining to the application's optimization, be it fronted processes or backend processes.
1.3.1.1 – API/Backend Optimization	Work unit	This work unit focuses on the development of API optimizations.
1.3.1.2 – Frontend Optimization	Work unit	This work unit focuses on the development of frontend optimizations.
1.3.2 – Usability Improvements	Deliverable	This deliverable focuses on the development of usability improvements and collaboration improvements for the Augmented Reality experience.
1.4 - Evaluation	Phase	In this phase, the application's developments will be evaluated, namely, the map loading performance, while justifications will be given as to what evaluation methods are used.
1.4.1 – Compare Performance	Deliverable	Various comparisons will be made, namely with the speed of both application versions.
1.4.1.1 – Gather data of previous implementation	Work unit	Data gathering of the previous implementation for comparison.
1.4.1.2 – Gather data of current implementation	Work unit	Data gathering of current implementation for comparison.
1.4.1.3 – Compare data	Work unit	Data comparison between the before and after, to see if there is any appreciable difference between implementations.
1.4.2 – Write conclusions	Deliverable	Conclusions based on the observations made in the previous Work unit.
1.5 - Conclusion	Phase	This phase signifies the conclusion of the project, where the dissertation report and the presentation are delivered.
1.5.1 – Dissertation Report	Deliverable	The dissertation report will be written alongside the development of the project, but will be delivered here, in the conclusion phase.

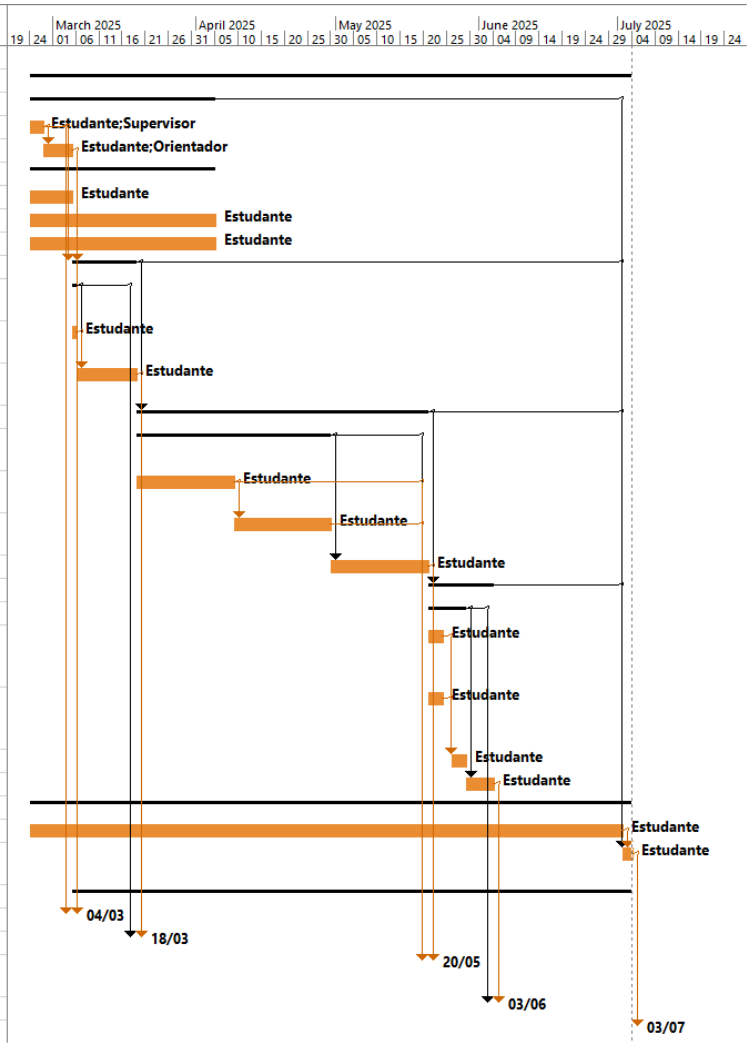
1.5.2 – Dissertation Presentation	Deliverable	The dissertation's presentation will be completed and delivered by this time. It will go through all project phases and explain what was done in each phase.

WBS Diagram


1. Dissertation
1.1 Planning
1.2 Design
1.3 Build
1.4 Evaluation
1.5 Conclusion
1.1.1 Planning Review
1.1.2 State of the Art Review
1.1.3 Skill Managment
1.1.3.1 Public speech Course
1.1.3.2 Stress Managment
1.1.3.3 Time Managment
1.2.1 Analysis of Implemented Solution
1.2.1.1 Indentify potential improvements
1.2.2 Design of Improvements
1.3.1 Application Optimization
1.3.1.1 API/Backend Optimization
1.3.1.2 Frontend Optimization
1.3.2 Usability Improvements
1.4.1 Compare performance
1.4.1.1 Gather data of previous Implementation
1.4.1.2 Gather data of current Implementation
1.4.1.3 Compare data
1.4.2 Write conclusions
1.5.1 Dissertation Report
1.5.2 Dissertation presentation

Planning Gantt Chart

ID	Task Mode	WBS	Task Name	Duration	Start	Finish	Predecessors
1							
2		**1**	**Dissertation**	**94 days**	**Mon 24/02/25**	**Thu 03/07/25**	
3		**1.1**	**Planning**	**30 days**	**Mon 24/02/25**	**Fri 04/04/25**	
4		1.1.1	Planning Review	3 days	Mon 24/02/25	Wed 26/02/25	
5		1.1.2	State of the Art Review	4 days	Thu 27/02/25	Tue 04/03/25	4
6		**1.1.3**	**Skill Managment**	**30 days**	**Mon 24/02/25**	**Fri 04/04/25**	
7		1.1.3.1	Public Speech Cours	7 days	Mon 24/02/25	Tue 04/03/25	
8		1.1.3.2	Stress Managment	30 days	Mon 24/02/25	Fri 04/04/25	
9		1.1.3.3	Time Managment	30 days	Mon 24/02/25	Fri 04/04/25	
10		**1.2**	**Design**	**10 days**	**Wed 05/03/25**	**Tue 18/03/25**	**4;5**
11		**1.2.1**	**Analysis of implemented solution**	**1 day**	**Wed 05/03/25**	**Wed 05/03/25**	
12		1.2.1.1	Identify potential improvements	1 day	Wed 05/03/25	Wed 05/03/25	
13		1.2.2	Design of Improvements	9 days	Thu 06/03/25	Tue 18/03/25	11;12
14		**1.3**	**Build**	**45 days**	**Wed 19/03/25**	**Tue 20/05/25**	**10**
15		**1.3.1**	**Application Optimization**	**30 days**	**Wed 19/03/25**	**Tue 29/04/25**	
16		1.3.1.1	API / Backend Optimization	15 days	Wed 19/03/25	Tue 08/04/25	
17		1.3.1.2	Frontend Optimization	15 days	Wed 09/04/25	Tue 29/04/25	16
18		1.3.2	Usablity Improvements	15 days	Wed 30/04/25	Tue 20/05/25	15
19		**1.4**	**Evaluation**	**10 days**	**Wed 21/05/25**	**Tue 03/06/25**	**14**
20		**1.4.1**	**Compare Performance**	**6 days**	**Wed 21/05/25**	**Wed 28/05/25**	
21		1.4.1.1	Gather data of previous implementation	3 days	Wed 21/05/25	Fri 23/05/25	
22		1.4.1.2	Gather data of current implementation	3 days	Wed 21/05/25	Fri 23/05/25	
23		1.4.1.3	Compare data	3 days	Mon 26/05/25	Wed 28/05/25	21;22
24		1.4.2	Write conclusions	4 days	Thu 29/05/25	Tue 03/06/25	20
25		**1.5**	**Conclusion**	**94 days**	**Mon 24/02/25**	**Thu 03/07/25**	
26		1.5.1	Dissertation Report	92 days	Mon 24/02/25	Tue 01/07/25	
27		1.5.2	Dissertation presentation	2 days	Wed 02/07/25	Thu 03/07/25	**19;3;10;14;26**
28		**1.6**	**Milestones**	**87 days**	**Tue 04/03/25**	**Thu 03/07/25**	
29		1.6.1	Reviews Complete	0 days	Tue 04/03/25	Tue 04/03/25	4;5
30		1.6.2	Design Finished	0 days	Tue 18/03/25	Tue 18/03/25	11;13
31		1.6.3	Application Features Developed	0 days	Tue 20/05/25	Tue 20/05/25	15;16;17;18
32		1.6.4	Performance Evaluatec	0 days	Tue 03/06/25	Tue 03/06/25	20;24
33		1.6.5	Final Presentation	0 days	Thu 03/07/25	Thu 03/07/25	27



Annex II – Analysis Annexes

This annex shows the full domain model of the previously developed application.