FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Web-based System for Visualizing Engineering Tasks on Multi-LOD Large-scale CAD Models

Ricardo Amaral Nunes

Mestrado em Engenharia Informática

Supervisor: Daniel Castro Silva

Supervisor at CERN: Fernando Baltasar dos Santos Pedrosa

July 28, 2023

A Web-based System for Visualizing Engineering Tasks on Multi-LOD Large-scale CAD Models

Ricardo Amaral Nunes

Mestrado em Engenharia Informática

Approved in oral examination by the committee:

July 28, 2023

Abstract

At CERN, many experiments are performed daily on several machines. For the machines to work correctly, they need regular maintenance and check-ups. These maintenance tasks can take weeks to complete and use different subsections of a machine. Scientists and engineers must consider these different variables when planning future experiments. This process takes precious time of their day and makes them focus more on trivial assignments than analyzing the results of their experiments and planning new ones. Moreover, if the tasks are not planned correctly, e.g., tasks that can't overlap do indeed overlap, it can delay future experiments and consume even more of the CERN engineers' limited time.

To solve this problem, we propose a 3D visualization system that displays the machine with the maintenance tasks for a given date. The developed system comprises two main parts: a preprocessing pipeline that converts and simplifies the accelerator model to the glTF format and a 3D visualization tool that displays the model, changes it according to the users' input, and provides the needed task information. The preprocessing pipeline uses Unreal Engine to convert the models and remove unnecessary details and Simplygon to perform the triangle decimation. The 3D visualization system is composed of six components, each one serving a main purpose, such as connecting to the API and managing the GUI. The application retrieves the task data from the CERN Scheduling Database through the Java API. Using this data, the model sections are colored according to the tasks, and the user navigates through the models and gets more detailed information about them by clicking on the section, which displays the detailed version of the model or the task information menu. We observed that this method of dividing the information into levels and letting the user choose the level of granularity of the displayed information is a quick and efficient method of displaying information. The evaluation of the tool was performed by conducting a controlled experiment and measuring loading times and performance. The results of the controlled experiment were highly positive, with virtually all participants preferring the 3D visualization tool to the existing 2D tool. Moreover, we obtained an average loading time for the detailed models of 2 seconds, which was hidden by the preview models, which had a loading time of only 0.5 seconds. The system maintains a stable FPS of 30, only dropping when a new model is loaded, and a time needed to render the next frame of 40ms. Besides, using Simplygon, we obtained a reduction of around 80% in the number of vertices for both types of models.

Keywords: 3D Visualization System, 3D Model Simplification, 3D Model Conversion, Task Scheduling, Multi-LOD 3D Model Visualization

Resumo

No CERN, são realizadas múltiplas experiências são realizadas diariamente em diversas máquinas. Para que as máquinas funcionem corretamente, é necessário fazer manutenção regularmente. Estas tarefas de manutenção podem levar semanas para serem concluídas e utilizar diferentes partes da máquina. Os cientistas e engenheiros devem considerar estas diferentes variáveis no planeamento de experiências futuras. Esse processo consome tempo precioso do dia a dia dos funcionários do CERN e faz com que se concentrem mais em tarefas mundanas do que na análise dos resultados e planeamento de futuras experiências. Além disso, se as tarefas não forem planeadas corretamente, isto pode atrasar experiências futuros e consumir ainda mais o tempo limitado dos engenheiros do CERN.

Para resolver esse problema, propomos um sistema de visualização 3D que exibe a máquina com as tarefas de manutenção previstas para uma determinada data. O sistema desenvolvido é composto por duas partes principais: uma pipeline de pré-processamento que converte e simplifica o modelo para o formato glTF e uma ferramenta de visualização 3D que exibe o modelo, altera-o de acordo com o input dos utilizadores e fornece as informações sobre as tarefas necessárias. O pipeline de pré-processamento utiliza a ferramenta Unreal Engine para converter os modelos e remover detalhes desnecessários, e a ferramenta Simplygon para realizar a decimação de triângulos. Obtivemos uma redução de 80.0% para o modelo global e uma redução de 80.2% para os modelos detalhados. O sistema de visualização 3D é composto por seis componentes, cada um com um objetivo específico, como conectar-se à API e gerir a interface gráfica. A aplicação obtém os detalhes das tarefas na base de dados do CERN através da API feita em Java. Com estes dados, as seções do modelo são coloridas de acordo com as tarefas, e o utilizador navega pelos modelos e obtém informações mais detalhadas sobre eles ao clicar na seção, o que exibe a versão detalhada do modelo ou um menu contendo mais informações sobre a tarefa. Observamos que este método de dividir as informações em vários níveis de detalhe e permitir que o utilizador escolha o nível de granularidade pretendido das informações exibidas é um método rápido e eficiente para exibir a informação das tarefas. A avaliação da ferramenta foi realizada através de uma experiência controlada e através da medição dos tempos de carregamento e do desempenho. Os resultados da experiência foram altamente positivos, com praticamente todos os participantes preferindo a ferramenta de visualização 3D em relação à ferramenta 2D existente. Além disso, obtivemos um tempo médio de carregamento para os modelos detalhados de 2 segundos, que foi ocultado pelos modelos de visualização prévia, que têm um tempo de carregamento de apenas 0,5 segundos. O sistema mantém um FPS estável de 30, apenas diminuindo quando um novo modelo é carregado, e necessita cerca de 40 ms para renderizar o próximo frame. Além disso, obtivemos uma redução de cerca de 80% em termos de vértices usando Simplygon.

Palavras-chave: Sistema de Visualização 3D, Simplificação de Modelos 3D, Conversão de Modelos 3D, Planeamento de Tarefas, Visualização 3D de Modelos com Múltiplos Níveis de Detalhe

Acknowledgments

To my parents and family, which provided me with the support and love needed to go through this last few years and become an engineer.

To the family I chose, my friends, for their companionship, laughter, and words of encouragement that have made this challenging journey more enjoyable and meaningful. Skål!

To CERN and Fernando Pedrosa for this incredible opportunity and for fulfilling my childhood dream.

Last but not least, I would like to thank my supervisor, Daniel Castro Silva, for his guidance, expertise, and support throughout the course of my thesis.

Ricardo Amaral Nunes

"The real danger is not that computers will begin to think like men,
but that men will begin to think like computers."

Sydney J. Harris

Contents

List of Figures

List of Tables

Abbreviations and Symbols

2D	Two-Dimensions
3D	Three-Dimensions
ANOVA	ANalysis Of VAriance
API	Application Programming Interface
ASCII	American Standard Code For Information Interchange
ASP.NET	Active Server Pages Network Enabled Technologies
AVG	Average
BDS	Building Description Systems
BIM	Building Information Modeling
CAD	Computer-Aided Design
CDN	Content Delivery Network
CERN	European Organization for Nuclear Research
COLLADA	Collaborative Design Activity
CPU	Central Processing Unit
DAE	Digital Asset Exchange
DB	Database
DCC	Digital Content Creation
DOM	Document Object Model
DXF	Drawing Exchange Format
FBX	Filmbox
FPS	Frames Per Second
GIS	Geographic Information System
GUI	Graphical User Interface
glTF	Graphics Language Transmission Format
glb	Binary Graphics Language Transmission Format
GPL	General Public License
HTML	HyperText Markup Language
HCI	Human-Computer Interaction
IDE	Integrated Development Environment
IFS	Industry Foundation Classes
IEC	International Electrotechnical Commission
IGES	Initial Graphics Exchange Specification
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LHC	Large Hadron Collider
LIDAR	LIght Detection And Ranging

LOD	Level of Detail
MIT	Massachusetts Institute of Technology
MRI	Magnetic Resonance Imaging
MS	Microsoft
MTL	Material Template Library
NASA	National Aeronautics and Space Administration
NPM	Node Package Manager
NWW	NASA Java World Wind
OBJ	Wavefront Object
OSG	Open Scene Graph
PCI	Pavement Condition Index
PDM	Product data management
PNG	Portable Network Graphics
PS	Proton Synchrotron
PUTQ	Purdue Usability Testing Questionnaire
QUIS	Questionnaire for User Interface Satisfaction
RAM	Random Access Memory
SDK	Software Development Kit
SPS	Super Proton Synchrotron
SQL	Structured Query Language
STEP	Standard for the Exchange of Product Data
STL	Standard Triangle Language or Standard Tessellation Language
SUMI	Software Usability Measurement Inventory
SUS	System Usability Scale
UE	Unreal Engine
UML	Unified Modeling Language
UNESCO	United Nations Educational, Scientific and Cultural Organization
URL	Uniform Resource Locator
USE	Usefulness,Satisfaction, and Ease of Use
VR	Virtual Reality
VRML	Virtual Reality Modeling Language
VSG	Vulkan Scene Graph
VTK	Visualization Toolkit
X3D	Extensible 3D
XML	Extensible Markup Language

Chapter 1

Introduction

Information visualization is the process of organizing and presenting data so the reader can interpret it and derive the needed information from it [Spence, 2001]. Its purpose and usage evolved through the centuries, from creating maps for navigation to representing statistical ideas and data graphics [Friendly, 2008]. The goal of this dissertation is to improve the 2D information retrieval system for the maintenance tasks of CERN machines by adding a third dimension and providing more information in an intuitive way.

1.1 Context and Motivation

Data visualization is the graphical representation of data and information. It involves creating visual elements to help people understand and convey complex data, making it easier to identify patterns, relationships, and trends. [Friendly, 2008] described the evolution of data visualization and divided it into several milestones. Its earliest application was in documenting celestial bodies and creating maps to aid navigation and exploration. In the 17th century, there was a surge of data visualization being applied in practical applications, such as in analytic geometry and coordinate systems, done by Descartes and Fermat, and the creation of probability theory by Pascal and Fermat. Due to innovations in design and technique in the previous centuries, the 19th century saw the birth of modern graphics. In this time period, all modern graphics were created, such as bar and pie charts, histograms, line graphs, and so on. Moreover, scientific publications began to make use of graphics to analyze natural and physical phenomena. After this surge, graphical visualization went through a dark age from 1900 to 1950, where its presence was diminished and was substituted by more numerical forms of representation. However, it had a resurgent in popularity and usage in the 50s. One of the causes was the development of computer science, which provided new paradigms for representing statistical ideas and data graphics. In the current days, new visualization systems have been developed to make graphics more dynamic and interactive and allow a more efficient manipulation and analysis of the data.

CERN, the European Organization for Nuclear Research, is an international organization based in Geneva, Switzerland [CERN, 2023]. It was founded in 1951 in a United Nations Educational, Scientific and Cultural Organization (UNESCO) meeting, with the goal to stop the "brain drain" to North America that had started due to World War 2 and to bring unity to a post-war Europe. It is the home to the Large Hadron Collider (LHC), the world's largest and most powerful particle accelerator. Moreover, it is home to the Proton Synchrotron (PS), the accelerator that this dissertation focuses on. At CERN, many experiments are performed daily on several machines. For the machines to work correctly, they need regular maintenance and check-ups. These maintenance tasks can take weeks to complete and use different subsections of a machine. Scientists and engineers must consider these different variables when planning future experiments. This process takes precious time of their day and makes them focus more on trivial assignments than analyzing the results of their experiments and planning new ones. Furthermore, if the tasks are not planned correctly, e.g., tasks that can't overlap do indeed overlap, it can delay future experiments and consume even more of the CERN engineers' limited time.

CERN has a website called Scheduling Tools which contains several options to visualize maintenance tasks. For example, there is the Gantt View, which displays the scheduled tasks in a Gantt chart, and the Calendar View, which displays them in the calendar. Besides, there is the Interactive View. This option displays a 2D image of the top view of a machine, with its section colored according to the ongoing tasks for a given day. However, this poses some problems. Due to the large scale of the machines, some sections are very small and difficult to see and identify. Furthermore, in order to view future tasks, one needs to select the wanted date and wait for the page to be refreshed and for a new image to be generated, which hinders the interactivity of the system.

CERN is divided into departments, each with several sections subdivided into groups. The project described in this dissertation was proposed by the Organisation, Scheduling, and Support section, of the Accelerator Coordination and Engineering group, from the Engineering department.

1.2 Objectives and Methodology

This dissertation aims to create a visualization system where CERN's staff can easily and efficiently see ongoing and future maintenance tasks and plan their experiments better. More precisely, it aims to develop a 3D visualization system that allows one to see all current, past, and future activities on the PS accelerator and download an image depicting this. From this goal, we can extrapolate two main research questions:

- **RQ1:** How large and complex can 3D models be displayed on the web, with low delay and unnoticeable interaction latency?
- **RQ2:** How do we integrate task schedule information into a 3D model in a way that allows efficient and easy interpretation of the information?

The first question concerns the rendering of large 3D models on the web. We consider a large and complex 3D model with millions of triangles and hundreds of megabytes of file size. More specifically, we have the example of the PS accelerator. The 3D models of the insides of this machine total 1.5GB in file size and around 30 million polygons. When rendering a model of these dimensions on a constrained environment, such as the Web, many problems arise. There is an upper limit on the data rate of the connection, which might become a bottleneck for transferring the 3D model and lead to unwanted delays. Moreover, applying actions and transformations such as translation and rotation to a large model may lead to a noticeable and undesired lag in the interaction. This is due to the need to apply the transformations to all vertexes of the model, and since there are millions of them, the delay may become noticeable.

Integrating the maintenance schedule into a 3D model is also not straightforward. We need to have a complete understanding of the users' information needs, i.e., what information they need at the moment and how they prioritize specific information needs in relation to others. For example, a user may not be that interested in knowing which team is performing the maintenance task in a sector only when the task is finished. Moreover, the user must quickly and efficiently extract this information from the 3D model with minimal input. We consider quick and efficient extraction of information one that doesn't invoke confusion or uncertainty on the user regarding on how to interact with the system and retrieve the needed information.

Considering the goal of the project and the possible obstacles described above, we propose a 3D visualization system that satisfies the information needs of CERN staff and overcomes the predicted problems. This system is divided into two parts: a preprocessing pipeline that converts the CAD model into a simplified glTF model [Khronos Group, 2023] and a visualization interface, where users will be able to interact with the accelerator model and retrieve the needed information. For the sake of guaranteeing that the interface works correctly and without any noticeable delay or lag, we constructed automated tests for every functionality and evaluated the system using performance metrics such as frames per second (FPS) and time to retrieve and display detailed information about a sector (amongst others). Furthermore, a controlled experiment testing the usability of the system was conducted to evaluate how well the system satisfies the users' information needs.

1.3 Document Structure

This dissertation is structured into the following chapters:

Chapter 2 provides a revision and analysis of the most recent and relevant papers and technologies about information visualization and 3D model conversion, simplification, and manipulation.

Chapter 3 presents defines the problem that this dissertation sets to solve and gives a general view of the proposed solution. Moreover, it contains the analysis of the requirements, the choice of technologies, the project planning, and the risk analysis.

Chapter 4 described in a more detailed way the preprocessing pipeline and the visualization tool by focusing on the major design decisions, workflows, and problems faced.

Chapter 5 provides a detailed description of the evaluation methods used to test the application, i.e. the controlled experiment and the loading and lag measurements, and an in-depth analysis of their results.

Chapter 6 concludes this dissertation by re-stating the main findings, summarizing to work performed, and discussing possible future work.

Chapter 2

Related Work

Information visualization is the process of presenting data so that the reader can interpret it and derive the needed information from it [Spence, 2001]. Information visualization deals with abstract data, i.e., data that usually don't have a spatial component. When representing data related to physical objects or mathematical structures and models, we use scientific visualization [Mazza, 2009]. This is the case for Computer-Aided Design (CAD), a software that allows engineers to create machines and their components with high precision. The field more closely related to the theme of this dissertation is Building Information Modeling (BIM). BIM is the usage of computer-aided modeling and simulation to perform a building design, construction, and facility management [Migilinskas et al., 2013].

CERN possesses CAD models representing the accelerators. These CAD models usually contain the machines, the exterior tunnel that encapsulates them, and some surface buildings. Moreover, they contain many details, such as fire extinguishers, signals, cables, electrical wires, etc., that are irrelevant to the goal of the application and/or don't provide any additional information. Furthermore, CAD models have detailed and complex geometries that are unnecessary for this application. For these reasons, the model needs to be simplified to make it easier to process and interact.

This section starts by giving a brief overview of the theory behind information visualization. Most specifically, how to create a good visualization system, what types of interactions exist, and how to evaluate the final result. Next, we investigate the process of the simplification of the geometries, including the different types and recently published articles and the conversion between CAD and the 3D format. This investigation is accomplished by analyzing the most recent and well-known papers alongside the most mature and known technologies. Then, we present the results of the search and analysis for the most recent and relevant articles about web BIM visualization systems. We end by analyzing and comparing the current 3D visualization frameworks of the market and the most well-known 3D file formats.

2.1 Information Visualization

[Bertin, 1983] defines a graphical representation as a sign system constructed by humans to better understand and communicate information. It is a monosemic system, i.e., every sign's meaning is known before observing them. In fact, the reader is not concerned with deciphering the meaning of the signs but with understanding the relationships between them.

In this section, we discuss guidelines on how to construct good visual representations and the basics of creating such representations. Moreover, we examine the most common interactions in 2D and 3D representations. We conclude by discussing evaluation metrics for this kind of representation.

2.1.1 Fundamentals of Graphical Representations

A graphical representation is usually composed of a set of components that possess a certain relationship that we, as readers, want to extract. These components are connected by an invariable common notion called the invariant [Bertin, 1983]. As an example, we can see Fig. 2.1, a graphical representation that depicts the trend of stock X on the Paris exchange in June 1959. In this case, the invariant is the quotation for stock X, cash payment, and closing price on the Paris exchange. The components are the quotation value, in francs, and the time, in days. Since there are two components, we need two visual variables, the two dimensions of the plane.

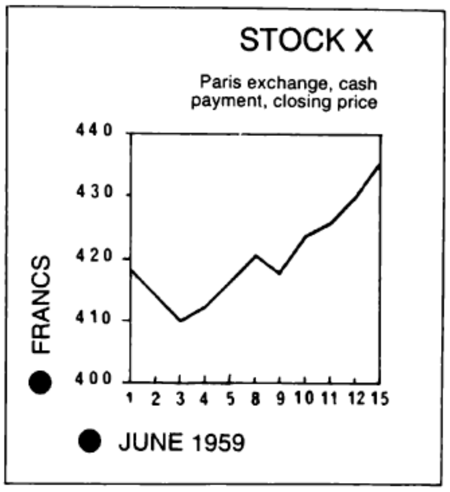


Figure 2.1: The trend of the stock X on the Paris exchange, taken from [Bertin, 1983]

The headings, which are composed of the legend and title material, are an essential part of graphics. They serve two main purposes: identifying the invariant and the components involved, called external identification, and identifying which visual variables correspond to which components. Examining the example present in Fig. 2.1 again, we can observe that "STOCK X" and "Paris exchange, cash payment, closing price" represent the invariant, while the "FRANCS" and "JUNE 1959" represent the components. The external identification of the components is also used as internal identification: the "FRANCS" position and direction indicate that the value increases from top to bottom, while the "JUNE 1959" indicate that the time increases from left to right [Berlin and Kay, 1991].

Another important aspect of graphical representations identified by [Berlin and Kay, 1991] is the components. To represent a certain amount of components, we need at least the same amount of visual variables. Every component has a length, i.e., the number of divisions that enables the viewer to identify. They can be classified as short components when the number of divisions doesn't exceed four and long components when the length exceeds fifteen divisions. It is important to note that length only applies when the series of possible values are discrete. Moreover, components can be qualitative, ordered, or quantitative. Qualitative components have no intrinsic order between values, ordered when the values can be ordered, and quantitative when the values are represented by a series of numbers. The visual variables associated with the components must have at least the same level of organization.

A good visual representation is one that satisfies the requirements and information needs of the end-user [Mazza, 2009]. The process of creating a good visual representation is not straightforward, and there is no order of pre-established steps to follow to achieve one. However, one can follow a set of criteria to help guarantee a good quality visual representation.

[Tufte, 2001] defended that, in order to achieve graphical excellence, the graphical displays should serve a clear purpose, make the viewer focus on the substance, and reveal the data at several levels of detail. Furthermore, graphic representations should have "visual integrity", meaning that they should never distort the data or aid in false interpretation. Data graphics should not distract the viewer from the substance of the data. For this purpose, [Tufte, 2001] proposed the data-ink ratio, which measures the "proportion of a graphic's ink devoted to the non-redundant display of data information.". Most graphic representations should strive to achieve a high data-ink ratio, meaning that every graphical element should convey new information. Moreover, it is important that the graphic is elegant. For the author, this means that it must have a defined format and design, with no decorations, that displays complex relations in an accessible way.

In the present day, there are some tools that help us create graphic representations, such as computers and visualization software. In the latter's case, although several different software programs and libraries exist, they all follow the same process, as identified by [Mazza, 2009]. This process, which can be observed in Fig 2.2, consists of preprocessing and transforming the raw data, performing the appropriate visual mapping, and creating the final view.

Preprocessing the raw data consists of extracting useful data and converting it to a structured format so that the visualization software can analyze it. The most common operations to perform on the data are filtering the unnecessary data, calculating new data to be added (like statistical data), and adding metadata.

Visual mapping is the process of defining the visual structures that we want to represent the data with. There are three visual structures to be defined: the plane, or spatial substrate, which is the dimensions in physical space where the visual representation exists; the graphical elements, also called marks, which encapsulate everything visible present in the substrate; and the graphical

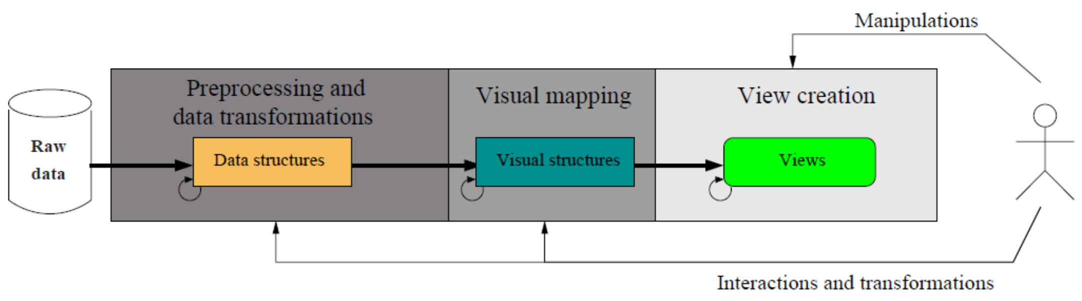


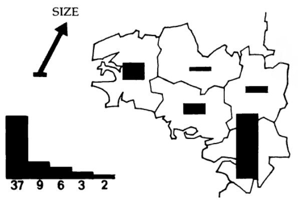
Figure 2.2: General methodology for visualization softwares, identified by [Mazza, 2009]

properties, or visual variables, which are the properties of the graphical elements that the human eye can process. The view is simply the mapping of the data structures and the visual structures to create the graphical representation [Mazza, 2009] [Bertin, 1983].

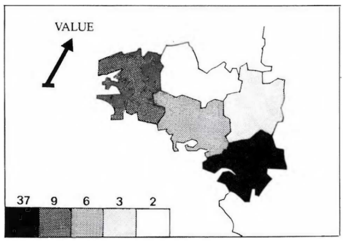
According to [Bertin, 1983], the different types of visual variables are position, size, value, texture, color, orientation, and shape. Special care should be taken when using color since it is the only property that its meaning can change according to cultural, linguistic, and physiological factors. [Ware, 2000] gives the example of the red and green colors. In Western culture, red is associated with danger, and green with renewal. However, this is not the case in China: red symbolizes good luck, while green represents death. According to the research done by [Berlin and Kay, 1991], the more globally recognized colors, called primary colors, are white, black, red, green, yellow, and blue. It is not advisable to use primary colors to represent quantitative data or to represent the order in a sequence of values since there may not be enough colors for the job, and there is no convention on the order of colors. However, if one still wants to use color for ordering, they can bypass this problem by using a color scale, where color gradually shifts from one color to another, or an intensity scale, i.e., change the intensity of color according to the value. One last consideration regarding color is that some people might be color-blind, having difficulties distinguishing between certain colors. Hence, the designer should take this into consideration and implement, if possible, a different choice for color mapping [Mazza, 2009].

The plane is the most basic part of any graphical representation. It is homogeneous and defines two dimensions where all the marks are located. Moreover, a mark is represented on a plane by a point, a line, or an area. Points and lines are independent of the characteristics of the mark they represent, except for the position. The meaning of an area depends on the size, shape, and orientation of the mark it represents [Bertin, 1983].

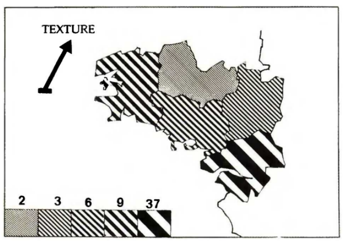
If we need to represent more than two components, we must use retinal variables. These variables correspond to the visual variables, excluding position (which is defined in the plane). Retinal variables are considered to be "above" the plane since they share the same physical area of the points, lines, and areas and represent a third component alongside its relationship with the other two. They give meaning to the marks of a plane in regard to quantities, categories, time, and space. Figure 2.3 contains examples for every retinal variable. As we can verify, any retinal variable can be used to represent a component. However, this doesn't mean that all retinal variables are equally appropriate to represent some component. The suitability of a variable depends on the type of component that we want to represent [Bertin, 1983].



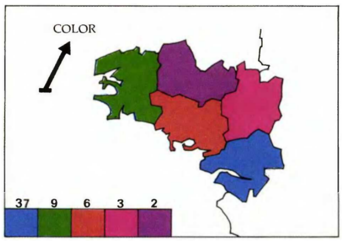
(a) Example of the retinal variable size



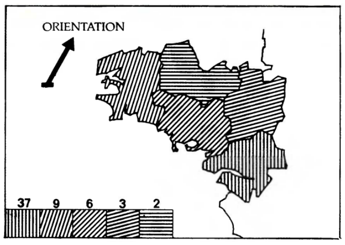
(b) Example of the retinal variable value



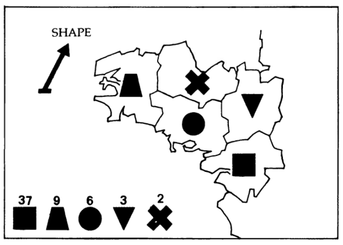
(c) Example of the retinal variable texture



(d) Example of the retinal variable color



(e) Example of the retinal variable orientation



(f) Example of the retinal variable shape

Figure 2.3: Examples of the retinal variables, adjusted from [Bertin, 1983]

According to [Mazza, 2009], the main problem encountered when designing a visual representation is creating a visual mapping that shows with fidelity the information contained in the data while satisfying the user's needs. The following procedure establishes a sequence of steps that one can follow to create a visual representation from abstract data, overcoming the previously defined problem. First, one should start by discussing their information and visualization needs with the final users. The designer should clearly understand the purpose and requirements of the graphical visualization. Then, we should understand what is the nature of the data to represent: if it is quantitative, ordinal, or categorical [1]. After this, the attributes of the data, i.e., the number of dimensions, should be determined. The attributes can be independent or dependent. The independent attribute is one that we change in order to measure the variation of the dependent attributes.

Counting on the number of dependent attributes, a collection of data can be univariate (one dependent attribute), bivariate (two dependent attributes), trivariate (three dependent attributes), and multivariate (four or more dependent attributes). Afterward, the structure of the data should be analyzed. More specifically, the data can be categorized as linear, i.e., when it is stored in linear data structures, like vectors and tables, temporal, spatial, or geographical, when it is related to the world or something physical, hierarchical, and network, i.e., when it describes relationships between entities. Finally, one should consider the type of interaction intended. This can be static, which means no interaction; manipulable when the user controls some parameters that change the

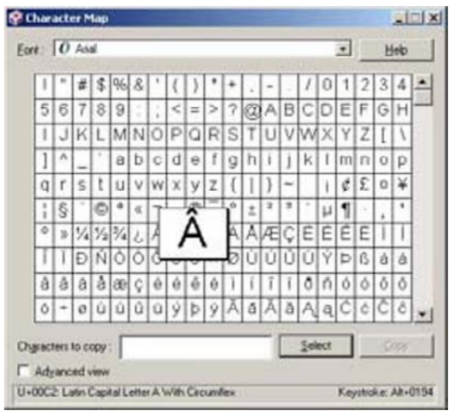
[1]quantitative refers to something that can be measured, ordinal when there is a logical order of values, and categorical when there is no order

view, and transformable when the user controls the modification and transformation of the data [Mazza, 2009].

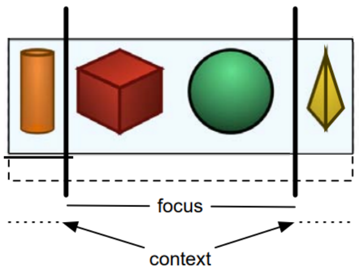
2.1.2 Interactions with Graphical Representations

When the amount of data that we want to display is large, the physical space available can become a constraint. We may not have enough space to display all the data or with all the details needed. To overcome this challenge, one can add interaction to their graphics so that the user can modify the input data, change the visual mapping or manipulate the view. According to the transformations that the user can perform, there are three types of representation: static if no transformation is allowed, manipulable if the user can manipulate the view, and transformable if the user can manipulate the input data [Mazza, 2009].

In manipulable representations, the most common techniques are zooming, rotation, and scrolling. Moreover, there is also an overview + details, which consists of showing the user the general structure of the information and allowing him to focus on certain data. There is also the focus + context technique, where the contextual and detailed information is presented in the same view. More precisely, it displays primarily detailed information while preserving the context where it's positioned. Figure 2.4 shows examples of both the overview + details and the focus + context techniques. On the other hand, we have transformable representations, which are useful when the user doesn't know the data very well and wishes to perform an exploratory analysis of it. Some common techniques include filtering, reordering, or performing dynamic queries on the data [Mazza, 2009].



(a) Example of the overview + details technique, used in the map of the characters used by Microsoft Windows



(b) Example of a focus + detail technique, most precisely, the bifocal display

Figure 2.4: Examples of manipulable representations, taken from [Mazza, 2009]

2.1.3 3D Representations

Data with a low number of dependent variables can be easily represented in 2D. However, when this number starts to grow, the spatial dimensions are insufficient, and it might become inefficient to map the attributes to only two spatial dimensions. According to [Mazza, 2009], one can include a third spatial dimension to overcome this challenge, making the mapping easier. Nonetheless, this addition comes with a few problems. First, the graphical elements on the front may occlude the ones in the back. Moreover, it can become difficult to understand the position of certain elements in relation to the axes. Both these problems can be reduced by allowing the rotation of the graphic or identifying the values of the data according to the axis. Another solution is removing the third dimension and mapping an attribute to graphical properties, such as color or shape. The 3D graphical representation may not be the best solution since it increases the cognitive load of the viewer. One should consider the needs of the viewer and the type of data being used. For example, 3D works very well when we want to represent a moving object or when the data has a three-dimensional spatial component.

[LaViola et al., 2017] defined the basic manipulation tasks, which all other more complex tasks are based on. These are the selection task, where the user chooses one or more objects from a set; positioning, rotation, and scaling tasks consist of changing an object's position, orientation, and scaling. Each of these manipulation tasks can be customized by changing the task parameters, such as distance to the target, initial position, initial orientation, etc.

2.1.4 Evaluation of Graphical Representations

Evaluating the produced graphical representation is very important to achieve a good quality in the final product. The evaluation process ensures that the final product meets the users' requirements and behaves as expected, with no errors. It is a continuous process that occurs during the design life cycle and helps guide the design process. Moreover, it can reveal problems in the early stages of the design that can be easily and quickly resolved [Dix et al., 2003].

To better understand the evaluation of visual representation, one can look to the scientific field of Human-Computer Interaction (HCI). [Dix et al., 2003] defines HCI as the design, implementation, and testing of an interactive system, considering the user's needs. A user can be any person or group of people. It defines two types of evaluation: formative, when performed in the design phase or with a functional prototype to verify if it is acceptable or needs improvements, and summative when performed at the end of the process with a sample of the final users. Both types of evaluation serve different purposes. The formative evaluation identifies potential future problems with the design and how to fix them. The summative evaluation identifies possible improvements to the final result.

The evaluation process has three main goals: to test the system's functionality, to assess the user's experience, and detect the system's problems. The system's functionality derives from the requirements set by the users, meaning that it should satisfy their needs. It is evaluated on how well the functionalities are exposed to the user, i.e., if the users can efficiently satisfy their needs. Another important goal is the user's experience with the system. This should be evaluated by testing the usability of the system, the user's experience with it, and how easy it is to learn. The final goal is to identify any problems with the design. The developed design may have some aspects that were thought to be clear in theory, but in practice, the user may find it confusing. It is important to identify every problem and confusing aspect so that they can be properly fixed or clarified [Dix et al., 2003].

As stated by [Dix et al., 2003], evaluation should be regularly performed throughout the design process since it helps identify and correct problems before they become costly and hard to fix. However, testing continuously with users is a hard and expensive process. Alternatively, one can use an expert to identify possible problematic aspects of the design that violate known cognitive principles or ignore known and accepted empirical results. This is called expert analysis, and it is flexible and cheap compared to evaluation with end-users. There are four types of expert analysis: cognitive walkthrough, heuristic evaluation, the use of models, and the use of previous work. The cognitive walkthrough method was first defined by [Polson et al., 1992] and consists of an expert performing a sequence of actions to complete a task while using a set of evaluation criteria to identify positive and negative aspects of the design. Heuristics evaluation, created by [Nielsen and Molich, 1990], is a simple method that uses a short set of simple design rules to identify and justify which parts of the design are good or bad. Moreover, there is the model-based evaluation, which consists of creating a cognitive model that simulates the user's behavior and compares this output with the real user behavior. The final type of expert analysis is the use of previous work, where the designer uses well-known studies and experiments to support or refute certain design aspects [Dix et al., 2003].

Another type of evaluation is the evaluation through user participation. This type of evaluation is usually performed in later stages of development, with a working prototype, but it can also be performed early in the process, like in requirements capture. They can be performed in a laboratory, where the user is in a controlled but unnatural environment, or in the user's work environment, where the observation may be difficult due to uncontrollable external forces, with the advantage of the user being in a natural environment [Dix et al., 2003].

According to [Dix et al., 2003], one of the most common ways to evaluate a design is through controlled experiments. These experiments consist of testing a hypothesis chosen by the evaluator using some measure attributes of the participant's behavior. Five main aspects need to be defined before starting the controlled experiment.

First, we have the participants that should be chosen to match the intended user group as closely as possible. Ideally, one should test with real users, but sometimes that is impossible. Moreover, the sample size chosen should be large enough to represent the population. Furthermore, we need to define the variables of the experiment, which are the variables we want to measure (the dependent variables) and the ones we vary to produce different results (the independent variables). Experiments can have more than one independent variable. Also, the dependent variable should only depend on the independent variable [Blandford et al., 2008].

Another important factor of an experiment is the hypothesis. The hypothesis is the predicted result of the experiment, which is that the variation of the independent variable will produce a variation in the dependent variable. The purpose of the experiment is to prove the hypothesis true. This is done by disproving the null hypothesis, i.e., that there is no relation between them. There is also the need to define the experiment method to use. There are two main ones: between-subjects (or randomized) and within-subjects (or repeated measures). In the between-subjects method, every participant is assigned to a condition. There are at least two conditions: the experimental condition, where the independent variable was manipulated, and the control condition, where it was not. Since a subject only participates in one condition, it cannot become familiarized with the condition and influence the result in this way. However, one should pick the participants carefully so that there is no significant variation between the condition's subject groups and consequentially not bias the result. The second type of experimental method is the within-subjects method, where the participant performs every condition. This means that the participant can learn how to do the condition and transfer that learning to the next condition. However, the transfer learning effect can be overcome by varying the order of the conditions between users. The within-subjects method requires fewer participants and suffers less from the variation between them [Blandford et al., 2008].

After performing the experiment and saving the data, we need to analyze it. This analysis depends on the type of data collected and the questions needed to answer. Table 2.1 presents the different statistical techniques to be employed depending on the type of the independent and dependent variable. The independent variable can be two-valued when it only has two possible values, discrete when it can only be a finite number of values, and continuous when it can take any value. The dependent variable can be discrete, continuous, and also normal when it follows a normal distribution [Dix et al., 2003]. Regardless of the type of analysis performed, it has the goal of answering the researchers' questions and proving the set hypothesis.

Table 2.1: Different types of statistical techniques to be employed depending on the type of variables of the experiment, according to [Dix et al., 2003]

Independent variable	Dependent variable	Technique
Parametric		
Two-valued	Normal	Student's t test on difference of means
Discrete	Normal	ANOVA (ANalysis Of VAriance)
Continuous	Normal	Linear (or non-linear) regression factor analysis
Non-parametric		
Two-valued	Continuous	Wilcoxon (or Mann–Whitney) rank-sum test
Discrete	Continuous	Rank-sum versions of ANOVA
Continuous	Continuous	Spearman's rank correlation
Contingency tests		
Two-valued	Discrete	No special test, see next entry
Discrete	Discrete	Contingency table and chi-squared test
Continuous	Discrete	(Rare) Group independent variable and then as above

Apart from the controlled experiments, there are several other types of evaluation with users. For example, we have observation studies, where the expert observes the participants performing some predefined tasks and records their actions. Participants can be asked to think aloud, i.e., to talk about what they are performing and thinking. This allows a deeper analysis of problems with the design and a clearer insight into how users use the interface. However, this feedback is subjective and may be biased due to the observation.

While observing the user, the interviewer needs to record the user's actions to analyze them after. There are several methods of recording the users' actions. The most simple one is note-taking on a piece of paper. It is a cheap option, but it is hard to capture more complex actions, and it's always limited by the note-taker speed. Other options are audio or video recording, where we can hear and see the users' interactions with the interface. Moreover, we have computer logging, where the users' input on the computer, such as keystrokes and mouse button clicks, are recorded. We also have user notebooks, where the user is asked to keep notes of the activities he performs and problems that he encounters. One is not bound to only one of these methods. They can be used in conjunction according to the need of the interviewer [Dix et al., 2003].

The final type of evaluation through users' input that we are going to discuss is the query technique. This method consists of asking the users about the interface. It can reveal some issues with the design that the designer never considered. However, the data collected is subjective and unnatural since the users are asked to think about and rationalize the interface. There are two main types of query techniques: interviews and questionnaires. Interviews have the goal of understanding the users' preferences, impressions, and attitudes. They can have a flexible structure, with questions being added according to the user response (such as Why do you think that, and What if) [Dix et al., 2003].

On the other hand, questionnaires are a set of questions prepared in advance that are asked of the participant. It takes less time to execute and can be analyzed more objectively. There are several different styles of questions. General questions have the goal of clustering the participants into demographic, geographic, or other types of groups. Open-ended questions are designed for the users to give their opinion about any aspect of the design. Furthermore, there are scalar questions that ask the user to specify on a numerical scale their judgment regarding a specific statement. The most common scales are 1 to 5 and 1 to 7. Another style of question is multiple-choice, where the user has to choose one or more answers from a set that better identifies their opinion or perception. Finally, we have the ranked questions that askes for the participants to order the items on a list according to their preferences [Dix et al., 2003].

According to [Dix et al., 2003], there are some best practices that should be followed when performing questionnaires. One should always perform a pilot study to identify problems with the questions and to iron out some details before it is sent to all other users. The participants should be as random as possible, and since the answer rate is low, one should send a lot more than actually needed.

Diving deeper into questionnaires, there are several standard questionnaires that one can use to evaluate a design. Standard questionnaires are questionnaires that can be applied to several use cases, with a set of predetermined questions in a certain order and format and rules on how to analyze the results [Sauro and Lewis, 2016]. A considerable amount of standard questionnaires exist, so we picked a few examples to analyze further.

We start with the USE questionnaire created by [Lund, 2001]. USE stands for Usefulness, Satisfaction, and Ease of Use. It was constructed by collecting a large set of items and eliminating or rewording the ones that could not be applied across hardware, software, documentation, and service domains. Moreover, [Lund, 2001] used standard psychometric techniques to refine the items. To further test the items, users were asked to rate their agreement with them, using a seven-point Likert rating scale, from strongly disagree to strongly agree. The author noticed that the users were evaluating the products mainly in terms of Usefulness, Satisfaction, and Ease of Use and that Usefulness and Ease of Use were correlated. Using all this information, a final set of items were selected.

Another common standard questionnaire is the Software Usability Measurement Inventory (SUMI) questionnaire, developed by [Kirakowski and Corbett, 1993]. It can be used to compare different products or different versions of the same product. It is composed of 50 items, and it should be performed on users with prior experience with the software being evaluated. The creators of SUMI recommend a minimum of 10 users to test with, but if the evaluation is for diagnostic purposes, it can be smaller. This questionnaire can be applied to any application that records user input through a keyboard or a mouse, displays it on a screen, and there is input/output between secondary memory and peripheral devices.

The third and final standard questionnaire analyzed is the SUS questionnaire, developed by [Brooke, 1996]. It is a simple and quick ten-item questionnaire that assesses the global subjective view of the usability of a design. It uses a five-point Likert scale, and the users are asked to indicate the level of their agreement for every item. The items were selected by identifying from a pool of several possible items, the ones that lead to an extreme response, and tested with two different software systems. Moreover, the questionnaire has an equal amount of questions where most users selected strongly disagree and questions where most users selected strongly agree. This is done to prevent response bias from users not having to think about the statement and automatically answering.

There are many more standard questionnaires that test the usability of software systems. For example, there is the Questionnaire for User Interaction Satisfaction (QUIS), which measures the users' subjective rating of the interface through a ten-point scale. Moreover, there is the Purdue Usability Testing Questionnaire (PUTQ), which constructs the questionnaire based on human factors extracted from HCI theory and the three stages of human information processing. However, this is not meant to be an extensive listing of these questionnaires. If the reader wants to know more about standard usability questionnaires, we recommend reading the literature review written by [Assila et al., 2016].

2.2 Model Conversion from CAD Format to 3D Format

CAD formats are encoded using mathematical functions that describe their surfaces. This type of encoding is not suited for web rendering, which mostly supports polygonal meshes [Gregory, 2018].
There are some libraries that render CAD formats on the web,such as OpenJSCAD [The JSCAD Organization, 202

an open-source tool for creating parametric surfaces using Javascript. However, directly rendering CAD format is not optimal since most graphical hardware doesn't provide support for them [Guthe et al., 2004]. For this reason, in this section, we analyze some papers and technologies that propose and implement an algorithm for model conversion between the CAD format and the 3D format.

2.2.1 Literature Review

CERN uses the software CATIA[2] to model their building and accelerators. This software exports the models to standard CAD formats, such as DXF [Autodesk, Inc., 2011], STEP [Sustainability of Digital Formats and IGES [ISO/IEC, 1997]. Most of the CAD formats are not compatible with 3D frameworks since they use 3D model formats, such as glTF [Khronos Group, 2023], FBX [Autodesk, 2023], and COLLADA [Barnes and Finch, 2008]. In this section, we analyze some of the more recent conversion systems proposed to solve this problem.

In the conversion process from a CAD model to a 3D model, some problems arise that need to be solved, as identified by [Raposo et al., 2006]. Firstly, the often unneeded complexity of the CAD model decreases the performance of the visualization. Moreover, CAD formats do not usually have materials and textures associated with them, making the model look less real, which can conflict with the application goal. Loss of geometric precision and project-related information can also occur when converting. Furthermore, changes made to the model in a 3D format usually can't be transmitted back automatically to the CAD model.

A great deal of work has been published in recent years to address some or all of these problems. [Bellalouna, 2020] defines a best practice process to transform CAD to 3D format in 6 steps (Fig 2.5). The first step is to convert the CAD data to a 3D format. Then, the number of polygons should be optimized to balance the graphical quality and the application's performance. Afterward, all unnecessary models, such as those inside other models, should be deleted and thus not visible to the user. After this, a re-assignment of the materials and textures must be done since they were deleted in the conversion step. Furthermore, the conversion step can have the undesirable side-effect of damaging an object's structure by dividing it into several sub-structures. So, it is necessary to merge these objects into a single structure. Finally, the assembly structure (the parent-child relationship) should be reinstated into the model.

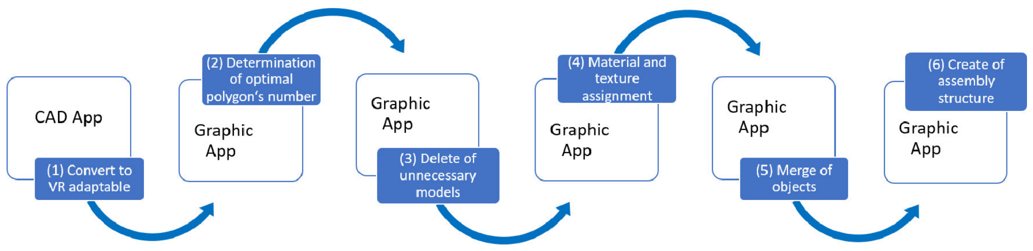

CAD App
(1) Convert to VR adaptable
(2) Determination of optimal polygon's number
Graphic App
Graphic App
(3) Delete of unnecessary models
(4) Material and texture assignment
Graphic App
Graphic App
(5) Merge of objects
(6) Create of assembly structure
Graphic App

Figure 2.5: 6-step conversion process developed by [Bellalouna, 2020]

[2]website available at `https://www.3ds.com/products-services/catia/`

[Lorenz et al., 2016] developed a CAD-to-VR automated system that simplifies and converts the model and preserves its kinematics. They identify one of the main problems of using VR to visualize CAD models in the conversion process. This is a manual process, where the user needs to use a digital content creation (DCC) to convert the model. Furthermore, the kinematic chains and animation defined in the CAD model are lost and must be redefined. To fix this problem, the authors propose an automated system that converts a CAD model to the VRML format while preserving kinematics and animations. The choice of the VRML format is the result of a comparison between some 3D exchange formats. However, only four formats were analyzed, and some commonly used formats, such as COLLADA [Barnes and Finch, 2008], FBX [Autodesk, 2023], and glTF [Khronos Group, 2023], were not mentioned. The system starts by exporting the CAD data to VRML format using a CAD system. The model is then reduced using the GPure tool (this tool isn't available anymore). NuGraf/PolyTrans [3] was also considered, but it was not chosen due to not performing geometry simplification or optimization. No other reduction tools or methods were discussed. The animation and kinematics are passed to a CAD adapter, which is implemented using a CAD system API, and merged into the VRML model. The proposed system's architecture can be found in Fig. 2.6. The system was evaluated only by constructing a prototype, using Siemens NX [4] as the CAD system, Instant Reality (we couldn't find this system) as the VR system, and Siemens Teamcenter [5] as the PDM system. No other evaluation method was performed, and the authors do not provide any result statistics.

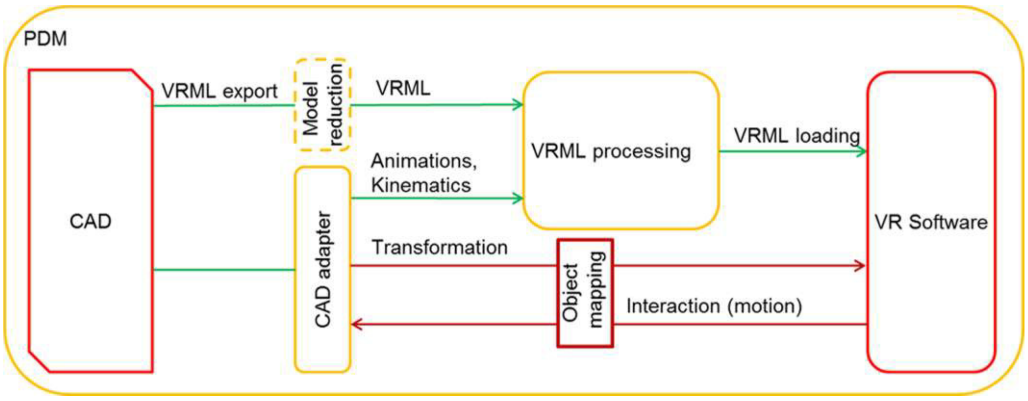


Figure 2.6: Architecture of the CAD-to-VR conversion system proposed by [Lorenz et al., 2016].

[Santos et al., 2021] created a tool called CADto3D that, as the name suggests, converts models encoded in CAD to the 3D format. This tool allows the user to control the level of detail of the output mesh interactively. Moreover, it also provides a method that automatically creates the texture coordinates for the generated meshes. The tool allows users to import a STEP file and export it to FBX. Besides, it also allows the re-tessellation of the mesh. In this context, tessellation is the process of converting the CAD surface that is defined using mathematical functions into polygonal

[3]available at `https://www.okino.com/nrs/nrs.htm`
[4]more information at `https://plm.sw.siemens.com/en-US/nx/`
[5]available at `https://plm.sw.siemens.com/en-US/teamcenter/`

meshes. The authors note that they first tried to apply a mesh decimation algorithm, but the results obtained weren't "visually satisfactory".

The CADto3D tool was constructed as a desktop application using Electron [OpenJS Foundation and Electron c They also used Three.js [Three.js community, 2022c] to create a scene and visualize the models. The import process was handled using OpenCascade, and the exportation to FBX using the FBX SDK. Both this process and the re-tessellation are Python subprocesses. The authors fail to clarify if the interactiveness of the tool comes from being able to export to FBX multiple times or seeing, at run-time, the result of the re-tessellation. Furthermore, they don't clarify which re-tessellation algorithm they decided to use. [Santos et al., 2021] compared the developed tool to the plugin Datasmith for the Unreal Engine tool [Epic Games, 2023a]. The results can be seen in Table 2.2. The authors say that Datasmith failed for Object 2 due to its size, but they didn't provide the value. Moreover, they don't explain why they didn't export Object 1 using Datasmith. [Santos et al., 2021] attributed the low re-tessellation time to using temporary files to store shape information. However, as said before, they don't indicate which algorithm they use or what these temporary files are.

Table 2.2: Result of the experiments of the simplification method proposed by [Salinas et al., 2015]. The experiments were performed on models of different complexity and compared against known approaches.

	Object 1		Object 2	
	Datasmith	CADto3D	Datasmith	CADto3D
Import	70s	75s		340s
Export		5s		25s
Import into Unreal		30s		160s
Total import time	70s	110s	failed	525s
Re-tessellation	60s	30s		130s

2.2.2 Existing Technologies

In addition to the papers discussed previously, some tools in the market perform the conversion between CAD and Mesh formats. We now present the result of an extensive search and analysis on the state of the art of conversion tools.

We first need to establish the analysis criteria. We analyze the conversion tools according to the possibility for automation and fine-tuning of the tessellation process and if they are free and open source. Automation in this context refers to the ability to write scripts that perform the conversion automatically with no user input. As already explained before, tessellation is the process of converting mathematically defined surfaces into polygonal surfaces [Gregory, 2018]. This process has some parameters that can be fine-tuned in order to generate more or fewer polygons. Considering these criteria, we now present the examined conversion tools. These are assimp

[Kim Kulling, 2023], PythonOCC [Paviot, 2023], Unreal Engine [Epic Games, 2023b], Blender [The Blender Foundation, 2023], and CAD Assistant [Open Cascade, 2023].

The Asset-Importer Library (Assimp) is a free and open-source library developed by Kim Kulling and several other contributors that focuses on the conversion between several 3D file formats. It is written in C/C++ but offers several ports option, such as Python, Javascript, and Rust. Assimp performs the conversion process by importing the models into an internal format called aiScene and defining custom exporters for every desired format. The conversion process is done by writing a script that automatically performs the operation. However, there are no options to fine-tune the tessellation process.

PythonOCC is a free and open-source wrapper for the OpenCASCADE C++ kernel developed by Paviot. It allows the conversion between several 3D formats, as well as the rendering of the models on WebGL [Mozilla Foundation, 2023b] and x3DOM [Fraunhofer-Gesellschaft, 2022d] renderers. Like Assimp, it allows the user to write a script in order to convert the 3D models automatically. It also doesn't provide any optimization options. OpenCASCADE [OPEN CASCADE SAS, 2023] is used as a kernel in this library, and is also a geometry library that allows surface and solid modeling, data exchange between several formats, and visualization of models.

Unreal Engine (UE) is a free and open-source game engine developed by Epic Games. It possesses a plugin called Datasmith [Epic Games, 2023a] that allows Unreal Engine to import several CAD formats, including some proprietary ones, such as 3ds Max and CATIA. UE allows scripting in Python or Blueprint, but the software needs to be open and running. Furthermore, it provides options for the user to optimize the tessellation process.

Blender is a free and open-source 3D creation software. It supports modeling, animation, rendering, and simulation, amongst many others. Like UE, it allows users to write a Python script to automate certain jobs, such as conversion. However, it does not provide any optimization for the tessellation process.

Finally, we have CAD Assistant which is a free and open-source CAD viewer and converter created by [Open Cascade, 2023]. Like pythonOCC, it uses the Open Cascade library as its kernel. It allows users to view 3D models encoded in CAD formats like STEP and IGES and convert them to several 3D formats, such as glTF, FBX, and COLLADA. Regardless, this is a manual process and doesn't allow scripting. Also, it doesn't provide any options to optimize the tessellation process.

There are several other libraries and tools that convert between CAD format and 3D format, but they were not mentioned in the analysis because they were very small and may cease to exist in the future (and hinder the maintenance of the proposed system) or didn't work with large CAD files (greater than 50MB). This is the case of the 3DModelConvertToGltf library, developed by [Wang, 2023] that converts STEP and IGES files to glTF, and the FreeCAD tool [The FreeCAD Team, 2023], a free and open-source tool to construct CAD objects.

2.2.3 Summary

In this section, we analyzed some papers and tools that proposed and developed algorithms and methodologies for conversion between CAD formats and 3D formats.

[Raposo et al., 2006] and [Bellalouna, 2020] gave a general description and best practices to take into consideration when designing a conversion algorithm. This allowed for a better understanding of how to tackle the problem. To the best of our knowledge, there seems to be a lack of papers trying to solve the conversion problem. Both papers analyzed used third-party tools as part of their methodology to convert the models.

A summary of the analysis can be found in Table 2.3. Even though there are more conversion tools, we believe that the analyzed ones are a good representative set of the current state of the art. In the next chapter, we further analyze these tools having the requirements of the project in mind.

Table 2.3: Summary of the comparison between several conversion tools. The comparison was made regarding if it is an automatic conversion, if the tools perform any geometry optimization, and if it is open source and free. Semi-automatic means that it allows scripting, but the software needs to be open and running.

Tool	Automatic	Optimization	Free and Open source
assimp	Yes	No	Yes
PythonOCC	Yes	No	Yes
UE	Semi	Yes	Yes
Blender	Semi	No	Yes
CAD Assistant	No	No	Yes

2.3 Simplification of 3D Models' Geometry

After the conversion process, the machines' models must be rendered on the web. However, these models have a very large geometry complexity that may increase the interaction lag. In this section, we analyze some of the most well-known and recent papers concerning model geometry simplification. We also discuss some of the existing technology on the market that can simplify 3D models.

2.3.1 Literature Review

Currently, 3D models have become more widely used. It is used in product design, medical data acquisition (e.g., for MRI machines), range scanners, satellite radar, and many other applications. However, these models have a high level of complexity and are composed of millions of triangles, which hinders the interactiveness of the models [Lindstrom and Turk, 1998] [Kanai et al., 2012].

In order to alleviate this problem, several simplification algorithms were proposed over the years, and we analyze some of them in this section.

One common way of encoding the geometry of a 3D model is by using polygon meshes, which are defined by [Foley, 1996] as "a collection of edges, vertices, and polygons connected such as each edge is shared by at most two polygons.". These types of meshes are well-suited to represent flat-faced objects but fall short for curved-faced objects. For this type of object, continuous parametric cubic curves and bicubic surfaces are better suited. A high number of polygons usually leads to lower performance, so it is important to simplify the model a priori.

According to [Hansen and Johnson, 2011], model simplification consists of constructing a hierarchy of model representations with decreasing model accuracy and increasing interactivity. The simplification algorithm needs to take into account the type of data. The authors define three types: 1) terrain visualization, data of terrains, usually used by flight simulators and acquired from satellite imaging and LIDAR; 2) 3D scanned models, which are highly detailed representations of physical objects; 3) medical and scientific visualization, normally 3D volumetric data, and 4) CAD and Synthetic Environments, that represent simple to complex machines with high geometric complexity. Next, the authors define the different types of hierarchies, starting with the simplest one, the discrete hierarchy, that encodes multiple levels of detail (LODs) of a model at a coarse granularity. There is also the continuous hierarchy, with a very high number of LOD, which can represent a continuous progression of the model complexity, and the view-dependent hierarchy that allows the user to select, at runtime, the desired LOD of the different parts of the model. The final type of hierarchy described is hierarchical LOD, which is similar to the view-dependent hierarchy but is usually more coarse, and the hierarchy is associated with the scene graph structure. Figure 2.7 provides an illustration of the types of hierarchies.

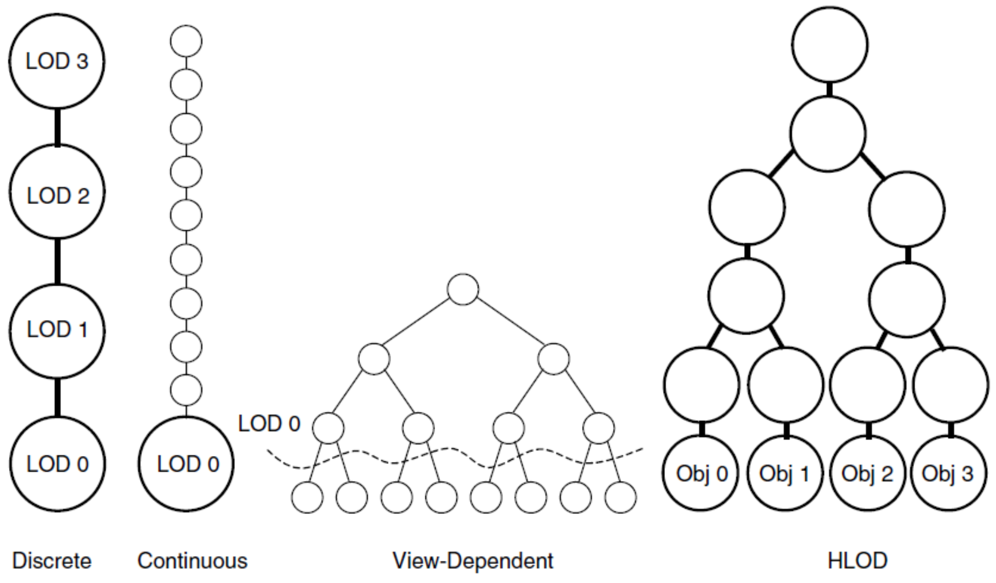


Figure 2.7: Types of model simplification hierarchies, taken from [Hansen and Johnson, 2011]

Most simplification hierarchies are built by starting with the original model and gradually reducing its complexity. For static models, this is done before the user interaction. Depending on what is desired, the simplification operations can produce an adaptive hierarchy, where some areas retain more primitives, hence more detail, or a regular hierarchy, where LOD is the same for all areas. An adaptive hierarchy is built by executing a sequence of simplifications operations ordered by a priority queue. These operators are usually vertex-merging operators that include the edge collapse, which merges two vertexes connected by an edge into a single one, discarding the primitives that become degenerate. This operator has several variants, such as the half-edge collapse, where the final vertex is one of the two source vertex, the vertex-pair collapse, which can merge any two connected or unconnected vertices and triangle collapse, where three connected vertexes are merged, amongst other variants. Another common type of simplification operator is vertex removal, where a vertex and the adjacent triangles are removed, and the created hole is triangulated with new triangles. Figure 2.8 shows a graphical representation of the different types of local decimation operators. The vertex-merging operators are more commonly used than the vertex removal operators because they are more easily computed, encoded, and interpolated. Moreover, the topological structure of the model can be preserved using edge collapse, triangle collapse, and/or vertex removal.

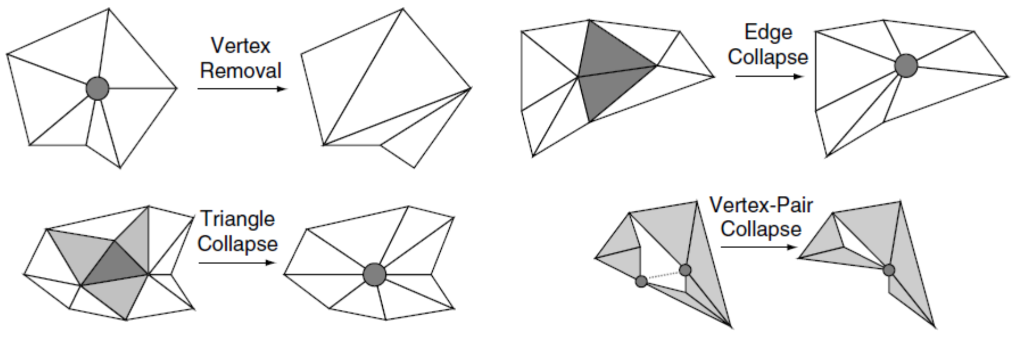


Figure 2.8: Simplification operators for performing local decimation, obtained from [Hansen and Johnson, 2011]

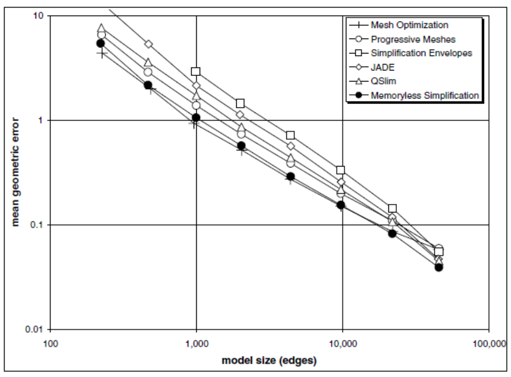
As stated before, the simplification operators are ordered in a priority queue. This priority can be calculated using a number of error metrics. These include curvature estimates, which start by removing vertices from areas of low curvature; point-surface metrics, which measure the mean squared error between specific points on the original surface and their corresponding points on the modified surface; vertex-plane metrics, where the error is measured between the vertices of the simplified surface and their supporting planes; amongst others. Moreover, some metrics measure the error of non-geometric attributes, such as colors and normals. Another simplification technique is an image-based simplification, where a 2D imposter is used in distant geometries to accelerate the rendering process.

Having discussed the basic terminology of model simplification, we now present some of the work done in this area. We start by [Lindstrom and Turk, 1998], which proposes a fast and memory-efficient method for polygonal simplification. They use the edge collapse operation for simplification because it helps preserve the original surface's shape. It is an atomic operation and does need re-triangulation, compared to vertex or face removal. The proposed algorithm repeatedly collapses the edge with the lowest cost and re-calculates the cost for all edges affected by this operation until the desired LOD is met. So, the first step is to calculate the cost of all edges in the mesh, which are maintained in a priority queue. In every iteration, an edge is popped off the queue and analyzed for removal. If there is no viable replacement vertex or it violates a previously established topological constraint, the edge is rejected, its' cost is set to infinity, and it is moved to the back of the queue. When a viable edge is selected, the collapse operation is performed, the cost of adjacent edges is recalculated, and the next iteration of the loop begins. The edge-collapsing process has two main design aspects: how to measure the cost of collapsing an edge and how to determine the position of the replacement vertex. The authors posed this as an optimization problem, most respectively, given an objective function $f_C(e,v)$ that gives the cost of replacing edge e with vertex v, v is chosen such as f_C is minimal. They chose f_C in a way that uses the volume and area information about the model. The proposed algorithm was tested using the Standford Bunny, a standard test object for simplification, and a model of the bones in a human hand. Table 2.4 demonstrates the simplification times for well-known algorithms at the time and the developed algorithm (the Memoryless Simplification). We can see that only the QSlim method is faster than the proposed method. However, it took ten times longer to run. As seen in Fig. 2.9, the "Memoryless Simplification" approach produces a lower mean geometric error than the other approaches. In future work, the authors identified the extension of the method so that any two vertexes can be merged and a more accurate measure of volume deviation, using additional information on each triangle/vertex.

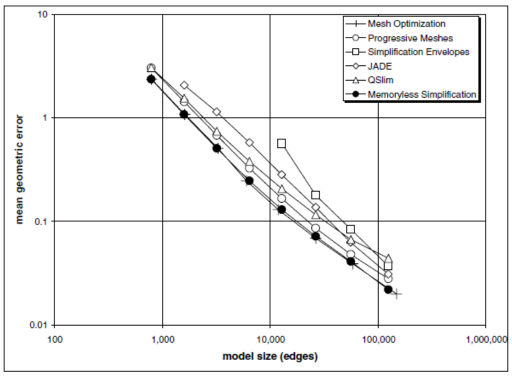
Table 2.4: Processing times for mesh simplification methods on the bunny and hand models, extracted from [Lindstrom and Turk, 1998].

Method	Bunny	Hand
Mesh Optimization	43m05s	10h51m45s
Progressive Meshes	8m20s	2h36m40s
Simplification Envelopes	9m26s	2h45m31s
JADE	5m25s	45m31s
QSlim	32s	3m56s
Memoryless Simplification	2m29s	20m59s

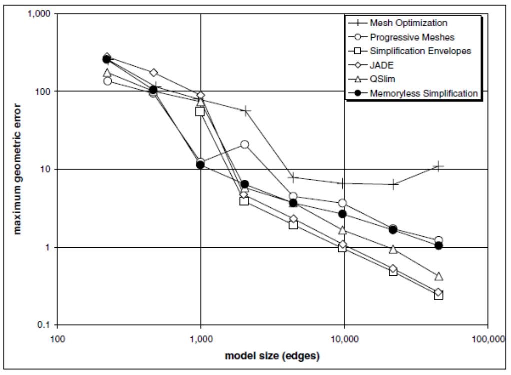
[Kanai et al., 2012] developed several appearance-preserving simplification methods for 3D CAD models by taking into account both the invisible parts and invisible form features of the parts. So, the methodology can be divided into two methods: the part simplification method, where the simplification is performed by removing invisible parts, and the feature simplification method, where invisible form features are removed. The authors also included an optional function that tasks into account movable parts that can change the visibility of some form features. The



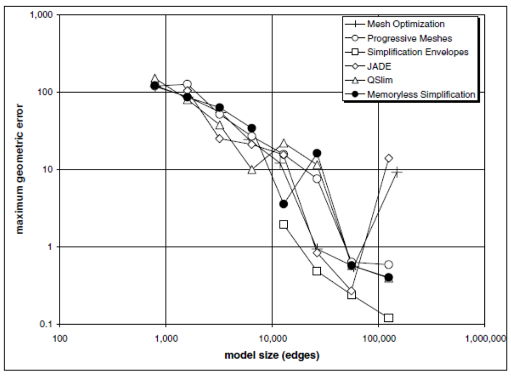
(a) Mean geometric error for the bunny model.



(b) Mean geometric error for the hand model.



(c) Maximum geometric error for the bunny model.



(d) Maximum geometric error for the hand model.

Figure 2.9: Mean and maximum geometric errors for the bunny and hand model, acquired from [Lindstrom and Turk, 1998]

simplification process is depicted in Fig 2.10. As a first step, the triangles' mesh modes per part and form feature are created and outputted to separate files. Then, the parts and features are rendered with different colors from multiple view directions using OpenGL, and the results, defined by four characteristics, are extracted from the frame buffer. These characteristics are then used by three evaluation indices to determine which parts and/or features are invisible. The invisible parts are then removed. Considering the invisible features, only those that are not needed for defining the visible form feature geometries can be removed. This verification is made by an API of the CAD system. The authors do not clarify which CAD system they are referring to, and it is unclear if they developed this API.

The results of the two simplification methods can be observed in Table 2.5. As we can see, when the model has few parts, the part simplification method fails to produce good results, while the feature simplification manages to reduce the model greatly. However, in the Boring machine and Stove burner, where the number of parts is substantially larger, the results of the simplification methods are similar. This suggests that feature simplification works much better than part simplification when the CAD model is composed of a few parts. Moreover, there is no comparison between the developed algorithm and other known simplification algorithms. In future works, the authors identify a possible extension of the proposed method to generate a cutaway model or 2D

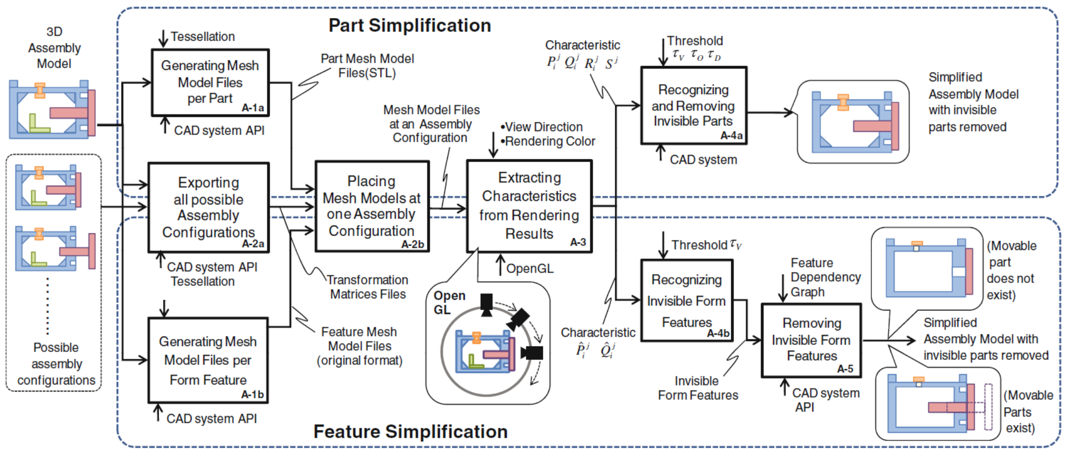

Part Simplification
Feature Simplification
3D Assembly Model
Possible assembly configurations
Tessellation
Generating Mesh Model Files per Part A-1a
CAD system API
Part Mesh Model Files(STL)
Exporting all possible Assembly Configurations A-2a
CAD system API Tessellation
Generating Mesh Model Files per Form Feature A-1b
CAD system API
Placing Mesh Models at one Assembly Configuration A-2b
Mesh Model Files at an Assembly Configuration
Transformation Matrices Files
Feature Mesh Model Files (original format)
•View Direction
•Rendering Color
Extracting Characteristics from Rendering Results A-3
OpenGL
Open GL
Characteristic
Threshold
Recognizing and Removing Invisible Parts A-4a
CAD system
Simplified Assembly Model with invisible parts removed
Recognizing Invisible Form Features A-4b
Feature Dependency Graph
Removing Invisible Form Features A-5
CAD system API
Invisible Form Features
Simplified Assembly Model with invisible parts removed
(Movable part does not exist)
(Movable Parts exist)

Fig. 2 Overview of the two appearance preserving simplification methods

Figure 2.10: Overview of the two appearance preserving simplification methods, taken of [Kanai et al., 2012]

drawings.

Another form of simplification model simplification is by simplifying the model mesh. [Salinas et al., 2015] proposed a method for this purpose that simplifies by decimating triangle surface meshes. They start discussing why extreme simplification of complex meshes is a difficult process. This is due to three main problems: the preservation of the coarse-scale structures, the filtering of the surfaces, and the presence of geometric noise and topological defects. The proposed method involves a pre-processing stage that detects a set of planar parts, called proxies, and estimates their adjacency via a proximity graph. This detection is restrained by an error tolerance from planarity specified by the user. So, the methodology developed takes as input a surface triangle mesh and a set of pre-detected proxies and generates a simplified mesh where the coarse-scale structures are preserved with regard to error metrics and specific rules. In addition to simplifying the mesh, this algorithm also preserves its structure and geometry. This is accomplished by incorporating the geometry of the proxies into the local error metric used for the edge collapses. The addition of the proxies makes the vertices automatically migrate to them, improves the robustness to noise, and reduces the local error. The proxies' structure is preserved during the decimation by establishing structure-preserving rules and not performing edge collapses that violate them. An overview of the procedure described above can be found in Fig. 2.11.

The method was implemented in C++ and tested using models of different complexity. Moreover, the results were compared against the approaches described in [Lindstrom and Turk, 1999], [Garland and Heckbert, 1997], and [Cohen-Steiner et al., 2004]. They can be found in Table 2.6. In general, the proposed approach takes significantly more time to perform the decimation than the others. However, this approach generated simplified models with fewer vertices and less mean error than the other approach. The authors identified some limitations of the approach, such as the final meshes may not perfectly align with the abstracted planes and the lack of usage of other decimation operators that could improve the quality of the results. Moreover, the proxies cannot

Table 2.5: Result metrics of the simplification processes, adapted from [Kanai et al., 2012]. PS refers to the part simplification method, while FS refers to the feature simplification method.

Model	Simplification method	# of parts	# of features	# of faces
Power supply	Original Model	6	161	1,800
	PS reduction's rate (%)	0	0	0
	FS reduction's rate (%)	33	64	70
Boring machine	Original Model	415	421	7,295
	PS reduction's rate (%)	54	54	48
	FS reduction's rate (%)	53	58	49
Sphygmograph	Original Model	59	538	9,153
	PS reduction's rate (%)	59	26	28
	FS reduction's rate (%)	68	56	69
Stove burner	Original Model	451	6,537	57,211
	PS reduction's rate (%)	68	57	56
	FS reduction's rate (%)	76	77	77

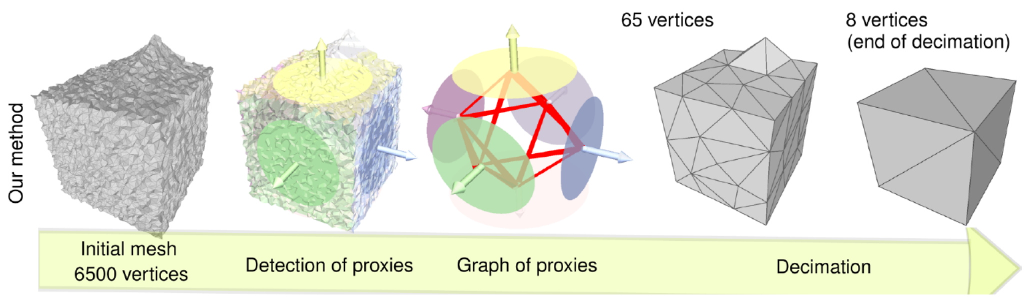


Figure 2.11: Overview of the simplification method by the decimation of triangle surface meshes, proposed by [Salinas et al., 2015]

Table 2.6: Result of the experiments of the simplification method proposed by [Salinas et al., 2015] (SLA). The experiments were performed on models of different complexity and compared against known approaches.

Input Mesh	No. vertices	No. simplified vertices	No. proxies	Decimation time (s)	Mean error	Method
Fandisk	12 000	81	41	2.9	0.0079	LT
				0.94	0.0064	GH
				10	0.0076	VSA
				5.3	0.0060	SLA
Chichen Itza	392 000	450	173	85	0.032	LT
				44	0.064	GH
				600	0.28	VSA
				250	0.034	SLA
Triumphal arch	13 631	115	70	2.9	0.11	LT
				0.65	0.33	GH
				5.3	0.042	SLA
Church	9 301	50	25	1.6	0.42	LT
				0.7	0.33	GH
				3.0	0.23	SLA
Paris	5 927 613	30 742	7 392	702	0.59	GH
				2, 990	0.52	SLA

approximate free-form or non-developable surfaces.

[Li and Nan, 2021] developed a automated simplification pipeline that filters and simplifies 3D building mesh models. The pipeline aims to reduce the geometric complexity of single buildings while preserving their structure. This method takes a triangulated mesh model as input and sequentially applies the following three main processes: mesh filtering, structure extraction, and mesh decimation. In mesh filtering, the vertex positions and normals are separated and treated differently. A bilateral filtering algorithm is applied to the normals, which are used then to update the mesh vertices. This process is then repeated from 10 to 20 times. The authors found that a good result could be achieved after ten iterations. After the filtering process, region and contour extraction is performed. The planar region extraction is done by executing a region growing method, where the triangles are traversed by their common edges and merged if their normals are similar enough. The algorithm stops when there are no more suitable candidates and a new growth stage starts from one of the remaining faces. For the contour feature extraction, the face edges are again traversed, and if two adjacent faces belong to different regions, a contour is identified. The final phase of the pipeline is a feature-preserving edge collapse. This operation is guided by hierarchical error quadrics obtained by analyzing neighboring faces and the extracted planer regions. All edges and associated costs are maintained in the priority queue, where the edge with the lowest cost is popped off the queue and collapsed. An overview of the above-described pipeline can be seen in Fig 2.12.

The proposed method was evaluated using several building mesh models generated by different techniques, such as photogrammetry techniques, surface modeling from LiDAR point clouds, and

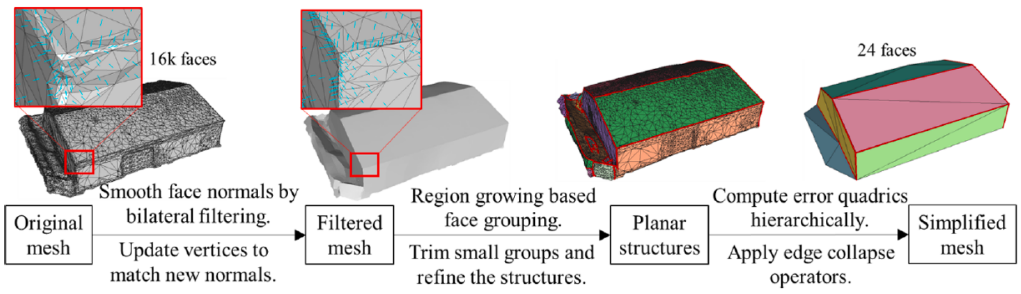

16k faces
Original mesh
Smooth face normals by bilateral filtering.
Update vertices to match new normals.
Filtered mesh
Region growing based face grouping.
Trim small groups and refine the structures.
Planar structures
Compute error quadrics hierarchically.
Apply edge collapse operators.
24 faces
Simplified mesh

Figure 2.12: Automated simplification pipeline developed by [Li and Nan, 2021]

Kinect fusion algorithm. Figure 2.13 depicts the results of the pipeline phases. It is important to point out that not all faces belong to a planar region. Moreover, Table 2.7 and Fig. 2.14 contain the statistical results of the proposed method. As we can see, the simplification achieves very low compress ratios and a low mean error. However, one limitation of this method is that it can only simplify one building at a time. Furthermore, when the target vertices are less than twice the number of planes set, the structure of the building begins to deteriorate rapidly. Also, the proposed method may not be this efficient when dealing with buildings that consist of large free-form surfaces. In future work, the authors identify a parser to transform the simplified building model into semantic components so users can more easily edit and analyze the models.

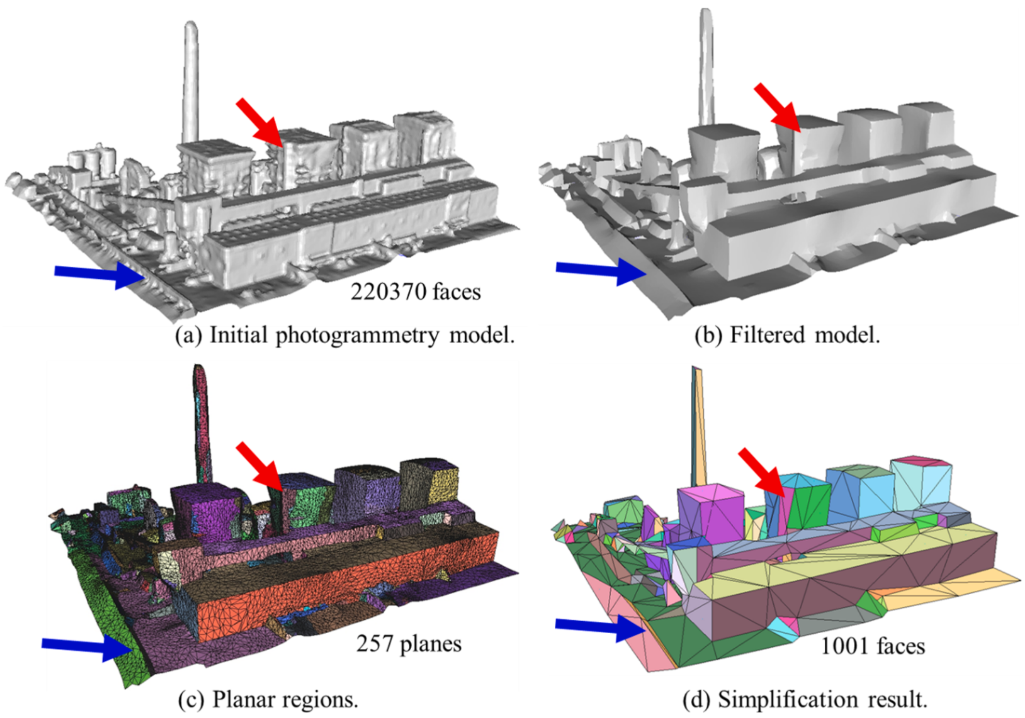


Figure 2.13: Example of the simplification method proposed by [Li and Nan, 2021], applied on the power station model

Table 2.7: Statistics of the simplification process results, adapted from [Li and Nan, 2021].

Model	Original vertex / face #	Result vertex / face #	Compress ratio
Office building	14,643 / 28,929	208 / 391	1.12%
Power station	100,543 / 220,370	514 / 1001	0.64%
Indoor flat	223,634 / 428,021	101 / 169	0.03%
Factory building	22,118 / 43,205	466 / 803	3.32%

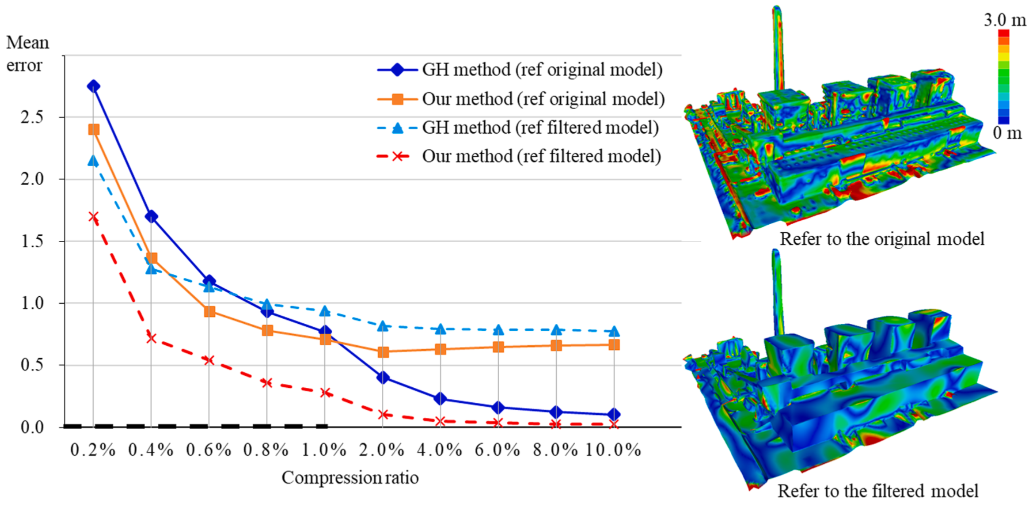


Figure 2.14: Comparison between the mean error of [Li and Nan, 2021] method and [Garland and Heckbert, 1997] method

2.3.2 Existing Technologies

In the previous section, we discussed possible technologies to convert the CAD model into a more suitable 3D format. This conversion process may generate a polygon mesh using a very large number of polygons to maintain the accuracy of the 3D model. Nevertheless, this level of accuracy is not needed or desired in this application. It would hinder the user interaction experience with the models. With this being said, in this section, we conduct an analysis of the existing 3D model simplification software according to the following criteria:

Simplygon is a simplification software owned by [Microsoft, 2023]. It isn't open-source and has two tiers: a free tier, where a user can perform up to two hundred processing a day, and paid tier, with unlimited processing. Simplygon simplifies the model by performing mesh decimation and can remove invisible objects. However, it can't remove objects by volume. It provides a Python API so that any user can automate the process.

PyVista [Sullivan and Kaszynski, 2019] is a free and open-source Python library for the visualization and analysis of 3D models. It uses at its core the Visualization Toolkit (VTK) [Schroeder et al., 2006], which is a visualization library written in C++. In addition to visualization, it allows the simplification of models by mesh decimation. It also allows a user to write scripts to automate the simplification process.

PyMeshLab is a Python library created by [Muntoni and Cignoni, 2021] for visualizing and editing 3D models. It achieves this purpose by providing a Python interface with the MeshLab software [Cignoni et al., 2008]. To perform the mesh decimation, PyMeshLab implements a variant of the [Garland and Heckbert, 1997] algorithm. It allows a user to write a Python script to decimate the mesh of a model and remove any object by volume.

All tools analyzed perform mesh decimation to simplify the model. PyVista and PyMeshLab are very similar, considering the set criteria. There are other simplification tools not discussed, such as OpenSceneGraph (OSG), a 3D graphics toolkit that can be used for simulation, scientific visualization, and modeling, amongst others. There is also the VulkanSceneGraph (VSG) tool, the successor of OSG, that uses the Vulkan API as its core to render and visualize the models. However, both these software have some problems: OSG is no longer being developed, and VSG is new and not mature enough.

2.3.3 Summary

Several papers have been written about model simplification and conversion in the past few decades. We will now briefly overview the papers discussed in this section and provide their main results, weaknesses, and limitations.

One of the main papers on geometry simplification is [Lindstrom and Turk, 1998]. This paper, alongside some others of the same time period, laid the groundwork for all future papers about model simplification. In fact, most of the recent papers are improvements or adaptations of their work.

The most common simplification operation used is the edge collapse since that, with a proper error metric, it can preserve the structure of the geometry being simplified. Most of the analyzed papers focused on simplifying the model by simplifying its geometry. Only [Kanai et al., 2012] proposed a method where the CAD model is simplified by removing its invisible parts and features.

There are a few papers discussing the conversion between the CAD and 3D formats. Only one paper [Lorenz et al., 2016] was analyzed, but it only allowed conversion to the VRML format and relied on third-party tools. One explanation for this lack of papers is that most CAD systems allow the conversion of the CAD model to multiple formats, some of which are 3D formats.

Table 2.8 provides the discussed papers' main results, strengths, and limitations. One of the main conclusions drawn from this study of existing papers is that there is a lack of a unified approach that simplifies the geometry, removes unnecessary parts and features, and converts the model to an appropriate format. With this in mind, we propose a preprocessing pipeline that fills that gap.

Considering now the discussed technologies, we verified that were not many mature and free tools. From the ones analyzed, we can observe that no tool satisfies all established criteria. In the

Table 2.8: Summary and comparison of the papers related to simplification and conversion of CAD models.

Reference	Results	Strengths	Limitations
[Lindstrom and Turk, 1998]	Fast and efficient method for geometry simplification through edge collapses	Memory efficient and fast. Preserves the shape of the model	Can only merge two connected vertexes
[Kanai et al., 2012]	Simplification of 3D models by removing invisible parts and/or features	Takes into account features and moving parts	Relies on 3th-party tool
[Salinas et al., 2015]	Decimation of triangle surface meshes by using a set of planar proxies constructed a priori	Small mean error, and high level of reduction	Output mesh may not be perfectly aligned. Can't approximate free-form or non-developable surfaces
[Li and Nan, 2021]	Automated pipeline for the filtering and simplification of triangular mesh building models	No user interaction. Low compression ratio and mean error. Applies filtering for noise removal	Only one building at a time. Doesn't work well with large-freeform surfaces. Low target vertices lead to loss of building's structure

next chapter, analyze the tools further by considering the requirements of the project. Table 2.9 presents a summary of the comparison performed between the analyzed tools.

Table 2.9: Summary of the comparison between several conversion tools. The comparison was made regarding if it is an automatic conversion, if the tools perform any geometry optimization, and if it is open source and free. Semi-automatic means that it allows scripting, but the software needs to be open and running.

Tool	Mesh decimation	Automatic	Removes obj. by vol.	Removes obj. by vis	Free and Open source
Simplygon	Yes	Yes	No	Yes	No
pyVista	Yes	Yes	Yes	No	Yes
PyMeshLab	Yes	Yes	Yes	No	Yes

2.4 BIM Projects Discussion and Analysis

This section is dedicated to BIM systems. BIM originated from the Georgia Tech School of Architecture in late 1970 [Latiffi et al., 2014]. It was first introduced as Building Description Systems

(BDS) with the purpose of storing complex physical information about buildings. Next, Graphical Language for Interactive Design (GLIDE) was introduced in 1977 as an expansion of BDS since it covered more building elements and could evaluate the structure design. In the following years, several systems were proposed that fixed problems or improved the previous building software by including more of the project information. Nowadays, BIM includes information on all of the phases of a building project, from design to maintenance [Latiffi et al., 2014].

We analyze several types of BIM, such as WebBIM, 4D BIM, and BIM+GIS integration. WebBIM concerns the rendering of BIM models on the Web. This can pose several problems since these models tend to be large and complex. 4D BIM focuses on displaying the BIM models with the temporal changes represented on the models, and BIM+GIS deals with the integration of Geographic Information System (GIS) data into the BIM model. We end by summarizing the strong and weak points of the discussed articles and the main features present in every proposed system.

In the last decade, several papers were published with the purpose of bringing BIM models to the web using mainly WebGL or a framework based on it. This brings several advantages, including cross-platform availability, easier visualization and manipulation of the models, and no demand for third-party plugins or software. This section presents some of the most relevant papers found.

In [Liu et al., 2021], a general methodology was developed to view differential expressions for airport pavement quality on the web. The presented methodology can be found in Fig. 2.15. It starts by processing the BIM model using Revit[6] and extracting it to the JSON format. Then, the JSON file was transformed into OBJ [Wavefront, 2022] and rendered by Three.js. The authors chose the OBJ format because it is lightweight, but there is no discussion of alternative formats that may be better suited for the task at hand. According to the authors, Three.js was chosen because it's easy to learn, supports several 3D file formats, and provides a set of functions to interact with the model. Moreover, the paper does not go into great detail about how the interface in Three.js was constructed.

The methodology was tested on an airport that is 584m long and 179m wide to display some of its pavement's properties, such as the pavement condition index (PCI). The result can be seen in Fig. 2.16. The user can move through the scene, export objects, and view and modify their properties. No objective metrics were provided. The paper concludes that displaying 3D models on the web efficiently provides the information needed for BIM management (most precisely, airport management) and calls for future work in loading JSON directly into WebGL.

[Prandi et al., 2015] developed a framework that allows efficient web-based visualization of large city models based on Open Geospatial Consortium (OGC) standards. For this purpose, a three-level architecture was created (Fig. 2.17). The application level renders the models on the browser. It allows user interaction, while at the information level, data is stored in CityDB format to more easily integrate data from different data sources and standard 3D formats for efficient

[6]available at `https://www.autodesk.in/products/revit/overview?term=1-YEAR&tab=subscription`

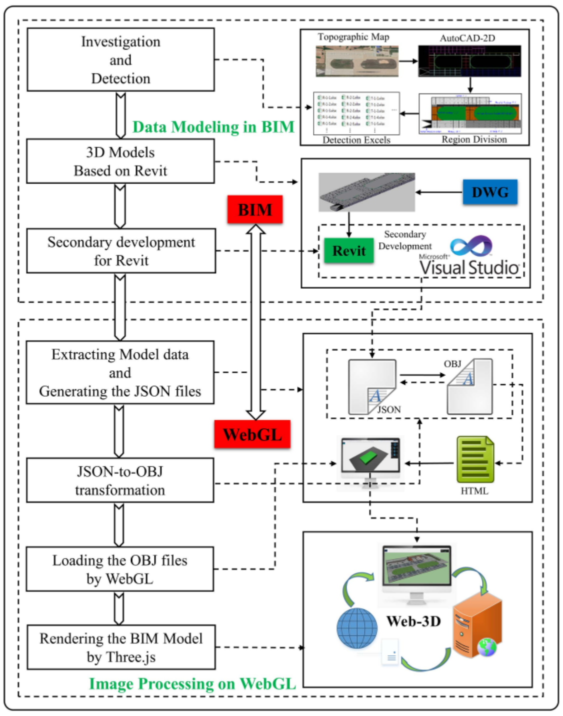
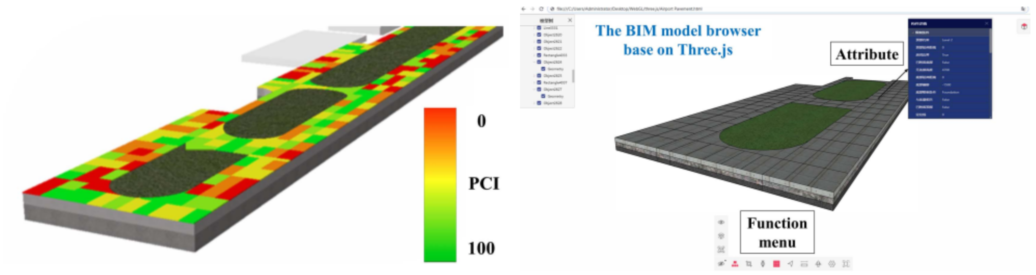

Investigation and Detection
Data Modeling in BIM
3D Models Based on Revit
Secondary development for Revit
Topographic Map
AutoCAD-2D
Detection Excels
Region Division
BIM
DWG
Revit
Secondary Development
Visual Studio
Extracting Model data and Generating the JSON files
WebGL
JSON-to-OBJ transformation
JSON
OBJ
HTML
Loading the OBJ files by WebGL
Rendering the BIM Model by Three.js
Web-3D
Image Processing on WebGL

Figure 2.15: Visualization methodology for airport pavements proposed by [Liu et al., 2021]

visualization of 3D models. Finally, the middleware level provides services for querying and manipulating data on the information level and provides the geometric data needed for visualization at the application level.

In this paper, three use cases were presented. The first case study uses Nasa Java World Wind (NWW), a Java-based visualization tool, to render a model of the city of Rotterdam. The CityGML file is converted to C3D, an OBJ-compliant file format, optimized for size and parsing speed, that contains hierarchical, geometry, and reference model data. This is then rendered by the NWW. The authors obtained an FPS greater than 60 while using models with tens of millions of triangles on an "average PC" using this approach. The specification of the PC is not disclosed. The second case study is similar to the previous one, with the difference that Cesium [Cesium GS, Inc., 2022b] replaces NWW in the client. This led to reduced performance and memory issues. The final case



(a) Visualization of PCI on BIM model (b) Functional interface of BIM browser

Figure 2.16: Visualization results obtained by [Liu et al., 2021]

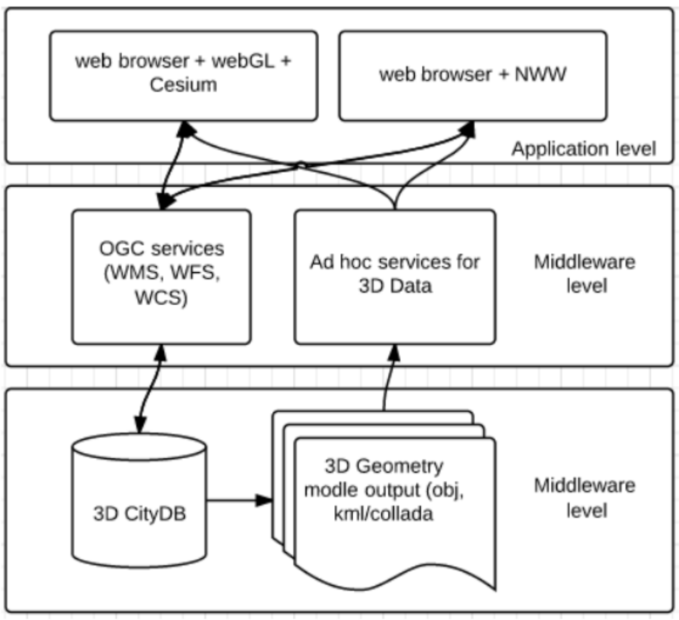
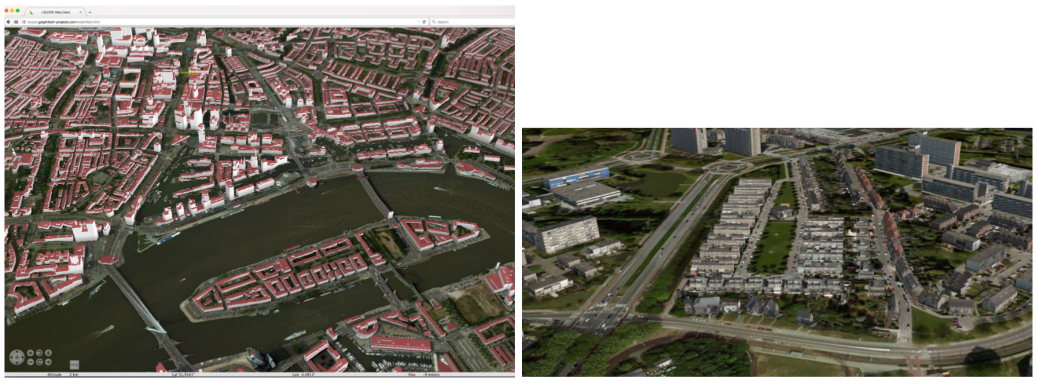

web browser + webGL + Cesium
web browser + NWW
Application level
OGC services (WMS, WFS, WCS)
Ad hoc services for 3D Data
Middleware level
3D CityDB
3D Geometry modle output (obj, kml/collada
Middleware level

Figure 2.17: Architecture of the CityDB models visualization system proposed by [Prandi et al., 2015]

study uses, instead of COLLADA, the glTF format for faster loading of the 3D models. The authors first convert the 3D CityDB files to COLLADA and then feed them to the COLLADA2GLTF converter (developed by the Khronos group). This converter has two outputs, the glTF files containing the models and a JSON map with geo-referenced data for each component. The case study results are presented in Fig. 2.18. The authors conclude by reiterating the performance and memory problems of using Cesium, and by requesting future work on combining glTF and Cesium as a way to solve those problems.



(a) Result of case study 1 (b) Result of case study 3

Figure 2.18: Adapted results of the CityDB models visualization system described in [Prandi et al., 2015].

A visualization system called WebBIM was developed by [Zhou et al., 2018], which uses Industry Foundation Classes (IFC) and WebGL to display BIM on web browsers. The paper starts by defining the system's characteristics, such as being light-weighted, since the data must be transported through a network. Moreover, it must be cross-platform since the increasing use of mobile devices and similar devices may lead to a need to visualize BIM models in all sorts of devices. It must be open, meaning it should comply to open standards. Considering the above characteristics, the authors recommend the usage of WebGL and IFC.

The proposed framework, present in Fig. 2.19 has three modules: TriCon, InsMod, and WebGL-RM. The TriCon receives as input the IFC file and transforms the geometry into triangles that WebGL can render. Then, InsMod compresses the triangles and reuses the geometry description for all triangles generated by the same faculty component. Finally, the WebGL-RM renders the compressed triangles into the browser.

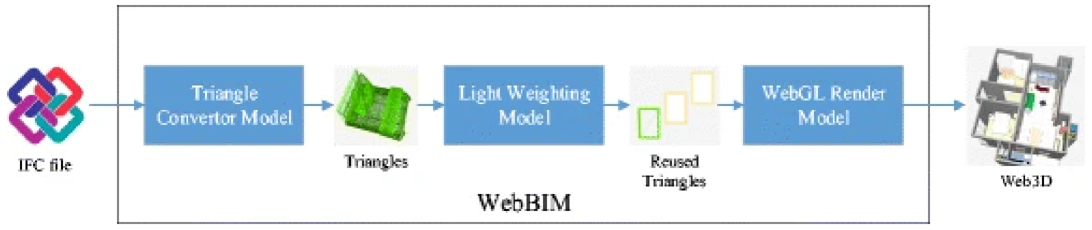


Figure 2.19: The WebBIM framework, proposed by [Zhou et al., 2018]

To test the efficiency of the framework, several empirical studies were conducted. 100 BIM files were compressed and loaded through the framework for this effect. The results can be found in Fig. 2.20. As we can see, depending on the selected BIM model, the results can vary greatly, mostly between a data compression rate of 20% and 45% and a load time declination rate of 10% and 25%. The authors do not address the high variance of the results or ways to mitigate it. They state that they could load a BIM file with millions of IFC objects, triangle faces, and vertices, maintaining an FPS of 25 while interacting with the model. There are no other mentions of different BIM files being tested. Finally, a cross-platform test was implemented using Chrome on a computer, an iPhone 6, and an Android Pad web browser. The authors do not mention which Android Pad nor the FPS obtained, only saying that " (...) WebBIM has an acceptable performance in all these devices.". Moreover, only basic visualization interactions are supported by the system. They conclude by noting that the intended goal of the paper was achieved and by mentioning that WebBIM can be applied to 4D simulation, safety analysis, indoor route planning, and other applications.

[Fassi et al., 2015] developed a web visualization tool to visualize and manipulate the model of the Milan Cathedral. It starts by establishing the desired characteristics, such as it must use open standards and software and must be able to visualize and provide interaction for large 3D models. Moreover, it should be user-friendly for experts and non-experts alike and be able to be used collaboratively. The built system, named BIM3DSG, has three main functionalities. First, 3D specialists can add new models or modify existing ones. Also, any user can visualize the complete model or parts of it in conjunction with the BIM information for every object. Furthermore, there exists a central database with all 3D models and additional files needed. There is no mention of

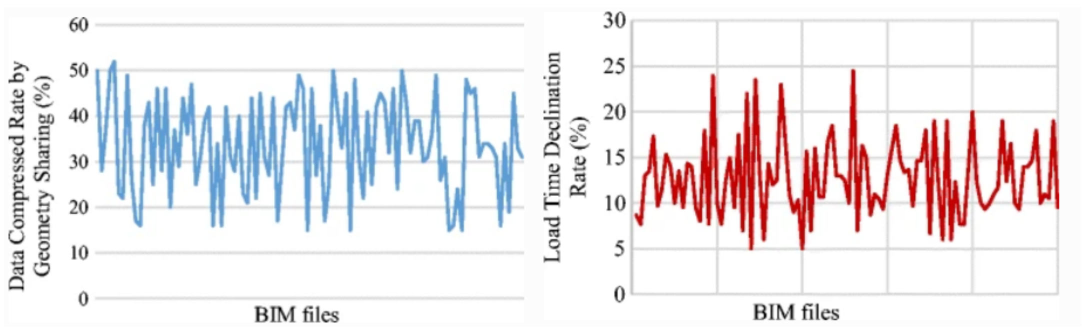


Figure 2.20: Adapted results of compression and loading obtained by [Zhou et al., 2018]

which CAD systems and formats were used. The system's schema can be seen in Fig. 2.21.

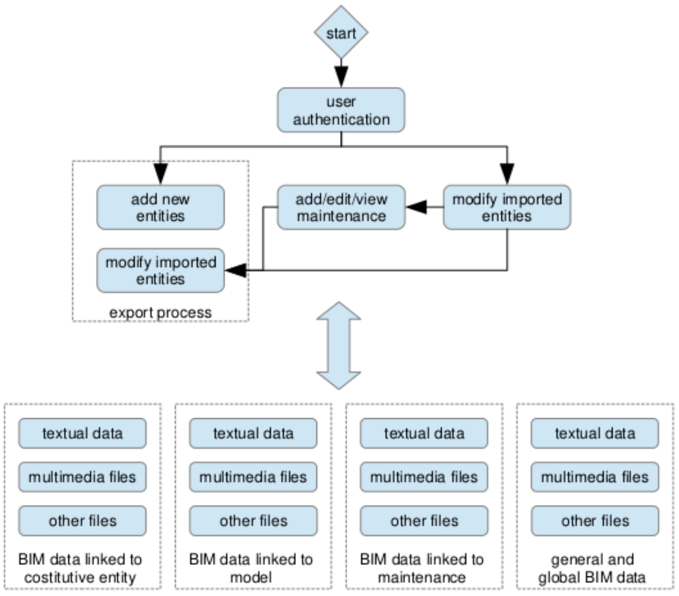


Figure 2.21: BIM3DSG schema from [Fassi et al., 2015]

According to the authors, what sets the system apart from others is the ability to split the model into different constitutive entities and store them separately in the database and the option to load only a specific user-requested part of the model. The article mentions an upgraded version (BIM3DSG v2), which is more efficient and adds new functionalities, such as text support and the dynamic manipulation of the information fields. However, no follow-up paper was found. A case study of the Milan Cathedral was presented, where the user could introduce maintenance activities with a description, manage and visualize maintenance objects with different colors according to their characteristics, perform queries to get detailed information about the model, and many others. An example of the utilization of the system can be found in Fig. 2.22. The paper concludes by affirming the built system meets the previously established criteria and that a future system extension for general architectonic and archaeological case studies is in progress.

Furthermore, [Wan-Li et al., 2016] created a collaboration system for facility management of BIM models called V3DM+. They set three research goals: the system should be able to integrate

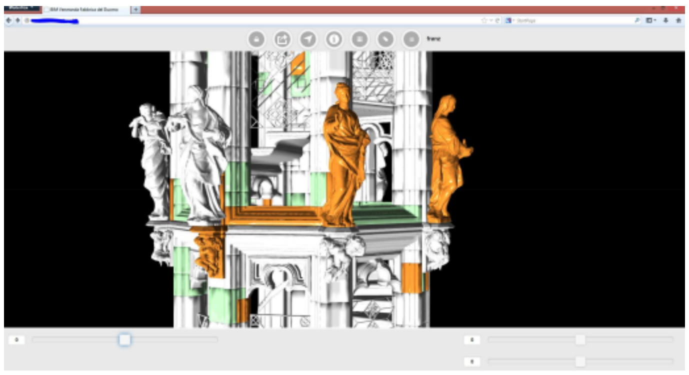


Figure 2.22: Example result of the case study from [Fassi et al., 2015]

3D models from different BIM tools, should be intuitive to use, and the users must be able to find the needed information efficiently and initiate a discussion about it.

The proposed system is composed of two main modules: the data arrangement and the data presentation modules. In data arrangement, the files from different BIM tools are converted into the same format, FBX. The reason for choosing this file format is not included in the article. Moreover, the models are integrated and standardized to a central model, with the associated information being stored in a database and linked to the models by a specific ID. The module also includes a mechanism to update modified data. The data presentation module consists of two main functionalities: data revealing and data collaboration. The data-revealing function focuses on presenting the information in an efficient and intuitive way. For this purpose, the authors chose a game engine to display the 3D models. It also allows users to execute queries to retrieve relevant information. The data collaboration function allows professionals and managers to identify problems with equipment and assign maintenance tasks. An overview of the system described above can be observed in Fig. 2.23.

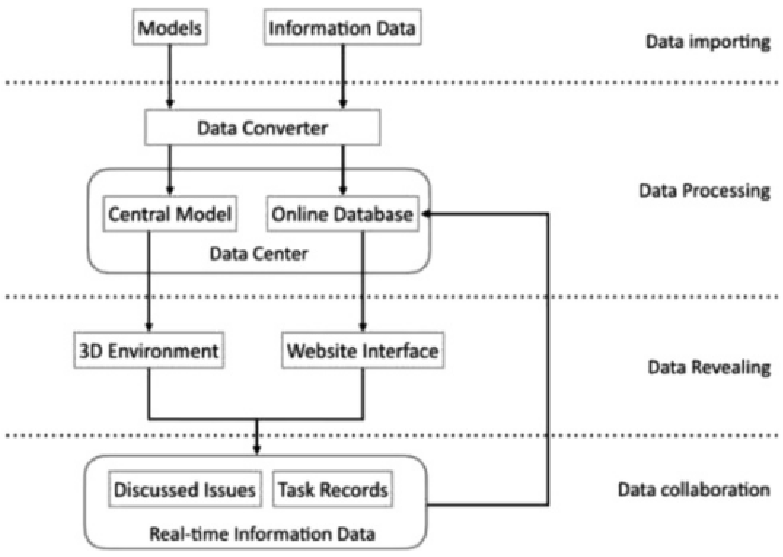


Figure 2.23: V3DM+ system architecture of [Wan-Li et al., 2016]

Likewise, [Wan-Li et al., 2016] conducted an accessibility analysis on the Tamsui Civil Sports Center to test the proposed system. An example of the result obtained can be seen in Fig. 2.24. They first analyzed the management demands and then implemented and optimized the system for the sports center. The performance was evaluated only subjectively, using the feedback of the center's manager, and no fault was identified. The authors called for future work in the automatic importing and presentation of the data.

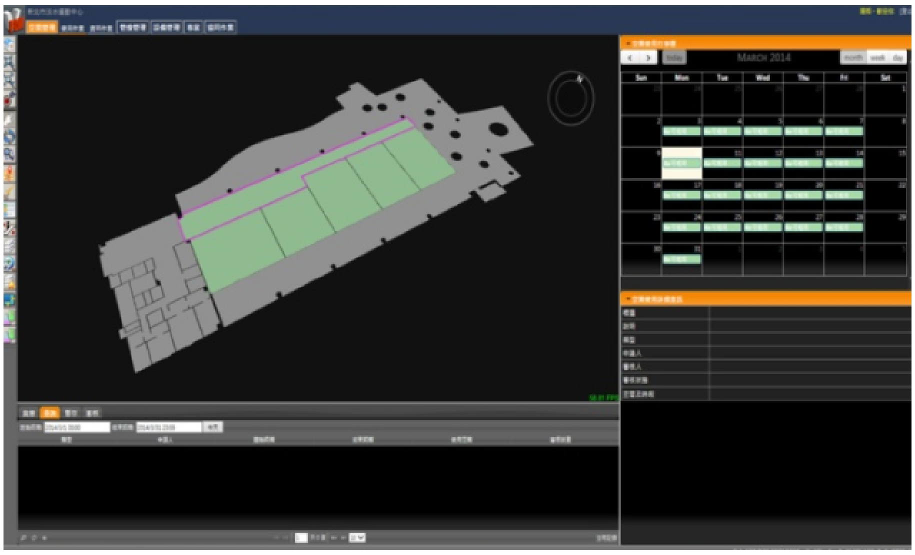


Figure 2.24: An example result of [Wan-Li et al., 2016]

4D BIM consists of integrating the project schedule into the BIM model. The 4D represents the three dimensions of the BIM model plus the time dimension. This allows users to visualize the project construction process over time and to plan and coordinate with each other.

In [Park et al., 2017], a methodology is developed based on a central database for the visualization of 4D BIM models. A UML diagram of the methodology can be seen in Fig. 2.25. The user starts by accessing the Web Viewer, which queries the database. The results are grouped into six different tables based on their progress status. This status is derived automatically from the construction schedule in the database. Dynamics views of the objects are constructed based on this information and sent to the Web Viewer for display. The construction schedule table is updated daily, and a backup is made of the previous date. For this methodology to work properly, the 3D BIM models must be in a common open format, such as IFC, and the project schedule must be updated daily.

The proposed methodology was applied to three case studies using the same construction project with the same BIM model and schedule data. Each status type is associated with a different color, except for the finished status, which has no highlighting color. The graphical interface was built using xBIM Toolkit modules, C#, and ASP.NET, and the database was built using MS-SQL Server 2014. The first case illustrates the construction model on May 2, 2016, along with the status of all tasks, e.g., the trowel finish work is on time. However, the exterior and interior metalwork is still under construction, amongst other tasks. The second case study shows the construction model

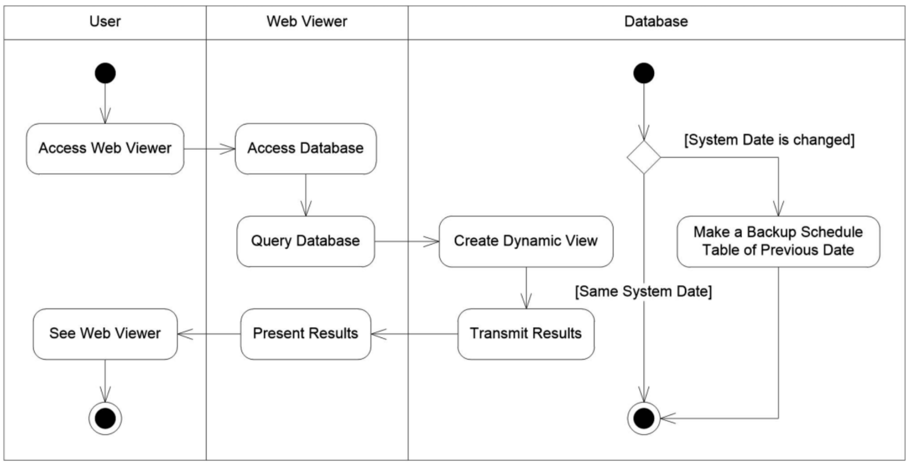
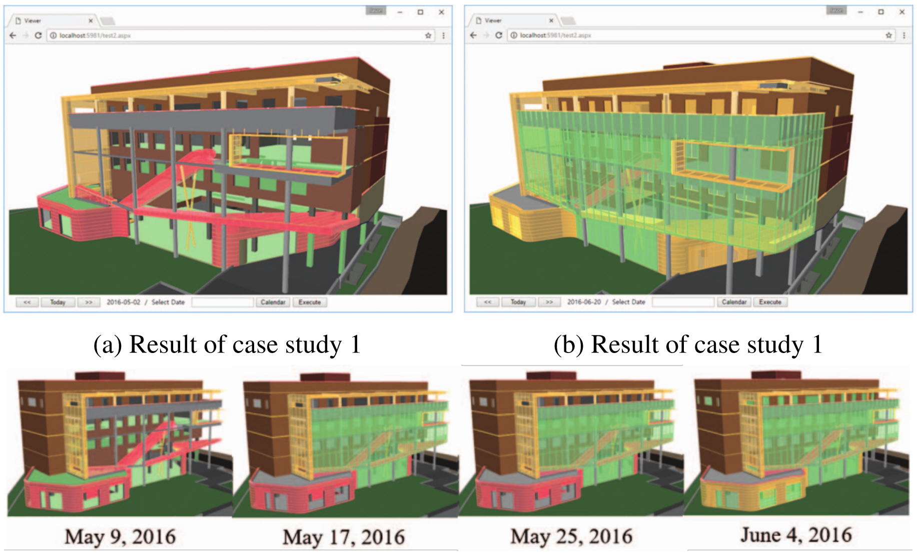


Figure 2.25: UML diagram of methodology proposed by [Park et al., 2017]

on June 20, 2016, with different statuses for the tasks. Finally, the third case study shows the evolution of the project from May 9 to June 4, 2016. [Park et al., 2017] calculated the program's efficiency by measuring the page-loading time without data downloading time, which was 1.4s. No explanation was provided for why data downloading time was excluded, nor any other metrics were discussed. The results can be observed in Fig. 2.26. The authors conclude that the proposed methodology helps the decision-making of construction project operations with information delay and data inconsistencies. They identify as future work an analysis of the cost of implementing the methodology versus its benefits and the automatic update of the status attribute of the schedule table by comparing a real photo with the BIM model.



(a) Result of case study 1 (b) Result of case study 1

(c) Result of case study 3

Figure 2.26: Results of the case studies presented by [Park et al., 2017]

Geographic Information systems (GIS) are systems that store, analyze and manage geographical data related to a certain location. Their integration with BIM models has been studied for years since it allows them to display detailed 3D models alongside their geographical data.

[Zhang et al., 2020] developed a method and a BIM/GIS Integrated Platform (BGIP) to integrate the project information and models into a unified environment without conversion of standards or third-party plugins. A diagram of the implemented method can be seen in Fig. 2.27. The BIM server is composed of three parts: an IFC parsing component, a database, and a data access interface. The parsing component extracts and parses all structures present in the IFC file and stores them in the database. Several interfaces were implemented for the web service to query the needed data. The GIS server contains geographical and terrain information about the project. The data from both servers are integrated and shown to the user via a WebGL container. The BIM and GIS data fusion is executed using four main technologies: 3D visualization of the GIS data, coordinate transformation between BIM and GIS, rendering of large models, and information retrieval, display, and interaction.

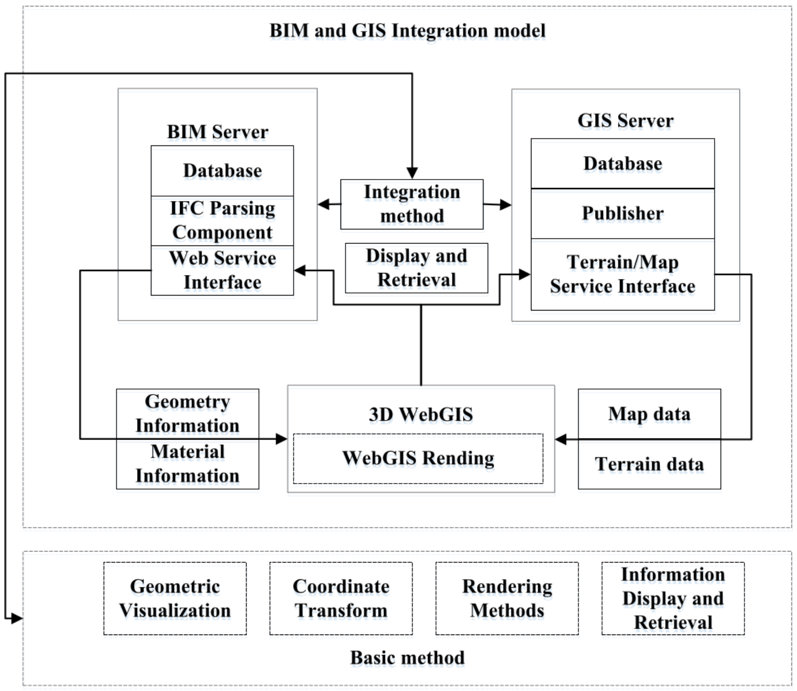


Figure 2.27: Implementation diagram for the BGIP developed by [Zhang et al., 2020]

A prototype system was implemented to test the proposed methodology. Its architecture can be seen in Fig. 2.28. The system is comprised of three layers. The presentation layer is responsible for displaying the BIM models and the corresponding GIS data and providing tools for interaction. The data layer stores the models and all related information. The application layer provides services for the presentation layer to access the needed information. [Zhang et al., 2020] applied the proposed system to the hydropower project Nuozhandu in Yunnan Province. The system provided several functionalities, including visualization and manipulation of the models (zooming, navigation, rotation, etc.), safety monitoring and analysis, and facility management. The results can be seen in Fig. 2.29.

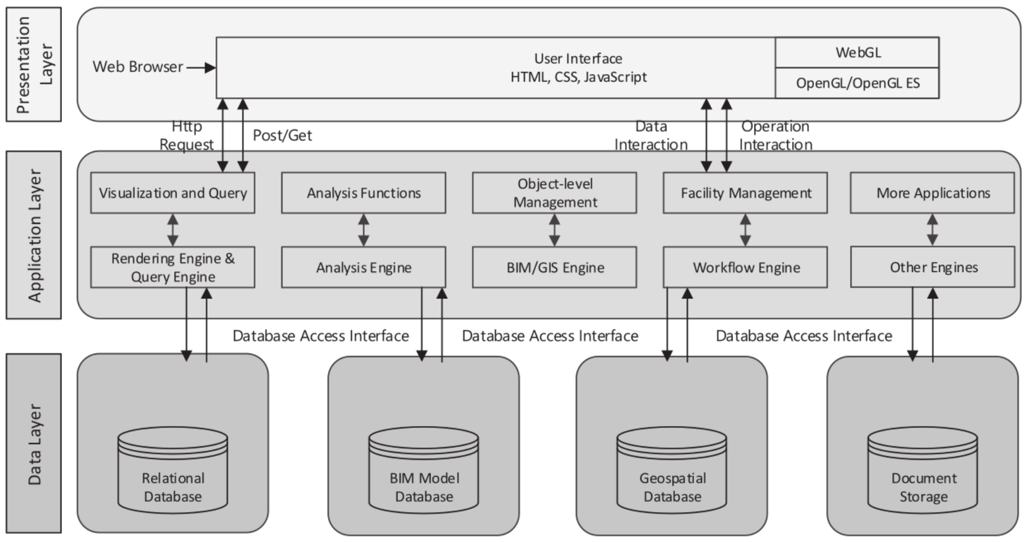
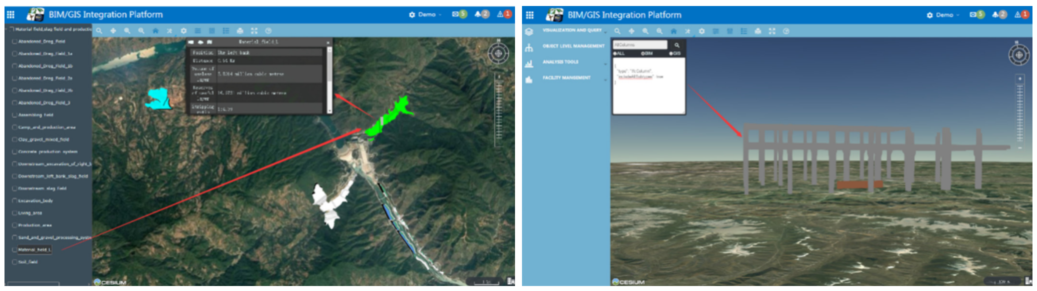


Figure 2.28: System architecture of [Zhang et al., 2020]

The authors conducted interviews and surveys to evaluate the implemented system with people with experience in BIM/GIS systems. The performances were evaluated using two criteria: the comparison of task efficiencies and benefit analysis. Overall, the results imply that the system is more efficient than the existing method. However, the system received low scores in terms of stability and operability. [Zhang et al., 2020] justify this by saying that the generation and rendering of the models were not simple for the users, and some previous habits may have hindered the experience. There was no explanation that considered the faults of the system, and no objective metric was provided. The authors identify some limitations, such as the available models in the GIS domain were not considered. Moreover, the system was implemented for a highly specific case, and the advantages provided may not hold for other types of projects. Also, they address the lack of measurement of the data streaming efficiency. As further work, [Zhang et al., 2020] called for research under more realistic conditions.

Having analyzed several articles about BIM, we now summarize this analysis, comparing each one's strong and weak points and identifying what is missing in the literature.



(a) Displaying a BIM object and its' properties (b) Query fetching all columns

Figure 2.29: Adapted results of the case study presented by [Zhang et al., 2020]

Starting with the conversion of CAD formats, most articles impose a neutral format (such as IFC) to the importing process, while others use propriety formats or provide an API for format conversion. Some articles perform model preprocessing in the form of model simplification (by removing unnecessary objects or simplifying geometries) and/or compression.

Most of the analyzed articles integrate some type of information into the BIM model. [Zhang et al., 2020] integrates GIS data into the model, [Park et al., 2017] integrates the objects' schedule, and the others mostly integrate maintenance tasks or specific object properties. This allows an efficient visualization and extraction of the needed information using only the model viewer. Only [Zhou et al., 2018] doesn't integrate any type of information since it focuses on efficient visualization of models through the Web.

Some articles provide this feature in terms of the automatic update of information, which means changing data in the database and being automatically displayed on the model. However, this is not a common feature in the articles analyzed. Besides, only three articles allow the visualization of the history of the properties of an object, and none display future predictions or scheduled tasks.

Most of the articles discussed developed complex visualization interfaces. This means these interfaces allowed simple visualization techniques, such as rotation, translation, zooming, etc., as well as some more evolved techniques, like performing queries and viewing an object and its properties in detail. Furthermore, all articles lacked testing. Some only evaluated their implementation with subjective metrics, and even the articles that used both types presented few metrics.

The results, strengths, and limitations can be seen in Table 2.10, and the main features in Table 2.11. To the best of our knowledge, there is no article or tool that has all the needed characteristics for the proposed project. So, the system developed will fill this "gap" by implementing all discussed features and meeting the given requirements.

Table 2.10: Summary of the analyzed articles in the literature review.

Reference	Results	Strengths	Limitations
[Liu et al., 2021]	Efficient display of PCI in airport pavement model based on Revit and Three.js	Integration of PCI information with BIM model	Intermediate conversion to OBJ. Dependence on Revit to generate JSON. No objective metrics
[Prandi et al., 2015]	Vis. of BIM model with geospatial data. Tested on Rotterdam City	Integration of geospatial data into a 3D model. Usage of standard formats	Intermediate conversion to Collada. Memory and efficient problems when using Cesium+Collada
[Zhou et al., 2018]	Vis. system based on IFC and WebGL. Avg. data compression of 30% and avg. loading time reduction of 12%. 25 FPS achieved	Neutral format. Data compression. Cross-platform	Other CAD formats and complex vis. techniques not supported.
[Fassi et al., 2015]	Tool to interact with BIM model on Web. Tested on Milan Cathedral	Automatic model update. Allows to add maintenance activities and differentiate by object features	Little mention of CAD compatibility. No objective metrics
[Wan-Li et al., 2016]	V3DM+ vis. system for Facility Management	Allows the database update and assignment of maintenance tasks	Limited testing. No model compression
[Park et al., 2017]	Vis. system for 4D BIM models. Page-loading time of 1.4s without data downloading	Daily update of status. Efficient vis. of an object by status using colors	Limited metrics used. Can't perform model queries
[Zhang et al., 2020]	Platform that integrates BIM models with GIS data	Efficient data fusion of BIM and GIS. No third-party software. Neutral format	Case study is very specific; benefits may not hold for others. No objective metrics

Table 2.11: Main features of analyzed articles. History is the ability to display both past and future properties of objects. Download refers to the possibility of downloading the current view of the model. Metrics refers to the type of metrics (objective, subjective, or both) used to evaluate the developed systems. Vis. (Visualization) complexity indicates which visualization techniques were implemented. Simple refers to rotation, translation, and zooming, while Complex refers to simple techniques, queries, and display of selected objects' information.

Reference	Model preprocessing	Integration of information	Automatic update	History	Download	Metrics	Vis. complexity
[Liu et al., 2021]	No	Yes	No	None	No	Subj	Complex
[Prandi et al., 2015]	No	Yes	No	None	No	Both	Complex
[Zhou et al., 2018]	Yes	No	No	None	No	Both	Simple
[Fassi et al., 2015]	Yes	Yes	Yes	Past	No	Subj	Complex
[Wan-Li et al., 2016]	No	Yes	Yes	Past	No	Subj	Complex
[Park et al., 2017]	No	Yes	Yes	Past	No	Both	Simple
[Zhang et al., 2020]	Yes	Yes	No	None	No	Subj	Complex

2.5 3D Visualization Frameworks

A high-level visualization framework is highly recommended for the visualization of complex 3D spatial-temporal data. It provides a high level of abstraction that speeds up the development and softens the learning curve. One may develop their application only using a low-level library, such as WebGL [Mozilla Foundation, 2023b], but it would be a slow, costly, and prone to error process.

In choosing the appropriate framework, we set a set of core requisites. Firstly, it should be a web-based framework since the project will be deployed to the CERN website and would significantly help with integration. It must also be compatible with most modern browsers.

As stated in the previous chapter, we need a conversion process to convert from the CAD format to the 3D format. This diminishes the importance of compatibility between CATIA and the chosen framework. However, it should still be compatible with a wide variety of 3D formats, preferably ones that allow fast and efficient loading and interaction of the models. The choice of file format will be discussed in the following section.

The framework documentation is an important choice criterion since it will be the main resource to answer and clarify questions about the framework. We evaluate it from three angles: the number of tutorials and examples, the existence of an API reference, and a built-in editor for development and testing. Besides this, the community should be active and large for the same reasons as the documentation. In the context of this analysis, we consider a sizable community one with more than 200 people. We assess this by analyzing the GitHub [7] statistics (the number of

[7] https://github.com/

forks and stars), which gives a sense of how popular the framework is, the existence of a forum, the number, and size of dedicated social groups like Discord [8] and Reddit [9], and the number of questions associated to the framework tag on Stack Overflow [10]. A high number of questions means that the framework is largely discussed, and it increases the probability that problems encountered were already discussed and solved.

Furthermore, the framework should be open-source. This provides an extra level of security and transparency since the code is available for everybody to analyze. It should also be, if possible, free.

According to the previously set criteria, six frameworks were analyzed.

Three.js is an open-source and free-to-use 3D visualization framework that is built on top of WebGL. A vast majority of browsers support WebGL, so Three.js must also be supported. It uses Javascript to visualize and interact with the 3D models [Three.js community, 2022c].

This framework has a big and active community, with a dedicated forum where anyone can ask and answer questions and a server on Discord with almost 10k members (available at `https://discord.com/invite/56GBJwAnUS`). It has a tag on Stack Overflow with around 20k questions [Stack Overflow, 2022d] and a Reddit Community with 12k members (available at `https://www.reddit.com/r/threejs/`). The framework currently has 33.2k forks and 86.3k stars on GitHub [GitHub, 2022].

Three.js has extensive documentation, a basic tutorial, an extensive list of examples and API references, a manual with more in-depth tutorials, and a built-in editor. Three.js is also compatible with several 3D formats, such as glTF, FBX, OBJ, and COLLADA [Three.js community, 2022a].

Three.js is licensed under the MIT License [MIT, 2023], meaning that commercial use of this framework is allowed [Three.js community, 2022b].

Babylon.js is an open-source web-based framework. It is compatible with WebGL and WebGPU [11] and uses Javascript to view and manipulate the 3D model [Microsoft, 2022c].

The framework provides a forum for its users to post and answer questions. However, it has a low presence in Stack Overflow, with only 450 questions [Stack Overflow, 2022b], and in Reddit, with only 782 members in the community (available at `https://www.reddit.com/r/babylonjs/`). Moreover, no sizable Discord server was found for this framework. At the time of writing, Babylon.js has 3k forks and 18.7k stars on GitHub [Microsoft, 2022b]. Overall, it is fair to say that Babylon.js has a medium size community.

Furthermore, this framework provides high-quality documentation with an API reference, a list with a sizable amount of examples, and several tutorials. Also, it provides an online IDE for development and a debugger tool called "The Inspector" [Microsoft, 2022c].

Babylon.js is compatible with the following 3D formats: glTF, glb, OBJ, and STL [Microsoft, 2022a]. Furthermore, it is licensed under Apache License 2.0 [The Apache Software Foundation, 2023], meaning that commercial use of this framework is allowed [Microsoft, 2022b].

[8] `https://discord.com/`
[9] `https://www.reddit.com/`
[10] `https://stackoverflow.com/`
[11] `https://developer.chrome.com/docs/web-platform/webgpu/`

Cesium is a free, open-source WebGL-based framework. Similar to the previous frameworks, it uses Javascript to wield the models and is compatible with Unreal Engine [Epic Games, 2023b]. Cesium is optimized for visualizing and handling 3D geospatial data, providing some functionalities for that effect, such as 3D tiles [Cesium GS, Inc., 2022b].

A forum is provided by the framework for the users to ask and answer each other's questions. In Stack Overflow, it has a tag with 794 questions [Stack Overflow, 2022c]. No sizable Discord server or Reddit community was found. At the time of writing, Cesium.js has 3k forks and 9.5k stars on GitHub [Cesium GS, Inc., 2022a]. Considering the above statistics, the framework possesses a medium-sized community.

Cesium has high-quality documentation, with an extensive list of examples and in-depth tutorials, and a full API reference. It also provides its own IDE for code development, called "Cesium Ion" [Cesium GS, Inc., 2022b].

Cesium.js supports several types of 3D models, such as OBJ, FBX, DAE, glTF, and gdb [Cesium GS, Inc., 2022b]. Moreover, it is licensed under Apache License 2.0, meaning that commercial use of this framework is allowed [Cesium GS, Inc., 2022a].

X3DOM is an open-source framework that provides a set of functionalities to manipulate the HTML5 DOM and make it display 3D content. This is a big shift in paradigm since the previous frameworks used WebGL, which is an imperative language. The framework is also supported by all modern mobile and desktop browsers [Fraunhofer-Gesellschaft, 2022d].

X3DOM has little to no community, with only 152 questions on Stack Overflow [Fraunhofer-Gesellschaft, 2022 and no sizable Reddit community or Discord server. At the time of writing, the framework has 262 forks and 752 stars on GitHub [Fraunhofer-Gesellschaft, 2022b]. On the other hand, it has extensive documentation, with several examples, tutorials, and API references, but doesn't possess an editor for easier development and testing [Fraunhofer-Gesellschaft, 2022a].

X3DOM only supports X3D [Web3D Consortium and X3D Working Group, 2023] files, making integrating the existing 3D models challenging. It is dual-licensed under the MIT and GPL [The Free Software Foundation, 2023] licenses, meaning it is free for commercial use [Fraunhofer-Gesellschaft, 20

A-frame is an open-source 3D graphic framework that specializes in constructing VR experiences. It is part of the group Supermedium (website at `https://supermedium.com/`) and is built on top of Three.js and focuses on providing a declarative way, through HTML, to edit the scene [Supermedium, 2022c].

This framework does not provide an in-site forum for users to ask questions. However, it has around 3k questions on Stack Overflow [Stack Overflow, 2022a] and a Discord server with almost 5k members (available at `https://discord.com/invite/tGYjkYr`). It is worth noting that the Discord server includes other projects from Supermedium, so not all members are users of A-Frame. On GitHub, it currently has 3.6k forks and 14.7k stars [Supermedium, 2022b]. Hence, considering the above statistics, it is fair to say that A-frame has a medium-sized community.

A-frame's documentation is full of extensive guides and API references and several examples to help the users learn the framework. However, there is no custom editor to help develop and test the code [Supermedium, 2022a].

The framework provides built-in loaders for the formats glTF and OBJ. However, custom loaders can be written to handle any file format, and user-written loaders are also available [Supermedium, 2022a]. It is licensed under the MIT license, meaning that it is free for commercial use [Supermedium, 2022b].

Blend4Web is an open-source web 3D graphic framework highly integrated with Blender [The Blender Foundation, 2023] that uses WebGL to display and handle 3D models. This framework uses the Blender model as input, defines custom interactions and features (like physics, materials, and sounds), and outputs a file that can be displayed on the browser [Triumph LLC, 2022c], [Triumph LLC, 2022b].

The aforementioned framework has a forum where users can debate topics related to it, but it isn't very active. There are no questions present on Stack Overflow, and the Reddit Community and Discord Server were not sizable. In GitHub, the framework holds 125 forks and 725 stars, and it is not updated in 4 years [Triumph LLC, 2022a]. Considering the above statistics, it can be affirmed that the framework has a weak and inactive community.

Blend4Web has extensive documentation, with a list of project examples, an extensive user manual, and a complete API reference. It also provides a Web Player to visualize and interact with the final model, but there is no editor for debugging and testing [Triumph LLC, 2022b].

This framework can only load 3D models that come for Blender, making integration with CATIA very difficult. It has two tiers: a free one, where the users must publish all code written, and a paid one, where there is no need to publish the code, and users get access to extra documentation and examples [Triumph LLC, 2022c]. Blend4Web is licensed under the GPL license [Triumph LLC, 2022a].

A substantial amount of 3D frameworks are present in the web ecosystem today, each one satisfying a specific market segment. It is unfeasible to analyze and compare all of them. However, some not discussed frameworks stand out from the remaining but don't meet enough requirements. This is the case for Verge3D [12], a 3D web framework that focuses on providing a tool for artists to expose their work on a web page but sacrifices functionalities for that. Moreover, there is CopperLicht [13], a 3D framework that focuses on creating models for games but has little documentation and has not been updated in some time. More information about these frameworks can be found on the provided links. Table 2.12 shows a summary of the comparison made between every framework.

[12]available at `https://www.soft8soft.com/verge3d/`
[13]available at `https://www.ambiera.com/copperlicht/`

Table 2.12: Summary of comparison results between frameworks. The scale goes from "Very small/low", meaning that the framework has the bare minimum, to "Very high/big", which means that the framework matches all set criteria.

Framework	Open source	Free	Community	Documentation	3D compatibility
Three.js	Yes	Yes	Very big	Very high	Very high
Babylon.js	Yes	Yes	Big	Very high	Very high
Cesium	Yes	Yes	Big	Very high	Very high
X3DOM	Yes	Yes	Small	High	Low
A-frame	Yes	Yes	Big	High	Very high
Blend4Web	Yes	Yes, but need to share code	Very small	High	Very

2.6 3D File Formats

In order to manipulate the 3D model, we need to store it. So, we need to decide which 3D file format to use. The chosen format must have the hierarchical information of the components (parent and child relationship) and must support materials, textures, cameras, animation, and vertex skinning. Vertex skinning is the process of animating 3D objects by attaching the vertexes to a skeleton. The objects can then be animated by manipulating the bones of the skeleton[James and Twigg, 2005]. Moreover, the formats will be compared using their file size and loading time in Three.js. We decided to analyze the following formats since they are the most commonly used: glTF, FBX, COLLADA, and OBJ.

glTF is a file format developed by the Khronos group. It encodes all data into several JSON files, and it was designed the be an efficient way to transfer 3D content over networks. glb is a variant of glTF that stores the data in binary format into a single file. glTF maintains the scene graph and supports all previously defined criteria [Khronos Group, 2023]

FBX is a file format created by Autodesk that saves all 3D data in an ASCII file with a custom format. Similar to glTF, it preserves hierarchical information and supports materials, textures, cameras, animation, and skinning [Autodesk, 2023].

Also developed by the Khronos group, COLLADA is a file format that uses XML to code all model's 3D data. Furthermore, it maintains the model's hierarchy and the needed features. The difference between this format and glTF is that COLLADA can contain much more information, such as multiple versions of the model. This makes this format less suitable for real-time applications [Barnes and Finch, 2008].

The last format is OBJ, a file format created by Wavefront. It codes all 3D data into an ASCII file with a specific format. It doesn't preserve the model's hierarchy, does not support cameras, and only supports materials and textures in a separate MTL file [Wavefront, 2022]. It also doesn't natively support animation and vertex skinning.

Table 2.13 provides a summary of the analysis made. glTF/glb, FBX, and COLLADA have very similar features, while OBJ only supports materials and textures with the presence of an MTL file. There are other file formats that were not analyzed due to deviating too greatly from the intended use case. This is the case for the USD format, developed by [Pixar Animation Studios, 2021], which is an open-source format designed to facilitate the interchange and collaboration of complex 3D scenes between different applications and platforms. However, it was built for artistic application, and provides several unnecessary features.

Table 2.13: Summary of comparison results of the main features of the 3D file formats that Three.js officially supports. The Format column refers to the underlying format that the files are coded in. Hierarchical means if the format supports scene hierarchy (nodes are children/parents of other nodes)

3D Format	Encoding	Hierarchical	Materials & textures	Cameras	Animation	Skinning
glTF/glb	JSON/Binary	Yes	Yes	Yes	Yes	Yes
FBX	ASCII	Yes	Yes	Yes	Yes	Yes
COLLADA	XML	Yes	Yes	Yes	Yes	Yes
OBJ	ASCII	No	Yes, with MTL file	No	No	No

Now that we discussed the characteristics of each format, we compare them in relation to the size of the file generated and the loading time. A 3D object (a chair, available at `https://tinyurl.com/4aa556jc`) was converted into the four file formats. The conversion process was done using Blender [The Blender Foundation, 2023]. For every format, we measured the file size (in KiloBytes) and the loading time (in milliseconds) in Three.js. These experiments were performed on a laptop with the processor Intel (R) Core(TM) i5-6300U, with a clock frequency of 2.40GHz, and the integrated graphics Intel(R) HD Graphics 520. The loading time was obtained by averaging five consecutive model loadings. The results of these experiments can be seen in Fig. 2.30. We can see that COLLADA and OBJ have large sizes but short loading times. glTF has a shorter loading time but a larger file size than FBX, but the difference is not large.

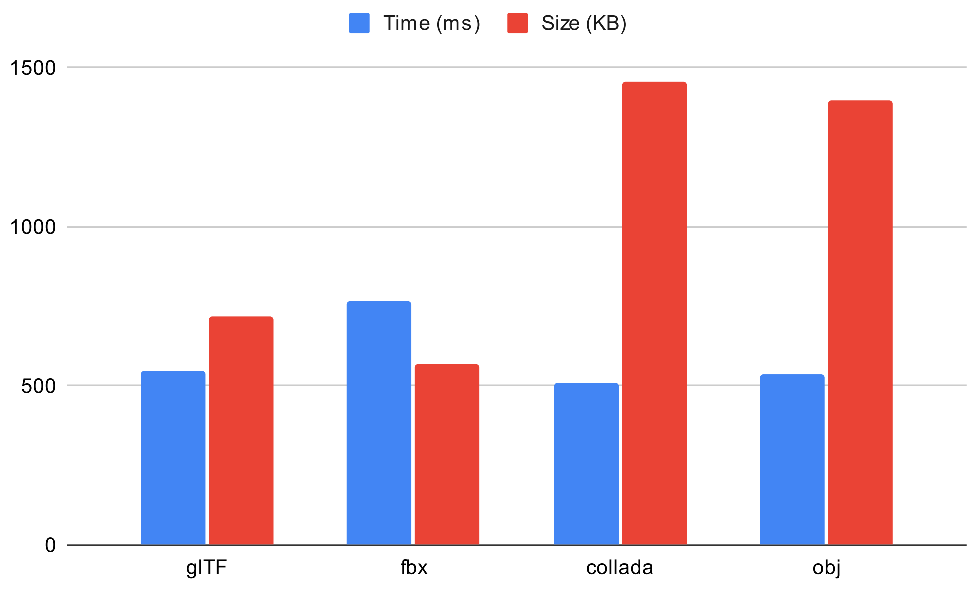


Figure 2.30: Chart depicting the file size and the loading time of different file formats. A model of a chair, available at `https://tinyurl.com/4aa556jc`, was use to measure this times.

Chapter 3

Problem and Proposed Solution

Having analyzed the related work and the existing technologies in the market, we now present the proposed solution to the problem. We start by defining the problem and analyzing the requirements given by CERN. Then, we propose a solution to the visualization problem that encapsulates a preprocessing pipeline and a visualization system. Next, we present the planning for the project, which is defined by four main stages: learning, developing, testing, and reporting. We end by analyzing the potential risks of the projects and suggest possible mitigations for them.

3.1 Problem Definition

CERN is the home of several well-known accelerators, such as the Proton Synchrotron (PS), the Super Proton Synchrotron (SPS), and the Large Hadron Collider (LHC). These accelerators perform several experiments a year and need constant maintenance and calibration. However, managing these tasks can be a complex job. CERN is organized into a hierarchical structure consisting of departments, sections, and groups. Departments serve as the highest-level units, each encompassing multiple groups. Each group is further divided into sections. Multiple distinct groups can have several ongoing tasks spanning multiple accelerator sections that can't overlap. Moreover, before scheduling a task, the group needs to consider the next one for the same accelerator sections since a delay on the task can impact them.

If a CERN employee wanted to see the planned tasks for a machine in an image, he could only use the Visualization menu that provides a 2D image of the selected machine. Other types of visualization exist, such as Gantt and Calender view, but they don't provide the spatial dimension. Figure 3.1 shows the tasks of the PS machine for the 22nd of May, 2023 (these tasks are mock tasks used for demonstration purposes). The different colors represent the responsible groups for the task. The 2D can be useful for a global understanding of the position of the tasks, but it lacks more detailed information about them and more fine-grained control for the view. CERN wanted to expand the 2D view by integrating a new dimension to ease the task visualization and allow more interaction from the user.

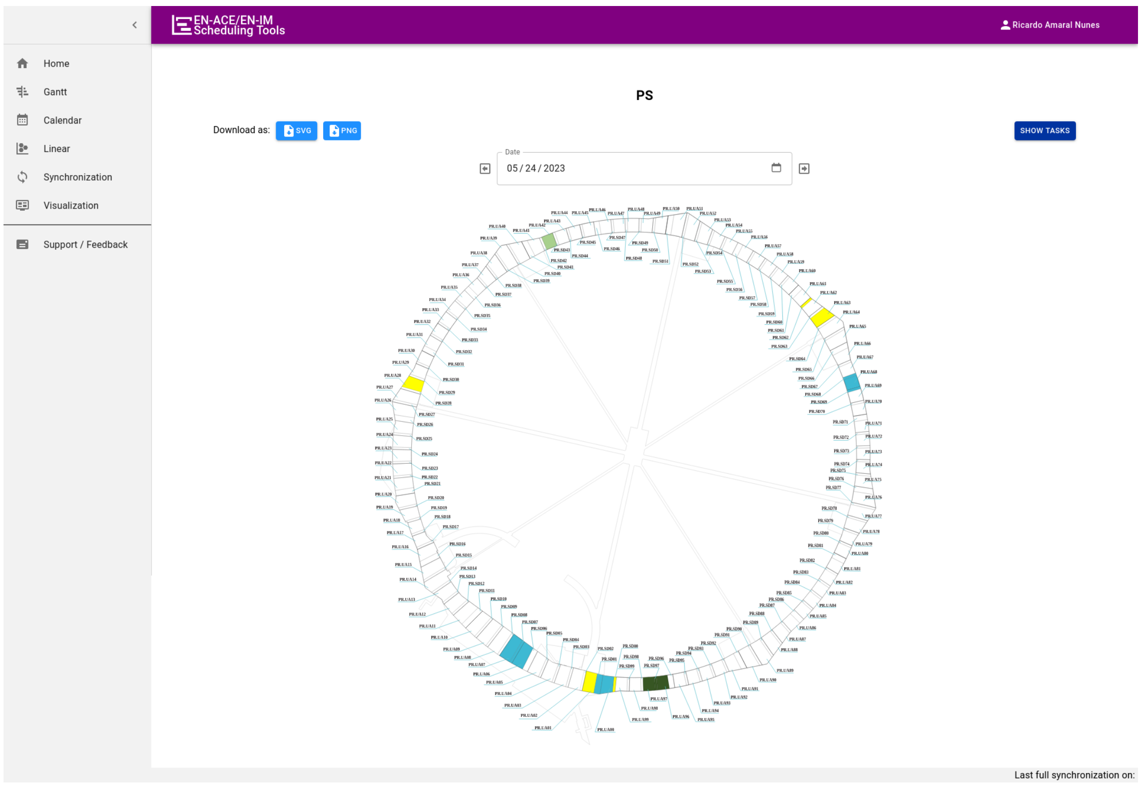


Figure 3.1: Existing 2D solution at CERN for visualizing maintenance tasks

Considering all these specifications and problems, we designed a 3D web visualization system that allows users to easily see the scheduled tasks and plan their own tasks accordingly. The visualization system uses the CAD models CERN already possesses for the machines. This system is divided into two parts: a preprocessing pipeline that simplifies and converts the model from a CAD to a 3D format, and a web visualization system, that allows the user to interact with the application and retrieve the needed task data. Furthermore, the web visualization tool has two main views: a global view, which displays the exterior of the whole machine, and a detailed view, which displays the detailed interior of a section of the machine.

The developed visualization system should be extensively documented to allow CERN employees to easily maintain it. Moreover, the created prototype is only supposed to work for the PS machine, but the system should be easily extended to allow the visualization of different machines.

3.2 Requirements Analysis

CERN provided a set of requirements that the proposed visualization tool should meet. These requirements were refined over time and extended according to the project pace. We divided the set into two categories: functional and non-functional requirements. The functional requirements have the goal of defining the intended behavior of a system [Malan et al., 2001] and are as follows:

- The visualization system should perform simple and complex visualization techniques, such as rotation and translation of the model, and provide the possibility of viewing the model in different levels of detail.
- Every section of the model should be distinctly colored, considering the maintenance tasks being performed. The sections that have no associated tasks should be colored white. If one section has more than one task, then it should use the default color.
- The user must be able to download the model view as an image.
- The visualization tool should allow the user to view information about past maintenance tasks as well as future tasks.

In addition to the requirements above, non-functional requirements were also defined:

- The interaction lag and the loading time of the machine models should be kept to a minimum or, preferably, not noticeable.
- The information available in the visualization system should be accessible in an intuitive and efficient way (related to RQ2).
- The visualization system must be integrated into the CERN website and use the CERN technology stack, which comprises Spring (website at `https://spring.io/`), React (website at `https://react.dev/`), and Oracle DB.
- The visualization tool should be responsive to different PC screens and adapt the layout accordingly.
- The visualization system should be compatible with CATIA 3D models or CATIA export formats.
- The visualization system should be synchronized with the MS Project (the software used to add new tasks) at least twice a day.
- The visualization system should be able to be expanded to other machines with minimal effort.

3.3 Proposed Solution

Having properly defined the problem at hand and analyzed all given requirements, we now propose a novel 3D web visualization system that solves the problem and meets all requirements. The developed solution comprises a preprocessing pipeline and a web visualization tool. The preprocessing pipeline takes as input the 3D model of the particle accelerator and reduces and converts it to the glTF format. The web visualization tool is in charge of all the user interactions with the 3D model and displaying schedule information.

3.3.1 Preprocessing Pipeline

Large and complex models with millions of polygons can introduce undesired latencies in user interactions [Lindstrom and Turk, 1998]. Moreover, most 3D web frameworks have no compatibility with models in CAD format. To diminish the impact of this factor, a preprocessing pipeline to simplify and convert the 3D model was proposed, present in Fig. 3.2.

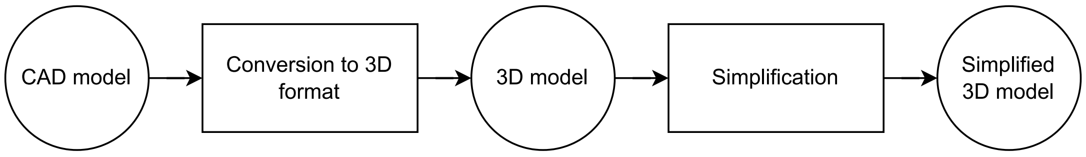


Figure 3.2: Proposed preprocessing pipeline

The pipeline takes as input a CAD model. It first converts the model into a 3D format. The 3D model is then simplified. This preprocessing occurs only once before deploying the system or when new machines are added. Both of these processes are performed using external tools. An analysis of the existing tools on the market is done in the following section.

3.3.2 Visualization System

The visualization system is responsible for handling all user interactions and providing all relevant task information. Moreover, the system needs to retrieve the data from CERN databases to satisfy the user information needs. The architecture for the system was constructed to deal with these demands and can be seen in Fig. 3.3.

The life cycle of the proposed system starts in the MS Project tool. This tool allows the facility coordinator, i.e., the person responsible for the tasks of a machine, to plan its ongoing and future tasks. After finishing the planning, the MS Project tool pushes the changes to the scheduling database, which is the main database for all tasks of all machines. The web visualization system then retrieves the scheduling data present in the database through a Java API.

The system is divided into three levels: the application level, the interface level, and the data level.

The application level is in charge of all user interactions. The user accesses the visualization system through the CERN Website. The web application retrieves the 3D model and the schedule information through the endpoint defined on the interface level and renders it, coloring each section according to the tasks present. It also provides more detailed information about the tasks, such as the responsible groups and start and end dates, if the user wants.

The interface level provides an endpoint called "visualize/interactive" where the web client can retrieve all necessary data. This endpoint constructs a query and sends it to the database, where it is executed. The result is then returned, and the endpoint sends it to the Application level in JSON format. The endpoint was developed by CERN using the Spring framework.

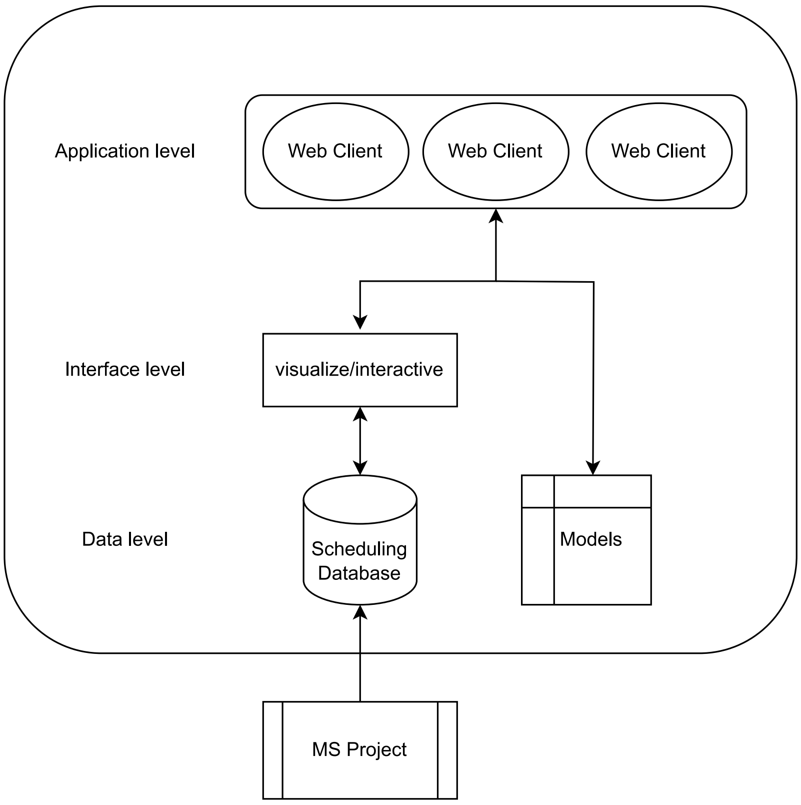


Figure 3.3: General architecture of the proposed web visualization tool

The data level is responsible for storing all the data. The models are stored in the server and are simply served upon request from the Web Client. All of the information about the tasks are stored on the Scheduling Database. This database was developed and is maintained by CERN.

3.4 Technologies Choice

In the previous chapter, we analyzed several technologies and papers describing algorithms and methodologies for model conversion and simplification. We decided not to implement an algorithm from a paper since it would take valuable time to develop the visualization tool, and it falls out of the scope of this dissertation. In this section, we further analyze the technologies previously discussed, now considering the established criteria, and choose the ones to use on the visualization system.

The analysis performed on the 3D web framework doesn't change when considering the requirements of the project. CERN affirmed that we should choose the framework that best suited our needs while being easily maintainable. Considering this and the analysis previously made, we decided to use Three.js since it has all necessary requirements and has the biggest community of all frameworks. Regarding the 3D formats and considering all characteristics discussed and the results of the experiments, glTF was chosen since it has all the needed features and has a low loading time and file size. Moreover, Three.js also recommends the usage of this format in its documentation.

Previously, we analyzed the conversion tools regarding automatization, optimization, cost, and open source. We now need to add more criteria to reflect the requirements and the choice of 3D format. We decided to add two more criteria: support for importing STEP files and exporting into the glTF format.

The conversion tools analyzed are assimp, pythonOCC, Unreal Engine, Blender, and CAD Assistant. Every one of these tools supports the import of STEP, with the exception of Blender. However, there is a paid plugin[1] on the Blender shop that adds the support for STEP. Regarding the exportation to glTF, both the assimp and pythonOCC libraries expressed in their respective documentation that they support this feature, but when tested, it didn't work. Unreal Engine, Blender, and CAD Assistant export to the glTF format without any problem.

A summary of the comparison made for the support of different formats can be seen in Table 3.1. Only Unreal Engine and CAD Assistant support both the STEP and glTF/glb format, but Unreal Engine provides more useful features (considering the previous analysis), and so it was chosen.

Table 3.1: Summary of the comparison between several conversion tools regarding support for STEP import and glTF/glb export

Tool	Supports STEP	Supports glTF/glb
assimp	Yes	No
PythonOCC	Yes	No
UE	Yes	Yes
Blender	No, only with paid plugin	Yes
CAD Assistant	Yes	Yes

The simplification tools were analyzed according to the following criteria: mesh decimation, automation, removal of objects by volume and visibility, cost, and open source. Considering the requirements and 3D format choice, we further analyze these tools according to support for the glTF format. The analyzed tools are Simplygon, pyVista, and PyMeshLab.

Both pyVista and PyMeshLab state in their documentation that they support glTF. However, when we tested it, the results proved the contrary. For pyVista, we opened a ticket on Github (available at https://github.com/pyvista/pyvista/issues/4164) regarding this issue, but no fix has been proposed to date. Simplygon supports glTF. A summary of this analysis can be found in Table 3.2. We decided to use Simplygon since it is the only tool that supports glTF and met most of the previous criteria established in the previous analysis.

[1]available in https://ambient.gumroad.com/l/stepper

Table 3.2: Summary of the comparison between several conversion tools regarding glTF support

Tool	Supports glTF/glb
Simplygon	Yes
pyVista	No
PyMeshLab	No

3.5 Project Planning

Planning the project's activities and tasks is an essential step of the dissertation process. It structures the project's tasks chronologically and defines a time limit for every task. This minimizes the risk of the project going over the time limit. The project plan was depicted in a Gantt Diagram, as shown in Fig. 3.4.

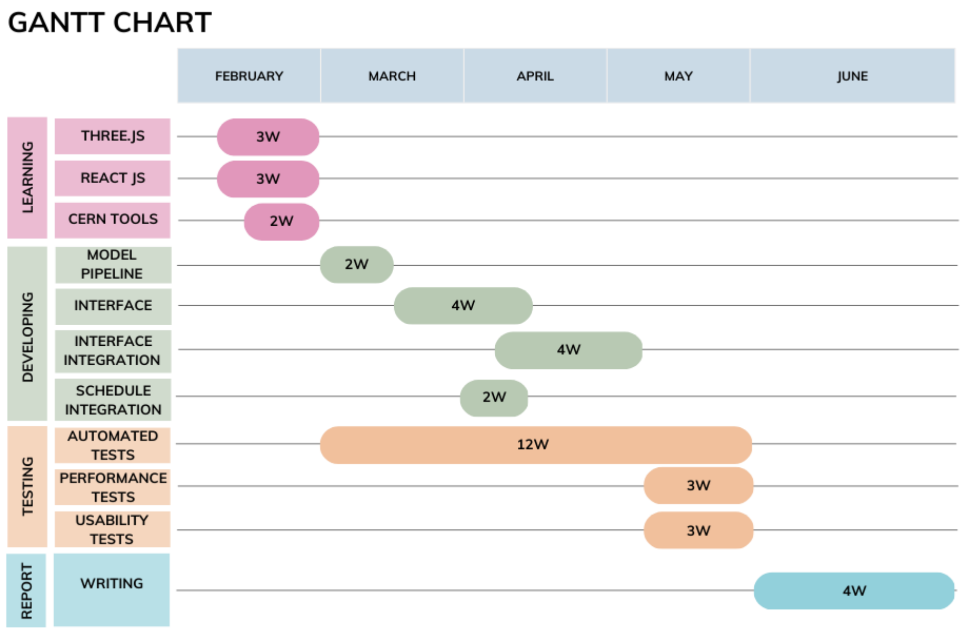


Figure 3.4: Gantt chart representing the project's planning

The project's first phase was the learning phase, where Three.js, React.js, and the CERN technology stack were studied. This was accomplished by reading and performing the tutorials of each framework/tool and constructing small projects with similar but simplified features of the project. It took place in the last three weeks of February.

The second phase, the developing phase, was planned to be executed from March to the first week of May at CERN. In this phase, the visualization tool was developed, starting with the preprocessing tool, followed by the interaction interface and the integration with the task schedule information. The final development step is integrating the tool into the CERN technology stack. There were some deviations from the plan in this phase. The pipeline took a week longer than it was supposed to, and the interface and the integration process took much less time than it was planned (around a few days of work). This extra time was used to add extra features to the application that were not planned.

After development, extensive testing was needed to ensure the tool met the requirements. Alongside the development phase, automated tests were created to verify if the prototype behaved correctly and if new changes didn't change the previous functionality. In May, final performance and usability tests were executed to evaluate the final state of the tool. The performance test subsists of measuring the downloading time of the model, the lag of the interaction, and the lag when changing the view. The usability test consists of an observational study, where the users are asked to perform some tasks and are asked to rate their opinion about certain design aspects. The tasks were timed, and we registered the number of clicks needed to perform the operation. Furthermore, we also used the SUS questionnaire [Brooke, 1996] to measure the usability of the visualization tool. The month of June was dedicated to writing the remaining of the dissertation and performing additional usability tests.

3.6 Risk Analysis

Risk planning consists of finding and analyzing the possible risks associated with the project. This is an essential step in any project since it helps to understand what risks can appear, their likelihood and severity, and possible mitigations. A summary of the proposed risks can be found in Table 3.3.

The first risk encountered is that the usability tests may not be finished by the time the internship at CERN ends. The CERN staff have busy schedules and may not have time to participate in the test in May. This would hinder the final quality of the project and make it harder to correct detected faults. To mitigate the risk, the tests should be communicated and planned with the CERN staff in advance. Despite all this, if the test still can't be performed at CERN, it should be adapted to be completed in a remote environment.

After development, the web visualization tool needs to be integrated into a specific part of the CERN Website, so it needs to be integrated into CERN's technology stack. However, this integration may not be straightforward since we don't know the specifics of how the technologies function and how they interact. Research on the integration process in the learning phase helps mitigate this risk. If it is still encountered, then a reduction of the testing time may be considered.

The writing of the report is planned for June. However, a month might not be enough to finish the dissertation. Extensive note-taking of the tasks performed at CERN can help the writing process. Moreover, the writing process can start in parallel with the testing phase.

Table 3.3: Summary of the analysis of the project's risks

Risk ID	Risk Description	Likelihood	Severity	Mitigation
R1	The tests may not be completed in time	Medium	Medium	Find candidates immediately. Do them remotely
R2	Integration may take longer than planned	Medium	High	Reduce testing time. Research integration in learning phase
R3	May not have enough time to write the report	Low	High	Take notes during the development. Start writing during testing.
R4	Chosen framework may not satisfy all proposed requirements	Low	Medium	Choose another one from the list.
R5	CERN API unavailability due to maintenance or failure	Low	Medium	Check for API availability during internship and plan around it.

In the previous chapter, several 3D visualization frameworks were analyzed to determine which best fit the project. Still, we don't possess prior experience with any of the frameworks, so the chosen framework may not provide enough functionalities to build the proposed tool. This is not probable to happen, but if it does, another framework from the analyzed list can be used instead.

The proposed tool uses schedule data from CERN API to color the 3D model appropriately. This API might go through maintenance and become unavailable for a week or more. If this occurs and coincides with the two weeks of schedule integration, it may delay the whole project. This can be avoided by planning the integration around the CERN API availability.

Chapter 4

Implementation

In the previous chapter, we introduced the problem at hand and gave a general approach to how we intend to solve it. In this chapter, we give a detailed description of the visualization system developed. We start by describing the preprocessing pipeline, more specifically, the conversion and simplification processes. We explain the chosen parameters for these processes and the script developed to automate them. Furthermore, we present the constructed web visualization tool alongside the main design decisions taken and the graphical interface created. Moreover, we describe the architecture of the visualization tool and explain how it was integrated into the Scheduling Tool website. We conclude by explaining how the system can be extended to other machines.

4.1 Preprocessing Pipeline

As discussed before, the CAD models need to be converted and simplified before their usage. For this purpose, we developed a preprocessing pipeline, as seen in Fig. 4.1. The STEP model is converted to glb using Unreal Engine. For this purpose, a Python script was written to automate the job. The model is simplified using Simplygon. This is also done automatically with a Python script. The result is a simplified glb model of an accelerator.

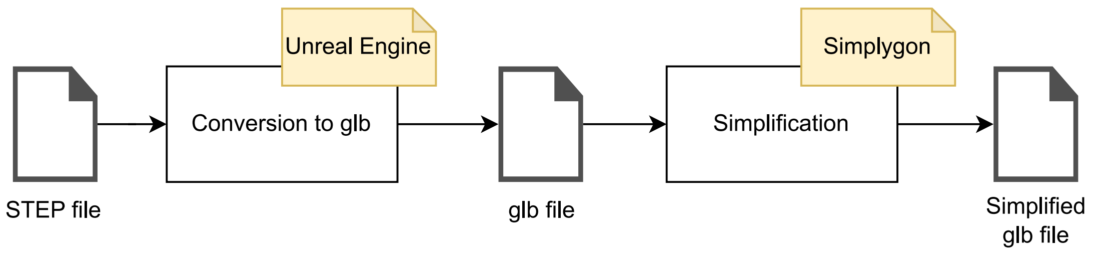


Figure 4.1: Preprocessing pipeline

In this section, we provide an in-depth analysis of the conversion and simplification scripts. We discuss the chosen values for the parameters and the main decisions. We also analyze some alternative paths that we decided not to take.

4.1.1 Conversion Process

The conversion process is divided into two steps: the import process from CAD to Unreal and the export process to glTF. A Python script was written to automate and simplify this process. In order to execute the script, the user needs to open Unreal Engine, create or open a project, and select the option to execute Python script. As input parameters, the user should insert the path for the STEP file and the desired output path.

We need to define three main parameters in the importing process: the chord tolerance, the maximum edge length, and the normal tolerance. Chord tolerance is the maximum distance between any point of the original surface and the corresponding point on the tessellated surface [Yong and Narayanaswami, 2003]. Max edge length corresponds to the maximum length on any edge of any triangle of the tessellated mesh. Finally, the normal tolerance is the maximum angle between two adjacent triangles in the tessellated mesh [Epic Games, 2023b]. The default values for these parameters are 0.2cm, 0cm, and 20º, respectively. Since we want a simplified model, we increased these values to 30cm, 0.8cm, and 45º. Values greater than these ones were tested but introduced faults in the model geometry. Moreover, any increased complexity in the form of redundant triangles can be eliminated in the simplification phase. Algorithm 1 represents the logic of the script of the import process to Unreal Engine.

Algorithm 1 Pseudo-code of the import process on Unreal Engine

```
$scene \leftarrow UE.createSceneFromFile(filename, sceneName)$
$tessellationOptions \leftarrow scene.getOptions()$

$tessellationOptions.chordTolerance \leftarrow 30$
$tessellationOptions.maxEdgeLength \leftarrow 0.8$
$tessellationOptions.normalTolerance \leftarrow 45$

$scene.startImport()$
```

The PS model in CATIA is divided into eight octants plus the center building. In the beginning, CERN provided us with the octants and center in different files because they were quite large. Only in the final month, we had direct access to CATIA. A representation of the structure of the model can be seen in Fig. 4.2. The model was composed of two main parts: the exterior walls and everything else that was inside. The exterior was simply extracted to be used as the global model. The interior possesses a lot of detail that doesn't provide any additional information and increases the file size of the model. Fire extinguishers, electrical cables, and pipelines are some examples of objects that, even though they give an increased realism to the model, don't add new information and would only serve to distract the user. These objects were removed manually since most were named according to a serial number (so there was no way to know a priori what they were) or were not grouped together. Moreover, the name of the sections was not correct. The modeling team used an unknown naming scheme, so we used the GIS portal to match the sections with the correct names. The GIS portal is a visualization tool that allows CERN staff to see the accelerator on a map and get a panoramic view of their interiors.

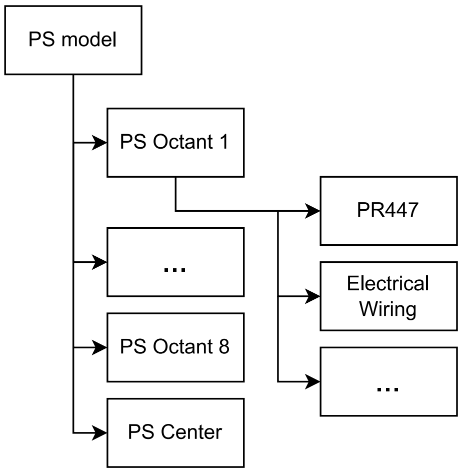


Figure 4.2: PS model structure. The PS is composed of eight octants and a center. Each octant contains several other groups/objects, some with a name conveying their purpose or an identifier

The next step is the exporting process to glb. There are several parameters to define in this process, but most of them remain on default. We decided not to export the texture, lights, and cameras since it would increase the file size and they can be added, if needed, at runtime. It is important to note that the option "Use Mesh Quantization" should remain set to false. Mesh quantization consists of applying the "KHR_mesh_quantization" extension[1] of glTF, which converts the vertex attributes, that are usually of float type, to 16-bit or 8-bit storage. This technique saves memory space by sacrificing precision, which would be useful for our application. However, this extension is not supported by all tools (Unreal can't import models with this extension, and neither does Simplygon), so we set it to false. Algorithm 2 shows the logic of the export process to the glb format.

4.1.2 Simplification Process

The simplification process is done by using Simplygon. The result of this process should be a simplified version of the input model, with fewer triangles but retaining a similar shape. To use the tool, we need to execute the Python script with three parameters: the path to the directory where the model is located, the output directory for the simplified model, and the name of the file containing the model.

We opted to use the reduction processor of Simplygon. This processor removes vertices and triangles of the model according to a set of heuristics. Since the software is proprietary, they don't provide much more information about the process. They allow users to give weights to geometry, UV mapping, and normals to indicate the level of importance of the reduction process. In our case, we give more priority to the geometry of the model. The reduction process stops when

[1]https://github.com/KhronosGroup/glTF/tree/main/extensions/2.0/Khronos/KHR_mesh_quantization

Algorithm 2 Pseudo-code of the export process from Unreal Engine to glb

```
$exportOptions \leftarrow UE.GLTFExportOptions()$ ▷ There are more export options, but they were left on default

$exportOptions.exportPreviewMesh \leftarrow False$
$exportOptions.includeGeneratorVersion \leftarrow False$
$exportOptions.exportVertexSkinWeights \leftarrow False$
$exportOptions.useMeshQuantization \leftarrow False$
$exportOptions.exportLevelSequences \leftarrow False$
$exportOptions.textureImageFormat \leftarrow UE.GLTFTextureImageFormat.NONE$
$exportOptions.exportLights \leftarrow 0$
$exportOptions.exportCameras \leftarrow False$

$UE.GLTFExporter.exportToGltf(filepath, exportOptions)$
```

the simplified model reaches a certain number of triangles or ratio. Simplygon provides several components a user can add to the reduction process, such as bone reducer, normal repairer, and visibility. We chose to use the visibility component, that removes invisible texture spaces and triangles.

Simplygon provides in its documentation an example script for mesh reduction. We used this script as the base for the simplification and performed some alterations to turn the code more modular and tailored to the application. We set three different reduction targets, depending on the purpose of the output model: 0.5-0.8% for the preview models since they are temporary and only viewed from afar, 4-5% for the details models since they have a lot of complexity that is not needed, and 20-30% for the global models. The types of models are discussed in depth in the next section. The ranges of values presented here were chosen by trial and error since there is no way to know a priori what ratio of simplification is appropriate for a given model. A model can have an unnecessarily complex geometry that requires a higher level of simplification, or it can have a pretty simplified geometry already. The values chosen are the ones that work best for the machines. Each simplification of the global models took around two minutes, while the simplification of details and preview models took around six minutes. The reduction targets are not the same for models of the same type because the reduction target also depends on the size of the model, which differs from model to model. The pseudo-code of the simplification process can be seen in Fig 3.

Algorithm 3 Pseudo-code of the simplification process performed by Simplygon

```
$sgScene \leftarrow LoadScene(filepath)$
$sgReductionProcessor \leftarrow CreateReductionProcessor()$

$sgReductionSettings \leftarrow sgReductionProcessor.GetReductionSettings()$
$sgVisibilitySettings \leftarrow sgReductionProcessor.GetVisibilitySettings()$
$sgReductionSettings.SetReductionTargetTriangleRatio(reductionTarget)$
$sgVisibilitySettings.SetCullOccludedGeometry(True)$    ▷ Remove all non-visible geometry.
$sgVisibilitySettings.SetRemoveTrianglesNotOccludingOtherTriangles(False)$    ▷ Don't remove non occluding triangles.
$sgVisibilitySettings.SetUseBackfaceCulling(True)$    ▷ Remove all back-facing triangles.

$sgReductionProcessor.RunProcessing()$
$SaveScene(sgScene, filepath)$
```

4.2 Web Visualization Tool

The proposed solution is composed of two parts: the preprocessing pipeline, which converts and simplifies the model, and the web visualization tool, which renders the model on a website and allows the user to interact with it. The visualization tool is the main part of the project since it allows CERN staff to visualize the maintenance tasks of the machines.

The initial concept for this dissertation was to develop the prototype for the PS machine but to develop it so that it could be easily extended to other machines. With this goal in mind, we developed a framework that can be modified to visualize other machines. The framework assumes two main views: a global view that encapsulates the whole model and a detailed view that presents the details of a section of the machine. On the global view, the sections are colored according to the tasks performed inside.

In this section, we discuss the created framework, its architecture, and how it was constructed to extend to other machines. Furthermore, we present the developed graphical interface and explain the main design principles. We also explain how the tool was integrated into the Scheduling Tool website.

4.2.1 Architecture

The web visualization tool is divided into five components. Each component has a set of responsibilities and serves a specific aspect of the tool. The components are essentially files that contain functions for the application logic to use. This makes it easier to expand the application to other machines since the state is not stored in any component. Furthermore, all the files described here (including the configuration files) are bundled together and sent as a single file. A diagram of the architecture of the tool can be seen in Fig. 4.3.

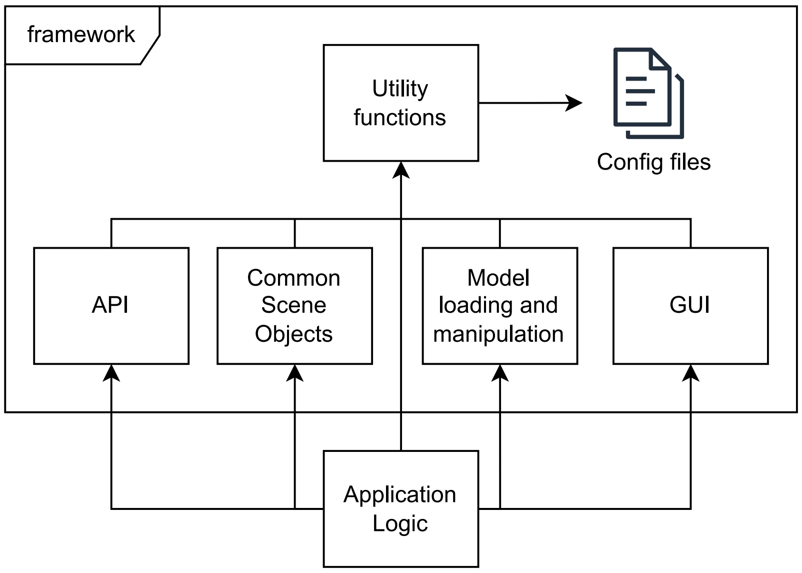


Figure 4.3: Overview of the system and its main interactions

The API component is responsible for communicating with CERN's API and parsing the response to a more suitable format. The API endpoint is called "visualization/interactive". It is used by other website pages and accepts two parameters: the facility ID and the date. This endpoint returns an object with a key equal to the date requested and a list of the sections with the maintenance tasks at that date. Each section has a name, a color, and a list of the tasks that are being performed on it. The most important attributes of tasks are the name, start, and finish date, and the responsible groups of the tasks. There can be more than one responsible group for the task. The other attributes are simply discarded.

The API response is converted from a list of sections with tasks into a list of tasks. This is done to ease further manipulation of the data. As explained before, there are low-level sections, the lowest granularity one can go when associating a task to a section, and high-level sections, which are comprised of several low-level sections or other high-level sections. A scheme of this hierarchical relationship can be found in Fig 4.4. These sections are organized in a hierarchical way, like a tree. The API can return any type of section. High-level sections are converted to a list of low-level sections, and a task is created for each section. This conversion is done by performing a breadth-first search for the high-level section and returning all "leaf nodes" (the low-level sections) associated with the section. For example, if there were a task being performed in Section 0, this would be converted into a list of two tasks, one for PR.UA00 and another for PR.SD00.

The common component contains functions to initialize and handle the basic elements of a scene, such as the renderer, the lights, the camera, and the control. These elements were grouped into one component because they are simple and don't require special interaction. Considering the renderer options, we decided to not use aliasing since it hurts performance. For the same reason, we didn't apply any post-processing effects.

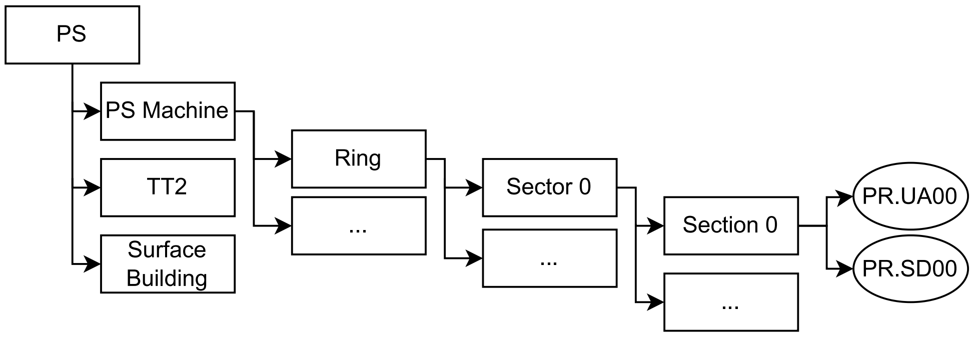


Figure 4.4: Diagram representing the hierarchical relationship of the PS sections. The square sections are high-level sections, while the round ones are low-level

Next, we have the model component that is responsible for handling the 3D models. There are three types of models: the global models, which are the exteriors of the accelerator; the detailed models, which contain the magnets and several machines (the interior of the accelerator); and the preview model. The purpose of the preview model is explained in the next section. One of the component's main functionality is to download the 3D models. The download options are defined in the configuration files. Every model configuration has the path to the model, the name, a flag indicating if it is a detailed model, and a set of geometric transformations to apply. The global model configurations also have the name of the detailed model associated with it (since each global model is the exterior of the detailed model). The model component is also responsible for coloring a model according to the task present in its sections. This is accomplished by transversing the model and changing the material of the model to one with the desired color.

The GUI component is responsible for registering the users' input and changing the graphical interface accordingly. We used the dat.gui package that comes with Three.js to create a custom control panel. This control panel gives the user the option to change the displayed model, change the view, focus on a section with a task when it is the only one on the detailed model, and filter the tasks by the responsible group. We only need to initialize the GUI with the options names and the callback functions. This component is also responsible for displaying the menus, i.e., the help and the task information menu, and managing the popup containing the section name.

Furthermore, we have the utility component that encapsulates all utility functions, constants, and data types used across all components. Moreover, it is also responsible for loading the configuration files and giving the correct model properties to the other components. Abstraction was achieved with the help of JSON configuration files. Every machine has a set of JSON files that define all necessary data to represent it, such as the download options, a tree of all locations of the machine, the cameras, and the general properties of the machine. This can be seen in Fig. 4.5. There is also a configuration for the application (the common file), with data that doesn't change with the machine, such as the background color, default color, and API's URL. New configuration files for other machines can be added by creating a folder inside the Config folder. In Appendix A, a more detailed view of the contents of these JSON files can be found.

Finally, we have the main component, which is responsible for all the logic and the state of

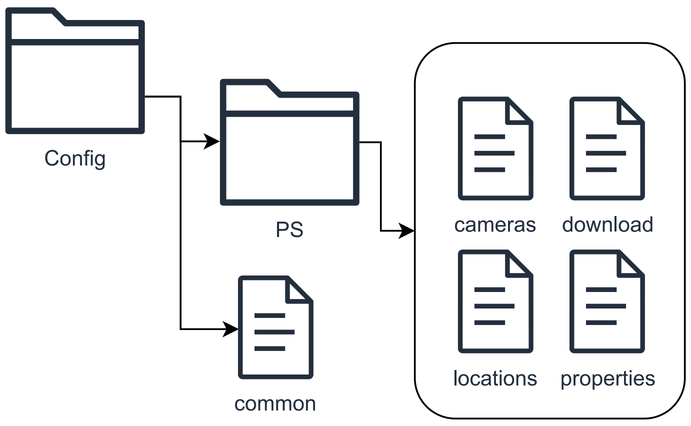


Figure 4.5: Folder and file structure of the JSON configuration files

the application. It initializes all the necessary scene elements, loads and maintains the state of the models, reacts to user input, and more. It is also responsible for the raycaster element, which is used to detect the objects that the user is selecting with the mouse cursor. This is accomplished by sending a ray from the position of the cursor and registering the objects that it collided with. The main component is also responsible for rendering the scene, which is done constantly. The scene could be rendered only when something in it changes since there are no animations, but the constant rendering doesn't create any lag, and it would increase the complexity of the application.

4.2.2 Design Decisions and Graphical Interfaces

The system developed is monosemic since the CERN employees know the meaning of the signs and want to extract the scheduling data. In this case, the invariant is the scheduling data, such as the location, date, and groups. We decided to use a minimalist approach and only display objects that convey useful information, as suggested by [Tufte, 2001].

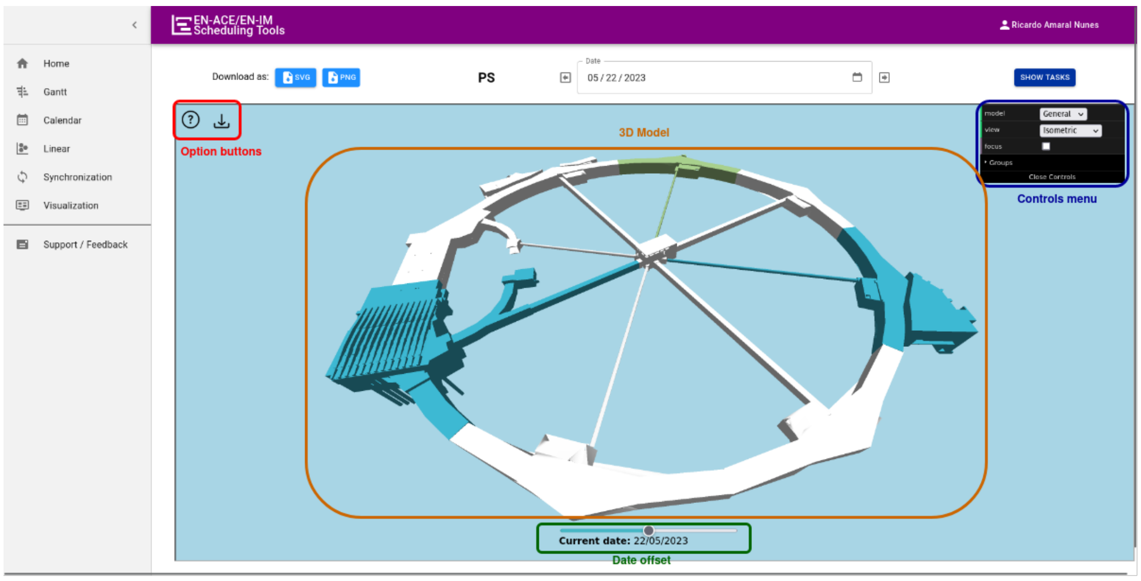
We only have two types of data: the task name, location, and responsible group, which are nominal, and the task start and finish date, which is temporal. Special care was given to the group and the dates. Sections are colored according to the groups that have a task on them. If there is a section with more than one group working on it, then specific color should be used (named in this context, for easier referencing, the "multiple" color). White was used to indicate that a section currently has no tasks. There are 54 groups in CERN, with some of the groups having the same color. Figure 4.6 represent different groups and the associated colors. Furthermore, we have the start and finish dates of the tasks, which can be considered the main component of the visualization. It is the one that users will vary in order to see the change in tasks. Even though we have several components and possible visual variables to map them to, CERN was very specific on which ones to use. We should only use the groups and colors to distinguish between different tasks.

Since we are locked in the marks and the visual variables to use, we can focus on developing different types of interaction with the application. Our system is transformable and manipulable since the user can change the date (and, consequentially, the input data) and manipulate the view.

BE-ABP #ffc000	●	BE-ASR #52de1d	●	BE-BI #bbb600	●	BE-EA #ffc000	●
BE-CO #ffc000	●	BE-GM #ffc000	●	Multiple #3db9d3	●	None #ffffff	○

Figure 4.6: Sample of CERN groups and respective colors. Multiple is the color used when different groups have a task in the same section. None is the color used when no task is present in the section

Considering the system's characteristics described above, we developed the graphical interface present in Fig. 4.7. It is composed of four main components: the Controls menu, the options buttons, the date offset, and the 3D model.


EN-ACE/EN-IM Scheduling Tools

Date 05 / 22 / 2023

model General

view Isometric

focus

Current date: 22/05/2023

Figure 4.7: Global view of the model on the 3D view

The controls menu, identified in blue, allows users to interact with the model and change some of its properties. We chose the drop-down for the model and view because the model name is a nominal variable with few available options, so the user can easily locate and pick the wanted one. The drop-down model lets the user change the current model to 1) the detailed version of any section, 2) the global view, or 3) a top view that contains the details of all sections. The drop-down view sets the view type, which can be "Isometric", the initial view, "Top", which displays a top view of the model, and "Custom" when the user interacts with the model. These drop-downs provide a secondary way to change the current model/view so that users can interact with the visualization tool in a way that is more convenient to them. The focus checkbox focuses the camera into a section that has a task when there are no other tasks in the section. This was implemented because when there is only one section with a task in a detailed model, it can be hard to see where it is. Furthermore, there is the group list that contains a checkbox for all the groups that have tasks at the specified date. This allows the user to filter the tasks being displayed by checking or unchecking the group boxes. This feature is an extra functionality requested by CERN.

The options buttons provide some extra functions that are not specific to the date or the model. The help button displays the help menu that provides instructions on how to interact with the tool and the meaning of the options of the Control menu. The download button lets the user download a PNG image that contains the current view of the model. If more image formats need to be added in the future, then a drop-down menu containing the format options can be implemented.

The date slider allows the user to change the current date. He can push the slider forward to increase the date or back to decrease the date. This makes the action of changing the date intuitive since moving to the left usually means going back and moving to the right going forward. The current date is always shown on the label below the slider. This date slider is helpful when the user wants to quickly see which tasks are planned after a specific task so he can better coordinate his own tasks. It also allows the user to quickly make big changes in the date while retaining the option of a more fine-grained control by using the arrow keys.

Finally, we have the main component, which is the 3D model. A user can perform basic interactions, such as zooming in and out, rotating, and translating the model. Rotation can be achieved by clicking the left mouse button and dragging the mouse. Translating the model is similar, but instead of the left mouse button, it is the right button. Zooming in and out can be done by using the middle mouse button. These commands are identical to other 3D software (such as CATIA), and we chose them because there are several CERN staff that use CATIA, and this way the interaction commands are homogeneous. Each section is colored according to the responsible groups of the tasks being performed there. If there is more than one group with a task in a section, then the default color is used. While hovering the mouse cursor over a section, a popup appears with the name of the section to help the user to get familiarized with the machine. To see the detailed view of a section, the user needs to double-click on the section. Double-click was chosen instead of just clicking because a user could accidentally click on a section while moving the model, leading to an unwanted action and a worse user experience.

The details of octant 1 of the PS machine can be seen in Fig. 4.8. Even after simplification, the 3D models of the details of the machine are very large and may take a long time to load. In order to avoid degrading the user experience by making the user wait, we created a preview model that is extremely simplified. This model is only meant to be seen from afar, and it is immediately replaced after the detailed model finishes downloading. Furthermore, to avoid downloading the same model more than necessary, we store the model so that if the user needs to see it again, the application doesn't need to download it again and it is consequentially faster.

A popup also appears while hovering the mouse cursor over a section. However, in addition to the name of the section, a list of the tasks present in the section is also displayed. If the user wants to learn more about a task, he can double-click on the task section to open the task information

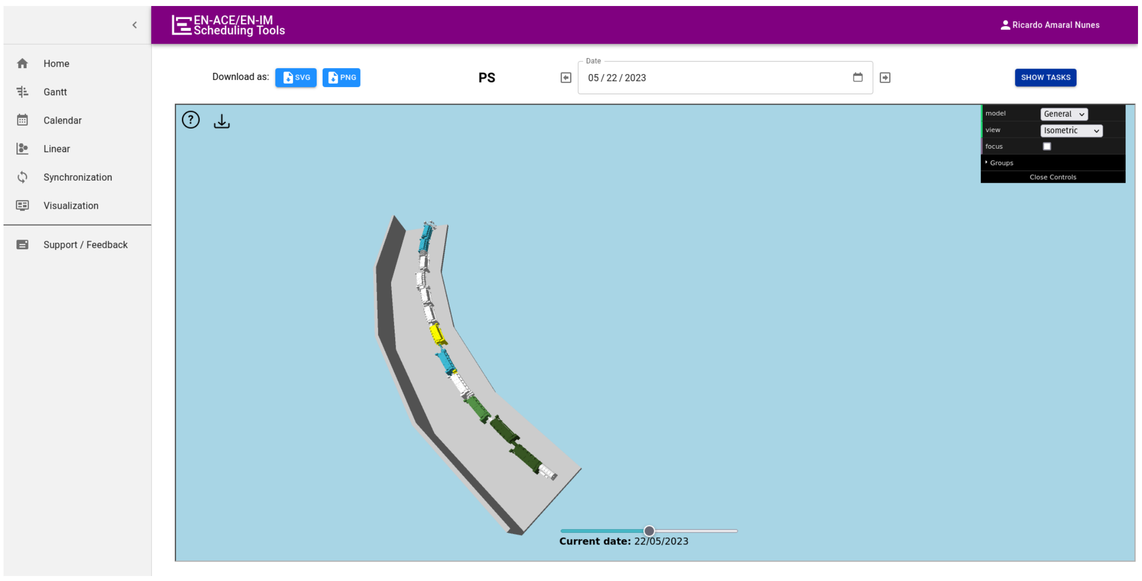
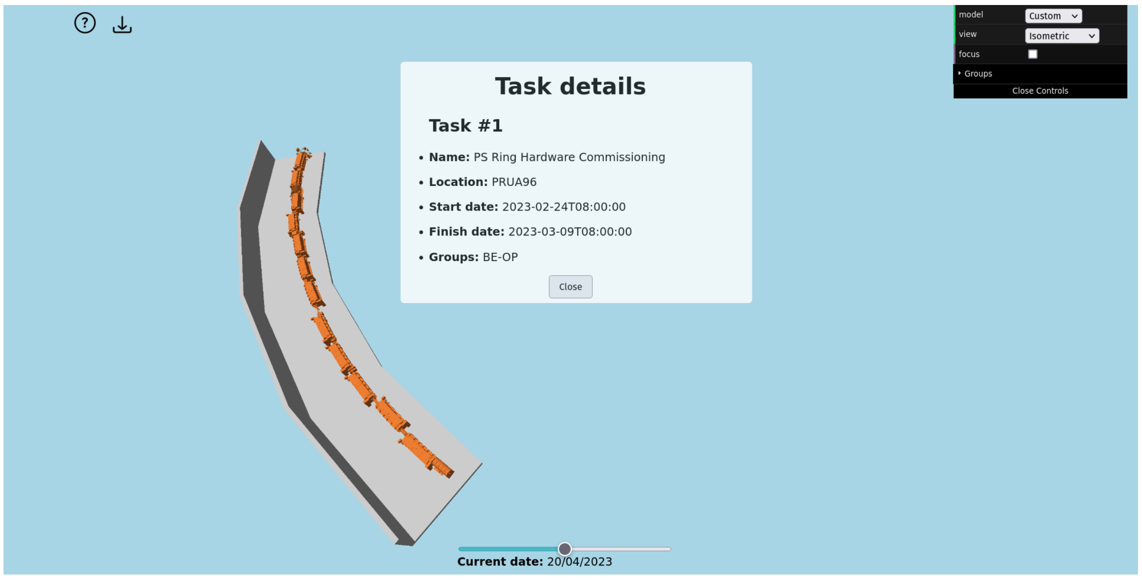


Figure 4.8: Detailed view of octant 1 of the PS machine

menu. This menu contains the name of the task, the localization, the start and finish date, and the responsible groups. If there is more than one task in the section, then the second task appears below, and the user needs to scroll to read its information. Figure 4.9 shows the task information menu for the mock task "PS Ring Hardware Commissioning".


Task details

Task #1

- **Name:** PS Ring Hardware Commissioning
- **Location:** PRUA96
- **Start date:** 2023-02-24T08:00:00
- **Finish date:** 2023-03-09T08:00:00
- **Groups:** BE-OP

Figure 4.9: Example of the task information menu

4.2.3 Project Development and Final Structure

Three.js can be installed using the NPM package manager [2] or by importing from a CDN. We decided to use NPM because we don't want to introduce extra third-party dependencies to the project. At the beginning of the project, some problems arose that had to be dealt with before starting to develop. Three.js is built using Javascript, and one of the requirements established by CERN is to use their technology stack, which includes Typescript. So, we needed a method to transpile our Typescript code to Javascript and bundle it together with Three.js. For this purpose, we decided to use Webpack[3].

Afterward, we started programming the application. Constant feedback from CERN staff allowed us to understand the product better and get immediate feedback on new features. As stated before, we finished the integration and the interface sooner than planned. This allowed us to add new features requested by CERN and to test the expansion to other machines. The two main features added were group selection, which allows filtering of the tasks displayed by the responsible group, and the preview models, which reduce the user's perceived loading time. Moreover, we spent some time improving some elements of the interface and the code.

The project is structured as described in Fig 4.10. The "dist" folder contains the results of the transpilation and compilation of the source code. It also contains the 3D models, the button icons, and the HTML index file. This folder was used as a test environment while developing the application. The "docs" file contains all relevant information about the system, including instructions on how to use it and code documentation automatically generated using Typedoc [Typedoc Community, 2023]. The "source" folder contains all the source code of the visualization tool. Every component described above corresponds to a single source file. Furthermore, this folder contains all configuration files in the "config" subfolder.

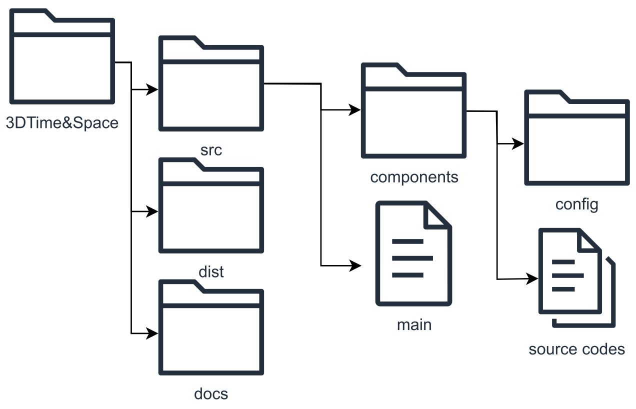


Figure 4.10: File structure of the project's code

The 3D models are structured as described in Fig. 4.11. The "models" folder contains the models of several machines. Inside the machine folder, there are three folders: the global folder, which contains the models for the global view of the machine; the details folder, which contains

[2]https://www.npmjs.com/
[3]https://webpack.js.org/

the models of the detailed interior of the machine; and the preview folder, which contains the models that are used as a placeholder of the detailed models when these are loading.

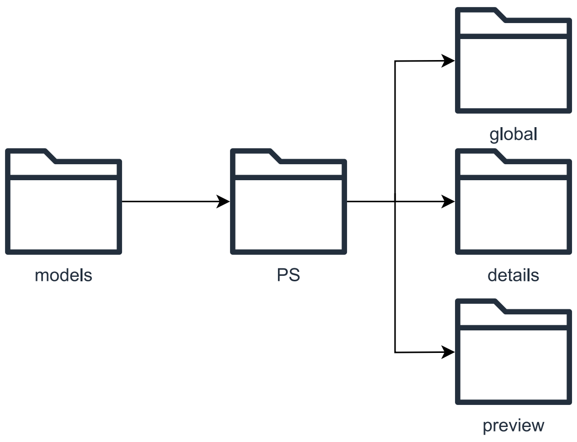


Figure 4.11: 3D models structure in a directory. An example of the PS machine is given

4.2.4 Integration

The 3D visualization tool was developed separately from the Scheduling website. As a result, we need to integrate the tool into the website. This process should be easy and automatic since it will be repeated every time there is a new version of the tool.

The first change to perform is the API endpoint to connect. When developing the visualization tool, we used a test database with a different endpoint. Before deploying, we must change the endpoint to the correct one. Moreover, there are some statistics about the application that are shown in the top left corner of the window and are only helpful for development, so they need to be removed.

We faced a problem when trying to integrate the code into the website. We tried first to put the code alongside the website code, which created several dependencies issues. In the end, we decided to use the iframe HTML tag that allows a developer to embed a webpage inside another webpage. This separates the application's code from the website's code and makes it easier to deploy. We need to put the javascript file in the server alongside the 3D models, the icons, and the application HTML file, and insert an iframe tag in the page that fetches the appropriate HTML file. The HTML file then fetches the needed resources. Between new versions of the visualization tool, only the resources that were changed need to be replaced. The tool is now fully integrated with the webpage.

We decided on this approach to make the website and the system independent. However, this also severely restrains the communication between them. If, in the future, there is a need to implement new features that require communication between the website and the system, one can convert the code to React Three Fiber [Poimandres, 2023]. This library enables the creation of 3D visualizations and virtual reality experiences using React and Three.js in a declarative way. Since there was no need for this now, we decided against using the library.

4.3 Expansion to Other Machines

One of the requirements established was that the visualization system should be easily adapted to other machines so that other CERN employees that work on different machines can use the system in the future. The machine that the system was supposed to support at the end was only the PS, but in order to prove that the visualization system could be done easily, we decided also to include the LINAC4 machine. This test is done in Chapter 5. In this section, we provide a guide on how to add new machines to the system.

The system makes only a few assumptions in order to work properly. It assumes that the user wants to display two different types of models: a model of the surface of a building and a more detailed model of the inside of the building. In fact, it doesn't need to be a building: it can be any object. If we ignore the intended semantic identity of the project, the system can display any two objects and make it so that when a user selects one, it substitutes it for the other. Furthermore, the name of the sections of the 3D model should be the same as the ones retrieved by the API. This allows the program to match the sections and color them appropriately. Also, the CAD model must be structured and divided into the appropriate sections.

To extend the system to other machines, we only need the CAD models and the configuration files. The conversion and simplification process stays essentially the same. The reduction target may need to be adjusted according to the model. For the visualization tool to display the machine, we first need to introduce the 3D models. A specific folder should be created with the name of the machine. This folder should follow the structure exemplified in Fig. 4.11. The preview models may not be needed if the detailed model is not very large in size (less than 5MB). Then, we have to create the configuration files for the machines. The new configuration files can be an adapted version of the files from other machines by changing the paths to the models and the GUI options names. The only file that is specific to a machine is the location tree file that contains a hierarchy of all locations of the machine. Afterward, we need to add to the utils component the configuration files imports and change the function that returns the current machine configuration to include the new machine. After that, the new machine should be visible on the screen, and we only need to fine-tune the cameras and lights. In Chapter 5, we evaluate this process by integrating the LINAC4 machine into the system.

Chapter 5

Evaluation

In the previous chapters, we defined the problem to be solved by establishing the requirements. After describing the implementation process, we must ensure the developed system meets the requirements. For this purpose, we created a controlled experiment to see how a user interacts with the system and to validate the design choices taken. Furthermore, we used the SUS standard questionnaire [Brooke, 1996] to measure the usability of the system. We then analyze the experiment's results to answer the research questions previously established, evaluate the design decisions taken, and identify possible issues with the system. Also, we perform several measures of the loading time of the models, the initial loading time of the application, and the interaction lag. We conclude with a description of the method of the addition of LINAC4 and a comparison between LINAC4 and PS machines' information retrieval process.

5.1 Evaluation Methods

To evaluate the developed system, we designed a controlled experiment to analyze the users' interaction and measure some statistics in order to answer the research questions. Moreover, we used the SUS standard questionnaire to quantify the system's usability and measured the interaction lag and loading times of the models. In this section, we give a detailed description of the experiment, the questionnaire, and the methodology of the time measurements.

5.1.1 Controlled Experiment

As described in Chapter 2, controlled experiments have five main aspects that need to be defined: the participants, the variables, the hypothesis, the method, and the analysis.

The participants of the experiments are the CERN staff, i.e., the real users and some non-CERN personnel. This last group is not a stakeholder in the project but can give some insight into the interaction with the system. Besides, it increases the sample size, and we can verify after executing the experiment if the CERN staff performs better due to possessing knowledge of the accelerators.

We want to measure the efficiency and usability of the system. So, we set as the dependent variable the time to complete the task and the number of clicks. Both these variables measure the usability of the system since a high number of clicks or a large time duration imply that the method to retrieve the needed information may be too complex or convoluted. The independent variable is the system being used. We test the participants using the developed 3D visualization system and the existing 2D view.

The hypothesis we want to prove is that the 3D visualization system is more efficient and intuitive than the existing 2D system. Of course, the 3D system should be efficient and intuitive on its own, but it also needs to be better than the 2D system to justify its usage. Furthermore, we chose the within-subjects method for this experiment. This is due to the small sample size. We vary the order of the experiment's tasks to identify if there is transfer learning between the 2D and 3D that might influence the results.

After performing the experiment, we need to analyze the collected data. We measure two variables: the time duration and the number of mouse clicks. We pondered several methods to record the users' data, such as recording and taking notes. We experimented with recording the participant's voice to measure the time and the number of clicks. However, the ambient noise hinders the ability to count mouse clicks accurately. So, we decided to write a script that times the participant and registers the number of clicks automatically. We compare the average value and the standard deviation of the results of both systems to verify which one is, in fact, the most efficient and intuitive. It is important to note that even though comparing the usability of the 2D and 3D systems is our main goal, we have other aspects we want to evaluate. We took notes of the participants' behavior and reactions and asked them subjective questions about some graphical elements. This is used to evaluate the design decisions and identify possible faults and issues with the system.

Considering all of the above, we design an experiment with several scenarios and tasks to evaluate the usability of the system. The full script for the experiment can be found in Appendix B. There are 20 participants, ten from CERN and ten external. Moreover, the participants are divided into two groups. One group starts with the 2D system (group A), and the other starts with the 3D system (group B). The experiment comprises seven scenarios, some exclusive to the 3D system. Every scenario has a specific goal, i.e., a specific aspect of the design to test. Scenario 1 tests if the user can find the webpage containing the visualization system inside the website. Scenario 2 is exclusive to the 3D system and tests if the user understands how to interact with the model. Scenario 3 evaluates the information retrieval process. Scenario 4 tests the functionality of changing the date to see previous and future tasks. Scenario 5 is only for the 3D system and tests the group filter. Scenario 6 tests only for the 3D system if the user can get a top-detailed view of the model and download a screenshot of the view. Finally, scenario 7 verifies if the process of retrieving task information is similar for other machines.

Alongside the tasks present in the scenarios, some opinion questions were asked to better understand the user's perception. These questions ask the user to rate their opinion from 1 to 5. The first two questions are identical and ask the user to rate the task information retrieval process.

More precisely, the questions are:

- **PQ1:** On a scale from 1 to 5, how easy was it to retrieve the task information?
- **PQ2:** On a scale from 1 to 5, how easy is it to detect the sections that had tasks?

These questions are asked after Scenario 3, both for the 2D and 3D systems, in order to quantify the users' opinions and allow for comparison between the two systems. The third question is asked after Scenario 7, only for the 3D system, and has the intention of understanding how similar is the information retrieval process between two different machines. The question is: "On a scale from 1 to 5, 1 being not similar at all and 5 being very similar, how similar was the process to retrieve the task data for the PS and for the LINAC4?". In addition to these close-ended questions, two open-ended questions are posed at the end to capture the users' opinions regarding both the 3D and 3D efficiency and usability and possible improvements to be made.

A standard usability questionnaire was asked at the end of the experiment. This allows us to better compare the developed system with other 3D visualization systems and to future versions of itself. From the pool of analyzed standard usability tests in Chapter 2, we decided to use the SUS standard questionnaire. This questionnaire is quick and easy to administer, which is important since the CERN staff is usually very occupied, making the time duration an aspect to be mindful of. We use the SUS score to quantify the results and the adjective ratings described by [Bangor et al., 2009] to analyze the results.

As a final remark, it is important to note that bias toward giving good reviews is a possible problem. The interviewer is known by most of the participants, and this may skew their opinion. In order to mitigate the risk, we affirmed at the beginning of the experiment that the participant should strive to give their honest opinion since it is the most helpful. They also remembered that before answering the posed questions.

5.1.2 Time Measurements

In order to evaluate the system more objectively, we propose to measure the time delay of the interaction with the 3D model and the time to download the different types of models. The measurements were made in a Lenovo Legion Y540-15IRH, with 16GB of RAM and an Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz. The measurements were made in São Miguel, Azores, which is very far away from the CERN server, which can hinder the measurements. We used the console.time and console.timeEnd [Mozilla Foundation, 2023a] to measure the times. The measurements were performed five times, and we extracted and analyzed the average and the standard deviation of the samples. The interaction lag was measured using the Stats panel from Three.js [Stats.js Community, 2023]. We measured the FPS and the time, in milliseconds, needed to render the next frame for the different types of models. Since Three.js requests the browser to render the scene, the refresh rate is aligned with the monitors rate.

5.1.3 Simplification of the Model's Geometry

In order to simplify the model's geometry, we used the simplification software Simplygon. We opted to use triangle decimation with a visibility remover that removes invisible triangles. To evaluate the performance of this software, we compare the reduction in the number of triangles, the reduction in filesize, and the average and standard deviation of the time taken to perform the reduction. Furthermore, we compare the three different types of models (global, detailed, and preview) using these statistics. The measurements were taken in a Lenovo Legion Y540-15IRH, with 16GB of RAM and an Intel(R) Core(TM) i5-9300HF CPU @ 2.40GHz. We used the process_time function of the library time in Python two measure the time [Python Software Foundation, 2023].

5.2 Results and Analysis

Having decided on how the evaluation methods, we now present their results. We start by presenting the results from the controlled experiment and using them to answer the research questions alongside some questions about the application regarding the design decisions. The full and raw unedited data can be found in Appendix C. Furthermore, we present the results of the time measurements and the inclusion of LINAC4.

5.2.1 Controlled Experiment Results

The experiments counted 20 participants: ten from CERN and ten from outside. Regarding sex, the participants are evenly distributed, and the average age is 28 years. 90% of the participants had never used the Scheduling Tools website before, and 65% had interacted with a 3D model.

First, we need to verify any differences between the groups from CERN and non-CERN, i.e., if having a better knowledge of the accelerators and the institution helps to interact with the application. For this purpose, we used the Mann–Whitney U test [Mann and Whitney, 1947] to check if the groups are similar. This is a non-parametric statistical test used to determine if there is a significant difference between two independent groups. For an $\alpha = 0.05$ (significance level) and group sizes of $n_1 = 10$ and $n_2 = 10$, we get a $U_{crit} = 23$ (the critical value). This means that only if our test statistic is smaller than the critical value can we reject the null hypothesis, i.e., the groups are similar. We performed the test on the sum of time taken to perform all tasks of the experiment and got a $U_{stat} = 28$. This means that we can't reject the null hypothesis, and the groups are, in fact, similar.

Furthermore, we need to address the possibility of transfer learning. A participant that starts with the 2D system (group A) may gain an understanding of the graphical elements and their meanings and apply it to the 3D system. This makes them perform better than the participants that started with the 3D system and bias the result. The same can be said for the participants that started with the 3D system (group B). We performed the Mann-Whitney U test to check this bias. We did two tests, one for the 2D system and another for the 3D system. For both systems, an $\alpha = 0.05$ and group sizes of $n_1 = 10$ and $n_2 = 10$, we get a $U_{crit} = 23$. Both tasks used the time taken to perform the tasks of the associated system. For the 2D system test, we got a $U_{stat} = 10$, and for the 3D system test, we got a $U_{stat} = 48$. This means the 3D system doesn't suffer from bias, but the 2D system does. In other words, participants that start with the 3D system seem to better apply the knowledge gained in it to the 2D system, while the contrary can't be said.

The next item to be discussed is the main one, which is: "Is there any difference between the 2D and the 3D systems?". To answer this question, we performed the Wilcoxon signed-rank test [Wilcoxon, 1945]. This is a non-parametric statistical test, similar to the Mann-Whitney U test, that is used to determine if there is a significant difference between two paired groups. According to this test, if the value calculated by it is higher than the critical value, then we can't reject the null hypothesis, i.e., the groups have no significant difference. There needs to be special care in choosing the tasks to be compared. The 3D system has more functionalities than the 2D, so it has more tasks. This makes that we can't directly compare the 2D with the 3D. We pick three tasks that are very similar in both systems. For the 2D system, we chose S3 (Scenario 3) T2 (Task 2), S4 T1, and S7. For the 3D system, we chose S3 T3, S4 T1, and S7. We summed the time and number of clicks for both systems. For both tests, we used chose an $\alpha = 0.05$. Considering the population size is 20, we have a critical value of $W_{crit} = 52$. After performing the test, we got a result of $W_{stat} = 79$ for the time and $W_{stat} = 41$ for the number of clicks. This means that the 2D and 3D systems are similar regarding the time to complete the tasks but differ in the number of clicks. We have a theory to explain the similarity in time. We believe that the 3D system has a higher learning curve than the 2D and that if the users repeated the tasks, they would perform much better. Table 5.1 shows the average and the standard deviation of the time duration and clicks for the tasks discussed above, plus the S4 T2. We can see that the standard deviation is much higher in the 3D system. In S3, the standard deviation is even larger than the average. This means some participants are much more comfortable with the 3D system than others. Moreover, if we see the results for S4 T2, essentially equal to S4 T1 but asked after, we can see that they improved dramatically, surpassing the 2D results. These indicate that after the participants learn the commands, they can retrieve the necessary information much quicker in the 3D system than in the 2D. We don't anticipate the higher learning curve being an issue since the system is intended for professional use, and it is expected for users to take some time to learn it. Also, the average number of clicks is lower for the 3D system, especially considering task 2 of Scenario 4.

We now analyze every scenario to determine the effect of the design decision on user interaction. Scenario 1 had the goal of verifying if a user could find the visualization page. We recorded that 60% went directly to the Visualization tab, 25% went to other tabs but eventually found the correct one, and 15% needed help to find the tab. Since 85% of participants eventually found the page without help and it is a one-time issue, this is an acceptable percentage. We also detected that 45% of the participants didn't understand that the tab "Interactive" concerned the 2D view. They were vocal regarding this issue, and we recommend changing the name to another more descriptive one.

Scenario 2 has the goal of checking if the participant understands the 3D system interaction commands. We observed that 90% of the participants didn't know how to translate the model.

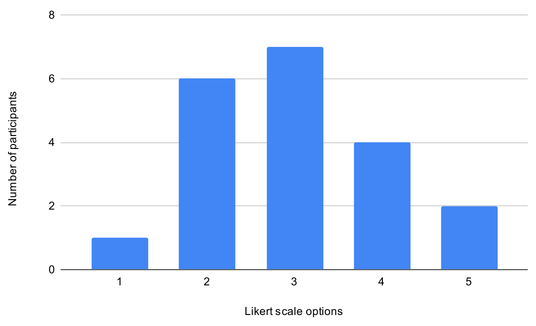
Table 5.1: Result comparison between the 2D and 3D systems regarding time and number of clicks. μ is the average and σ the standard deviation. S is short for Scenario, and T is short for Task

	S3		S4			S7	
	2D	3D	T1 2D	T1 3D	T2 3D	2D	3D
μ_{time}	1.685	0.83	0.918	1.575	0.374	0.9385	1.056
σ_{time}	0.779	0.886	0.426	1.174	0.263	0.457	0.431
μ_{clicks}	35.35	16.75	17.15	21.7	6.55	26.25	21.25
σ_{clicks}	20.327	19.981	8.152	18.968	7.178	10.891	10.140

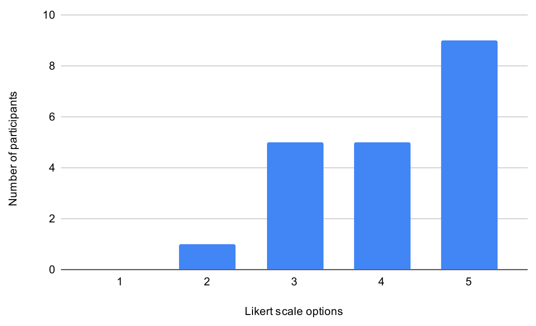
This is a weak point of the system since one may need to reframe the model to view it better. One solution to fix this is to detect when the user is manipulating the model a lot in a short period of time and display a short animation explaining the command. 70% of the participants saw the buttons in the beginning, and 90% saw the Controls menu. When asked to explain the options of the control menu, only one participant couldn't do it. All other participants could correctly identify the functionality of the options, with the exception of the "focus" option. This could be fixed with a popup with a short description of the option displayed when the user hovers the mouse above it.

Moving on to Scenario 3, which has the purpose of evaluating the process of information retrieval for both the 2D and 3D systems. In the 2D system, 85% of participants could not correctly count the number of tasks present in PS octant. There are two possible reasons for this. First, the color scheme is not intuitive for most people and may lead to participants counting more than one time the same task. The second possible reason is the lack of a clear separation of the sections. This was identified as an issue by some of the participants. Furthermore, we analyzed if the participants could correctly identify the section's name on the 2D drawing. 55% could not do this because the labels of the drawing were too small to easily read. Several participants squinted their eyes and got closer to the monitor to see the section names. The 3D system obtained better results. 60% of the participants identified all sections correctly. The other 40% missed a small section that had a task, which can become a problem. To fix this, we could add an extra graphical element on the scene identifying the section with a task, e.g., the floor below the section could change to the color of the responsible group for the task. Another problem is that 75% of participants didn't discover the task information menu on their own. Most people tried to click on the section but didn't move to double-click. To fix this, we can put an animation of a mouse double-click when the user tries to click on a section with a task more than once in a short time period.

As stated before, two questions were posed to the participants after Scenario 3 (PQ1 and PQ2) for both the 2D and 3D systems. The responses can be seen in Fig. 5.1 and in Fig. 5.2. In the 2D system, PQ1 had worse responses, with most participants answering 2 and 3. However, users found it easy to identify the sections with tasks. The 3D system obtained better scores for both PQ1 and PQ2. Most users answered 4 and 5 for both questions.



(a) Column chart of the responses: "On a scale from 1 to 5, how easy was it to retrieve the task information?"



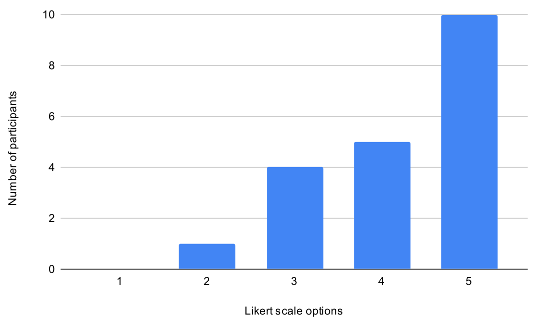
(b) Column chart of the responses to: "On a scale from 1 to 5, how easy is it to detect the sections that had tasks?"

Figure 5.1: Responses to PQ1 and PQ2 regarding the 2D system

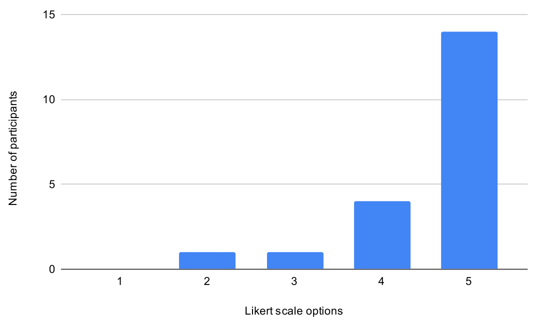
Scenario 4 tests if the user understands how to use the date slider correctly. There are two ways of changing the date: by the top date selector, which refreshes the webpage, and the bottom date slider, which doesn't need to reload the webpage to change the date. 90% of users used the bottom date slider and understood its usage, so there is no need to improve it.

Scenario 5 evaluates the group selection and filter functionality. 60% used the Controls menu group option. The other 40% used the "Show Tasks" functionality, which displays a list of the tasks present at the date, or went to the detailed view of every octant and counted the different groups. Only one participant got the answer wrong because he thought that only the groups that he unchecked and changed the color of the global model were the groups that had tasks. This is not the case since most sections had other tasks and were painted with the "multiple" color.

At the end of the experiment, two general questions about the project were asked. Regarding the difference between the 2D and 3D systems according to efficiency and usability, the consensus was that the 3D system is more efficient in information retrieval, better for visualizing the sections, has more information, and provides it in a more understandable way. However, three participants



(a) Column chart of the responses: "On a scale from 1 to 5, how easy was it to retrieve the task information?"



(b) Column chart of the responses to: "On a scale from 1 to 5, how easy is it to detect the sections that had tasks?"

Figure 5.2: Responses to PQ1 and PQ2 regarding the 3D system

said the 2D can be better at providing a global view of the tasks in the accelerator. The main improvements suggested are:

- inside the task information menu, have a link to other web pages that contain more information about a task. For example, the Gantt view, Calendar view, or the GIS portal.
- make the information inside the task information menu customizable. Add a configuration menu where the user can select the data he wants to see in the menu. The application should remember the preferred configuration for the user. Also regarding customization, have the option to store the view of the model for a later date.

After the open-ended questions, we gave the SUS questionnaire to the participants to answer. The results can be seen in Fig. 5.3. The questions alternate from positive (e.g., from 1 to 5, how intuitive was the interface?) and negative (e.g., from 1 to 5, how difficult was it to learn the commands?). As we can see, the results were mostly positive. We got an average SUS score of 86.8 and a minimum and maximum score of 72.5 and 100, respectively. Using the adjective rating system [Bangor et al., 2009], we understood that 65% of participants rated the system as "Excellent", and all participants rated it at least as "Good".

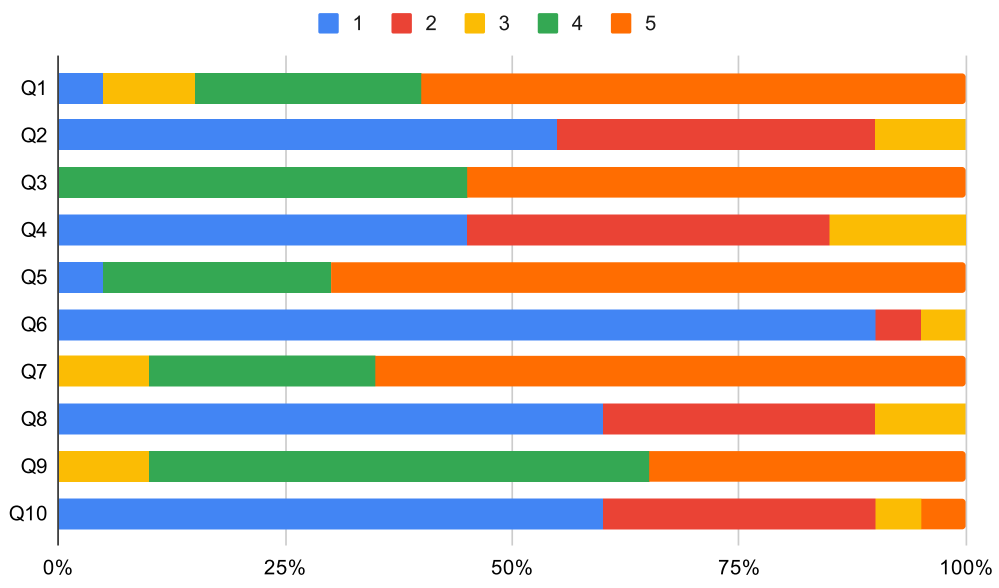


Figure 5.3: Results of the SUS questionnaire

As discussed above, the time difference between the 2D and 3D systems was not significant. However, this is due to the higher learning curve of the 3D system, as can be seen by the much lower completion time of a similar already performed task, in Scenario 4. Moreover, the open-ended answers and the SUS questionnaire results indicate that all participants preferred the 3D system. Considering all of these, we can answer RQ2 by saying delivering information efficiently in a 3D system can be accomplished by providing several levels of detail and granularity that the user can easily manipulate to get the information he wants.

5.2.2 Time Measurements Results

We now present the results of the time measurements taken. These can be seen on Table 5.2. We noted that when the system was evaluated with the users on the controlled experiment in mainland Portugal, the models loaded faster. We obtained an initial time delay of 0.5s (when starting the application). This is a lower waiting time than the one obtained by [Zhang et al., 2020]. This should be mainly due to the increased distance from the server when performing the measurements. As we can see from the results, some models take longer to load than others. This is due to the fact that some octants have increased details (extra sections that only serve to add realism, which was kept by CERN's request), which results in a larger file size. The octant details model takes around 1 to 2 seconds to load, which would be noticeable to the user. This is why the preview models are important: they reduce the waiting time of the user to visualize the model. The preview models only take around half a second to load. In the controlled experiments, no users noticed the delay time or even the existence of the preview model. Regarding the FPS, we recorded an average of 30 FPS in the global model. When loading a detailed octant model, the FPS drops to 11 and then stabilizes at 25 FPS. The lower FPS value comes from the increased size of the detailed model. This is similar to the one recorded by [Zhou et al., 2018]. Considering the time needed to render the next frame, we measured an average of 40 ms for the global model. When loading the detailed octant model, the time spikes two times to values over 150ms: when the preview model is loaded and when the detailed model is loaded. It then stabilizes at 40ms again. Considering these results and that the participants of the controlled experiments didn't notice any type of delay, we can answer RQ1 by saying that large and complex 3D models can be displayed on the web with low delay and lag by simplifying the geometry of the 3D model and providing an extremely simplified version of the model while it's loading so the user doesn't need to wait a lot to see the model on the screen.

Table 5.2: Loading times, in milliseconds, of the different 3D models. Avg refers to the average, and the STD to the standard deviation. The global model doesn't have a preview model

Model	Avg. Model Time	STD Model Time	Avg. Preview Time	STD Preview Time
Octant 1	2212.6	731.739	648.4	244.005
Octant 2	1035.4	197.410	654.4	767.780
Octant 3	2188.8	162.347	429.6	39.151
Octant 4	1568.8	285.329	479.6	62.919
Octant 5	1244.6	110.497	436.8	109.691
Octant 6	2151	253.999	417.8	69.182
Octant 7	1669.6	310.914	527.6	118.875
Octant 8	1620.8	542.888	499.8	197.870
Global	508	31.185	-	-

5.2.3 Simplification of the Models' Geometry Results

The results of the simplification of the models' geometry can be found in Table 5.3. Simplygon takes a similar amount of time to the algorithm described in [Salinas et al., 2015], but reduces much less the geometry of the model, with possibly much less accuracy (Simplygon doesn't provide the option to measure the Mean error). The results of the simplification were satisfactory, but in the future, it may be advantageous to try other algorithms and other techniques

Table 5.3: Results of the simplification process using Simplygon. #Vert refers to the number of vertices and orig. and simp. mean original and simplified, respectively. The filesize was measured in Megabytes. VPerc is the reduction percentage of the number of vertices, and FPerc is the reduction percentage of the filesize. μ_{time} is the average of the reduction time, and σ_{time} its standard deviation

Model	Orig. #Vert	Simp. #Vert	VPerc.	Orig. filesize	Simp. filesize	FPerc	μ_{time}	σ_{time}
Global	20881	4176	80.0%	10.3	2.26	78.1%	6.026	0.6242
Detail	542912	107248	80.2%	171	82	52.0%	297.297	140.654
Preview	542912	13405	97.5%	171	26.8	84.3%	307.037	149.936

5.2.4 Expansion to LINAC4

The first step of the inclusion process of LINAC4 into the 3D system is to get the CAD models. We downloaded the model directly from CATIA. The model is divided into two parts: the surface buildings and the machine itself, located underground. The interior of the surface building didn't have enough detail to separate it into sections, so we didn't create a detailed model for it. The interior of the machine was missing a part. We tried to discuss it with the CERN modeling group, but they responded that the CATIA model had the supposedly missing parts. The model remains incomplete.

We then removed all unnecessary details from the machine, such as the electrical wires and pipelines, and converted them to glb. Afterward, we simplified the model using Simplygon. This process remains identical.

To add LINAC4 to the 3D visualization web tool, we need to create a folder in the "models" folder and put the 3D models there. The 3D models should follow the structure represented in Fig. 4.11. We then adapted the PS configuration files and used them for LINAC4. The process of adaption consists of changing the names and paths to the 3D models of the "download" configuration file, replacing the "locations" file for one containing the LINAC4 locations (extracted from CERN API), and changing the "properties" file to represent the properties of LINAC4 properly. The final step is to add the configuration files to the function that returns the configurations for the current machine. This function is present in the utils component. Now the LINAC4 model can be seen on the Scheduling Tools website.

Scenario 7 serves the purpose of testing if the process of retrieving task information is identical for the PS and for the LINAC4 machine. The results of the question asked after the task are shown in Fig. 5.4. As we can see, every participant thought that the process of information retrieval was identical in both machines.

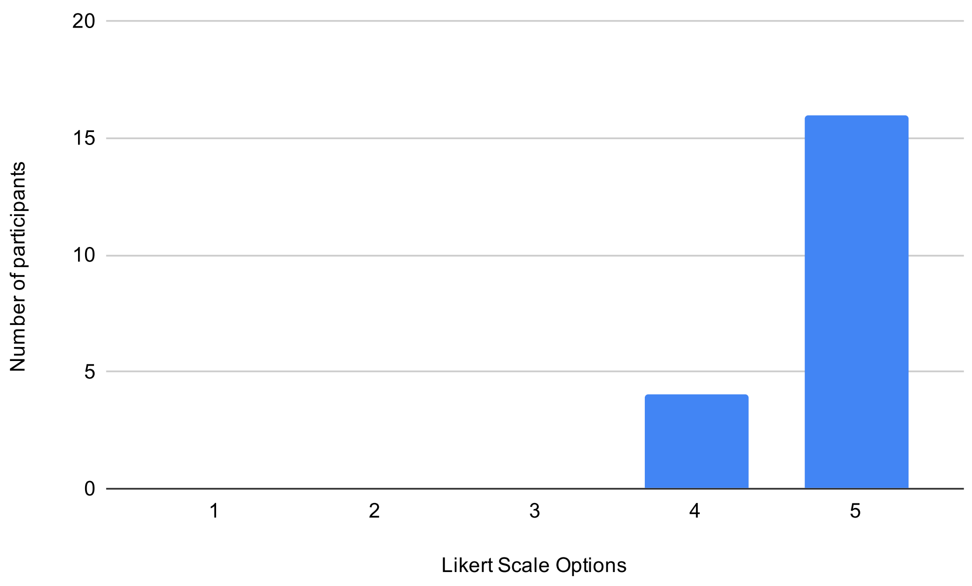


Figure 5.4: Results of the post question of Scenario 7

Chapter 6

Conclusion and Future Work

We end this dissertation by stating the problem faced and briefly describing the system developed in order to solve it. Moreover, we present the main results of the evaluation process applied to the system. Finally, we suggest some possible future work.

6.1 Conclusions

CERN possesses several accelerators and machines that need constant maintenance to work properly. CERN staff need to plan their experiments and future maintenance tasks around the scheduled ones. Verifying which activities are scheduled for which sectors of the accelerator and planning future tasks might be a complex and time-consuming task. In this dissertation, we proposed a 3D visualization system to display information about the maintenance tasks of the PS machine.

The developed solution consists of two parts: a preprocessing pipeline that converts and simplifies the model in a semi-automatic way and a web visualization tool that renders the model and provides the needed information. We started by extracting the global and detailed models from the CAD model, removing objects that couldn't be used to provide the task information, and converting it to glb. The model was then simplified by performing triangle decimation and removing invisible triangles. The web visualization tool comprises six main components, each with a single responsibility. The main component stores the state, while the others provide helper and utility functions to view the model, interact with the user, and provide needed information. The constructed tool was integrated into the Schedule Tools website as a new way to visualize maintenance tasks. The addition of new machines was accomplished by using configuration files. These files contain all information specific to the machine and are served to the application according to the current machine being visualized.

The visualization system was evaluated by performing a controlled experiment and measuring interaction lag and loading times. The controlled experiment is composed of seven scenarios, each with the goal of testing a different aspect of the system and the design decisions made. Some scenarios had post questions that asked the participant to rate a specific aspect of the 3D system. Two open-ended questions were asked at the end of the experiment to capture the opinion of the user regarding the system's efficiency. Moreover, we used the SUS questionnaire to quantify the usability of the developed system and allow comparison between other systems and future versions. The results obtained showed that, even though we didn't capture a significant difference between the 2D and 3D systems regarding time, when the user gets more familiarized with the 3D tool, the time spent and the number of clicks needed reduced drastically. Furthermore, the participants' consensus was that the 3D visualization tool had more information, was more efficient in providing it, and was more user-friendly than the 2D tool. So, we concluded that an efficient way of encoding information into a 3D model is by providing multi-levels of detail and granularity that the user can manipulate easily to get the information he needs. Regarding the taken time measurements, even though the distance from the main server influenced them, we observed that the waiting time was low and not noticeable to the participants. Besides, the application maintained a stable FPS, with the exception of when it was loading the detailed models. From this, we concluded that complex and large 3D models can be displayed on the web with low latency by simplifying the geometry of the model and providing an extremely detailed version of the model while the principal model is loading.

6.2 Future Work

The developed system is functional and satisfies all of the established criteria. However, we identified some possible improvements and features that, if implemented, would increase the efficiency of the system and add new functionalities. In this section, we analyze some of these possible alterations.

The current solution uses the web browser to render the model and process user interaction. Since the browser has limited computing power, we needed to simplify the model in order for the user to interact with it without any noticeable lag. However, the simplification process can alter the geometry of the model and may not be enough for future bigger machines. If one wants to keep an accurate depiction of the model's geometry, one can implement a rendering server. The browser would connect to the rendering server and send the user's input. The server would render the model, manipulate it according to the received user's input and send the result to the browser. This way, if the server has a high enough computing power and the internet is fast enough, one may not need to simplify the model and obtain maximum geometry fidelity. One can even use the Pixel Streaming functionality of Unreal Engine for this purpose [Epic Games, 2023b]. However, the server needs to have enough computing power to manage several requests at once.

Currently, the model is converted to glb and simplified using third-party tools. Nonetheless, the dependency on external software may not be desired since we don't control it, and it can change in a way that hinders or completely halts the system. For this purpose, we propose as a possible future thesis theme a conversion and simplification algorithm that optimizes the model for web visualization and can be easily integrated into CATIA as a plugin. This would simplify and automate the conversion and simplification model and allow a quicker and more efficient addition of new machines to the system. The proposed algorithm could also automatically generate, from the CAD model of the machine, the global and detailed version of the model without unnecessary details.

The conceived 3D system is integrated only into the Scheduling Tools website. Nevertheless, CERN possesses other websites that provide different types of visualization of the machines. This is the case with the GIS portal, which is a website that provides a panoramic view of all machines and CERN buildings. As a possible future work, our 3D system could be integrated into the GIS portal by providing a list of the ongoing tasks of the section displayed in the panoramic view. This can provide the user with a better perception of the machine, allow finer granularity of the task location inside the section, and provide a simple demonstration animation of the work scheduled.

Bibliography

[Assila et al., 2016] Assila, A., De Oliveira, K. M., and Ezzedine, H. (2016). Standardized Usability Questionnaires: Features and Quality focus. *HAL (Le Centre pour la Communication Scientifique Directe)*. Available at https://uphf.hal.science/hal-03400437.

[Autodesk, 2023] Autodesk (2023). Fbx documentation. Available in https://help.autodesk.com/view/FBX/2020/ENU/?guid=FBX_Developer_Help_welcome_to_the_fbx_sdk_what_new_fbx_sdk_2020_html, last access on 24 of February of 2023.

[Autodesk, Inc., 2011] Autodesk, Inc. (2011). *DXF Reference*. Autodesk, Inc.

[Bangor et al., 2009] Bangor, A., Kortum, P., and Miller, J. H. (2009). Determining what individual SUS scores mean: adding an adjective rating scale. *Journal of Usability Studies archive*, 4(3):114–123. Available at http://uxpajournal.org/wp-content/uploads/pdf/JUS_Bangor_May2009.pdf.

[Barnes and Finch, 2008] Barnes, M. and Finch, E. L. (2008). *COLLADA – Digital Asset Schema Release 1.5.0*. The Khronos Group Inc. and Sony Computer Entertainment Inc., Clearlake Park, CA 95424, U.S.A.

[Bellalouna, 2020] Bellalouna, F. (2020). Industrial Case Studies for Digital Transformation of Engineering Processes using the Virtual Reality Technology. *Procedia CIRP*, 90:636–641. DOI:10.1016/j.procir.2020.01.082.

[Berlin and Kay, 1991] Berlin, B. and Kay, P. (1991). *Basic color terms: Their universality and evolution*. Univ of California Press.

[Bertin, 1983] Bertin, J. (1983). *Semiology of graphics*. University of Wisconsin press.

[Blandford et al., 2008] Blandford, A., Cox, A. L., and Cairns, P. (2008). Controlled experiments. In *Research Methods for Human Computer Interaction*. Cambridge University Press.

[Brooke, 1996] Brooke, J. H. (1996). *SUS: a 'Quick and Dirty' usability scale*. CRC Press. DOI:10.1201/9781498710411-35.

[CERN, 2023] CERN (2023). Cern webpage. Available in https://home.cern/, last access on 09 of January of 2023.

[Cesium GS, Inc., 2022a] Cesium GS, Inc. (2022a). Cesium github. Available in https://github.com/CesiumGS/cesium, last access on 06 of November of 2022.

[Cesium GS, Inc., 2022b] Cesium GS, Inc. (2022b). Cesium website. Available in https://cesium.com/, last access on 06 of November of 2022.

[Cignoni et al., 2008] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association. DOI:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[Cohen-Steiner et al., 2004] Cohen-Steiner, D., Alliez, P., and Desbrun, M. (2004). Variational shape approximation. *ACM Transactions on Graphics*, 23(3):905–914. DOI:10.1145/1015706.1015817.

[Dix et al., 2003] Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (2003). *Human-computer interaction*. Pearson Education.

[Epic Games, 2023a] Epic Games (2023a). Datasmith webpage. Available in https://www.unrealengine.com/en-US/datasmith, last access on 01 of May, 2023.

[Epic Games, 2023b] Epic Games (2023b). Unreal engine webpage. Available in https://www.unrealengine.com/en-US, last access on 10 of April, 2023.

[Fassi et al., 2015] Fassi, F., Achille, C., Mandelli, A., Rechichi, F., and Parri, S. (2015). A NEW IDEA OF BIM SYSTEM FOR VISUALIZATION, WEB SHARING AND USING HUGE COMPLEX 3D MODELS FOR FACILITY MANAGEMENT. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-5/W4:359–366. DOI:10.5194/isprsarchives-xl-5-w4-359-2015.

[Foley, 1996] Foley, J. D. (1996). *Computer graphics: principles and practice*, volume 12110. Addison-Wesley Professional.

[Fraunhofer-Gesellschaft, 2022a] Fraunhofer-Gesellschaft (2022a). X3dom documentation. Available in https://doc.x3dom.org/gettingStarted/index.html, last access on 08 of November of 2022.

[Fraunhofer-Gesellschaft, 2022b] Fraunhofer-Gesellschaft (2022b). X3dom github. Available in https://github.com/x3dom/x3dom, last access on 06 of November of 2022.

[Fraunhofer-Gesellschaft, 2022c] Fraunhofer-Gesellschaft (2022c). X3dom tag on stack overflow. Available in https://stackoverflow.com/questions/tagged/x3dom, last access on 29 of November of 2022.

[Fraunhofer-Gesellschaft, 2022d] Fraunhofer-Gesellschaft (2022d). X3dom webpage. Available in https://www.x3dom.org/, last access on 06 of November of 2022.

[Friendly, 2008] Friendly, M. (2008). *A brief history of data visualization*. Springer. DOI:10.1007/978-3-540-33037-0_2.

[Garland and Heckbert, 1997] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 209–216, USA. ACM Press/Addison-Wesley Publishing Co. DOI:10.1145/258734.258849.

[GitHub, 2022] GitHub (2022). Three.js on github. Available in https://github.com/mrdoob/three.js/, last access on 23 of November, 2022.

[Gregory, 2018] Gregory, J. (2018). *Game engine architecture*. CRC Press.

[Guthe et al., 2004] Guthe, M., Balázs, A., and Klein, R. (2004). Real-time out-of-core trimmed nurbs rendering and editing. In *International Symposium on Vision, Modeling, and Visualization*, pages 323–330.

[Hansen and Johnson, 2011] Hansen, C. D. and Johnson, C. R. (2011). *Visualization Handbook*. Elsevier.

[ISO/IEC, 1997] ISO/IEC (1997). *Information technology - Computer graphics and image processing - The Virtual Reality Modeling Language*. ISO/IEC.

[James and Twigg, 2005] James, D. L. and Twigg, C. D. (2005). Skinning mesh animations. *ACM Transactions on Graphics*, 24(3):399–407. DOI:10.1145/1073204.1073206.

[Kanai et al., 2012] Kanai, S., Iyoda, D., Endo, Y., Sakamoto, H., and Kanatani, N. (2012). Appearance preserving simplification of 3D CAD model with large-scale assembly structures. *International Journal On Interactive Design And Manufacturing (ijidem)*. DOI:10.1007/s12008-012-0145-0.

[Khronos Group, 2023] Khronos Group (2023). gltf™ 2.0 specification. Available in https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html, last access on 24 of February of 2023.

[Kim Kulling, 2023] Kim Kulling (2023). Assimp webpage. Available in http://www.assimp.org/, last access on 05 of June, 2023.

[Kirakowski and Corbett, 1993] Kirakowski, J. and Corbett, M. J. (1993). SUMI: the Software Usability Measurement Inventory. *British Journal of Educational Technology*, 24(3):210–212. DOI:10.1111/j.1467-8535.1993.tb00076.x.

[Latiffi et al., 2014] Latiffi, A. A., Brahim, J., and Fathi, M. S. (2014). The development of Building Information Modeling (BIM) definition. *Applied Mechanics and Materials*, 567:625–630. DOI:10.4028/www.scientific.net/amm.567.625.

[LaViola et al., 2017] LaViola, Jr, J. J., Kruijff, E., McMahan, R. P., Bowman, D., and Poupyrev, I. P. (2017). *3D user Interfaces*. Addison-Wesley Professional.

[Li and Nan, 2021] Li, M. and Nan, L. (2021). Feature-preserving 3D mesh simplification for urban buildings. *Isprs Journal of Photogrammetry and Remote Sensing*, 173:135–150. DOI:10.1016/j.isprsjprs.2021.01.006.

[Lindstrom and Turk, 1998] Lindstrom, P. and Turk, G. (1998). Fast and memory efficient polygonal simplification. In *Proceedings of the Conference on Visualization '98*, VIS '98, page 279–286, Washington, DC, USA. IEEE Computer Society Press. DOI:10.5555/288216.288288.

[Lindstrom and Turk, 1999] Lindstrom, P. J. and Turk, G. (1999). Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):98–115. DOI:10.1109/2945.773803.

[Liu et al., 2021] Liu, Z., Gu, X., Dong, Q., Tu, S., and Li, S. (2021). 3D visualization of airport pavement quality based on BIM and WebGL integration. *Journal of transportation engineering*, 147(3). DOI:10.1061/jpeodx.0000280.

[Lorenz et al., 2016] Lorenz, M., Spranger, M., Riedel, T., Pürzel, F., Wittstock, V., and Klimant, P. (2016). CAD to VR – A Methodology for the Automated Conversion of Kinematic CAD Models to Virtual Reality. *Procedia CIRP*, 41:358–363. DOI:10.1016/j.procir.2015.12.115.

[Lund, 2001] Lund, A. (2001). Measuring usability with the use questionnaire. *Usability and User Experience Newsletter of the STC Usability SIG*, 8.

[Malan et al., 2001] Malan, R., Bredemeyer, D., et al. (2001). Functional requirements and use cases. *Bredemeyer Consulting*.

[Mann and Whitney, 1947] Mann, H. J. and Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, 18(1):50–60. DOI:10.1214/aoms/1177730491s.

[Mazza, 2009] Mazza, R. (2009). *Introduction to information visualization*. Springer London. DOI:10.1007/978-1-84800-219-7.

[Microsoft, 2022a] Microsoft (2022a). Babylon.js documentation. Available in `https://doc.babylonjs.com/`, last access on 03 of November of 2022.

[Microsoft, 2022b] Microsoft (2022b). Babylon.js github. Available in `https://github.com/BabylonJS/Babylon.js`, last access on 05 of November of 2022.

[Microsoft, 2022c] Microsoft (2022c). Babylon.js mainpage. Available in `https://www.babylonjs.com/`, last access on 05 of November of 2022.

[Microsoft, 2023] Microsoft (2023). Simplygon webpage. Available in https://www.simplygon.com/, last access on 06 of June, 2023.

[Migilinskas et al., 2013] Migilinskas, D., Popov, V., Juocevicius, V., and Ustinovičius, L. (2013). The benefits, obstacles and problems of practical BIM implementation. *Procedia Engineering*, 57:767–774. DOI:10.1016/j.proeng.2013.04.097.

[MIT, 2023] MIT (2023). Mit license. License available in https://mit-license.org/.

[Mozilla Foundation, 2023a] Mozilla Foundation (2023a). console: time() method. Available in https://developer.mozilla.org/en-US/docs/Web/API/Console/time last access on 04 of June, 2023.

[Mozilla Foundation, 2023b] Mozilla Foundation (2023b). Webgl documentation. Available in https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API, last access on 23 of June, 2023.

[Muntoni and Cignoni, 2021] Muntoni, A. and Cignoni, P. (2021). PyMeshLab.

[Nielsen and Molich, 1990] Nielsen, J. and Molich, R. (1990). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 249–256, New York, NY, USA. Association for Computing Machinery. DOI:10.1145/97243.97281.

[Open Cascade, 2023] Open Cascade (2023). Cad assistant webpage. Available in https://www.opencascade.com/products/cad-assistant/, last access on 06 of June, 2023.

[OPEN CASCADE SAS, 2023] OPEN CASCADE SAS (2023). Opencascade technlogoy. Available in https://dev.opencascade.org/, last access on 22 of June, 2023.

[OpenJS Foundation and Electron contributors, 2023] OpenJS Foundation and Electron contributors (2023). Electron documentation. Available in https://www.electronjs.org/docs/latest/, last access on 23 of June, 2023.

[Park et al., 2017] Park, J., Cai, H., Dunston, P. S., and Ghasemkhani, H. (2017). Database-Supported and Web-Based visualization for daily 4D BIM. *Journal of the Construction Division and Management*, 143(10). DOI:10.1061/(asce)co.1943-7862.0001392.

[Paviot, 2023] Paviot, T. (2023). Pythonocc github. Available in https://github.com/tpaviot/pythonocc-core, last access on 05 of June, 2023.

[Pixar Animation Studios, 2021] Pixar Animation Studios (2021). Universal scene description. Available in https://openusd.org/release/index.html last access on 25 of June, 2023.

[Poimandres, 2023] Poimandres (2023). React three fiber documentation. Available in https://docs.pmnd.rs/react-three-fiber last access on 26 of June, 2023.

[Polson et al., 1992] Polson, P. G., Lewis, C., Rieman, J., and Wharton, C. (1992). Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International journal of man-machine studies*, 36(5):741–773. DOI:10.1016/0020-7373(92)90039-n.

[Prandi et al., 2015] Prandi, F., Devigili, F., Soave, M., Di Staso, U., and De Amicis, R. (2015). 3D web visualization of huge CityGML models. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-3/W3:601–605. DOI: 10.5194/isprsarchives-xl-3-w3-601-2015.

[Python Software Foundation, 2023] Python Software Foundation (2023). Python time package documention. Available in https://docs.python.org/3/library/time.html last access on 04 of June, 2023.

[Raposo et al., 2006] Raposo, A., Corseuil, E. T. L., Wagner, G. N., dos Santos, I. H. F., and Gattass, M. (2006). Towards the use of cad models in vr applications. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and Its Applications*, VRCIA '06, page 67–74, New York, NY, USA. Association for Computing Machinery. DOI:10.1145/1128923.1128935.

[Salinas et al., 2015] Salinas, D., Lafarge, F., and Alliez, P. (2015). Structure-Aware mesh decimation. *Computer Graphics Forum*, 34(6):211–227. DOI:10.1111/cgf.12531.

[Santos et al., 2021] Santos, B. F., Rodrigues, N. B., Costa, P., and Coelho, A. (2021). Integration of CAD Models into Game Engines. In *VISIGRAPP*. DOI:10.5220/0010201701530160.

[Sauro and Lewis, 2016] Sauro, J. and Lewis, J. D. (2016). *Quantifying the User Experience, Second edition: Practical Statistics for User research*. Morgan Kaufmann. Available at http://dl.acm.org/citation.cfm?id=3044726.

[Schroeder et al., 2006] Schroeder, W., Martin, K., and Lorensen, B. (2006). *The Visualization Toolkit*. Kitware, 4th edition.

[Spence, 2001] Spence, R. (2001). *Information visualization*, volume 1. Springer.

[Stack Overflow, 2022a] Stack Overflow (2022a). A-frame tag on stack overflow. Available in https://stackoverflow.com/questions/tagged/aframe, last access on 29 of November of 2022.

[Stack Overflow, 2022b] Stack Overflow (2022b). Babylon.js tag on stack overflow. Available in https://stackoverflow.com/questions/tagged/babylonjs, last access on 23 of November, 2022.

[Stack Overflow, 2022c] Stack Overflow (2022c). Cesium tag on stack overflow. Available in https://stackoverflow.com/questions/tagged/cesiumjs, last access on 29 of November of 2022.

[Stack Overflow, 2022d] Stack Overflow (2022d). Three.js tag on stack overflow. Available in https://stackoverflow.com/questions/tagged/three.js, last access on 23 of November, 2022.

[Stats.js Community, 2023] Stats.js Community (2023). Javascript performance monitor github. Available in https://github.com/mrdoob/stats.js/, last access on 01 of July, 2023.

[Sullivan and Kaszynski, 2019] Sullivan, C. B. and Kaszynski, A. (2019). PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). *Journal of Open Source Software*, 4(37):1450. DOI:10.21105/joss.01450.

[Supermedium, 2022a] Supermedium (2022a). A-frame documentation. Available in https://aframe.io/docs/1.3.0/introduction/, last access on 08 of November of 2022.

[Supermedium, 2022b] Supermedium (2022b). A-frame github. Available in https://github.com/x3dom/x3dom, last access on 06 of November of 2022.

[Supermedium, 2022c] Supermedium (2022c). A-frame webpage. Available in https://aframe.io/, last access on 08 of November of 2022.

[Sustainability of Digital Formats, 2023] Sustainability of Digital Formats (2023). Sustainability of digital formats. Available in https://www.loc.gov/preservation/digital/formats/fdd/fdd000448.shtml#identification, last access on 23 of June, 2023.

[The Apache Software Foundation, 2023] The Apache Software Foundation (2023). Apache license 2.0. License available in https://www.apache.org/licenses/LICENSE-2.0.html.

[The Blender Foundation, 2023] The Blender Foundation (2023). Blender webpage. Available in https://www.blender.org/, last access on 06 of June, 2023.

[The Free Software Foundation, 2023] The Free Software Foundation (2023). Gnu general public license. License available in https://www.gnu.org/licenses/gpl-3.0.en.html last access on 23 of June, 2023.

[The FreeCAD Team, 2023] The FreeCAD Team (2023). Freecad webpage. Available in https://www.freecad.org/, last access on 06 of June, 2023.

[The JSCAD Organization, 2023] The JSCAD Organization (2023). Jscad github. Available in https://github.com/jscad/OpenJSCAD.org, last access on 25 of June, 2023.

[Three.js community, 2022a] Three.js community (2022a). Three.js documentation. Available in https://threejs.org/docs/index.html#manual/en, last access on 04 of November, 2022.

[Three.js community, 2022b] Three.js community (2022b). Three.js github. Available in https://github.com/mrdoob/three.js/, last access on 05 of November, 2022.

[Three.js community, 2022c] Three.js community (2022c). Three.js website. Available in `https://threejs.org/`, last access on 04 of November, 2022.

[Triumph LLC, 2022a] Triumph LLC (2022a). Blend4web github. Available in `https://github.com/TriumphLLC/Blend4Web`, last access on 09 of November of 2022.

[Triumph LLC, 2022b] Triumph LLC (2022b). Blend4web user manual. Available in `https://www.blend4web.com/doc/en/index.html`, last access on 09 of November of 2022.

[Triumph LLC, 2022c] Triumph LLC (2022c). Blend4web webpage. Available in `https://www.blend4web.com/en/`, last access on 09 of November of 2022.

[Tufte, 2001] Tufte, E. R. (2001). *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT.

[Typedoc Community, 2023] Typedoc Community (2023). Typedoc website. Available in `https://typedoc.org/` last access on 28 of June, 2023.

[Wan-Li et al., 2016] Wan-Li, L., Tsai, M.-H., Yang, C.-H., Juang, J.-R., and Su, J.-Y. (2016). V3DM+: BIM interactive collaboration system for facility management. *Visualization in Engineering*, 4(1). DOI:10.1186/s40327-016-0035-9.

[Wang, 2023] Wang, J. (2023). 3dmodelconverttogltf github. Available in `https://github.com/wangerzi/3d-model-convert-to-gltf`, last access on 06 of June, 2023.

[Ware, 2000] Ware, C. (2000). *Information Visualization: perception for design*. Morgan Kaufmann. Available at `http://ci.nii.ac.jp/ncid/BA45754744`.

[Wavefront, 2022] Wavefront (2022). Obj webpage. Available in `https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml`, last access on 08 of December of 2022.

[Web3D Consortium and X3D Working Group, 2023] Web3D Consortium and X3D Working Group (2023). X3d specifications: Schema and doctype validation. Available in `https://www.web3d.org/specifications/` last access on 23 of June, 2023.

[Wilcoxon, 1945] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80. DOI:10.2307/3001968.

[Yong and Narayanaswami, 2003] Yong, T. and Narayanaswami, R. (2003). A parametric interpolator with confined chord errors, acceleration and deceleration for NC machining. *Computer Aided Design*, 35(13):1249–1259. DOI:10.1016/s0010-4485(03)00043-5.

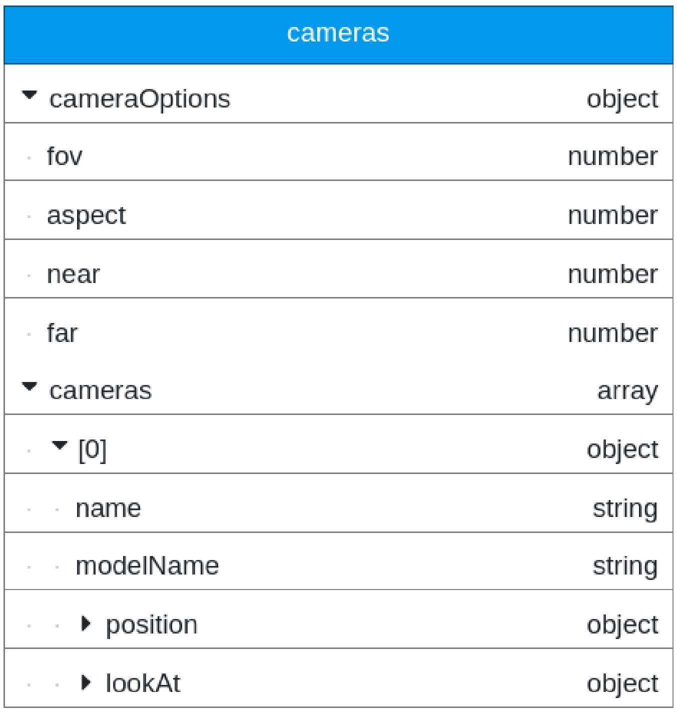
[Zhang et al., 2020] Zhang, S., Hou, D., Wang, C., Pan, F., and Yan, L. (2020). Integrating and managing BIM in 3D web-based GIS for hydraulic and hydropower engineering projects. *Automation in Construction*, 112:103114. DOI:10.1016/j.autcon.2020.103114.

[Zhou et al., 2018] Zhou, X., Wang, J., Guo, M., and Gao, Z. (2018). Cross-platform online visualization system for open BIM based on WebGL. *Multimedia Tools and Applications*, 78(20):28575–28590. DOI:10.1007/s11042-018-5820-0.

Appendix A

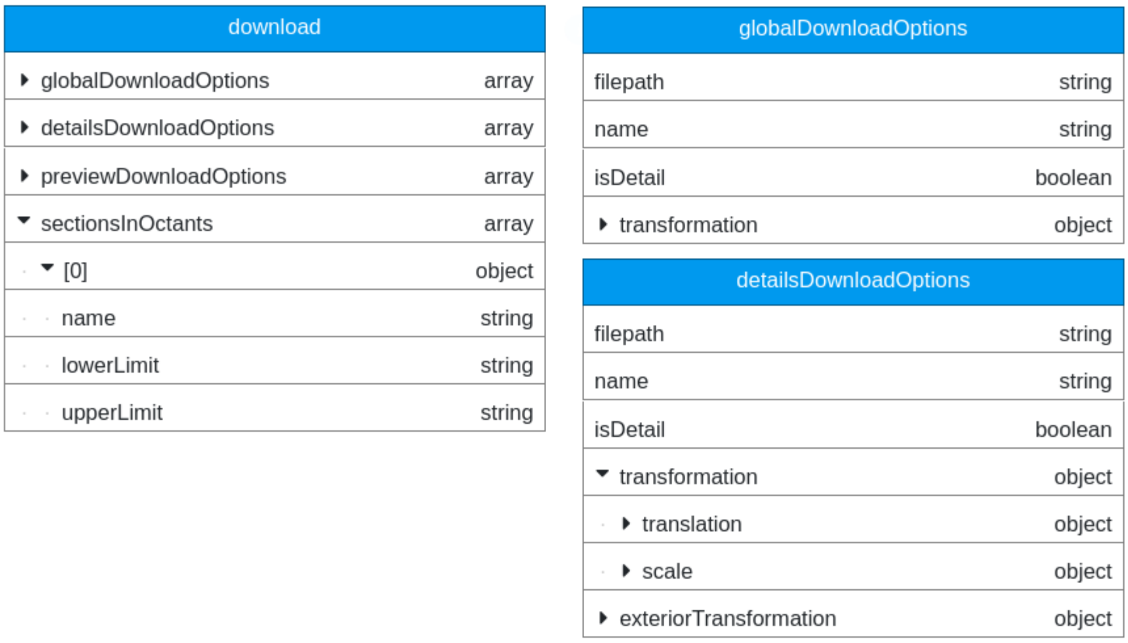
Schema of the JSON Configuration Files

In this appendix, the structure of the configuration files can be found. The location configuration file is not included here because it is a very large file, and a diagram depicting its structure is already present in the report.



cameras	
▼ cameraOptions	object
· fov	number
· aspect	number
· near	number
· far	number
▼ cameras	array
· ▼ [0]	object
· · name	string
· · modelName	string
· · ▶ position	object
· · ▶ lookAt	object

Figure A.1: Schema of the cameras JSON file

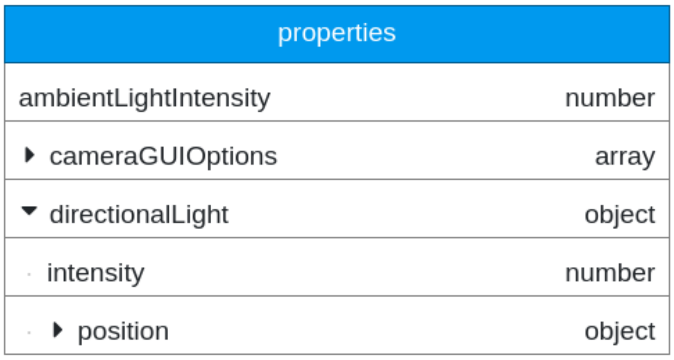


download	
▸ globalDownloadOptions	array
▸ detailsDownloadOptions	array
▸ previewDownloadOptions	array
▾ sectionsInOctants	array
· ▾ [0]	object
· · name	string
· · lowerLimit	string
· · upperLimit	string

globalDownloadOptions	
filepath	string
name	string
isDetail	boolean
▸ transformation	object

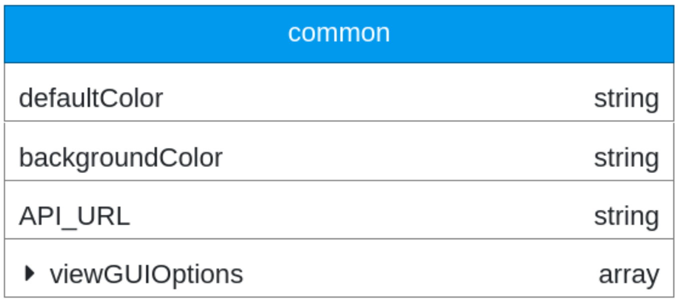
detailsDownloadOptions	
filepath	string
name	string
isDetail	boolean
▾ transformation	object
· ▸ translation	object
· ▸ scale	object
▸ exteriorTransformation	object

Figure A.2: Schema of the download JSON file



properties	
ambientLightIntensity	number
▸ cameraGUIOptions	array
▾ directionalLight	object
· intensity	number
· ▸ position	object

Figure A.3: Schema of the properties JSON file



common	
defaultColor	string
backgroundColor	string
API_URL	string
▸ viewGUIOptions	array

Figure A.4: Schema of the common JSON file

Appendix B

Controlled Experiment Script

This appendix contains the list of the questions asked in the controlled experiment. We start with the 2D system and then move on to the 3D.

Symbology

- NT - Not timed
- 2D - only done on 2D view
- 3D - only done on 3D view

Scenario 1 [2D]

- **Question:** *[Go to the main page]* Consider that you want a 2D view of what tasks are being performed at the PS machine for the date 22/05. How do you proceed?

Scenario 3 [2D]

- **Questions:**
  - *[Go to 22/05]* Could you tell me how many tasks Octant 1 has (indicate what Octant 1 is on the screen)?
  - Please tell me the end date of the “Calibrate sensors” task.
  - **[NT]** Please go back to the date 22/05. Please tell me the end sections (the terminal sections) of the task “Change cables”.
  - Please tell me the sections with a task from Group TE-VSC.
- **Post questions:**
  - On a scale from 1 to 5, how easy was it to retrieve the task information?
  - On a scale from 1 to 5, how easy is it to detect the sections that had tasks?

Scenario 4 [2D]

- **Questions:**
  - *Go to date 22/05.* What is the next task scheduled after the “Paint exteriors” task for the same sections?

Scenario 7 [2D]

- You now need some information about the ongoing tasks of LINAC4. Using the 2D viewer, can you tell me the name and the end date of the task or tasks present in LINAC4 on 22/05?

Scenario 1 [3D]

- **Question:** *[Go to main page]* Consider that you want a 3D view of what tasks are being performed at the PS machine for the date 22/05. How do you proceed?

Scenario 2 [3D]

- **Questions:**
  - **[NT]** Can you describe what you see?
  - By interacting with the model, change its position so that you see it from the top, and center the model, like this (show Figure 1). It should be very close to the image.
  - Now change the view so that you see the model from the bottom.

Scenario 3 [3D]

- **Questions:**
  - *[Go to 22/05]* Could you tell me which octants have tasks on them?
  - How many tasks are present on Octant 6? In which sections?
  - Please tell me the start and end date of the “Change Wires” task.
- **Post questions:**
  - On a scale from 1 to 5, how easy was it to retrieve the task information?
  - On a scale from 1 to 5, how easy is it to detect the sections that had tasks?

Scenario 4 [3D]

- **Questions:**
  - What is the next task scheduled after the “Change cables” task in the same sections? This task is located inside Octant 1.
  - Using the date slider, what was the previous task scheduled for the “Change cables” task on Octant 1 in the same sections?

Scenario 5 [3D]

- **Questions:**
  - **[NT]** Go to date 22/05.
  - How many groups have tasks at this date?
  - Please perform the needed operations to view only the tasks from the SCE-SAM group.
  - How many tasks does the TE-VSC group have?

Scenario 6 [3D]

- **Questions:**
  - You need an image of the ongoing tasks to add to a presentation. The image should present a detailed view of all octants at the same time. How do you proceed?

Scenario 7 [3D]

- You now need some information about the ongoing tasks of LINAC4. Using the 3D viewer, can you tell me the name and the end date of the task or tasks present in LINAC4 on 19/05?
- **Post questions:**
  - On a scale from 1 to 5, 1 being not similar at all and 5 being very similar, how similar was the process to retrieve the task data for the PS and for the LINAC4?

Final Questionnaire

- In your opinion, what were the main differences between the 2D and 3D tools regarding efficiency and usability?
- Do you have any suggestions on possible improvements?

Appendix C

Raw Results of the Controlled Experiment

In this appendix, we can find the raw results of the time duration and number of clicks of the controlled experiment and the result of the SUS questionnaire. The time is represented in minutes. The first ten results are from CERN staff. The results with an odd index are from version A of the controlled experiment, while the results with a pair index are from version B.

Table C.1: Raw results of Scenario 1 and 2 in the 2D and 3D systems

	Scenario 1				Scenario 2			
	2D		3D		Task 2		Task 3	
Subject	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks
1	1.23	7	0.18	7	0.17	4	0.27	4
2	0.33	5	0.35	6	0.3	9	0.13	3
3	1.85	17	0.45	10	1.02	7	1.13	25
4	0.22	7	0.47	5	0.07	2	0.07	1
5	0.55	12	0.28	7	0.17	2	0.05	1
6	0.83	12	1.67	21	0.25	4	0.1	3
7	1.58	23	0.3	10	0.07	2	1.97	30
8	0.5	12	1.28	17	0.11	3	0.11	2
9	1.08	13	0.18	7	0.46	8	0.18	2
10	0.31	5	0.46	7	0.35	5	0.06	1
11	2.02	31	0.92	19	0.07	2	0.72	12
12	0.69	20	0.49	9	0.14	4	0.12	2
13	1.6	18	0.29	8	0.07	2	0.34	7
14	0.47	12	0.36	10	0.83	19	0.09	1
15	2.08	36	0.52	15	1	14	0.17	3
16	0.42	8	1.48	19	0.45	3	0.13	3
17	2.57	28	0.4	14	0.07	2	0.41	11
18	0.9	18	1.47	32	0.59	7	0.07	1
19	1.74	25	0.53	9	0.18	3	0.07	1
20	0.34	10	1.9	18	0.11	5	0.09	7

Table C.2: Raw results of Scenario 3 in the 2D system

	Task 1		Task 2		Task 4		Task 5		Post questions	
Subject	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks	Q1	Q2
1	0.65	0	2.23	45	1.1	15	0.3	4	2	4
2	0.75	3	0.87	22	1.27	22	0.58	7	2	2
3	0.48	3	1.57	22	1.47	18	0.22	1	5	5
4	0.48	15	0.68	28	0.4	9	0.52	10	3	3
5	1.3	39	1	26	1.02	12	0.4	2	3	5
6	0.22	0	1.75	19	2.73	26	0.2	3	3	3
7	0.28	1	2.22	28	0.93	19	0.6	7	4	4
8	1.3	40	1.5	48	0.38	30	0.4	3	2	4
9	0.3	2	1.46	20	0.8	19	0.21	2	4	4
10	1.36	24	1.04	14	0.48	9	0.25	1	4	5
11	0.63	5	1.12	23	1.1	35	1.83	19	3	3
12	0.21	0	1.78	47	0.27	1	0.28	3	2	5
13	1.15	2	1.98	33	2.7	16	0.7	18	1	3
14	0.97	15	0.93	21	0.54	5	0.44	2	2	3
15	1.49	6	3.22	60	3.4	57	0.49	3	5	5
16	0.15	0	1.87	42	0.54	3	0.32	2	2	5
17	0.04	0	1.69	36	2.88	60	0.31	4	3	5
18	0.84	29	1.46	45	0.68	17	0.51	4	3	5
19	4.85	60	3.88	104	1.17	20	0.64	5	4	5
20	0.21	0	1.45	24	0.74	17	0.26	1	3	4

Table C.3: Raw results of Scenario 3 in the 3D system

	Task 1		Task 2		Task 3		Post questions	
Subject	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks	Q1	Q2
1	0.18	0	1.5	13	0.33	1	5	5
2	0.1	0	0.6	7	0.1	3	5	5
3	0.13	0	0.85	11	0.93	18	5	5
4	0.05	0	1	13	0.23	1	5	5
5	0.47	0	1.13	18	0.85	17	2	5
6	2.03	41	1.63	20	1.47	16	5	4
7	0.13	0	1.6	8	0.17	1	3	3
8	0.11	1	1.01	10	0.35	7	4	5
9	0.08	0	1.26	2	0.28	5	3	5
10	0.1	0	1.21	13	0.23	1	4	4
11	0.08	0	0.43	9	0.38	9	5	5
12	0.09	0	0.91	10	0.13	4	4	5
13	0.12	0	1.03	17	0.82	18	5	4
14	0.21	0	1.16	25	2.42	65	4	5
15	0.15	0	1.25	9	0.79	11	5	5
16	0.09	0	0.58	4	3.42	54	3	5
17	0.57	7	0.67	8	0.25	4	5	5
18	0.14	0	0.75	3	2.11	59	3	4
19	0.19	1	0.8	7	0.51	10	5	5
20	0.08	24	0.41	21	0.83	31	4	2

Table C.4: Raw results of Scenario 4 in the 2D and 3D systems

	2D		3D Task 2		3D Task 3	
Subject	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks
1	0.25	25	4.08	40	0.22	2
2	0.67	12	2.1	16	0.13	3
3	1	12	0.98	10	0.63	17
4	0.45	14	1.42	25	0.05	1
5	0.4	8	0.58	11	0.2	3
6	0.92	7	4.48	55	0.57	7
7	1.72	27	0.58	5	0.28	7
8	1.26	28	1.51	21	0.3	2
9	0.58	7	0.3	3	0.25	5
10	0.75	12	0.88	17	0.75	3
11	0.93	14	0.83	14	0.33	5
12	0.82	22	1.18	3	0.4	7
13	1	11	0.83	14	0.55	10
14	0.42	10	2.64	80	0.28	4
15	0.95	17	0.92	4	0.39	7
16	0.97	16	0.89	12	0.27	2
17	2	28	1.56	26	0.43	5
18	1.17	33	3.32	34	1.18	33
19	1.06	13	0.99	21	0.17	4
20	1.04	27	1.43	23	0.1	4

Table C.5: Raw results of Scenario 5 in the 3D system

	Task 2		Task 3		3D Task 4	
Subject	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks
1	0.7	8	0.18	7	0.45	10
2	0.12	1	0.08	7	0.12	3
3	0.1	0	0.08	5	0.15	3
4	0.1	0	0.08	5	0.8	17
5	0.07	0	0.08	5	0.15	3
6	0.15	0	0.22	8	0.38	7
7	0.18	0	0.07	5	0.18	4
8	0.11	0	0.13	5	0.31	6
9	0.05	0	0.08	5	0.08	2
10	2	80	0.66	12	0.06	5
11	0.38	11	0.13	5	0.25	4
12	0.58	26	0.09	5	0.14	6
13	0.69	12	0.12	9	0.22	8
14	1.16	54	0.06	5	0.18	3
15	1.03	6	1.16	20	0.65	11
16	0.17	0	0.15	5	0.29	6
17	0.68	14	0.12	5	1.28	46
18	0.17	0	0.06	5	0.33	10
19	1.53	25	0.14	5	0.4	8
20	0.23	13	0.08	5	0.19	5

Table C.6: Raw results of Scenario 6 and 7 in the 2D and 3D systems

	Scenario 6		Scenario 7				
	3D		2D		3D		
Subject	Time	Nº clicks	Time	Nº clicks	Time	Nº clicks	Q1
1	0.4	10	0.92	30	0.78	17	5
2	0.2	4	0.63	19	0.52	8	5
3	0.6	5	0.93	22	0.97	10	5
4	0.32	8	0.42	24	0.57	11	4
5	0.5	7	0.77	21	0.67	7	5
6	0.33	9	0.68	14	0.75	13	5
7	2.15	43	1.27	34	1.52	24	5
8	1.46	22	0.8	19	1.06	17	5
9	0.43	5	0.65	15	0.83	16	5
10	0.21	3	0.51	18	1.45	24	5
11	0.62	10	0.85	24	1.63	38	4
12	0.16	3	1.03	39	1.44	38	5
13	0.42	17	1.28	27	0.68	20	5
14	0.55	12	0.49	16	0.8	29	5
15	1.1	17	1.24	30	1.08	24	5
16	0.27	4	0.85	23	1.89	20	5
17	0.74	15	2.56	63	1.23	31	5
18	0.82	11	0.82	31	1.35	37	4
19	0.78	8	1.15	27	1.56	31	5
20	0.74	15	0.92	29	0.34	10	4

Table C.7: Responses to the SUS questionnaire

Subject	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	5	1	5	1	5	1	5	1	5	1
2	4	2	5	2	4	2	3	2	3	2
3	5	3	5	3	4	1	4	1	4	2
4	4	1	4	1	1	1	4	2	4	1
5	5	1	4	1	4	1	5	2	5	3
6	4	1	5	3	5	1	3	1	4	1
7	1	1	5	1	4	1	4	1	4	1
8	3	2	4	1	5	1	4	1	5	1
9	3	2	5	2	5	1	5	3	4	5
10	4	1	4	2	5	1	5	1	4	2
11	5	2	4	1	5	1	5	1	4	2
12	5	1	4	1	5	1	4	1	4	1
13	5	1	5	1	5	1	5	1	5	1
14	5	2	5	2	5	1	5	1	5	1
15	5	1	4	2	4	1	5	2	3	2
16	5	3	4	1	5	1	5	2	4	1
17	5	1	5	3	5	1	5	1	4	1
18	5	2	4	2	5	1	5	2	4	1
19	5	1	5	2	5	1	5	1	5	1
20	4	2	5	2	5	3	5	3	5	2