TU WIEN Informatics

AR-Schulungs-Anwendung und -Editor für 3D-BIM-Visualisierung im Bauingenieurwesen

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Media and Human-Centered Computing

eingereicht von

Konstantin Höbart, BSc
Matrikelnummer 01027938

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.-Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistant Prof. Dipl.-Ing. Dr.techn. Christian Schranz, M.Sc.
Projektass. Dipl.-Ing. Dr.techn. Christian Schönauer
Projektass. Dipl.-Ing. Harald Urban

Wien, 15. Oktober 2020

Konstantin Höbart	Hannes Kaufmann

TU WIEN Informatics

AR Training Application and Authoring Tool for 3D BIM Visualization in Civil Engineering

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media and Human-Centered Computing

by

Konstantin Höbart, BSc
Registration Number 01027938

to the Faculty of Informatics

at the TU Wien

Advisors: Univ.-Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistant Prof. Dipl.-Ing. Dr.techn. Christian Schranz, M.Sc.
Projektass. Dipl.-Ing. Dr.techn. Christian Schönauer
Projektass. Dipl.-Ing. Harald Urban

Vienna, 15th October, 2020

Konstantin Höbart	Hannes Kaufmann

Erklärung zur Verfassung der Arbeit

Konstantin Höbart, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 15. Oktober 2020

Konstantin Höbart

Acknowledgements

First, I want to thank my thesis advisors Univ.-Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann, Projektass. Dipl.-Ing. Dr.techn. Christian Schönauer, Assistant Prof. Dipl.-Ing. Dr.techn. Christian Schranz, M.Sc., and Projektass. Dipl.-Ing. Harald Urban for their continuous support and for providing valuable feedback in the process of this work. Furthermore, I would like to thank everyone who took part in the usability studies or helped in the process of finding participants. Special thanks go to my friends & colleagues at 7reasons, my parents, and my partner for the motivation and ongoing support.

Kurzfassung

Diese Diplomarbeit behandelt die Visualisierung von 3D-Gebäudeinformationsdaten (BIM) in Augmented Reality (AR) für die Bauingenieurausbildung. Verwandte Arbeiten, relevante Technologien und Standards wurden analysiert und basierend auf diesen Kenntnissen wurde ein Framework bestehend aus einem Desktop-Editor und einer Handheld-AR-Anwendung entworfen und mit der Unity Game Engine und Google ARCore implementiert. Der Editor ermöglicht es, Dozenten IFC-Dateien zu importieren, Anmerkungen in Form von Bildern, Videos und Text hinzuzufügen und die resultierenden Projekte für die Verwendung innerhalb des AR-Viewers zu exportieren. Die Entwicklung wurde von Mitgliedern des Zentrum Digitaler Bauprozess (E234, Fakultät für Bauingenieurwesen, TU Wien) unterstützt, die während des Entwicklungsprozesses Anforderungen und Feedback lieferten. Die Usability der entstandenen Prototypen wurde mit qualitativen und quantitativen Methoden im Rahmen von moderierten in-person lab sessions evaluiert. Die Ergebnisse wurden interpretiert, und für jede Anwendung wurde eine Reihe von Usability Problemen identifiziert. Verschieden Lösungsansätze für diese Probleme wurden als Mockups und Implementierungsideen formuliert. Abschließend wird ein Ausblick auf die mögliche Zukunft dieses Projekts gegeben.

Abstract

The following master's thesis analyzes the visualisation of 3D Building Information Modeling (BIM) data in Augmented Reality (AR) for civil engineering education. Related work, applicable technologies, and relevant standards were analyzed. Based on this knowledge, a framework consisting of a desktop editor and a handheld AR application was designed and implemented utilizing the Unity game engine alongside Google's ARCore. The desktop editor lets lecturers import IFC files, add annotations in the form of images, videos and text to them, and export the resulting projects for the use inside the AR viewer. This development has been supported by members of the Center of Digital Building Process (E234, Faculty of Civil Engineering, TU Wien), who provided real-world requirements and feedback throughout the process. The usability of the two resulting prototypes was evaluated using qualitative and quantitative methods as part of moderated in-person lab sessions. The findings were interpreted and a set of usability problems was identified for each application. Different solutions to these issues were developed alongside implementation ideas and mockups. Lastly, an outlook on the possible future of this project is given.

Contents

CHAPTER 1

Introduction

1.1 Motivation & Problem Statement

The communication of the three-dimensional composition of buildings and the interaction between components are significant challenges in the education of civil engineers. Virtual and Augmented Reality (VR & AR) can be useful visualization technologies in this problem space to improve understanding [Sam+10] [Din+17]. This thesis examines if the addition of annotations and explanations by lectures into AR 3D Building Information Modeling (BIM) visualizations can improve students' understanding and facilitate knowledge transfer for teachers. The development is supported by members of the Center of Digital Building Process (E234, Faculty of Civil Engineering) at the TU Wien, who will utilize the applications to present BIM data examples during future lectures.

1.2 Aims & Objectives

A visualization & authoring framework consisting of two interconnected applications is developed, which allows importing and viewing structured 3D models in the Industry Foundation Classes (IFC) file format as well as extending them with annotations and explanations. The resulting desktop editor and Android viewing app are evaluated in separate usability studies in which qualitative and standardized quantitative methods are applied.

The editor application is built to be used on a desktop computer by lecturers and features an import for 3D models in the IFC 2x3 file format. The imported model is displayed and one can inspect it with a traditional 3D orbiting camera. The system displays the IFC properties of each building element and allows filtering them. As a centeral feature, lecturers can add comments to building elements and media annotations on the 3D model's surface. Additionally, the editor features a hierarchy view of the loaded

elements and their sub-components for easy selection and navigation. Multiple IFC files can be imported at once and positioned to create a scene of objects. The objects and their components can be assigned to layers, allowing control over object visibility in the client AR application. The user can then export their scene, including all changes and annotations, as a project to an online project repository.

The viewer application features two AR viewing modes which allow students to position building models inside the real world. One downloads a project from the project repository, then the AR system displays it inside the camera view and integrates it with reality. Students can then explore the 3D models, select components, view attached IFC properties, and access the lecturer's annotations. Additionally, a feedback tool allows students to attach a comment to the project, which a lecturer can access inside the project repository.

I implemented both parts of the framework using the real-time Unity game engine, as it provides a lot of the needed functionality out of the box. External libraries will facilitate the import of IFC files at runtime. The editor application's export results in a well structured and portable format that future viewing clients could implement quickly. The viewing application is designed for the use on Android tablets but could support iOS devices in the future with minor modifications.

In collaboration with the Center of Digital Building Process, I evaluated the resulting prototypes through moderated in-person usability tests that feature tasks of example scenarios. These tests collected qualitative data in the form of participant feedback and quantitative data in the form of usability scores from post-study questionnaires.

1.3 Outline

In the following chapter, *Related Work*, I analyze prior publications to provide an overview of the problem space. The chapter *Technology* describes the different building blocks used during the implementation of the applications. Following that, I provide a detailed account of the design and implementation of the framework. The usability test of these implementations is covered in the chapter *Evaluation* and their results are analyzed in the chapter after that. Finally, a summary and an outlook on possible future work and improvements is given in the last chapter.

CHAPTER 2

Related Work

The next sections provide an overview of relevant topics for this thesis. Selected publications with related problem statements are discussed and prior work that delivers background information for the established themes is reviewed. Also, different usability evaluation approaches are explored to introduce relevant concepts and techniques for the later chapters.

2.1 Building Information Modeling (BIM)

The ISO 19650-1:2018 standard defines building information modeling (BIM) as "use of a shared digital representation of a built asset to facilitate design, construction and operation processes to form a reliable basis for decisions" [ISO18b] where an asset is defined as "item, thing or entity that has potential or actual value to an organization" [ISO18b].

Eastman et al. compiled a BIM Handbook [Eas+11] which discusses the BIM approach and its benefits over traditional alternatives that rely on 2D or 3D drawings. BIM technology allows creating digital models of buildings, which contain precise geometric information, data on the construction process, and information on the building's complete lifecycle. Stakeholders in architecture, engineering, construction, and facility management industries (AEC/FM) faced several challenges as designs and requirements evolved over the last decades. BIM can support efforts in all phases of the design & building process, resulting in improved performance and quality. BIM technologies can help these stakeholders assess financial requirements continuously and facilitate collaboration and analysis, as they offer a single source of information. Even after the construction process, BIM systems can assist operation and management procedures by providing real-time monitoring data on top of an up-to-date building plan that includes all updates of the construction process. The authors also stress that BIM should be considered as a process instead of a tool; it has to be integrated across the whole project lifecycle to reach optimal

efficiency. [Eas+11]
So BIM technology aims to replace traditional workflows that rely on static information, e.g., 2D and 3D CAD plans, with a dynamic process and data structure that can interface with players and workflows across industries.

Bryde et al. [BBV13] explored 35 construction projects from 2008 to 2010 that implemented BIM to analyze possible benefits and drawbacks. They formulate a set of criteria from each project: cost & time benefits as well as management & communication improvements. Derived Key Performance Indicators for these areas allow quantitative examination. The authors identified cost reduction as a crucial benefit in 21 of these case studies; only 2 mentioned a negative effect on project costs. Time was the second-highest scoring metric; 17 projects mention this as a positive aspect of implementing BIM, as most of them report faster design phases. Furthermore, 13 case studies highlight communication improvements with the use of BIM systems. Other benefits include quality, coordination, risk reduction, and scope clarification. Negative aspects were also reported, especially on the project's software side, as 20% of the projects report issues with software solutions. [BBV13]
This high rate of problems shows that there is still much space for improvements at BIM's software side, either by offering features to represent real-life workflows more efficiently or by making existing features more accessible.

In a more recent paper, Ghaffarianhoseini et al. [Gha+17] explore current BIM adoption. While many big corporations already implement BIM technology in their projects, most small firms are reluctant to adopt it. A lack of interoperability between different software vendors alongside high upfront costs often outweighs the potential benefits of implementing BIM processes for smaller contractors. Other factors include the lack of experience with BIM projects and skilled personnel. The authors postulate that the separate education of architects, civil engineers, and other AEC players results in inefficiencies and conflicts upon collaboration across sectors via BIM technologies.[Gha+17]
An increased focus on BIM systems during AEC students' education could lower the barrier of entry across sectors, resulting in higher adoption rates of BIM and the included benefits.

2.2 BIM Data in Real-Time Environments

The use of BIM 3D data in a real-time engine has been a focus of recent publications in the AEC sector. Du et al. [Du+18] built an automatically synchronizing pipeline between the BIM software Revit [Aut20b], an intermittent cloud-based BIM server, and a VR client application using Unity3D but relying on the FBX 3D file export from Revit to incorporate the geometric information from the BIM data into Unity3D. The metadata and BIM data is exported and transferred via a custom-built Revit Plugin to the server. The system can handle multiple users interacting and changing the BIM data, either from Revit or inside the VR application. All changes are automatically propagated to active clients, and the displayed information is updated at run-time. [Du+18]

Similar to this system, the developed framework in this thesis handles building information data in the Unity Game Engine, but the new framework won't rely on a vendor-specific plugin to include BIM information. Instead, the system utilizes IFC format capabilities, which do not rely on proprietary software.

An earlier approach by Dalton & Parfitt [DP13] described multiple workflows for visualizing BIM data in a fully immersive CAVE using Unity3D but disregarded the contextual meta information of the BIM data and only displayed the exported 3D meshes. Starting from various BIM & CAD tools like Revit, ArchiCAD [Gra20], Bentley Microstation [Ben20], the authors explored various ways to convert different output files through traditional 3D tools like 3ds Max [Aut20a] or Cinema 4D [MAX20] to end up with an FBX file, including textures, which was finally usable in the Unity Game engine. [DP13]
The struggles of converting and exporting data from different 3D tools can still be a problem today, but the situation has improved since its publication in 2013. Many of the mentioned tools now feature direct export possibilities to open formats but displaying them in a game engine can still be a struggle. Manual cleaning of the exported 3D mesh is needed in many cases to obtain a clean topology and a good compromise between vertex count and quality.

In more recent publications, Zhou et al. [Zho+19] formulated a browser-based WebGL visualization system for BIM data. They developed a pipeline that translates IFC files into JSON files containing the geometric information and compressed them by removing entries of duplicate building objects and referencing a single version of that object instead. [Zho+19]
The described workflow features JSON as a universal and open format for bringing the IFC file's geometric information into the browser. The non-geometric information of the IFC file is not included in the visualization.

Nandavar et al. [Nan+18] implemented a similar system to the one proposed in this thesis, utilizing VR together with IFC files to build walkthrough experiences that allowed adding pieces of IFC data to the original BIM model, thereby featuring a bidirectional data channel between the original model and the VR application. This implementation handled IFC files by translating the non-geometric and geometric information to a structured XML file utilizing the xbim .NET toolkit [LBČ17].

2.3 Augmented Reality (AR)

Azuma defines Augmented Reality as a technology that "supplements reality, rather than completely replacing it" [Azu97] as it is the case with Virtual Reality systems. Additionally, virtual objects are registered in three dimensions and integrated into the real world, and the system allows interactions in real-time. This AR definition is not limited to the sense of sight and allows for AR systems that alter reality via hearing, touch, or smell [Azu+01]. While VR is an immersive experience that completely replaces the real world, AR builds upon reality and adds to it. However, mixtures of these approaches are possible as well, leading to the encompassing term mixed reality (MR) or

XR [Azu19], Milgram et al. defined the reality–virtuality continuum (Figure 2.1), which includes graduations or variations of reality, from a real environment to a completely virtual one [Mil+95].

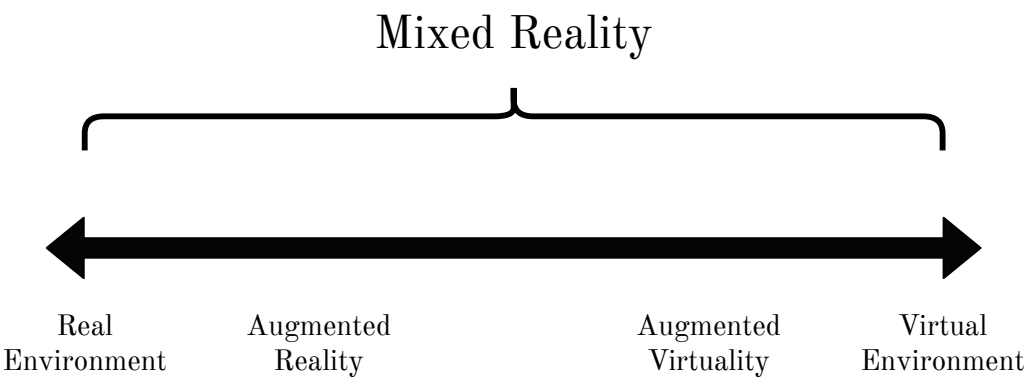


Figure 2.1: The Reality–Virtuality continuum by Milgram et al. [Mil+95]

A wide range of AR systems exists today that feature different approaches to displaying virtual objects inside reality. In a recent paper, Azuma reviewed current AR approaches and provided an outlook on the technology's future [Azu19]: Head-worn or head-mounted displays (HMD / HWD) either employ a see-through video approach displaying an augmented camera feed or an optical see-through approach that allows an unobstructed view and adds to it via projection on a transparent material. According to Azuma, these systems have the highest potential of providing a ubiquitous AR experience for end-users, but there are still some challenges to overcome, like the possibility of visual occlusion, a wide field-of-view and reducing the size of such a system to fit inside traditional eyewear. [Azu19]
Future client implementations of this thesis' framework could feature support for HWD like the Microsoft HoloLens[1]. Due to these systems' limited availability, this thesis' prototype features an AR approach on smartphones or tablets instead.

Smartphones and tablets AR experiences also follow a see-through video approach. The integrated camera stream is displayed and can be augmented either through overlaying visuals or by 3D compositing techniques. While these systems are readily available for end-users, they force the user's attention on the screen, limiting awareness of their surroundings. Additionally, they occupy one or both hands, making interactions difficult and tiring after more prolonged periods. [Azu19]
This thesis' AR system operates on this principle and unfortunately suffers from the same drawbacks.

[1]Microsoft HoloLens: `https://www.microsoft.com/en-us/hololens`

2.4 AR and Civil Engineering Education

AR is regarded as a tool with a lot of possibilities and potential in the field of civil engineering. Building documentation is created digitally in three dimensions and usually transferred to a traditional 2D plan for the use on the construction site. With AR civil engineers can view and explore the original 3D information directly [MTD14]. Many publications cover use cases of AR on construction sites. For example, Dunston and Shin [DS09] described that inspection, coordination, and interpretation, along with communication, are areas in which AR could facilitate and improve traditional workflows. They reflected that a suitable AR system for construction sites should be lightweight, small, reasonably accurate, and easy to operate.[DS09]
Since AR is being explored as a valuable technology on construction sites, its use during civil engineers' education can provide first experiences and help students get familiar with this visualization method.

Dinis et al. [Din+17] developed and evaluated multiple game-based VR & AR applications for the use in civil engineering education with the goal of offering a motivating and engaging experience for young students. Groups of master students developed multiple prototypes, which were then used to present civil engineering topics to K-12 students, who rated the VR & AR interfaces as significant or even essential to understand the presented concepts and as highly motivating.

Similar results can be observed with older students in the publication by Turkan et al. [Tur+17], where they aimed to facilitate teaching structural analysis topics. They formulated a lecturing approach that incorporates mobile AR to visualize typical structural problems. Utilizing a marker-based tracking approach, they developed an iPad app that allowed students to manipulate loads and inspect the real-time results interactively. An evaluation of the application showed that the AR app's learning results were similar to traditional textbook learning approaches. Still, students rated the AR experience as enjoyable and useful.

A journal paper by Ayer et al. [AMA16] reported on a study of an educational mobile AR game. It was evaluated by comparing the learning results of different groups of architecture and civil engineering students. The students were tasked with redesigning a part of a building. One group did so with the AR app, the second one did the assignment on a blank sheet of paper, and the third group did their design on a paper version of the AR app. The app assignment was rated as enjoyable by the participating students, but the other approaches achieved similar scores. The authors recognized that the AR app's limited material palette could constrict the students' design freedom but simultaneously, a broad collection of materials could allow students to explore new ideas and concepts.

These case studies show that AR can be used successfully as a visualization method to supplement traditional civil engineering education approaches. While the use of an AR system does not automatically increase learning results, students find the technology engaging as it allows explorations of civil engineering problems in a novel way.

2.5 Usability Studies of AR Applications

To provide a standardized and systematic way of evaluating handheld augmented reality (HAR) applications, Santos et al. [San+14] developed a Usability Scale specifically for these systems called Handheld Augmented Reality Usability Score (HARUS). The researchers structured their scale similarly to the System Usability Scale (SUS) by Brooke et al. [Bro96], shifting the focus from the factors usability and learnability to measuring the comprehensibility and manipulability of the tested system. Common pitfalls were identified based on a review of HAR usability studies and distilled into a questionnaire containing 16 statements, which are rated on a 7-point Likert scale [San+14].

Polvi et al. [Pol+16] used the HARUS evaluation questionnaire to assess the manipulability and comprehensibility of their AR interaction method *SildAR*. The developed system featured ray cast positioning in Simultaneous Localization And Mapping (SLAM) AR environments. Additionally, users could fine-tune virtual objects' positions in the scene by sliding objects along epipolar lines. They evaluated their solution with respect to an alternative approach where the user had to move the device to re-position the virtual object. The results proved the SlidAR system to be significantly more usable.

This thesis applies the HARUS scale to analyze the proposed AR viewer application, resulting in a score that can be compared to other HAR applications and future versions of the AR viewer.

CHAPTER 3

Technology

The following sections provide detailed information on the technologies, systems, and formats that were used to build the IFC Editor & IFC AR Viewer. Possible alternative technology candidates are also explored as well as reasons as to why one was chosen over the other.

3.1 IFC BIM File Format

The International Alliance for Interoperability (IAI) initially developed the IFC file format. They are a consortium of software vendors in the sectors of AEC/FM consisting of 600 members from over 20 countries [HFV01]. The IAI restructured to form the buildingSMART alliance and still develop new versions of the IFC standard, defined under ISO 16739-1 [ISO18a]. Major corporations and software vendors in the field of AEC/FM, like Graphisoft, Bentley, Nemetschek, and Autodesk, are part of the organization and implement IFC support in their tools [SDT12].

The IFC schema is defined as a data model that stores the identity and semantics, characteristics, and relationships for corresponding objects as well as processes, involved people, and abstract concepts like cost or performance [bui20b]. The schema is defined using the EXPRESS data specification language [ISO04] and in an equivalent XML schema definition. The data itself can be delivered using various formats, but the main ones are the Standard for the Exchange of Product Data (STEP) clear text format [ISO16] or the XML format. The schema contains definitions for 327 data types, 653 entity definitions, and 317 different properties [SDT12]. It combines the geometric information of a 3D model with the non-geometric information.

For example, the IFC Type *IfcSlab* is a building part that can be used to represent a floor, a roof slab of a building, or a similar horizontal building element. It either inherits its material by the assigned *IfcSlabType*, which can encapsulate common materials and

properties of slabs or has a material assigned directly to it, e.g., concrete or insulating material. Most IFC Types include definitions for its base quantities, like their width or volume. An IfcSlab contains quantities on its length, area, volume, and weight. Additionally, a set of properties can be assigned to a slab, defined as the property set *Pset_SlabCommon*, which contains data on its acoustic rating and fire rating, whether its an external building element load-bearing, and other attributes. The specific slab geometry can be defined in various ways, e.g, by its boundary representation or solid sweep [bui].

In Table 3.1 an excerpt of an IfcSlab's geometric definition is presented. The last line (#265) defines the IfcSlab and references a shape (#253) that consists of a Bounding Box (#249) and a Boundary representation (Brep) (#215), which is defined by a closed shell (#213) with six faces, each constructed by a poly loop (#208) made up from four points (#196).

```
#196 = IFCCARTESIANPOINT((4.93139276018,5.30009502262,0.));
#208 = IFCPOLYLOOP((#196,#189,#182,#180));
#210 = IFCFACEOUTERBOUND(#208,.T.);
#211 = IFCFACE((#210));
#213 = IFCCLOSEDSHELL((#178,#187,#194,#201,#206,#211));
#215 = IFCFACETEDBREP(#213);
#225 = IFCSHAPEREPRESENTATION(#165,'Body','Brep',(#215));
...
#247 = IFCCARTESIANPOINT((0.,0.,-0.35));
#249 = IFCBOUNDINGBOX(#247,4.93139276018,5.30009502262,0.35);
#250 = IFCSHAPEREPRESENTATION(#245,'Box','BoundingBox',(#249));
...
#253 = IFCPRODUCTDEFINITIONSHAPE($,$,(#225,#250));
#265 = IFCSLAB('(...)',#12,'Decke-001',$,$,#163,#253,'',.FLOOR.);
```

Table 3.1: Example of the geometry definition via Brep and Bounding Box of an IfcSlab in STEP format.

Currently, the IFC 2x3 specification is the most supported and widely used IFC format. It was created as a Publicly Available Specification (PAS) and has since been upgraded to a full ISO standard with IFC version 4, which is slowly becoming the new standard. However, support for it is still lacking in tools and libraries [Noa+20] [bui20a]. This thesis' framework has the necessary capabilities to be extended towards IFC 4 support, and this inclusion is on the roadmap for future versions.

A recent review of the IFC ecosystem analyzed 31 different IFC tools and their import & export capabilities. Only a small amount of the evaluated tools managed to import real-world IFC models, and even less were capable of exporting them fully [Noa+20]. The interoperability that the IFC file format promises is still in its early phases, but is improving as the IFC version 4 results were significantly better than the scores for the IFC2x3 version. Future IFC versions aim to simplify the data model, allowing a more straightforward implementation of the standard [bui20c].

IFC files are an open industry standard for exchanging BIM data between different software solutions. Still, there are proprietary alternatives like Autodesk Revit files and similar formats by other vendors, which are used more commonly to exchange data in real world applications [Noa+20]. However, these formats don't feature an open data schema and are, therefore, not usable outside of the vendors' software ecosystem.

3.2 Unity Game Engine

The Unity game engine is one of the most widely used game engines with a broad range of capabilities to build 3D or 2D applications for various platforms. While initially developed as a pure game engine, Unity now also aims to support other use cases in the Architecture and Construction industry and Automotive sector or Film & Cinematic production. Unity offers a wide range of ready-made assets and functionalities that can speed up the development process, e.g., camera controls, visual effects, physics, and various 2D or 3D materials as well as automatic Quality Levels to provide support for lower-end devices. Additionally, Unity includes an asset store with free and paid extensions for all parts of the engine. Lastly, there is the option to implement your own features using the C# programming language or via native platform libraries.

Unity features the option to compile an application for many different platforms, including Windows, macOS, Linux, iOS, and Android. Additionally, AR and VR headsets by Oculus and Microsoft are supported alongside several game consoles. Unity achieves this cross-platform approach by compiling the project code assemblies to native libraries. In the past, this was solely handled by the open-source Mono[1] software platform that provides a compatible .NET API for other operating systems than Windows. Unity implemented an inhouse application called IL2CPP (Intermediate Language To C++), which provides an alternative pipeline, converting the project to C++ and then compiling native assemblies from that [Pet15].

As an alternative to the Unity engine, this thesis project could also have been developed using its biggest competitor, the Unreal engine. It features a higher fidelity rendering pipeline, as well as a different approach to providing reusable components as so-called blueprints, a visual scripting system. Still, users can also add custom functionalities via the C++ programming language. Comparing both engines, Dickson et al. [Dic+17] tried to find the most suitable engine to teach in a university course. They concluded that Unreal could provide better or more satisfying results, but Unity was the more stable and better-documented solution. Unreal also does not offer a direct option for an IFC import at runtime, and no suitable plugin or extension was found during the research phase of this thesis, apart from importing a converted variant, e.g., an FBX, of the original IFC file.

Another promising approach would be developing a prototype with a well-established C# open-source BIM framework like xBIM [LBČ17] or similar alternatives. This could

[1] Mono Software Platform: `https://www.mono-project.com/`

provide a better usability experience for the desktop application since Unity lacks the integration of native OS window elements, an application written in the .NET framework could be easily replicate the look and feel of native Windows applications. But this approach would yield completely different solutions for the desktop editor and the mobile AR viewer, thus requiring more coding and duplicated work for each platform.

3.2.1 Unity Packages

Unity added a Package Manager in its recent versions, which enables Unity to handle dependencies in a well-structured manner similar to the dependency management of other programming environments like the NuGet package manager for .NET or NPM, the package manager for JavaScript projects. By default, Unity only displays packages that have been published by the Unity team, but one can add custom package repositories by modifying a manifest file.

In earlier versions, plugins and extensions had to be integrated into a Unity project's folder structure. In most cases, the developer had to handle upgrades of plugins manually. Using the package manager, updates can be applied automatically as packages are forced to adhere to a particular project structure and link to a repository where new versions can be published alongside changelogs and documentation. Additionally, the package manager offers capabilities to resolve conflicts between packages and provides a central user interface to handle internal as well as external dependencies in a unified way.

The IFC Editor & Viewer framework uses this approach to manage external and internal dependencies. The data model of a project export is hosted as a separate public git repository and is imported by both Unity projects via the Unity Package Manager. This approach allows future viewing applications to implement the same data model quickly.

3.2.2 IFC Import

Unfortunately, Unity does not include capabilities to import 3D files at run-time out of the box. Many free and commercial assets and extensions exist that provide importing support for FBX, OBJ, or other file formats. But in the case of IFC files, there are only a few extensions that allow an import at run-time, and all of them require either a license fee per developer or a subscription service. The decision to use the TriLib model loader package was made as it's the cheapest plugin and allows the import of a wide range of other 3D formats, which could prove useful in future extensions of the framework.

Boeykens described an alternative approach in a blogpost [Boe18] where he tried to import IFC files into Unity by converting them into Wavefront OBJ or Collada DAE and an additional IFC XML for the non-geometric information with the *IfcConvert* tool of the IfcOpenShell [Ifc20] project. He then imported the 3D files into the Unity Editor, which offered good results for his test cases but only works during the development of an application and not in a final build since the Unity Editor import functionalities are not included in an export. Unfortunately, our test files caused IfcOpenShell to crash frequently while not producing usable error codes. The latest official release on the

project's GitHub page is from 2017[2], although various unofficial builds include updated functionality. But the post's XML workflow was adopted in this framework as it produced stable results with every IFC test file and could even convert big (> 100 MB) 3D models quickly.

Another solution for the import of 3D IFC data could be the use of the *Open-Asset-Importer-Lib* (Assimp)[3] which can read many 3D file-formats including IFC 2x3 and IFC4. TriLib, the Unity package mentioned above, uses this library under the hood, but an out-of-date version. Also, the original developer of TriLib has indicated that he will not be updating his library in the future as he will focus the development on a new major version (2.0) of TriLib, with no plans to support the IFC file format. To facilitate the use of Assimp in a Unity project, a quite recent open-source library *Assimp for C#/Unity*[4] could be adapted, but this remains untested.

3.2.3 Cross-Platform Development

Unity supports a wide range of platforms, but the most relevant for this thesis are Windows in the case of the editor application and Android for the viewer application. Still, iOS compatibility was kept in mind during the design of the viewer app's structure to allow future developments in that direction. While Unity enables easy switching between target platforms, developers need to take extra steps to ensure their code's portability and their included dependencies.

The TriLib model loader is compatible with Windows, Android, and iOS, allowing for a cross-platform importing API for IFC files' geometric information and possible other file formats. The IfcConvert tool of the IfcOpenShell project is not available for mobile platforms. To address this problem, the non-geometric information of the IFC files is stored as part of the project's JSON file alongside custom properties and attributes like comments and annotations during a project export. The editor application's export consists of a folder named after the project title and containing all relevant IFC files and the JSON file named again after the project's title.

As many mobile devices, especially older ones, do not support native AR capabilities, a fallback solution was implemented using the Vuforia library for Unity. Users can select this secondary viewing mode to get a good AR experience on older devices based on image target tracking.

Unity's canvas UI system was utilized throughout the editor and viewer application to allow for different screen sizes and resolutions. This system provides different scaling modes for UI components, either keeping them at a constant physical size across devices or scaling them based on a reference resolution. Additionally, it features a number of pre-built UI components and manages their state, e.g., checkboxes, input fields, and sliders. To enhance the rendering of text elements the package *Text Mesh Pro* can be

[2]IfcOpenShell Releases: `https://github.com/IfcOpenShell/IfcOpenShell/releases/`
[3]Assimp: `https://github.com/assimp/assimp`
[4]Assimp for C# / Unity: `https://github.com/intelligide/assimp-unity`

integrated which provides more options for rendering text and provides a constant look across different display resolutions and densities.

Another approach to achieve a proper cross-platform solution would be a re-implementation with modern web technologies. WebXR is a new standard that slowly gains traction across mobile operating systems. Unfortunately, Google's Chrome browser is the only major one with partial support of the API[5]. The editor application could run on a WebGL framework like three.js and allow users to create viewable IFC projects without installing additional applications, relying solely on a modern web browser. Using this approach, the IFC file format's support is unclear, but a server-side converting solution could handle IFC files similar to the current framework. Unfortunately, WebXR support is not universally present in mobile browsers and considered experimental, but a WebXR viewer would be an excellent future addition to the framework.

3.3 Handheld Augmented Reality

The development of Handheld Augmented Reality (HAR) started with mobile devices being equipped with powerful processors, precise motion sensors, and capable cameras. Early work in this problem space relied on external cameras and first-generation personal digital assistants (PDAs) but could superimpose and track 3D geometries on the camera image with constant but low frame rates [WS03].

Modern mobile devices bring AR & VR to the masses as most recent phones and tablets include the necessary sensors delivering good quality signals, e.g., an inertial measurement unit (IMU) and an RGB camera [NLZ17]. Some concepts also include a depth component via an infrared camera and projector to further increase tracking reliability on surfaces that lack texture and trackable feature points [TUI17], a technology that is present in modern top-of-the-line iOS devices[6]. Utilizing SLAM concepts, the camera pose is estimated continuously by identifying and tracking feature points in the camera's stream of images and then solving the n-point perspective pose problem by relying on the calibrated camera's specifics [NS07].

Instead of HAR, the IFC viewer application could be deployed on a head-mounted AR device like the Microsoft HoloLens[7] or in various VR setting either head-mounted like the devices of Oculus lineup[8] or in the form of a handheld magic window approach. Exploring different visualization mediums is an area of interest in this framework's future as it could provide an additional way of visualizing BIM data and making it tangible and explorable in a walk-through-like setting, but this is out of scope for this thesis.

[5] WebXR Browser support: `https://caniuse.com/#feat=webxr`
[6] Apple FaceID technology: `https://support.apple.com/en-us/HT208108`
[7] Microsoft HoloLens: `https://www.microsoft.com/en-us/hololens`
[8] Oculus VR headsets: `https://www.oculus.com/`

3.3.1 ARCore & ARKit

Google and Apple both integrated AR frameworks into their respective operating systems. They can detect flat surfaces, position 3D objects into a coordinate system that matches reality, estimate reality's lighting conditions, and apply them to virtual models. Additionally, they feature face detection and tracking algorithms, image recognition procedures and can handle occlusions of tracked objects [Goo20] [App20]. Developers can integrate these libraries into their apps and thereby create an AR experience that enriches common smartphone apps or derive innovative applications. One can easily set up ARCore and ARKit with Unity, which allows for many possibilities to spawn 3D models, scenes, and effects into an AR view.

In a recent comparison of these to frameworks by Nowacki and Woda [NW20] showed that Google's ARCore yielded more accurate planes in a tracking test than Apple's ARKit but also resulted in more false positives while being slower. Apple devices also achieved 60 frames per second consistently while ARCore's performance was heavily device-dependent and was limited to 30 frames per second. It is also important to note that all current iOS devices come with a bundled version of ARKit while ARCore only runs on a few selected and validated devices and has to be installed manually in most cases. But the list of compatible devices is growing at regular intervals as the development team validates them. Unfortunately, only a handful of tablets are supported right now[9]. Limited parts of the ARCore library also feature support for iOS devices, mainly the Cloud Anchor API, which allows users to share generated 3D scenes and models with other users, creating a multi-user collaborative AR experience.

To allow the use of the IFC viewer application on older or unsupported devices, a secondary viewing mode using the Vuforia library was included, which requires an image target printout that serves as a tracking target. Besides Vuforia, a small number of other frameworks facilitate the production of handheld AR apps. Wikitude[10] is an Austrian company that offers a cross-platform AR software development kit (SDK) with similar features as ARCore and ARKit but includes a suite of cloud features and therefore requires a perpetual license to operate and receive SDK updates. EasyAR[11] is another AR SDK that can be used with Android, iOS, and Windows devices. VisionStar Information Technology, the company that develops the SDK, is based in Shanghai and offers a free license for personal use which includes a watermark, but there is an older version, v3.1.0[12], released January 2020 which can be used freely, but will not receive further updates.

3.3.2 Vuforia

The Vuforia platform, developed by PTC Inc., provides a wide range of AR features, including the ability to recognize predefined but custom image targets in the real world

[9] ARCore list of supported devices: https://developers.google.com/ar/discover/supported-devices
[10] Wikitude: `https://www.wikitude.com/`
[11] EasyAR: `https://www.easyar.com/`
[12] EasyAR Basic Version 3.1.0: `https://www.easyar.com/view/downloadHistory.html`

and deriving the camera's pose from this information, enabling the accurate positioning of 3D models in relation to the image target as long as the target is partially visible in the camera picture. Easy integration of the Vuforia platform is possible via libraries for Java, C++, and Unity applications. The use of these integrations requires a license, but a free option covers the limited use of image target recognition in scenarios where the target is defined on build time. In this thesis, Vuforia is used as a fallback option in cases where ARCore is not available on the mobile device or in a situation where the desired model position lacks texture, e.g., on a uniform colored table.

After registering an account at the Vuforia developer portal[13], one can upload an image to prepare it for the use as a target with the library. The portal then calculates feature points for the provided image (Figure 3.1), provides a score for the quality of the target, and provides a downloadable package of the feature set. For this thesis's framework, I chose the IFC2x3 Logo as an image target since it contains hard lines and good contrast, it reached 4 out of 5 stars on the developer platform. In hindsight, this proved to be a sub-optimal target since, besides the feature points at the bottom of the target (text), most points are mirrored across the x and y axes, which resulted in frequent flipping of the x and z axes when positioning the 3D model in the AR view. Thereby, the Vuforia developer portal's rating algorithm does not necessarily reflect the quality or stability of the target.

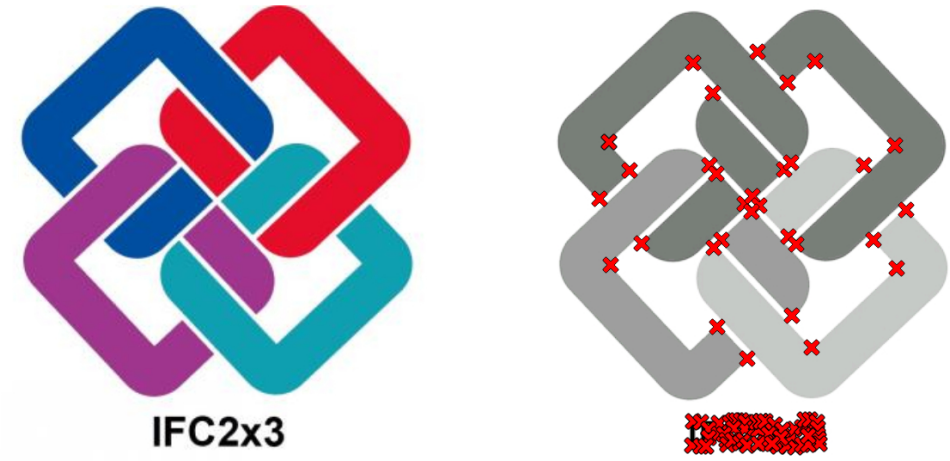


Figure 3.1: The chosen image target (left) the detected feature points by the Vuforia developer portal (right)

The target can then be downloaded as a Unity package and integrated into a project. It is placed inside a scene alongside a Vuforia ARCamera. As a default behavior, Vuforia renders all hierarchical children of a target image upon detection. During runtime, the IFC Viewer application first downloads and parses a selected IFC project, then positions

[13]Vuforia developer portal: `https://developer.vuforia.com/`

it as a child of the IFC Logo image target and disables all renderers before the rendering of the first camera frame. When the image target comes into the cameras' view, Vuforia enables all renderers again.

Compared to the tracking of ARCore, Vuforia yields a very good result when the whole image target is in view but has trouble when one tries to view the building model from lower angles as the system looses tracking frequently.

3.3.3 AR Foundation

In recent versions of the Unity game engine, the development focused on a unified API for VR, AR, and Mixed Reality (MR) technologies. This newly developed system supports Android and iOS phones or tablets as well as headsets like the Microsoft HoloLens resulting in the cross-platform Unity XR plug-in framework. This thesis' implementation uses the included AR package called *AR Foundation*, which provides an interface to various AR implementations, including ARCore and ARKit.

The features of the AR Foundation package are organized into subsystems. The following ones are relevant for this project: The *XRPlaneSubsystem* offers capabilities to track real-world horizontal and vertical planes and relies on the *XRDepthSubsystem*, which generates point clouds from feature points inside the camera's view, they are visualized inside the AR view to provide feedback of the tracking progress to the user. The *XRSessionSubsystem* manages the entering and leaving of the respective XR modes and includes information on the state of the tracking system and, finally, the *XRRaycastSubsystem* is queried when a user tries to position the downloaded IFC scenes via touch gestures and returns ray cast intersections with AR planes.

Google also offers an ARCore SDK for Unity[14], which could be used directly to access Android AR capabilities and supports older versions of Unity, but one would miss out on the cross-platform aspects of AR Foundation.

3.4 JSON File Format

The JavaScript Object Notation formulates a widespread, lightweight, and human-readable data exchange format. It consists of two structures, unordered key-value-pairs that form objects and ordered lists of values or objects [Cro00] [Nur+09]. While initially inspired by the JavaScript language, it is now used across a wide range of programming languages and has become a default exchange format for modern web APIs. Although the format itself lacks namespaces or a schema, this can be alleviated by providing a JSON Schema Documentation [jso19] alongside an application's API. It defines a data type for each field, marking them required or optional, and the ability to add descriptions and other validation constraints.

[14] ARCore SDK: `https://developers.google.com/ar/develop/unity`

An alternative to the JSON export would be an XML-based approach, but for debugging and testing in the early stages of development, JSON was more suitable since mistakes could be spotted more easily. Additionally, there would probably be a possibility to include all of the JSON's project data directly into the IFC schema, resulting in a single IFC export file. This approach was out of scope for this thesis as this would require the implementation of a standards-compliant IFC exporter for the use inside Unity.

CHAPTER 4

Design & Implementation

This chapter focuses on the conception and implementation process of the developed applications. Also, the exchange of data between the two applications is discussed. Design decisions that were made during the development to either support new features or alleviate design and usability problems are covered as well. Lastly, the current prototypes' software limitations are laid out, along with possible solutions for future development.

4.1 Concept & Structure

The development started in October 2019 by discussing the requirements and possible implementations in meetings with Christian Schranz and Harald Urban, both members of the Center of Digital Building Process at the TU Wien. They acted as stakeholders as they will use the final applications in civil engineering courses to visualize building processes. Christian Schönauer was also present during these meetings and advised on the thesis process and technical problems.

The initial requirements were to build a framework of two programs, a desktop editor, and an Android viewer, that should be able to process 3D models in various formats (IFC, FBX, and OBJ). The editor should allow lecturers to enrich a loaded model with comments, annotations, and background information on the building process's various steps. A basic animation tool should allow editor users to hide and reveal certain parts of a building, move building parts between positions and, thereby, visualize the construction process. Building parts should be assignable to user-defined layers, grouping them, and offering collective visibility control over them.

While editing, users should have access to the 3D model's attached BIM information model alongside filtering options. These filters should result in a concise set of essential attributes by hiding or deleting unneeded BIM properties. Bulk operations on certain

types of BIM data should be possible to speed up the editing process. One should be able to combine multiple objects to form a scene; this should be primarily used to show different aspects of a building, e.g., the structural model alongside an electrical layout. Users should then have the ability to export these assembled scenes to an online repository, allowing the viewer application to access them. Additionally, these online projects should be editable again inside the editor to allow for incremental changes.

The viewer application should be able to display the projects from the online repository inside an AR view. Students would use the application on a smartphone or tablet device and download projects from a server. The included annotations and comments should be easily accessible alongside the reduced BIM information on each building element. Animations and layers should allow for an interactive exploration of the 3D scene. Finally, the viewer application should offer students the option to leave feedback; this feature could be extended to allow direct interaction between lecturers and students. Assignments or quizzes could be generated inside the editor application, and students could submit solutions inside the AR viewer.

The resulting applications should be well documented as future projects could build upon the code base or access the online project repository. A viewing application for a VR headset like the HTC Vive would be an example of a possible future client.

During the first tests with various 3D model loaders, the project team decided that IFC files will be the primary import format. The included BIM metadata is key to the civil engineering courses' discussed problems. To convey more information during lectures on building processes, the stakeholders asked to have an annotation system that allows adding photos and videos alongside information texts to various parts of the imported buildings. Furthermore, the layer system was enhanced to allow the definition of groups of layers, which adds broader visibility control. The planned animation system was considered for later implementation. It did not have a high priority for the stakeholders and would be out of scope for the framework's initial prototype. The animation system's primary use would have been to visualize the building process, but it became apparent that this can be achieved with the proposed improved layer system.

These requirements resulted in the design of a system (Figure 4.1) that processes and builds upon IFC files via custom JSON extension files that include all annotations and changes to the original IFC file efficiently while preserving the source data. Different workflows inside the applications were identified and structured as tools. The selection tool allows interacting with building elements and their BIM information in multiple ways. An annotation tool offers capabilities to attach comments and media files to parts of a model. An additional transform tool can change the position, orientation, and scale of imported files. Each tool provides its functionality via one or multiple views inside the user interface, which are revealed to the user when the respective tool is selected.

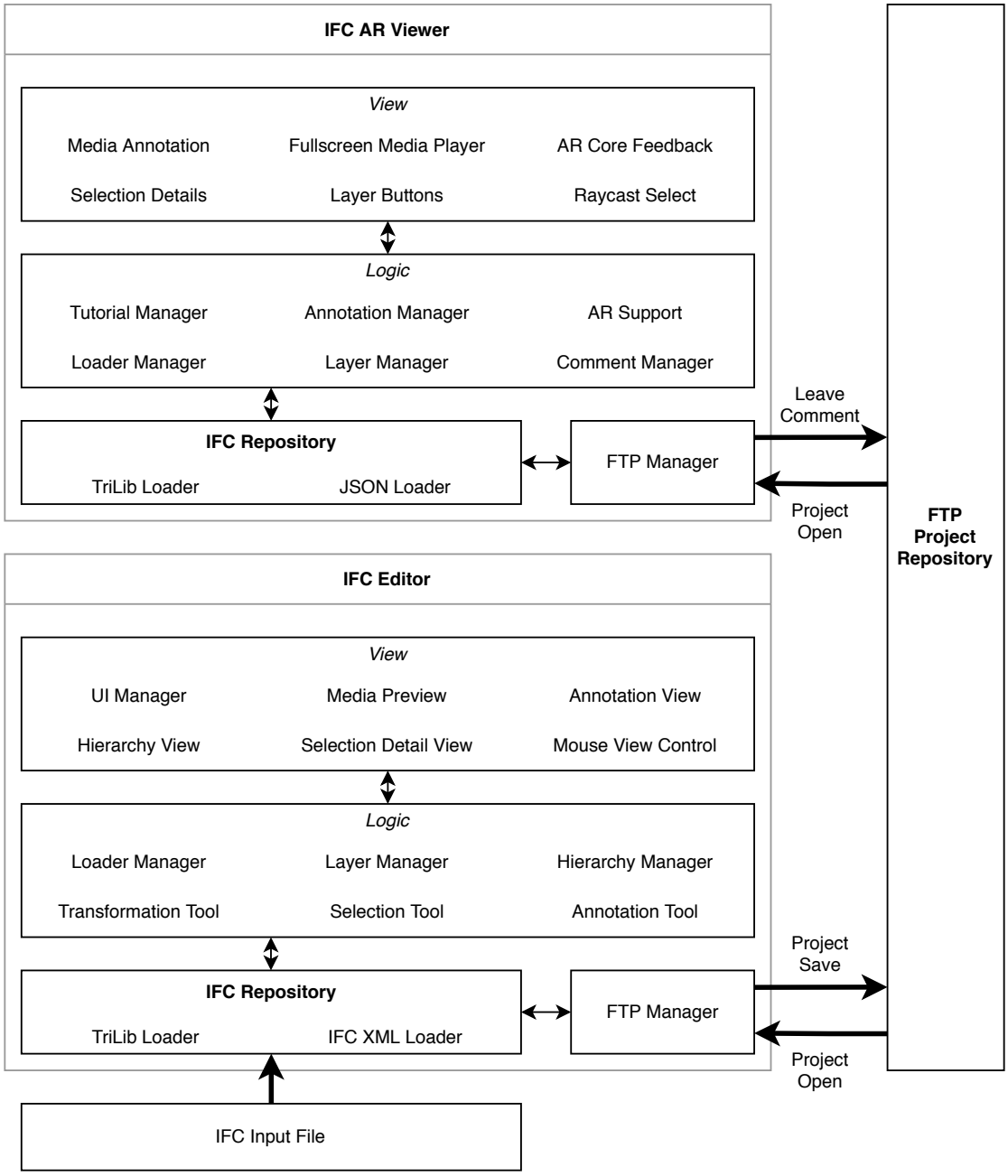


Figure 4.1: The framework's architecture with selected key components

4.2 Development Process

During the first meetings a development process similar to *Evolutionary Prototyping* [Dav92] was proposed. Versions of the editor and viewer application should be designed iteratively while gathering stakeholders' feedback and integrating improvements. The applications and requirements were discussed in regular personal meetings every 2–4 weeks until March 2020, then digital meetings were preferred due to the COVID-19 pandemic. Critical requirements, like the IFC import functionality and tool behaviors, were implemented first, resulting in incomplete but usable and testable prototypes.

The IFC desktop editor's development began by blocking out a user interface in Unity 3D with a layout similar to traditional 3D applications (Figure 4.2). Following the mock-up, the individual features were implemented and tested with 14 different IFC models provided by the stakeholders. To present a uniform UI and to visualize functionality, the open-source *feather icons*[1] library was used whenever icons could convey the functions of UI element. After completing the editor's initial implementation, the AR viewer's development started to allow testing the complete import and export workflow. In this phase, it was also necessary to extract the IFC desktop editor's underlying data model into a separate package to reuse the data type definitions inside the IFC AR Viewer application. Overall the IFC desktop editor repository received 79 commits across ten months, and the IFC AR Viewer repository received 14 commits across six months. The additional IFC types repository was updated seven times during the last two months of development.

This work resulted in the first beta release of the prototype in April 2020, which included most of the requested features and was sent to the stakeholders to evaluate the implementation while creating a detailed test report of all functionalities. Additionally, each issue was marked with a priority to determine its importance. Analyzing the report's remarks resulted in several technical bugs and usability problems alongside new requirements. These mostly concerned convenience functions, like bulk operations or workflow optimizations, but also some unforeseen interactions between tools, e.g., placed annotations, were not affected by the transform tool. In the next iteration cycle, most of these issues were addressed, resulting in a revised version of both applications. Using this version, the stakeholders created another set of test projects to confirm the improvements. These test models were later used during the usability study.

Two quick-start guides were created after the evaluation to give future users an overview of the applications' features (Appendix: Quick-Start Guides).

4.3 IFC Desktop Editor

The IFC desktop editor was created with the game engine Unity 3D and features an IFC import, several tools to manipulate or extend the imported files, and an export to upload

[1] Feather icons library: `https://feathericons.com/`

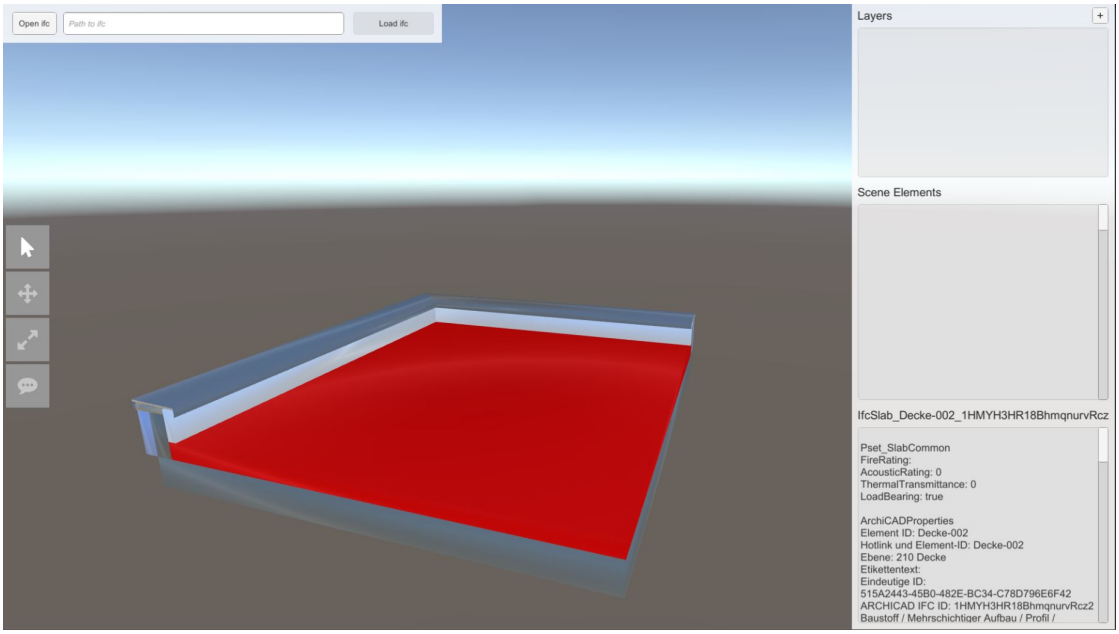


Figure 4.2: The initial concept of the editor UI (current selection highlighted)

the IFC files alongside a JSON file containing additional information and optional media files. Additionally, the editor can open existing projects from the server, allowing users to update them or build upon them to create a new project.

The editor's UI is split into four parts, the top button row, which contains system functionalities for opening and exporting. The scene view contains the loaded geometries and can be manipulated using an orbital camera, right click-drag to rotate, and scroll wheel to zoom. Coordinate axes mark the origin of the scene view. The right side of the window contains a sidebar. Its contents vary based on the selected tool. Finally, the toolbar on the left side of the application allows switching between different interaction tools: Selection, Transformation, and Annotation (Figure 4.3).

4.3.1 IFC Import

The import of an IFC file is split up into two pipelines: importing the geometric information and importing the non-geometric information. (Figure 4.4

The geometric data is read via a commercial Unity package called TriLib, which provides bindings to the popular 3D library Assimp, which enables uniform handling of many 3D file formats as well as Unity specific methods for it. This proved to be the easiest and most robust way of loading the geometric data while also being an extendable approach. The package also supports a wide range of other 3D formats that could be integrated with future versions. Once the import file is read, the TriLib loader spawns a new game object representing the loaded geometry.

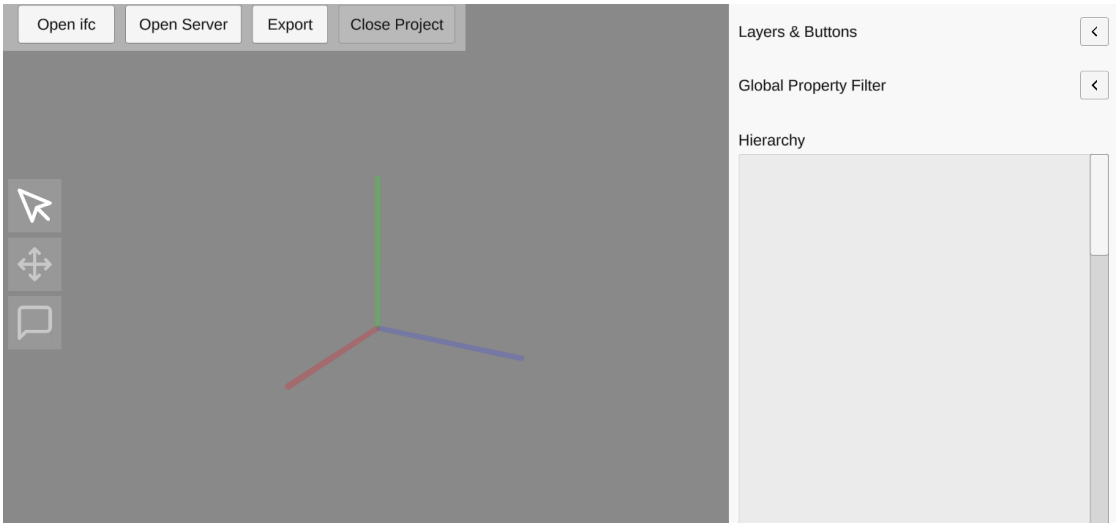


Figure 4.3: The initial view when launching the application. Button row for opening and exporting (top), toolbar (left), sidebar (right)

To ensure a usable environment, the loaded model is re-centered in the coordinate origin by finding the central point of all loaded meshes and raising the model to sit precisely on the x-z-plane, making it easier for the viewer application to position the model (Listing 4.1).

Additionally, all loaded meshes' scales are uniformed, as Unity has problems with rendering very big or tiny objects. This has to be done on a vertex level since Unity's scale transforms fail with very small objects. For this, all meshes inside the IFC file are enumerated, and a common bounding box is calculated. Based on the average

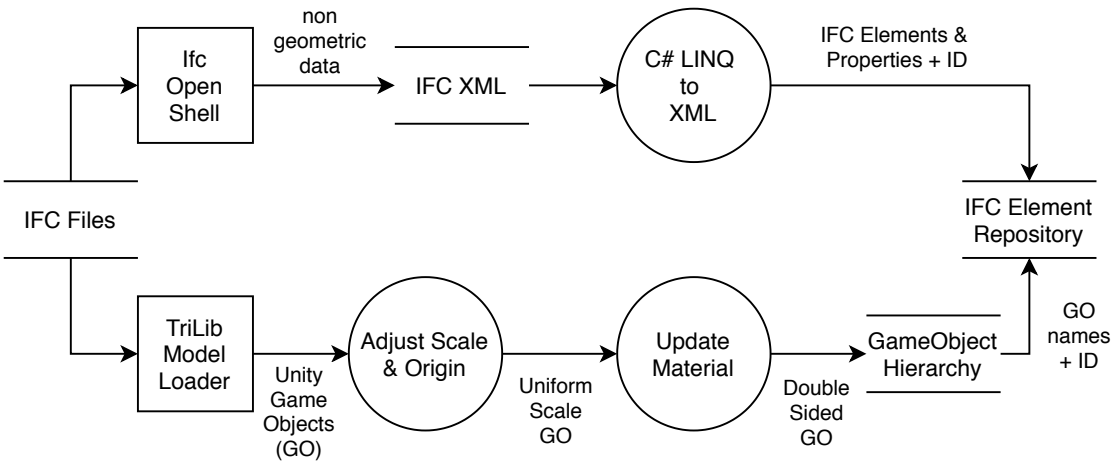


Figure 4.4: Dataflow Diagram of the IFC Import process

```
CenterRoot(Transform root)
{
    Vector3 center = new Vector3();
    if (root.childCount > 0)
    {
        Renderer[] allChildren =
           root.GetComponentsInChildren<Renderer>();
        foreach (Renderer child in allChildren)
            center += child.bounds.center;

        center /= allChildren.Length;
    }
    root.position -= new Vector3(center.x, 0, center.z);
}
```

Listing 4.1: Defining a new origin for the root GameObject.

extent of the bounding box, all vertices are re-positioned in a normalization step. After this operation, the mesh's normals and bounds have to be recalculated by calling the respective Unity functions. Then, the new mesh is assigned to the renderer and the physics collider. In a final step, each mesh's local position is also normalized to preserve the original relative positions (Listing 4.2). Ideally, this would be done inside the TriLib Importer. It does expose a scale parameter in its import function, but unfortunately, it is ignored[2].

Another problem of the TriLib loader is that the generated geometry's normals are not always calculated correctly, and an object might only appear partially. To alleviate this, a two-sided shader variant of the standard unity material shader was applied to all imported objects, which disables back-face culling and draws the polygons of objects regardless of the camera's viewing direction.

To import the non-geometric information, the IFC file is first converted with the IfcOpenShell command-line tool to an XML variant of the original IFC STEP file, which discards the geometric data. This XML is then parsed using a custom-written importer, which relies heavily on the C# Language-Integrated Query (LINQ) to XML capabilities to retrieve the stored information (Listing 4.3). The file contains the scene tree in the decomposition section. Each IFC element is listed in this structure relative to its scene parents, siblings, and children. They contain references to IFC Properties, which are sequentially listed in another section of the XML document and have to be retrieved by matching the reference ID with the property ID.

When an IFC element is queried, it is retrieved by its ID, and the corresponding references are resolved, e.g., as part of the UI selection process. The resulting properties are then

[2]TriLib global scale issue Unity forum thread: `https://forum.unity.com/`

```
ScaleMeshes(Transform root, float scale)
{
    MeshFilter[] allChildren =
       root.GetComponentsInChildren<MeshFilter>();
    foreach (MeshFilter child in allChildren)
    {
        var original = child.sharedMesh.vertices;
        var vertices = new
           Vector3[child.sharedMesh.vertices.Length];
        for (var i = 0; i < vertices.Length; i++)
            vertices[i] = original[i] / scale;

        child.sharedMesh.vertices = vertices;
        child.sharedMesh.RecalculateNormals();
        child.sharedMesh.RecalculateBounds();
        child.sharedMesh.Optimize();
        child.GetComponent<MeshCollider>().sharedMesh =
           child.sharedMesh;
        child.mesh = child.sharedMesh;
    }
    Transform[] children =
       root.GetComponentsInChildren<Transform>();
    foreach(var t in children)
        t.localPosition /= scale;
}
```

Listing 4.2: Scaling the loaded models to a uniform size.

mapped by their name. Their attributes are concatenated inside a string, which allows them to be displayed compactly inside the UI selection section. This mapping is done on the fly by the IfcRepository class when the application's selection is updated. An initial complete conversion upon importing was deemed to slow during testing. A larger test IFC file with more than 30.000 building parts was tested with the full transformation from XML to the internal data format, which took longer than two minutes. This was mainly caused by the single-threaded nature of Unity's MonoBehaviours. Although most of the import can be done in an asynchronous and multi-threaded way, there are still crucial parts where the import needs to integrate with Unity's procedures, e.g., updating the UI. This leads to the approach of only loading the properties of objects when they are selected. The lengthy conversion was deferred to the export procedure, where long waiting times are more acceptable, and internal Unity functionality does not need to be updated. The current set of test models consists of 14 IFC files, ranging from architectural models to 3D HVAC and plumbing diagrams. 12 of these models load without issues, one can

```
/// <summary>
/// Looks for an XML Element by ID in a given XML Document.
/// </summary>
/// <param name="source">The XML Document to filter.</param>
/// <param name="linkId">The ID to look for</param>
/// <returns>A XML Element with ID or null.</returns>
XElement FindXmlElementWithID(XDocument source, string linkId)
{
    XElement found = (from el in source.Descendants()
                      where el.Attribute("id")?.Value == linkId
                      select el).FirstOrDefault();
    return found;
}
```

Listing 4.3: Example of a LINQ approach to find an XML element by ID.

not be opened at all due to an importing error from TriLib and a second one is missing certain building parts. Additionally, this is the biggest test model and has a file size of 113 MB. The initial load takes under 2 minutes on a 4-year-old laptop.

The information of the geometric and non-geometric loaders is then merged by the id of each object. TriLib appends the IFC ID of each object at the end of the game object name after an underscore, and the XML identifies each IFC Element with an ID field always containing a globally unique string of 22 characters.

The licensed TriLib plugin's code was restructured in later stages of the development and hosted in a private repository to allow easy integration in Unity projects. A potential new developer would need access to this repository to develop new versions of the framework. An existing TriLib license could be transferred to them, or a new one purchased. After this, they could request access to the repository from the repository owner. The mentioned repositories are hosted publically on a free gitlab.com account[3].

Other examples of external dependencies include the open-source JSON parsing and formatting library *Json.NET* by Newtonsoft[4]. It is hosted by Unity in a different repository than the default packages as its use is usually only required in more advanced cases since Unity already includes a rudimentary JSON tool-chain. The IFC types repository[5] also adheres to the standards for Unity Packages. It includes a package.json file that specifies attributes like the name, description, author, package version, and information on the package's dependencies. The relevant scripts are located in a *Runtime* folder.

[3]Project Repository Page: `https://gitlab.com/konhoe/master-thesis`
[4]Json.NET - Newtonsoft: `https://www.newtonsoft.com/json`
[5]IFC Types Repository: `https://gitlab.com/konhoe/ifc-editor-types`

4.3.2 Tools

The editor application features a set of tools that one can use to modify an IFC project. Every tool extends the abstract Tool class, which provides template functions for activating and deactivating a tool alongside its assigned UI. Each tool is represented by a button inside the toolbar on the left of the screen. Selecting a different tool updates the screen's right side to display the information needed for that tool's operation.

Selection Tool

The application launches with an activated selection tool UI (Figure 4.5). In this mode, the user can select building objects either by clicking on them in the scene view, invoking a ray-cast selection that can cycle through overlapping building objects, or selecting an element inside the hierarchy view. Either selection method updates the scene view representation of that object by coloring in red and by focusing the hierarchy on it. A new selection also updates the selection information below the hierarchy view, displaying the respective building element's IFC details.

Using the hierarchy selection, one can also select a group of building objects revealing separate IFC details. This is not possible with the scene view selection. Additionally, the hierarchy view features a button on each building element to delete that element. Pressing the F key with an object selected focuses the scene camera on it and makes it the center of the orbital controls, pressing the F key again resets the camera. This functionality can be used to locate the current selection if initiated from the hierarchy view and is not visible in the scene right away.

The ability to cycle through overlapping building parts while selecting inside the scene view from the same or a similar ray cast position is achieved through gathering all intersected objects in a list sorted by their distance from the ray origin. If a previously selected object exists and is inside this intersection list, the element behind the previous selection is the newly selected one. The current implementation stops the selection cycle at the farthest object. To select the nearest object again, the user must either select a different object first or clear the selection by clicking into empty space.

One particular set of building elements is generated without IDs in their names, these are of the IfcMappedItem type, which points to an existing representation of an object. To allow selecting these objects, they are treated in a special case, and their parent object's ID is used for the IFC detail lookup.

The sidebar's selection information also contains buttons that allow adding the selected object to a layer, leaving a comment and bulk deactivating and activating its IFC properties. Only active IFC properties are included in the project export so that this functionality can hide unneeded information. To simplify this hiding of information further, a global property filter was added in later versions of the editor. It displays a list of all property types grouped by their type name and allows hiding these properties

for all building elements, causing them to be hidden for the export and in the editor application.

A common SelectTool was created to provide setter and getter methods for the current selection to keep the different selection methods working next to each other. It also keeps a list of all active selection methods that implement the ISelector interface and invokes their update method in case of a selection change. This then triggers a redraw of the relevant UI & scene elements to indicate the new selection. The update method also receives a Boolean value indicating whether a single object was selected or the parent of a group of objects.

The controls for the layer system are also located inside the selection UI. A user can create layers and groups of layers that can control the visibility state of building objects. One layer can be included in many groups allowing the user to define new groups without having to reassign all of the layers. Additionally, a building element can be assigned to many layers. The order of layer groups can also be changed as they usually represent a sequence of building processes and are visualized as a sequence of buttons inside the AR viewer. The user can toggle the visibility of layer group elements on and off to validate that their layer assignments are working as expected.

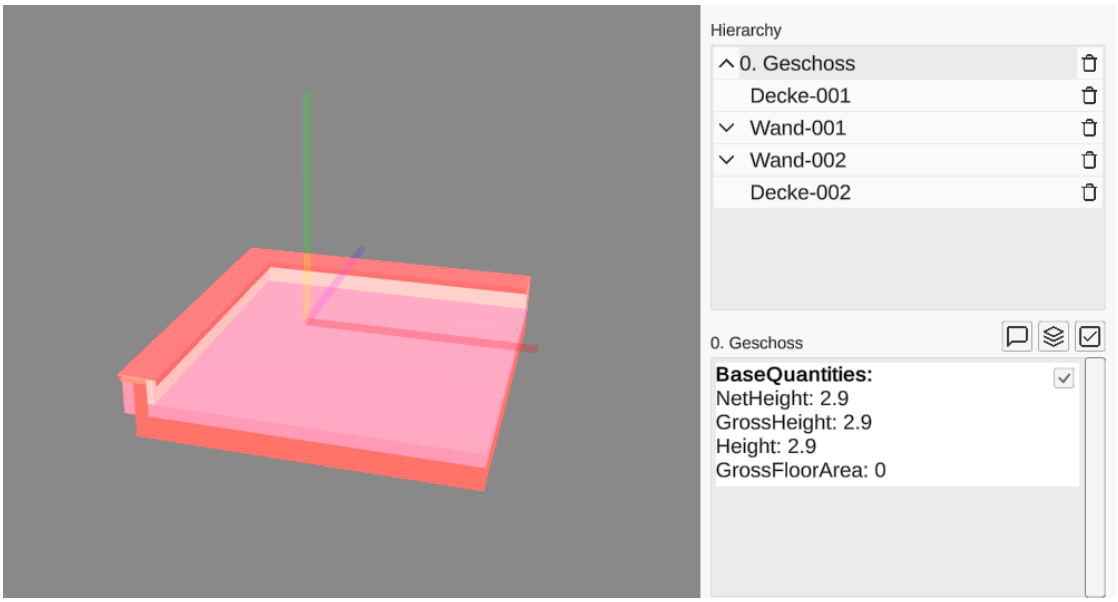


Figure 4.5: The Selection Tool Interface. Sidebar containing hierarchy view and selection details

Transform Tool

The transform tool can change the position, rotation, and scale of the imported IFC files along the three axes (Figure 4.6). This can be necessary when multiple IFC models represent different building information, e.g. one file for the building elements and another

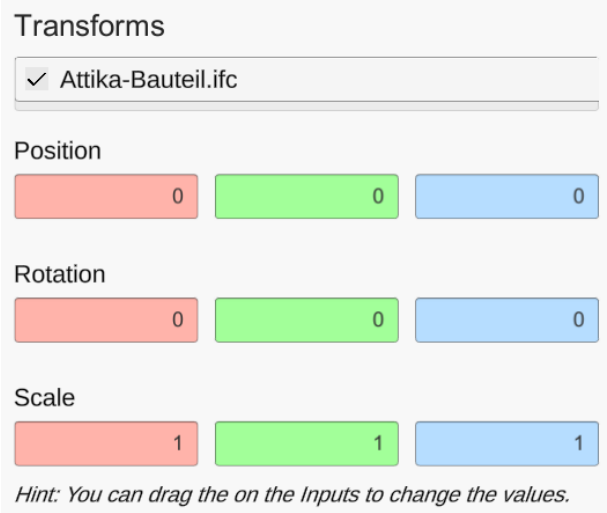


Figure 4.6: The sidebar containing the transform tool's settings.

one containing just the water pipes. The input fields are colored like the coordinate system inside the scene view to make their use more intuitive.

Annotation Tool

Finally, the annotation tool allows the user to add media or text annotations to the imported models. An annotation can contain a title, a description, and an image or video file. They also can be assigned to layers, and different visualizations can be chosen, either a flag or a ball. These functionalities are accessible via the sidebar and updated based on the selected annotation (Figure 4.7). Additionally, the user can change an annotation's position after the initial placement via the input fields and change its size by manipulating the slider.

These annotations can be used by lecturers to add background information to certain building parts or the model as a whole, e.g., a video showing the construction process or an image of the floorplan.

4.3.3 Export

After finishing all edits, a user can export the IFC project to the FTP server where it is available for clients. This is done by packaging all imported IFC files into a folder and creating a JSON export file. This file contains a top-level object containing a global comment field, currently unused, and several lists for IFC elements, transform, annotations, layers, layer groups, and filtered properties (Figure 4.8).

The IFC elements array has entries for each component with IFC properties inside the project. Each entry stores its referenced element ID, name, layer assignments, comments, properties with an additional active flag and a deleted flag. The transform array contains

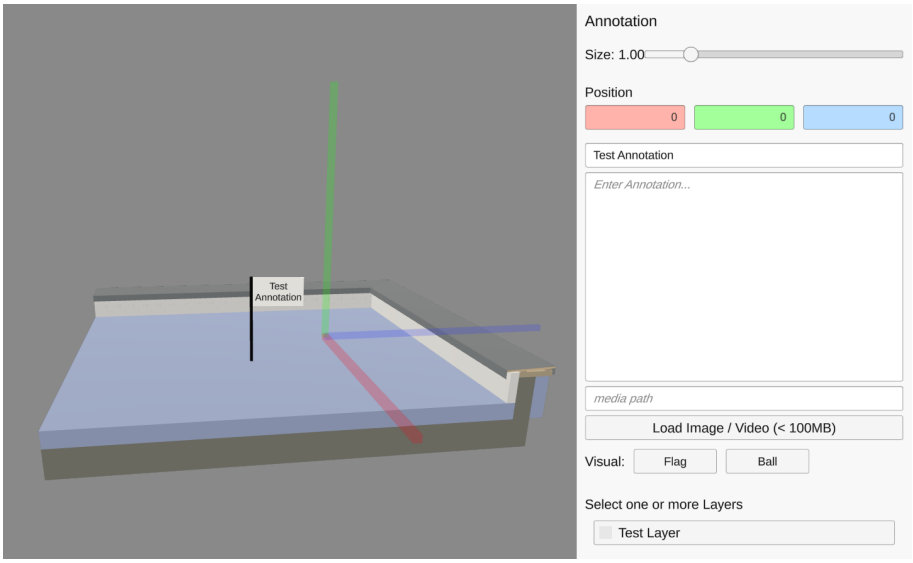


Figure 4.7: Annotation tool UI

an object for each loaded IFC file identified by its file name and stores the assigned transformations.

The annotation list contains objects with the respective annotation fields like its title, description, visual representation, a path to an optional media file, and position & scale.

The layers are stored by their name and an additional array field containing all assigned layer groups' names.

The export folder is then uploaded to an FTP server. For this use case, a credential management system was built that stores the last used credentials across application restarts. Additionally, the user can change the root directory and specify a name for the exported project. A new folder with that name will be created, containing all project files, with the JSON file again named after the project name.

4.3.4 Limitations

The editor application currently exhibits long load times for IFC files with many objects in a flat hierarchy as the hierarchy tree has to be generated for many elements. This could be alleviated by designing a queue like spawning system that distributes the spawning of hierarchy elements across multiple frames if necessary.

The UI of the editor application was built for a resolution of 1920 pixels by 1080 pixels. Although its window is resizeable, there are some edge cases in which some UI elements get occluded. This could be fixed by testing the application across other resolutions.

The importer currently assumes that an IFC file and its components are unique. Thereby an IFC model can not be imported multiple times. This would crash the application in

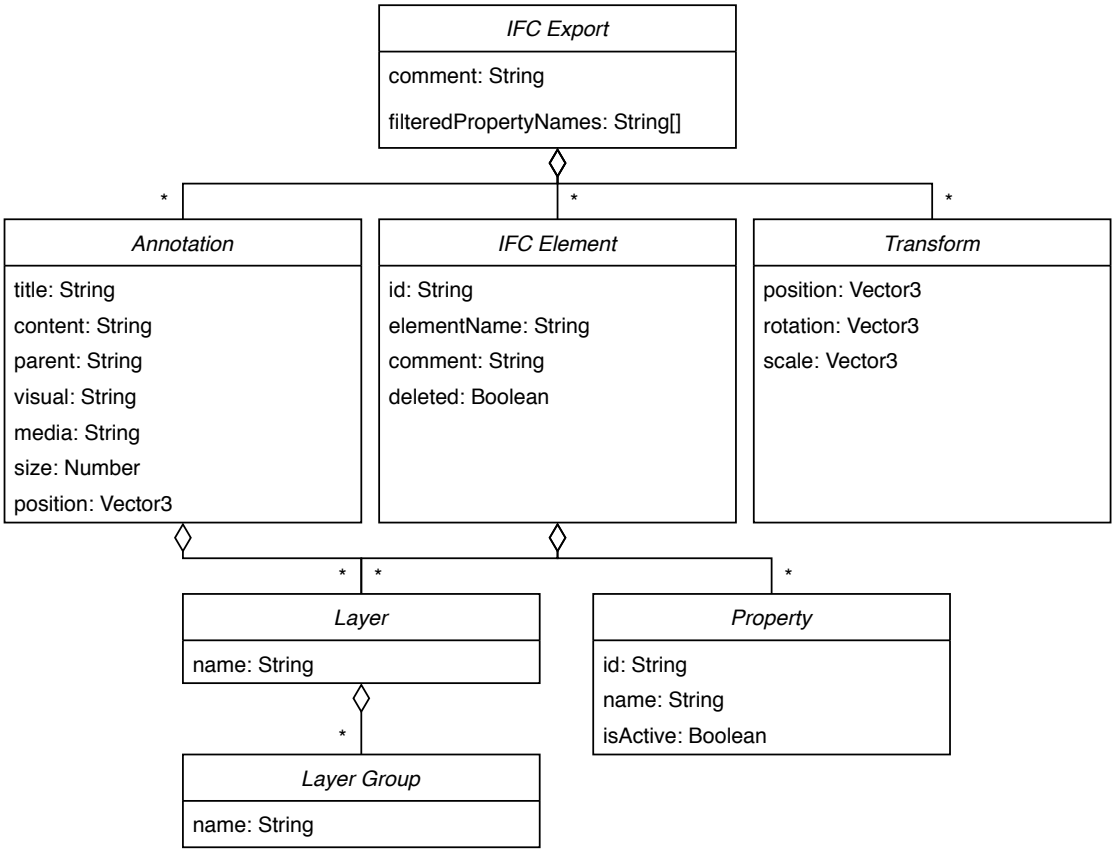


Figure 4.8: A Class Diagram of the IFC Export JSON file.

earlier versions, but this error is now captured and displayed in a dialog window. In future versions, IFC elements could be identified internally by a different generated ID, which would have to match the results of both the geometric and non-geometric importer. The matching of information is currently done via the game object name as TriLib includes the ID of a building element in its game object name, and the non-geometric loader fetches the same ID from the XML document.

Some IFC files can not be loaded at all since the TriLib loader fails during the geometry generation phase. Unfortunately, no error codes are returned, which made debugging the problem impossible in the scope of this thesis. Additionally, some test models passed the generation phase successfully but were missing building parts, again without error information. This could only be fixed by replacing the TriLib loader with a different mesh loading pipeline, which was experimented with at the beginning of the project, but no suitable replacements were found at the time. Since then, a new open-source repository[6] was discovered that integrates a more recent version of Assimp into Unity, which could yield better results.

[6]Assimp for C# / Unity: `https://github.com/intelligide/assimp-unity`

The current importer setup does not include textures for the loaded meshes as an application material catalog usually provides them. Although the IFC format supports the inclusion of texture files, this is mostly unsupported by software vendors[7]. This could be fixed by creating a custom catalog for common materials like concrete, wood, and other surfaces.

The FTP upload dialog does not feature traditional FTP browser actions like switching directories interactively. Furthermore, the credentials to the FTP server are currently saved as a plain text JSON file. Also, the file structure is in no way insured as other users could access the projects via a standard FTP browser and change their file layout. To alleviate this, a separate project management application could be built that provides access to an API with Authorization and Authentication standards like JSON Web Tokens (JWT) to manage access rights better. Additionally, the current project structure does not include a version field. The user has to manually delete and download a project again to access the latest changes.

Usability issues and limitations are discussed in the Evaluation chapter, as they were mostly uncovered during user testing and are disregarded in this section.

4.4 IFC AR App

The IFC AR Application was primarily created for Android tablet devices using the Unity game engine. The reference device which was used during testing is a Samsung Galaxy Tab S4 10.5, as it was one of the few tablets on the market that supports Google's ARCore functionalities. Other devices with different aspect ratios and resolutions should be capable of running the application, too, as the app was frequently tested on a Samsung Galaxy S8+ smartphone during the development process. Since the ARCore library is a rare feature for modern tablets, a fallback mode based on image target recognition was implemented using the Vuforia SDK. The app accesses IFC projects saved on an FTP server by the editor application. After downloading a project, the user is prompted through a set of steps to position the model in the AR view. When this is done, one can select different building parts, explore the IFC scene, and view attached IFC data plus video and image annotations.

4.4.1 Opening Projects

When users launch the application, they are prompted to choose one of two viewing modes (Figure 4.9). The ARCore mode features natural feature detecting and tracking algorithms which find stable feature points and calculate planes that align with real-world surfaces. The building models can then be spawned on these planes. Alternatively, the Vuforia mode features an image tracking system where models are positioned on a tracked real-world image print out.

[7] Relevant post on `https://forums.buildingsmart.org/(...)`

After this initial selection, the system displays an FTP browser where a user can connect to different servers or download a project. The user can change the default FTP credentials, which is persisted across application restarts. After the user selects a project, the system pulls it from the server and stores it locally on the device, making it available for future offline use. An existing version with the same project name may be overwritten. The project can then be either deleted or viewed inside the AR view.

A selected project is loaded by parsing the included project JSON and launching the TriLib geometry generator with the referenced IFC files and their respective transform offsets. After this phase, a scaling operation is necessary to convert the IFC mesh scale into a usable range. As with the editor import, this has to be done on a vertex level since the TriLib loader lacks internal scaling support.

Additionally, the application generates the annotation visuals and fills them with their respective contents. The resulting game objects are then deactivated until the user finishes the AR view's initialization steps.

Figure 4.9: The start screen of the IFC AR viewer app.

4.4.2 AR View & Interaction

If users select ARCore as viewing mode, they are prompted by an on screen tutorial to point the tablet's camera towards the area where the 3D model should be positioned. Viewing the area from different angles causes the ARCore algorithms to detect feature points which are rendered inside the AR view to give users some indication of the tracking process. When the system recognizes a reasonable amount of points on a flat surface, a plane is detected and rendered as an orange polyline surface with black outlines. This

pauses the tracking process, and the system prompts users to place the loaded model on the plane by dragging one finger across the surface or by rotating the model with two fingers. When users are satisfied with the model's position and orientation, they can lock it with a button press.

Now the instructions, plane indicator, and feature points disappear from the screen, and the opened IFC model remains to be explored. On the left side of the screen, several buttons appear that represent the layer groups created inside the editor application (Figure 4.10).

As soon as the user touches a building part, it is selected and highlighted with a red emission shader. This also causes the IFC detail panel to appear on the right side of the screen, filled with the building part's attributes and properties. The sidebar also includes buttons to increase and decrease the IFC detail's font size.

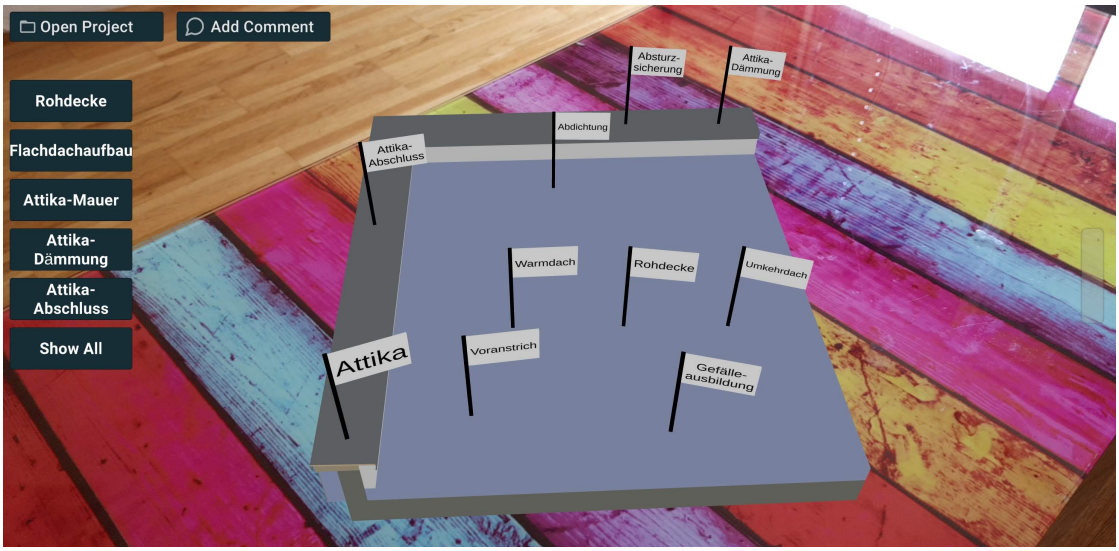


Figure 4.10: The initial view of the "Attika" test project inside the IFC AR viewer.

As an annotation is selected, the detail panel fills with the title and description of that annotation. The system displays the attached media above the annotation in the AR view (Figure 4.11). The user can select the media and view it full-screen, in case of a video player controls appear and the video starts playing.

A user also has the option to leave a comment for the whole project by opening the comment dialog, entering a name and comment content. Upon saving, the system transfers this comment to the FTP server as a new JSON file with the current timestamp plus the user's name as the filename. These files are currently not viewable inside the IFC editor but can be accessed directly on the FTP server. As a project is removed from the server, all submitted comments are lost since their parent folder is deleted.

Users can select the secondary viewing mode in the app's start screen by clicking the

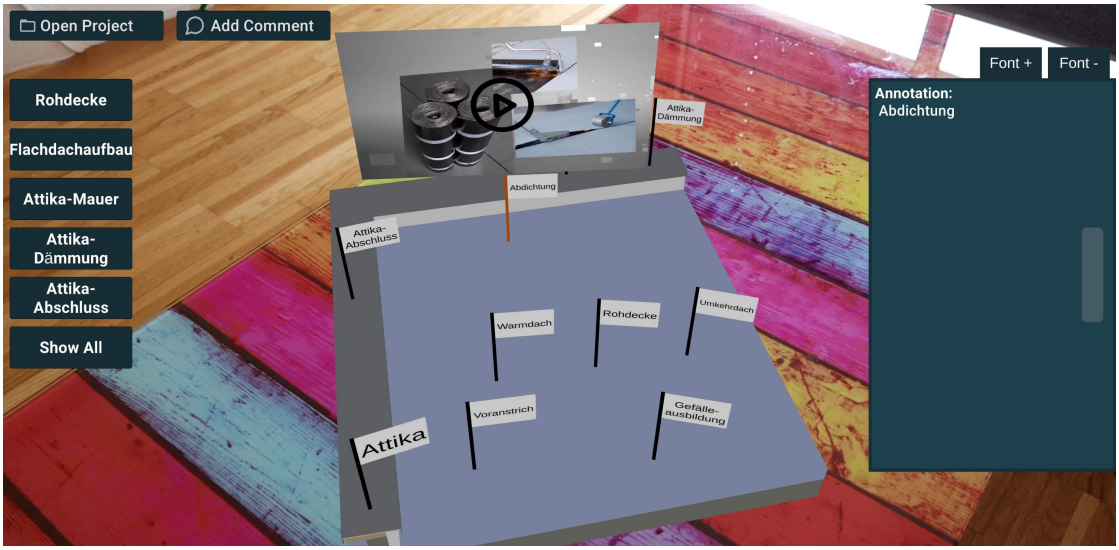


Figure 4.11: Open annotation in the IFC AR viewer.

Vuforia button. After selecting and downloading a project, the user is directed to the AR view and has to point the device's camera at the image target of the IFC2x3 Logo. Differing from the ARCore version, there are no positioning steps required by the user. Other than that, the user can explore the model in the same way as in the primary viewing mode. As an added benefit, the user can manipulate the target position, e.g., by rotating the printout, which causes the AR view to update accordingly.

4.4.3 Limitations

Online and downloaded projects currently do not have versioning information attached. Therefore users have to delete and re-download a project in case of a newly published version.

There is no progress indicator for the download process, which can confuse users when downloading bigger projects. The system could show a progress bar at the bottom of the FTP browser or include a percent counter next to the project name to address this issue.

The ARCore viewing mode does only allow positioning the model at the start of the AR view scene. To change the locked position, the user must close and reopen the project while again going through the initial surface scanning and plane detection steps. A separate button that unlocks the model's position could help in situations where the users realize that the initial positioning was not ideal.

Although the app was tested on multiple tablets and phones, there are still some resolutions where parts of the UI are hidden behind other elements or extend too far off-screen. Fixing this requires redesigning the mobile UI and testing it at many resolutions either on real devices or inside the Unity Editor. Unfortunately, the Unity Editor allows setting

a resolution but does not give an option to change the screen's physical size, the DPI, or the screen's shape. Some modern smartphones include a camera notch that extends inside Unity's render area. This can occlude buttons and text, Unity has a built-in option to circumvent this problem by reducing the render area by displaying black bars across the notched area, but this has not been tested.

CHAPTER 5

Evaluation

The developed framework's usability was evaluated by conducting moderated in-person usability tests, incorporating each application's main features, and collecting qualitative and quantitative feedback. The qualitative feedback was the main focus of this study.

5.1 Methodology

The study was formulated by mixing quantitative and qualitative questions inside two questionnaires, one for the IFC Editor (Appendix 7.2) and one for the AR Viewer (Appendix 7.1). The tests were conducted as moderated in-person lab sessions with a mix of experienced lecturers of the Center of Digital Building Process and students of informatics, 3D modeling, and civil engineering with some to no prior knowledge on the relevant topics of AR and BIM.

Together with stakeholders, a set of tasks that represent the typical use of the IFC Editor and the IFC AR Viewer were formulated. These were put into a questionnaire alongside a five-point scale where participants could rate the difficulty of the task from easy to hard.

Direct instructions were avoided in the task descriptions in favor of general tasks statements, e.g., "Create a new annotation with the title 'Test'" instead of "Change into the annotation mode by clicking the respective button and create a new annotation by clicking on a spot in the model and enter the title 'Test' into the title input.".

While this made the tasks more difficult for users, it caused them to explore the application more naturally, resulting in more and better feedback.

In some cases, the participants got stuck on a task either by not finding the right method to complete it or focusing too hard on the wrong part of the UI. In such a case, the author prompted the participant to close all open sub-windows and re-read the task statement carefully, which led them to solve the task in most situations successfully.

It was assumed that testing the AR Viewer first and then IFC Editor would yield more practical results since the typical user, e.g., a civil engineering student, won't have a lot of additional information on the project, but users of the editor app, e.g., lecturers, have a lot of prior knowledge and are more familiar with the underlying process.

Each questionnaire was pilot tested, which revealed some issues with the test setup, the questions themselves, and minor bugs that were fixed before conducting the study.

At the beginning of the session, the participants got a short verbal introduction:
"This study is done as part of a diploma thesis on an AR Training Application and Authoring Tool for 3D BIM Visualization at the faculty of informatics of the TU Wien. Today, you will be testing two applications, one AR app on a tablet and a traditional desktop application. These applications work by creating a scene of IFC 3D files inside the editor, which are then viewable on the tablet. During this test, you will fill out the provided questionnaire. The results are used and analyzed inside the diploma thesis. You will remain anonymous and are not named. To start things off, please read through the consent form on the first page of the questionnaire and feel free to ask questions at any time."

After the first set of questions on the demography and prior knowledge on IFC, BIM, and AR were finished, the user was handed the tablet to start the IFC AR Viewer test.

The questionnaire was either printed out or answered on a separate laptop. In the case of printed questionnaires, the testers struggled with handling the tablet device and the questionnaire at the same time. To alleviate this problem, the tasks were read aloud so that the testers could focus on solving them.

After solving all tasks, the participants filled out the quantitative survey on the application's usability. Then they gave feedback in an open conversation alongside listing pros and cons of the application and offering opinions on possible future improvements. The author added these feedback points to the notes for each session.

Additional notes were taken on the participants' actions as they tried to solve a problem, struggled with a task, or commented on a functionality.

With the resulting qualitative data, several usability problems were gathered and weighted based on the number of participants who encountered them. Possible solutions for each usability problem were formulated alongside advantages and disadvantages.

SUS & HARUS

The SUS is a widely used usability indicator. It features ten statements on different usability aspects of a system. Participants have to mark their agreement with a statement on a five-point Likert scale ranging from Strongly Agree to Strongly Disagree; positively and negatively worded statements are alternated in the survey to prevent response biases [Bro96]. A single usability score can be calculated from the results, ranging from 0 to 100. A score above 68 [LS18] or 70 [BKM08] corresponds to an average or satisfactory usability

experience. Empirical research in 2009 by Lewis and Sauro [LS09a] of 19 SUS tests lead to the result that the SUS scale could be split into two factors, usability and learnability. This was later refuted by the same researchers in 2017 as they analyzed over 9000 surveys and recommended that the SUS should be used as a single dimension score [LS17]. The original scale from 1996 was only adjusted slightly throughout history to replace certain out-of-date wordings [BKM08]. It is recommended that a SUS survey includes 12–14 participants, as the results typically asymptote quickly with 12 or more participants [TS04], although sample sizes ranging from five to nine are also not uncommon as a quick evaluation of prototypes [PPP13].

The HARUS questionnaire, developed by Santos et al. [San+14] is a relatively new usability score. It focuses on perceptual and ergonomic problems common in AR applications but not evaluated in more traditional usability questionnaires. It features 16 questions that are again alternatingly formulated positively and negatively. Alongside the usability, the HARUS scale assesses the manipulability and comprehensibility of an application. The questions can be split into two subsections, which each target one of these aspects. The HARUS scale tries to mimic the SUS scale in its design. However, instead of a five-point Likert scale, a seven-point one is applied—the authors of HARUS reason that their participants can distinguish more subtle differences due to their experience. Again relying on the established SUS questionnaire, the authors propose a similar interpretation of the HARUS results with a score over 70 being acceptable [San+14].

The SUS and HARUS tests' quantitative results were used to solidify the usability issues sourced from the qualitative data and provide a comparable score of both applications' usability. Future versions of the framework could be analyzed similarly, allowing for a direct comparison to this study's results. Additionally, noteworthy scores of singular questions were highlighted and analyzed with the context provided by qualitative data.

5.2 User Study

The user study took place on three dates at two separate locations. The first three tests were done at the author's office in a closed-off meeting room and the remaining four tests were done at the Center of Digital Building Process also inside a meeting room. Several days before the study, a pilot test was conducted at the author's home to reveal possible issues of the questionnaire, the study setup, or the applications.

As we conducted these sessions during the COVID-19 pandemic, devices and workspaces were disinfected after each session. All required safety measures were conducted, a minimum distance of one meter was kept between the participant and the author while sitting or standing more than two meters apart during most of the questionnaire.

5.2.1 Participants

The study included seven participants and one pilot study participant. Although this small number of participants is sufficient for uncovering most usability problems through observation [Vir92], a higher sample size would be preferable for the quantitative parts of the questionnaire. Of these seven testers, two were students of informatics, one was a 3D artist, one was a civil engineering student, and three were former civil engineering students and now work in the industry or were lecturers at the Center of Digital Building Process. Three of these participants were coworkers of the author, whereas four had not met the author before the usability test.

The pilot study tester was a female informatics student aged 25–29 and had some experience with AR applications and no experience handling BIM data.

The rest of the participants were between 18 and 34 years old, one woman and six men, all with some experience with AR applications and three with little to no prior knowledge on building information modeling.

5.2.2 IFC AR Viewer App Usability Test

The AR viewer test was split into three sections, a consent form, a part on demographics, and prior knowledge on BIM data alongside a question on the tester's experience with AR applications. In the next part of the survey, nine tasks (Table 5.2 on page 50) were to be completed, and participants rated the task difficulty on a five point scale from easy to hard. Additionally, we recorded the start and end times. After this, the HARUS questionnaire (Table 5.1 on page 49) followed with 16 questions on the usability, comprehensibility, and manipulability of the application. Finally, participants could express overall feedback on things they liked and didn't like or suggest new functionality.

The questionnaire was pilot tested, which resulted in small adjustments to the test setup. We skipped the initial selection screen of the viewing method to preselect the ARCore viewing variant. The test scene was downloaded in advance to reduce waiting time and avoid connection issues. We recognized during the pilot test that the target surface should provide enough feature points for fast plane detection. To facilitate this, a newspaper sheet or random print out was added to future test setups' work surfaces. A run-through of the test should take around 15 minutes.

The tasks of the AR viewer questionnaire represented an exemplary use case, starting with opening a project, scanning the environment to detect planar surfaces, and position the loaded 3D model on them. Next, a specific building part had to be be discovered, selected, and inspected to access a particular property. The project contained different layer groups represented by buttons on the left side of the screen. The participant had to activate a layer and select an annotation with an embedded video. The video should be opened, played, and closed again. As the last task, participants could explore the building model for as long as they wished and finally leave a comment with some random text.

Again notes on the interactions with the system were taken during this part of the questionnaire. The test setup was not constant across all sessions since they took place at different locations. Still, in every case, a work surface was created with sufficient feature points, and testers could walk around the setup and explore the building model from different sides (Figure 5.1).

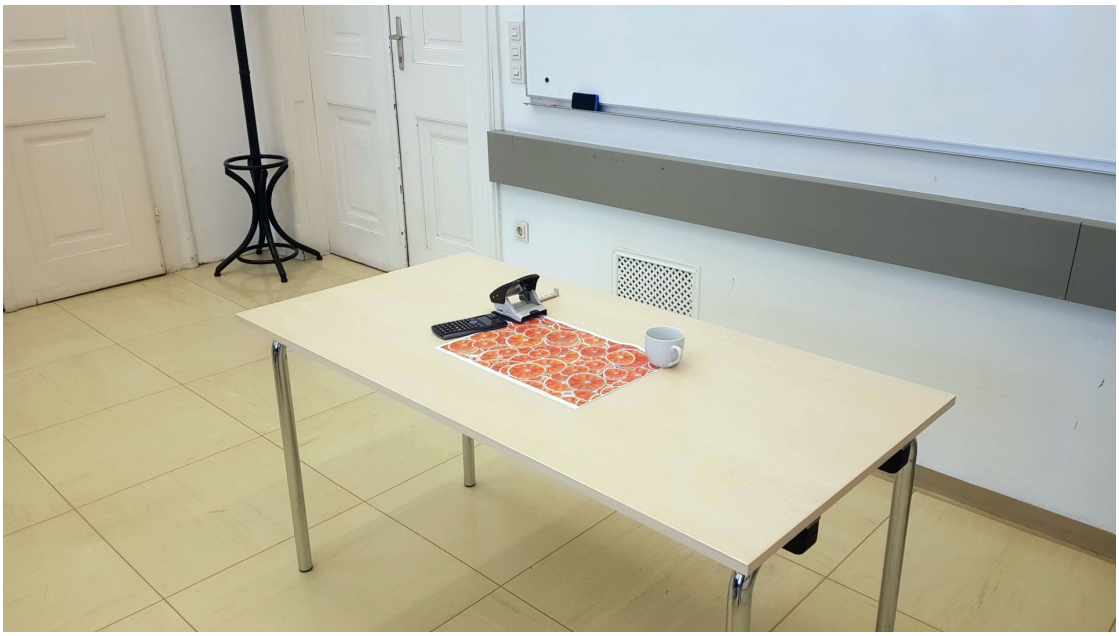


Figure 5.1: The AR viewer test setup at the Center of Digital Building Process.

5.2.3 IFC Desktop Editor Usability Test

The IFC desktop usability test (Figure 5.2) was structured into three sections, starting with a section on the consent form, basic demographics and prior knowledge on BIM of the participant. The second section consisted of 17 tasks (Table 5.4) alongside their difficulty rating and a start and end time. The third section contained the 10 SUS questions and a general feedback question.

The questionnaire was pilot tested and took around 30 minutes. During the pilot study, the participant identified a bug: deleted building parts were not marked as deleted during the export in some edge cases. During discussions with the author, some improvements to the test setup were found. The desktop of the test laptop was cleared of other files to make it easier to identify the relevant IFC & image file in the opening dialog. Also, the application was started in full-screen mode to avoid some issues with overlapping UI elements. Start and end time inputs were also added after the initial pilot test. During the pilot test, the participant was curious about the word *cumbersome* in the eighth question of the SUS part of the survey. This lead to the discovery of a widely accepted and updated version of the SUS questionnaire that replaces *cumbersome* with the more

modern alternative *awkward* [BKM08]. Similarly, the change from the word *system* to *product* is often proposed [LS09b], but this was disregarded as the tested application is better described as a system and not as a product since it's not publicly available.

The tasks formulated a small use case which incorporates most of the editor's functionality starting from the import and ending with export to the FTP server. The selection tool was tested in two ways as the scene selection can only select one building part, and groups of building parts have to be selected via the hierarchy. Finding and disabling properties on a building part was tested. Participants had to disable the *ArchiCADProperties* of a building element, an often unnecessary property added only by ArchiCAD. The creation and assignment of buttons and layers were tested in multiple steps as it was a complex task across numerous parts of the user interface. The transform tool was only tested with a single question on its move function since it worked similarly for all transforms. Creating an annotation was also split up into several parts to test whether the separate processes were easy to discover. Participants were observed as they solved these tasks. Notes were taken when they faced problems or commented on a function or their approach to a task. These notes act as additional qualitative data and support the uncovering of usability problems and their possible solutions.

Figure 5.2: The editor test setup at the office of the author.

5.3 IFC AR Android App Results

This section provides a detailed description of each test session of the IFC AR viewer application and the obtained results. The individual responses are then summarized and a list of identified usability problems is formulated alongside possible solutions. The outcome of the HARUS questionnaire is also calculated and discussed.

5.3.1 Sessions

Participant 1

The first participant was aged 30–34, male and had little experience with AR application (2 out of 5), and no expertise in handling BIM data (1 out of 5). He solved the first two tasks without any troubles and then he started to explore the building by selecting one building part after the other. After some time, he selected the right building element and discovered the width attribute after some scrolling. Working with the layer and annotation system was also relatively easy for the participant and the remaining tasks were solved in quick succession. After completing the task section, the tester answered the HARUS questionnaire and took a few minutes to formulate feedback. He would have liked to receive a message or alert when sending the comment form as currently the form stays open for a few seconds and closes after the upload is complete. He also discovered a bug: a video annotation sometimes keeps playing its audio even if another annotation is selected. Observing the participant also showed that he tried to enlarge the building model quite frequently with a two-finger zoom gesture. The same gesture was attempted with the full-screen view of images & videos.

In total, the questionnaire took eleven minutes. The participant rated all steps as easy (1 of 5) except for finding and selecting the specific building part (2 of 5) as he had to select a few different ones until finding the right one. The HARUS questionnaire results in a score of 95.8.

Participant 2

The second participant was aged 25–29, male, had experience with AR application (3 of 5), and had never worked with BIM data before. Opening and positioning the object in the real world went smoothly. The participant struggled with selecting an object as he only selected annotations, as they feature prominent titles inside the scene. Upon re-reading the question, he tried to select parts of the building and quickly found the right element. The other steps were no problem, but the participant was confused by the lack of confirmation when sending a comment. He also mentioned right at the beginning that handling the device and filling out the questionnaire could be quite hard, so the tasks were read aloud instead so he could focus on the steps. Additionally, he found it hard to hold the tablet device without obstructing the camera while holding it one-handed as the other hand is needed to select building elements. The tasks took seven minutes to complete.

He finished the whole questionnaire in eleven minutes; the HARUS calculation resulted a score of 90.6. The tester found the device quite hard to hold and not comfortable. He also would have liked some visual feedback when saving a comment.

Participant 3

The third participant was aged 25–29, male, and had more than average experience with AR applications (4 of 5) and some with BIM data (2 of 5). The first tasks were relatively easy for him, but he put the tablet down as he listened to the next task description. That caused the application to lose its tracking capabilities and the project had to be restarted. After this issue, the tester was a bit confused because a layer and annotation share the same name, and he did not know which one to activate. After playing around with the functionality, he realized which one was a layer and which one an annotation, although he would have liked a descriptive title above the layer buttons. He did not face any further problems and finished the tasks in six minutes.

The complete questionnaire took 14 minutes. In his final feedback, the participant suggested that the font size buttons should also affect the menu UI's font size and not only the element details. Additionally, he mentioned that it's helpful that the button for locking the object's position is colored red as he would have probably overlooked it otherwise. Finally, the HARUS questionnaire resulted in a score of 96.9.

Participant 4

The fourth tester was 25–29 years old, female, and had more than average experience with AR applications (4 of 5) and experience with BIM data (3 of 5). While opening and positioning the project was relatively easy for her, she struggled to find the right building element, but she was successful after some exploring. Finding the object's width was also quite hard since she expected such basic information to be further up in the list of properties. The layer buttons also irritated her shortly since she wasn't sure of their functionality. She suggests a title above the buttons. While entering a comment, she discovered that adding line breaks was impossible, a new bug that didn't come up during testing. She solved the remaining tasks without problems. The task section took ten minutes.

The full questionnaire took 17 minutes, and her final feedback was that she found the application quite clearly structured apart from a missing title for the layer buttons and the layout of the properties. The HARUS of this session resulted in 92.7 points.

Participant 5

Participant 5 was aged 18–24, male, had experience with AR applications, and with handling BIM data (both 3 of 5). He managed to solve the first tasks quickly, but he mentioned that he would have liked to have the tutorial steps more in the center of the screen so that they appear more prominent. He commented that holding the tablet

device with one hand only as he needed the second hand to select scene elements was inconvenient during the following tasks. While selecting an annotation and switching layers, he mentioned that it was odd that annotation popups stay opened when changing layers. He also would have liked to see a title above the layer buttons. The tasks took six minutes.

The total survey lasted 12 minutes, and the participant thought that the video annotations were a useful feature and provide a lot of context to the building model. As already stated, he felt it was hard to hold the device steadily while interacting with the AR scene. The HARUS calculation resulted in 78.1 points.

Participant 6

The sixth participant was aged 25–29, male, had experience with AR applications (3 of 5), and more than average experience with BIM data (4 of 5). The tester solved all tasks without any issues but was unhappy that he could not reposition the model while exploring. He often tried to enlarge the object with a two-finger zooming gesture. As he was browsing the properties of IFC elements, he mentioned that a search or categorization of the properties would be handy. While using the built-in comment dialog, he tried to click the save button multiple times as it lacks feedback. Additionally, he was confused about where the comments end up and how he could edit them. The task part of the questionnaire took seven minutes.

In total, the questionnaire took 17 minutes. The participant suggested that a filtering method of the 3D models could be beneficial, e.g. when looking for load-bearing building elements or elements of a specific material. This filtering could be done based on the IFC properties and highlight matching elements in different colors. He also would have liked to see an export or saving function where a user could capture a screenshot and annotate certain parts. This could then be used as feedback or communication channel for the project's creator, e.g., a lecturer. The lecturer could formulate tasks, and students would answer them using this new export or commenting function. The HARUS measured 91.7 points.

Participant 7

The final participant was aged 30–34, male, had some experience with AR applications (2 of 5), and no experience with BIM data. After opening the project, the tester was confused by the wording of the tutorial as he was unsure what "Position the model on the plane..." meant. He did not identify the orange surface inside the AR view as a plane, the author explained it briefly, and he could continue with the task. Upon finishing the positioning routine, the tester started to explore the building by walking around the tracking surface and looking at the object from many different sides, while switching the layers but not selecting any elements. After some time, he accidentally clicked on a building element and discovered this functionality. He solved the remaining tasks without

problems as he was exploring the functionalities while trying to solve prior tasks. The task section took twelve minutes to complete.

The questionnaire took 19 minutes to finish, and the tester also explored an additional test project after the test as he showed a lot of interest. He would have liked a more detailed description of the UI elements and scene elements, e.g., by giving examples of annotations, building elements, and layer buttons inside the tutorial and providing an image for the mentioned plane to make it's meaning more apparent. He also noted that the text on annotation flags was sometimes hard to read as it was too small. He also felt that it was quite hard to hold the device after exploring for some time. He suggested adding a case with a hand strap or grip to the tablet. Lastly, the HARUS resulted in 76 points.

5.3.2 Discussion & Summary

By analyzing the participant's feedback and comments alongside the session notes and quantitative questionnaire data, several specific usability problems can be derived, in addition to some overreaching issues with the IFC viewer application. In the following paragraphs, I analyze each usability problem and propose possible solutions.

The final HARUS calculation results in 88.8 points (SD=7.9) out of 100, no question received a perfect score (Figure 5.3), and some of the problems which were identified by the mentioned qualitative methods are also reflected in the results of one or more HARUS questions. Due to the two-factor nature of the HARUS questionnaire separate scores for the *manipulability* and *comprehensibility* of the application can be deduced [San+14], these result in 86.0 (SD=9.7) and 91.7 (SD=4.6) points respectively. This shows that users had more issues with the ergonomics of the application or the device than with the perceptual aspects of using it. The HARUS questionnaire is a relatively new tool for evaluating applications, but the authors postulate a similar scoring mannerism as with the SUS score [San+14], which would put the IFC AR Viewer in the category of "better products scoring in the high 70s to upper 80s" [BKM08]. While this is a good result, the low number of just seven participants does not allow for a broadly valid claim. The participants were able to solve all usability study tasks apart from participant 3, who was helped when the tracking was lost by restarting the application for him.

The tasks were overall very easy as their average difficulty was 1.32 (SD=0.69) with task 3 (*Find & Select the building part with the title "Decke-001".*) and task 4 (*Find the width of the element "Decke-001".*) being the hardest ones with an average difficulty of 2.43 (SD=0.97) and 1.57 (SD=1.13) respectively (Figure 5.4). Due to the small sample size of seven participants, no statement on the correlation of prior knowledge to their results can be made. On average, the participant took 7.4 minutes (SD=2.7 min) to finish the task section and in total 14.4 minutes (SD=3.3 min) to complete the whole questionnaire, which is in line with the initially assumed duration of 15 minutes.

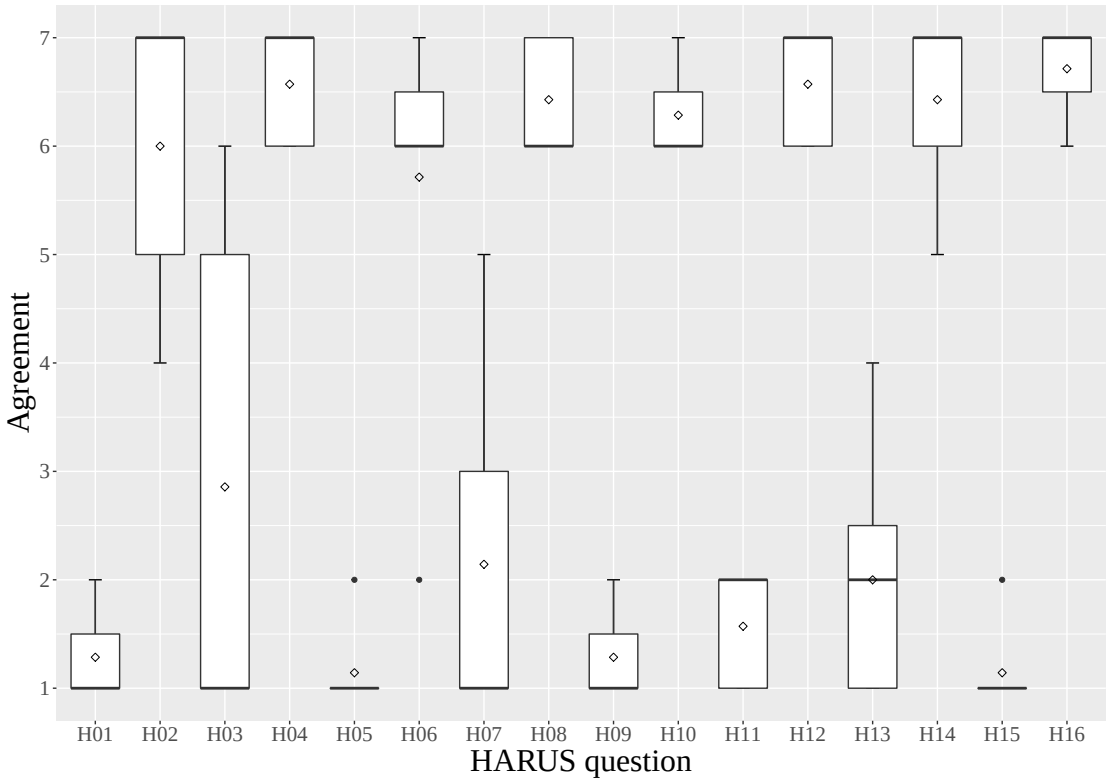


Figure 5.3: Box plot of the individual HARUS question results.

	HARUS question
H01	I think that interacting with this application requires a lot of body muscle effort.
H02	I felt that using the application was comfortable for my arms and hands.
H03	I found the device difficult to hold while operating the application.
H04	I found it easy to input information through the application.
H05	I felt that my arm or hand became tired after using the application.
H06	I think the application is easy to control.
H07	I felt that I was losing grip and dropping the device at some point.
H08	I think the operation of this application is simple and uncomplicated.
H09	I think that interacting with this application requires a lot of mental effort.
H10	I thought the amount of information displayed on screen was appropriate.
H11	I thought that the information displayed on screen was difficult to read.
H12	I felt that the information display was responding fast enough.
H13	I thought that the information displayed on screen was confusing.
H14	I thought the words and symbols on screen were easy to read.
H15	I felt that the display was flickering too much.
H16	I thought that the information displayed on screen was consistent.

Table 5.1: Questions of the HARUS questionnaire.

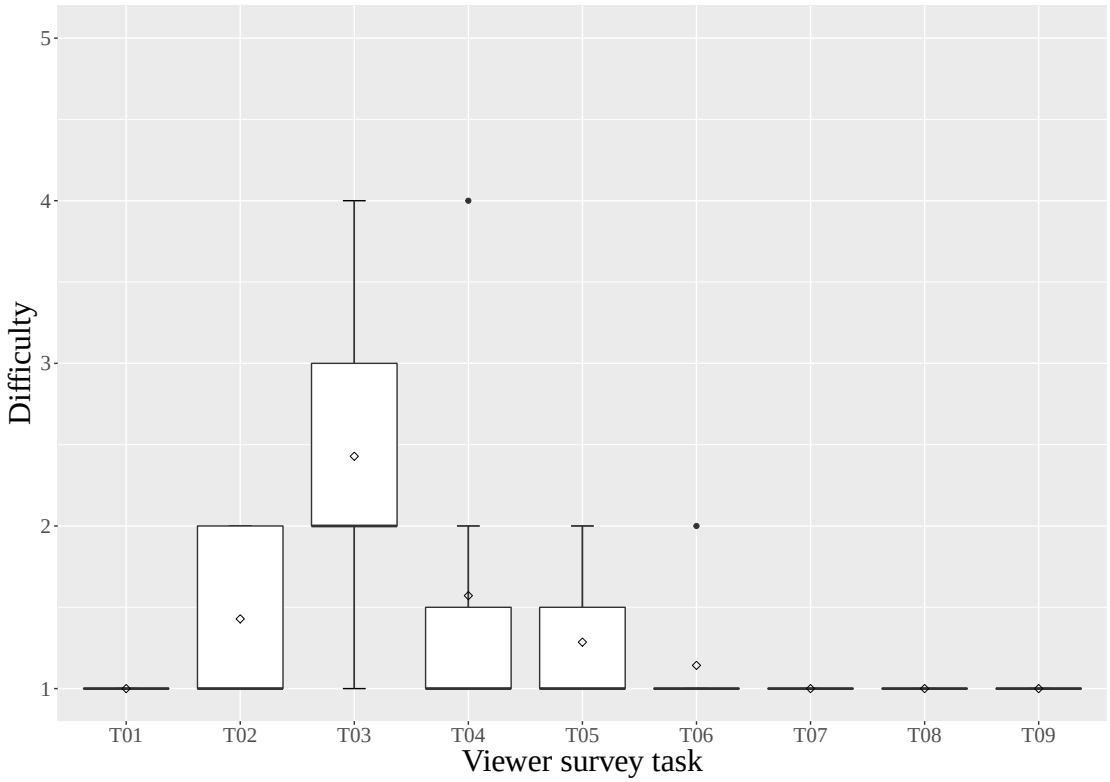


Figure 5.4: Box plot of the task difficulties during the viewer evaluation, rated from 1 (Easy) to 5 (Hard).

	Task Description
T01	Open the project "Attika-Demo-3".
T02	Follow the instructions on the screen and lock the position.
T03	Find & Select the building part with the title "Decke-001".
T04	Find the width of the element "Decke-001".
T05	Activate the layer "Rohdecke"
T06	Find and Select the Annotation with the title "Abdichtung".
T07	Open & Play the embedded video.
T08	Close the video.
T09	Now you can explore the other layers and annotations if you want, at the end add a comment with your name and a short text.

Table 5.2: Tasks of the viewer questionnaire.

Tutorial Improvements

The current tutorial implementation consists of a text field at the top of the screen that displays steps on initializing the AR view by scanning the environment and positioning the 3D model inside. Currently, this tutorial is displayed every time a new project is loaded with the following steps:

1. Open Project
2. Move the camera around to detect a surface
3. Place the 3D model by touching & dragging on the plane
4. Adjust the scale of the object by pinching with two fingers – Lock the object with the Button on the right

The system displays one step and switches to the next step after recognizing that the user has completed the previous one (Figure 5.5).

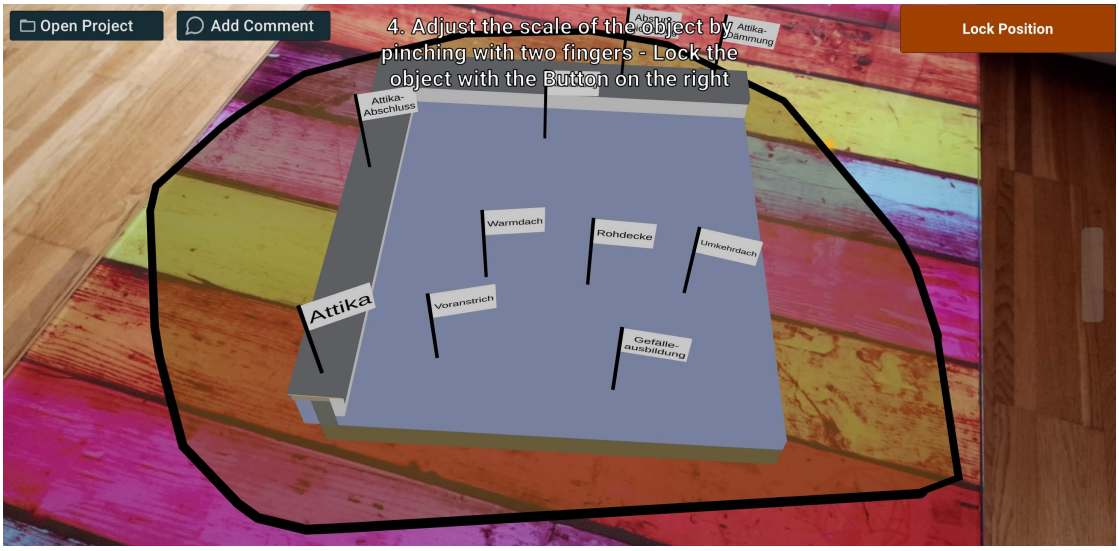


Figure 5.5: The fourth and final step of the current tutorial.

Four of the seven participants faced problems with the current tutorial implementation. For two testers (no. 2 & 3), the tutorial failed to convey the difference between building parts, annotations, and layers, which is also reflected in the higher difficulty rating of the corresponding tasks (2.4 of 5). Participant 7 was unsure how to interpret the tutorial's wording, which caused him to explore various parts of the project before finding the right building element, resulting in a 4 out of 5 for this task's difficulty. Participant 5 would have liked the tutorial to be more prominent in the center of the screen, resulting in a

higher average difficulty rating for the first task of following the on screen tutorial (1.4 of 5).

A solution to these problems would be to rework the tutorial system to incorporate images or even animations to describe the initialization steps more effectively and show the user isolated examples of the app's elements alongside their descriptions. This could be built as a full-screen first-time-use tutorial that is also always accessible via a separate help button. Multiple pages could visualize the functionalities of the app and provide the user with the necessary prior knowledge.

Layer Title

Similarly to the new tutorial system, adding headers or short descriptions to critical sections of the app's interface could improve first-time user experience.

Four out of the seven participants (no. 3, 4, 5 & 7) complained about the lack of a descriptive title on top of the button row on the screen's left side.

Alternatively, the system could hide the buttons inside a menu that folds outwards from the left side of the screen with an additional indicator on which layer is active. As the IFC editor usability test showed that participants had no issues identifying an abstract layer icon, this could be used to shrink that menu down even further. This approach could free up the screen space for a clearer view of the AR scene but could be tedious if a user wants to switch between layers frequently.

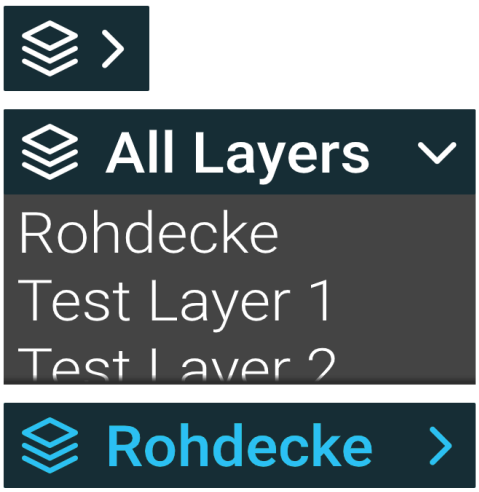


Figure 5.6: An example for an improved layer selection UI

Comment Dialog

The comment function was irritating to three participants (no. 1, 2 & 6) as they were unsure whether the system saved the comment or not, often hitting the save button multiple times.

The current implementation starts the asynchronous upload of the comment to the server when the save button is clicked but does not update the UI. When the upload finishes, typically after three to five seconds, the window is suddenly closed.

Fixing this problem could be achieved by marking the send button as not interactable after the first click. This would not allow additional click events and update the button's visual to a greyed-out version. Additionally, the button's text could be changed to indicate the upload process or even the progress in percent, but this would probably jump from zero to 100 percent instantly most of the time as the transfer usually only consist of a single or very few packets. A similar change could indicate an upload failure in case of poor or no network connection, which currently is not handled. These changes would align the UI with the IFC editor version, which already features this kind of behavior in its upload dialog.

Participant 6 was also unsure where this comment is submitted to and what its use was. A better description in the comment UI should indicate that this comment is used as feedback to the project's creator. A change of the submitted comment should also be possible alongside a UI indicator that a comment is present. This could be achieved by changing the button text of the *Add Comment* button to *Change Comment* after a comment has been submitted.

Device Handling

Three out of the seven participants (no. 2, 5 & 7) found the tablet hard to hold while exploring the AR scenes. Most of the time, users held the device with one hand and used the other to interact with the scene. Although some users also opted to grip the device with both hands and select elements by using their thumbs, given the size of the test tablet, this is not an ideal solution as some building elements could be out of reach. Additionally, one has to be careful not to cover the camera when shifting the tablet around as this can have adverse effects on the quality of the AR tracking. Also, the HARUS test included questions on the difficult handling of the device, which received relatively high average agreement scores of 2.9 (*I found the device difficult to hold while operating the application.*) and 2.1 (*I felt that I was losing grip and dropping the device at some point.*) out of 7.

This problem is best addressed on the hardware side of the project and not by changing the software. Choosing a tablet is a trade-off between screen size and weight. A bigger screen allows for a better AR experience but makes the device unwieldy or awkward to hold after some time. So one could select a smaller test device, e.g., a phone instead of a tablet, or one could try to improve the handling of the bigger tablet by adding a suitable

Figure 5.7: A tablet case with an attached hand strap. [Ama]

case with an integrated hand strap (Figure 5.7). This could have the added benefit of protecting the device from falls.

On the software side, one could build upon interaction methods that don't require touch input, like selecting objects by looking at them for a specific duration. The initial position could be done via a ray cast from the center of the AR Camera instead of a touch position, and users could scroll through the lists of IFC details with their thumb. Only the comment dialog has to be filled out using one or both hands, but as this is intended to be a final feedback form, users could put the device down at the end of a session and use both hands to interact with it.

Adjusting Model Position

Two of the seven participants (no. 1 & 6) tried two adjust the already placed building model's size by gesturing a two-finger pinch zoom. In the current implementation, the building model is placed and locked during the initialization phase and can not be changed during a session. The only option to reposition a loaded project is to open it anew.

As this two-finger gesture is very prevalent in modern UIs, the participants assumed it would work inside the AR view and the full-screen annotation media files. This could be easily enabled as the functionality is already implemented for the initial positioning but was disabled for the rest of the session to prohibit accidental scaling while trying to select building objects. The scaling implementation will have to account for the single-touch select feature and only trigger when the user intends to scale the model. A separate

button that switches the current session into a repositioning mode could fix this issue, which would allow the user to drag the model again on the recognized AR plane and pinch with two fingers to adjust the scale. A more seamless integration would feature a short delay when two touches are recognized so that the system doesn't enter the scaling mode immediately.

Font Size

The current implementation features a font adjusting functionality for the detail section of selected elements. Two of the participants mentioned that other parts of the UI were also hard to read. Participant 3 found the other menu items hard to read, and participant 7 had a hard time reading the annotation flags. This is also reflected inside the HARUS questionnaire as the relevant question (*I thought that the information displayed on screen was difficult to read.*) received an average agreement score of 1.6 of 7. The current implementation of the menu UI has a fixed font size dependent on the screen size and should look similar on a lot of device resolutions. The annotation implementation tries to display an annotation title as big as possible inside the flag, up to a given threshold.

One could implement bigger font sizes in the menu part of the application in a straightforward manner. Still, the surrounding UI elements like buttons and input fields would need to have flexible sizes to accommodate that. In the case of the annotation flags, the titles could grow beyond their enclosing flags, which could improve readability but clutter the AR view. Another approach would be to limit annotation titles to a certain number of characters and optimizing the space inside the flag visual based on this. One could move further details of the annotation inside the description field, which is viewable by selecting the annotation. This new adjustability would be accessed through the same buttons as the details window, but they would be integrated as static UI buttons as they are currently only revealed if an object is selected.

Details Structure

Two of the seven participants (no. 4 & 6) voiced their issues with the application's details section. But most of the testers had to look for the requested attribute, the width of the selected object, for quite some time. Participant 4 was quite frustrated with this and opted for a difficulty of 4 out of 5 for the task. The example in the test case was also an extreme one since none of the default IFC properties were filtered and two testers expected such a fundamental attribute to be on the very top of the list.

This could be fixed by offering a search field alongside the details view where a user can type a search term and a full-text search is performed on the detail list. The system highlights matching properties or rearranges them based on a calculated matching likelihood.

Another approach would be the filtering of the objects based on categories. These could be either custom defined by the lecturer inside the IFC editor or deduced by the IFC property names. This could yield many categories, but it would be a good addition for

more experienced users and novices to get an overview of the included IFC property types.

Both approaches could be implemented and combined to offer a highly customizable search inside the AR viewer.

5.4 IFC Desktop Editor Results

This section describes the IFC editor usability test sessions. Features and functionalities with usability issues are identified through analyzing the questionnaire results and the provided feedback. Additionally, the SUS questionnaire score is calculated and discussed.

5.4.1 Sessions

Participant 1

The first tester (30–34, male, no BIM experience) completed the first two steps quickly but struggled with finding the option to delete a building element. Upon discovering it, he was irritated by the lack of feedback as the deleted object just disappeared. After selecting the next building, he had difficulty finding the required attribute, looking in various subwindows and not in the attribute section. He also tried to disable the attribute globally for all building objects, which was not the solution. He noticed his mistake and began scrolling through the long list of properties, nearly giving up before reaching the right attribute yielding a difficulty score of 4 out of 5 for this task. He finished the rest of the tasks without significant problems. This task section took the participant 15 minutes to complete.

In total, this questionnaire took 27 minutes. The only challenging task was finding and disabling a specific attribute. The SUS calculation resulted in 80 points. The participant suggested adding a save button to the comment window of the object as he was unsure whether his changes were sent. A note that comments were saved automatically would also have been a solution for this problem. He also was irritated by the hierarchy view and would prefer a more traditional tree view where a user can keep an overview of all elements and their structure. To alleviate the hard part of the survey, finding a specific attribute, he suggested hiding the details of an attribute and only displaying their title to condense the displayed information.

Participant 2

The second participant (25–29, male, no BIM experience) had no problems with opening and finding a building part; deleting one took some searching, but he identified the right button. The user tried to delete by pressing the *Del* button on the keyboard first. He was irritated by the autosave of the comment window and suggested a save button. He struggled a bit with the layer system as he could not find the right subwindows at first. He also tried clicking on the title *Layers & Buttons* instead of the open button next to it. He finished the tasks section in nine minutes and mentioned that although he struggled with some parts, he thought that they were quite clear after the fact.

In total, the questionnaire took eleven minutes. The user kept the overall feedback brief as he thought that most points were already covered during the task phase but mentioned that an autosave tick icon similar to WhatsApp's indicator for sent and read messages could be useful. The SUS resulted in 95 points.

Participant 3

Participant number three (25–29, male, little BIM experience) started without any problems but tried to delete an object by right-clicking and expecting a context menu. Additionally, he tried to move the subwindows around, which is not possible. Finding the right attribute to disable also took him quite some time. He also wanted to disable it through the global property filter but expected it to hide building elements with the disabled properties. After a lot of scrolling, he managed to disable the right IFC property. A small break of eight minutes was taken as the tester received a phone call. When the user moved to upload the project, he struggled to find the right input for the project name as the inputs are not labeled clearly, and he expected the input to be the first one, but instead, the FTP access inputs are first. The tasks took 19 minutes. The short break was excluded from this duration.

The whole questionnaire lasted for 29 minutes, again excluding the break. The tester mentioned in his feedback that tooltips would be an excellent addition, as they could help to explain inputs and buttons. He also thought that the hierarchy view looks unconventional, but he understood it quite quickly. Lastly, he stated that the distinction between an active tool button and an inactive tool button is too little. The SUS amounted to 97.5 points.

Participant 4

The fourth participant (25–29, female, BIM experience) had no real issues with the first tasks. She mentioned that some more visual feedback would be nice for selecting objects in the hierarchy. The current implementation darkens the background slightly. As she was unchecking a property checkbox inside the UI, she was unsure whether this meant disabling it. She also tried to save a comment by pressing the Enter key but instead discovered that a line break wasn't possible. While looking for the layer assignments, she tried to click the title *Layers & Buttons* instead of the button next to it. She was unsure whether a button creation was successful, as the system did not provide any feedback. During the transform task, she noticed that dragging inside an input moved an object along that axis, but she wanted to select the number to delete it. While adding an image to an annotation, she clicked inside the path input, which didn't have any effect. After a few seconds, she discovered the button to open an image underneath the input. At the last task, she was unsure where to enter the project name, but identified it by its placeholder text after a short time. Finishing the tasks took 15 minutes.

She completed the questionnaire after 20 minutes. Her final comment was to add more titles and tooltips to the UI and add more feedback to the saving and modifying functionalities. The SUS section resulted in 95 points.

Participant 5

The fifth participant (18–24, male, experience with BIM) had no issues while opening the project and selecting building elements. As he deleted a building element, he was

surprised as no confirmation dialog appeared. Searching for a specific IFC property of a component took him longer than he liked. The property section was not structured clearly. As he finished writing a comment, he was unsure whether the system saved his changes. During moving the object along an axis, he accidentally dragged inside an input field that caused the model to disappear from the screen, but he quickly corrected the wrong offset. The task section took eight minutes to finish.

Sixteen minutes were needed to complete the questionnaire. He suggested adding a heading in the property section of the select tool. The three icons to add a comment, open the layer window, and bulk disable properties should have been a bit bigger. To get a clearer view of the individual properties, one could add different colors to them. Additionally, labels on the axes inside the scene view could be helpful. Changing the heading of the Transform tool to *Move, Rotate & Scale* could make its functionality clearer. He also would have increased the title's font size across the whole application. Finally, he mentioned that the application would be much easier to use on the second try. The SUS amounted to 90 points.

Participant 6

Participant number six (25–29, male, lots of experience with BIM data) started with some issues when rotating the opened 3D model. He was unsure which key combination was supposed to be pressed as he works with many different applications, and all of them feature other methods. He discovered the right click rotate after several tries but stated it was not apparent right away. He got used to the scene selection quite quickly but did not initially use the hierarchy selection and was confused by its behavior. While creating the layer group buttons, he was not sure whether their changes were saved. As he created an annotation, he first entered a name and then placed an annotation visual on the model. This did not work and is a bug in the annotation system. The tester then created a few annotations to make sure the other functionalities of the annotation system work. The task section took 19 minutes to finish.

The complete questionnaire took 31 minutes. The tester would have liked to see more selection methods in the scene view, e.g., a rectangle drag selection or a bulk selection by pressing the shift key and selecting multiple elements. He also mentioned that exploring the application would be easier alone as he would progress at his own pace. The SUS resulted in 80 points.

Participant 7

The final participant (30–34, male, no BIM experience) opened and selected building elements without issues but deleted the wrong element and was unhappy that no undo function exists. The project had to be reopened, and the right element was deleted. As he finished writing the comment on an element, he tried to save it with the Enter key. While moving the building model along an axis, he was confused as he was unsure which axis was which. While creating annotations, he accidentally deselected the newly created

annotation and filled out the name input, which did not affect it. He then deleted the empty annotation and began anew. The task section took 14 minutes.

The questionnaire took 22 minutes, and the tester recommended the addition of keyboard shortcuts for deleting elements and an undo action. He also stated that adding labels to the axes would be a good idea and that he would like to move the camera around more freely, e.g., by centering the rotation around a different point, which is already possible by setting the rotation origin to a sub-element, but he did not discover this during his session. During the SUS part of the questionnaire, he mentioned that the application is straightforward to use, but he struggled slightly as it is his first time using it. The SUS results in 100 points.

5.4.2 Discussion & Summary

Again, the participant's feedback is analyzed together with their comments and the author's notes to end up with a set of usability problems, supported by the quantitative methods' results. The next paragraphs cover each usability problem alongside possible solutions.

The SUS questionnaire yielded a total usability score of 91 (SD=8.1) out of 100, which would be an excellent result, according to Bangor et al. [BKM08]. Still, again a higher number of participants would be needed to make a valid claim. Question eight (*I found the system very awkward to use.*) received the worst score with an average agreement of 1.7 but also a wide spread across the scale (SD=1.3) (Figure 5.8). Apart from participant 7, who accidentally deleted a wrong building element, all participants solved all tasks of the usability study successfully.

When looking at the average task difficulties (Figure 5.9), task 5 (*Disable the attributes with the title "ArchiCADProperties" on the building element "Wand-001".)* was the hardest for most of the participants with an average difficulty of 2.6 (SD=1.5) while also being the only task that received a rating of 5 by one participant. In total, the average task difficulty was 1.5 (SD=0.8), while participants took an average of 15.3 minutes (SD=6.5 mmin) to solve these tasks and 23.5 minutes (SD=9 min) to finish the whole questionnaire, which was initially assumed to take around 30 minutes.

Modification Feedback

Four out of the seven participants (no. 1, 4, 5 & 7) complained about or mentioned the lack of feedback when deleting a building element. They either expected a confirmation dialog or another form of indication that the object was indeed deleted. In addition to the lack of feedback, one participant (no. 7) was frustrated by the lack of an undo function upon accidentally deleting the wrong object. Also, the rated difficulties of tasks that incorporated these functionalities were higher, e.g., task 3 (*Delete the selected building element "Decke-001".*) with an average rating of 1.8 (SD=0..9).

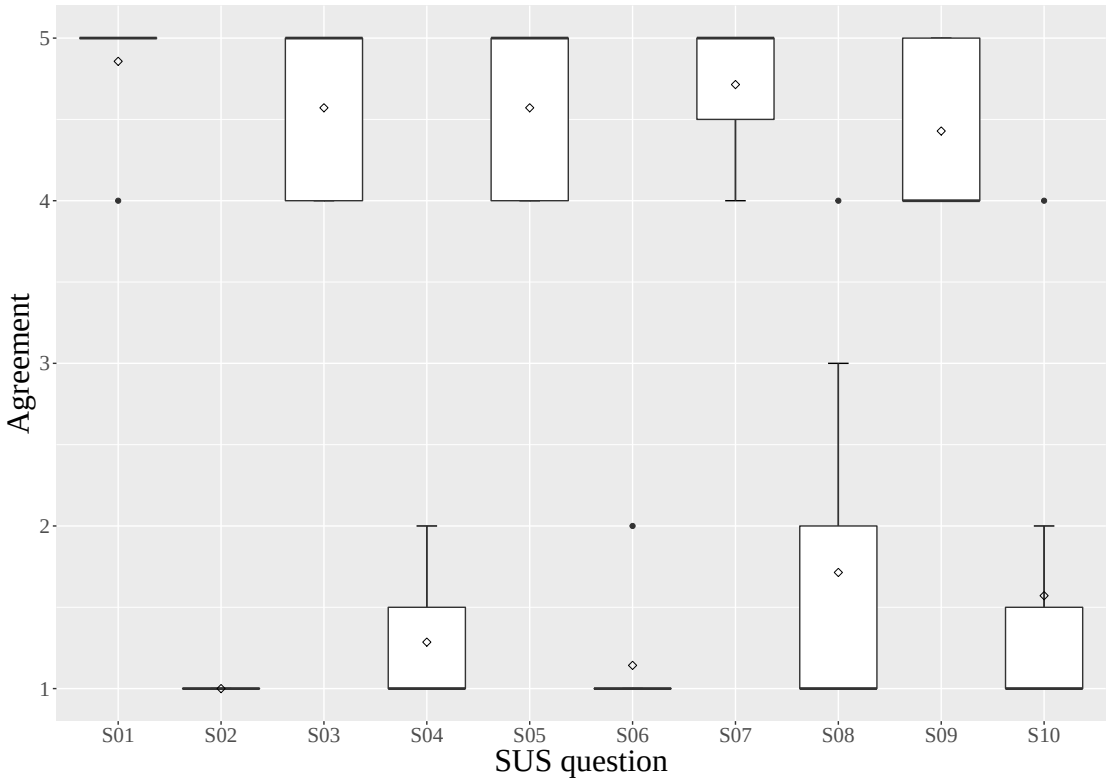


Figure 5.8: Box plot of the individual SUS question results.

	SUS question
S01	I think that I would like to use this system frequently.
S02	I found the system unnecessarily complex.
S03	I thought the system was easy to use.
S04	I think that I would need the support of a technical person to be able to use this system.
S05	I found the various functions in this system were well integrated.
S06	I thought there was too much inconsistency in this system.
S07	I would imagine that most people would learn to use this system very quickly.
S08	I found the system very awkward to use.
S09	I felt very confident using the system.
S10	I needed to learn a lot of things before I could get going with this system.

Table 5.3: Questions of the SUS questionnaire.

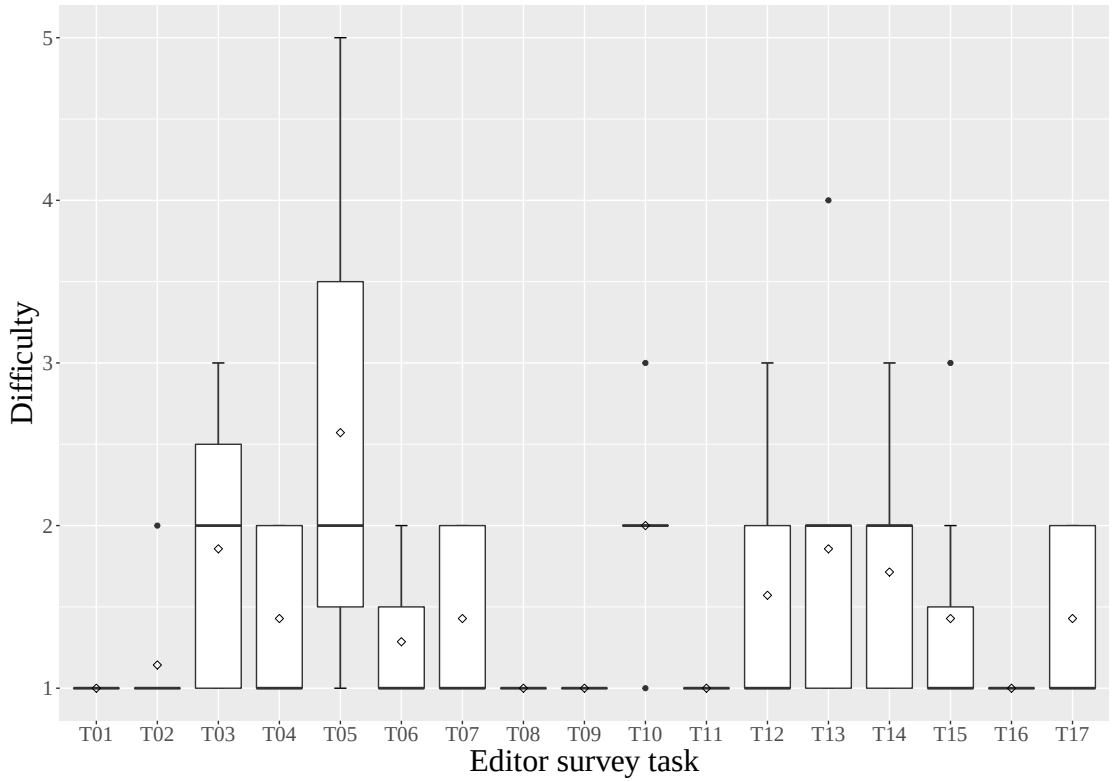


Figure 5.9: Box plot of the editor questionnaire task difficulties, rated from 1(Easy) to 5 (Hard).

	Task Description
T01	Open the IFC file attika.ifc from the Desktop.
T02	Find and select the building element with the title "Decke-001".
T03	Delete the selected building element "Decke-001".
T04	Find a way to select the group of building elements "Wand-001".
T05	Disable the attributes with the title "ArchiCADProperties" on the building element "Wand-001".
T06	Add the comment "test comment" to the building element "Wand-001".
T07	Create two new Buttons called "Button 1" and "Button 2".
T08	Create a new Layer called "Layer 1" and assign the "Button 1" to it.
T09	Create a new Layer called "Layer 2" assign the "Button 1" and "Button 2" to it.
T10	Add the building element Wand-001 to the new "Layer 1".
T11	Add the building element "Wand-002" to the new "Layer 2".
T12	Check if the Buttons control the visibility of the assigned Elements.
T13	Move the whole model by 1 along the x-axis.
T14	Add a new Annotation with the title "Test Title" at a random spot in the model.
T15	Add the image "test.jpg" from the Desktop to the new Annotation.
T16	Change the appearance of the Annotation to a ball.
T17	Export the project as "Test Project".

Table 5.4: Tasks of the editor questionnaire.

Similarly, two participants (no. 4 & 6) criticized the lack of feedback when creating new layer buttons and four participants (no. 1, 2, 4 & 5) expected some feedback when the application saves a comment.

The collection of these problems leads to the editor application's biggest usability problem, a lack of feedback when modifying the application state. The current prototype saves most changes automatically. A user stops manipulating a field or value and the application persists the modification after a specific timeout or when the user interacts with a different UI element. Although this seems very efficient since a user does not need to signal for a save on every edit, the underlying mechanism is not transparent for the user and is missing an abort or undo functionality, which would probably increase the user's confidence in the system.

To alleviate this, a save or submit button could be added to each modification element. With the delete button, one could add an input that makes sure that a user intended to delete the selected object and also gives the option to abort the deletion (Figure 5.10). A user could trigger the confirmation by hitting the *Enter* key and the abort action with the *Escape* key. A submit button could be added adjacent to the layer button creation's input field, submitting an edit could cause its appearance to change from an input field to a text field that becomes editable with a separate button (Figure 5.11). A similar solution could be applied to the comment field as four participants (no. 1, 2, 4 & 5) had trouble with this functionality.

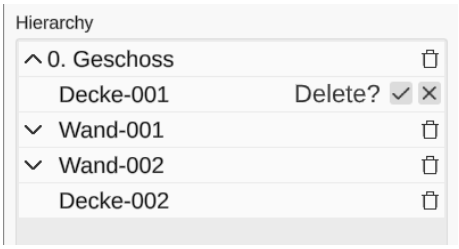


Figure 5.10: A confirmation dialog when deleting an object.

Figure 5.11: A possible Button Creation design, that includes a confirmation and a separate edit button.

Additionally, to address both the lack of an undo function and the missing feedback, one could add a history view to the UI that displays each modification in a list and allows users to jump to any given edit. Image editing applications like Adobe Photoshop usually provide such a feature; thereby, users could already be familiar with it (similar to Figure 5.12). Standard hotkeys like *Ctrl+Z* and *Ctrl+Shift+Z* could be used to step backward and forward in this history timeline.

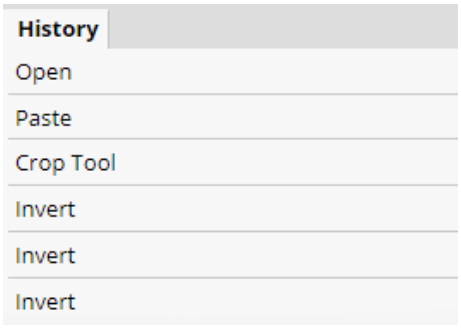


Figure 5.12: The history panel of the online image editing tool photopea.com.

Properties Section

Four participants (no. 1, 3, 4 & 5) were confused by the property section in one way or another. Task 5 of the editor questionnaire required the use of this section and received the highest difficulty rating among all tasks (Figure 5.9). The biggest problem being the long list of different IFC properties for each building element. This issue was already discovered early in the development of the IFC editor, which caused the addition of the global property filter, which users can edit to reduce the number of properties across the whole application. Unfortunately, this aspect was not part of the usability test; some participants tried to interact with the property filter, but they didn't manage to reduce the number of properties. Participant 5 suggests to color the properties differently, and participant 1 would have liked a more condensed view of the properties, hiding everything but their title. Different colors for different IFC properties would be a good enhancement as long as there would be a reasonable amount of colors. Unfortunately, the categorization of properties would have to be done manually as an automatic classification would probably not yield the desired results. Hiding the contents of properties would increase the readability. Still, in some cases, users need the data inside the property as attributes, e.g., whether a wall is a load-bearing one, are very relevant factors when exploring a model. Additionally, participant 5 suggested adding a heading to the properties window to indicate its use. As a side note, participant 4 was unsure whether a property was disabled after deselecting the checkbox next to it.

Another solution would be to integrate a search feature, similar to the proposed changes to the details window inside the viewer application (Section 5.3.2), which would provide a similar experience inside the mobile and desktop app. During development, it was considered to hide attributes with no values, but even empty attributes sometimes carry useful information, so this cannot be applied to all properties. A static heading next to the element name, which changes based on the current selection, could provide the user with some context (Figure 5.13). A placeholder text inside the search input could further identify this section of the sidebar as property section. To indicate that properties can be disabled and enabled via checkboxes, one could add a tooltip that describes the current state. Additionally, disabled properties could appear as crossed out text.

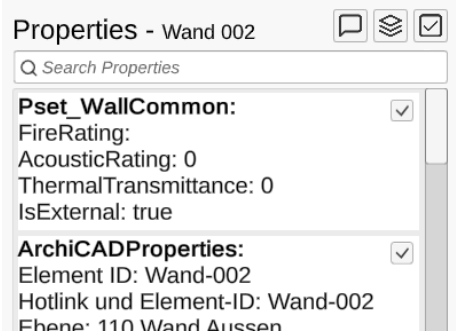


Figure 5.13: An improved properties panel with added search and heading.

Hierarchy

One major issue was the hierarchy view. Four participants (no. 1, 3, 4 & 6) were irritated by the current design of the hierarchy view and expected a more traditional tree view but were able to use it without too many issues. Participant 4 additionally mentioned that the current selection was not indicated clear enough.

The hierarchy view is structured in a way that tries to reduce the number of items inside of it. It resembled a traditional tree view in an initial implementation but was poorly optimized since it still spawned unrevealed items. Additionally, the hierarchy could get quite deeply nested, which caused the hierarchy view to become unusable as the user had to use the horizontal scroll bar to navigate to the right element. The hierarchy view was changed to adapt based on the current selection and display a maximum of three levels to fix this. But as users seem to rely on a more traditional representation, this should be revisited.

Different file explorers suffer from similar problems, as they visualize files across deeply nested structures. The Windows File Explorer shows only one hierarchical level while allowing users to navigate between levels via breadcrumbs. Apple's OS X features the same behavior but additionally has a mode where it displays multiple levels at once inside a column view. Since space is constricted in the application's window, as the hierarchy is a secondary view next to the 3D scene, the first approach would better fit our purposes. So by adding breadcrumbs to the hierarchy view, we would give users the option to jump up a few levels inside the hierarchy quickly (Figure 5.14).

Figure 5.14: An improved hierarchy view with added breadcrumbs.

Keyboard shortcuts

Two of the seven participants complained about the lack of keyboard shortcuts, specifically they where missing Delete, Undo, and Submit or Save hotkeys. Since all of these feature standard keyboard shortcuts, it would be most useful to implement them according to best practices. Additionally, one should add custom key-combinations for frequently used tools and operations, e.g., for the Select, Transform, and Annotate tools. These could also be made re-assignable in a future iteration to allow users to change them freely. The Unity engine already implements most of the needed functionality since this is common practice in video games.

Opening Subwindows

Multiple participants (no. 2 & 4) tried to open the subwindows for the *Layers & Buttons* functionalities by clicking on the title instead of the open button next to it. Fortunately, this is easily fixable, and one should also apply this to the *Global Property Filter*, which suffers from the same issue but was not opened often during the usability tests (Figure 5.15).

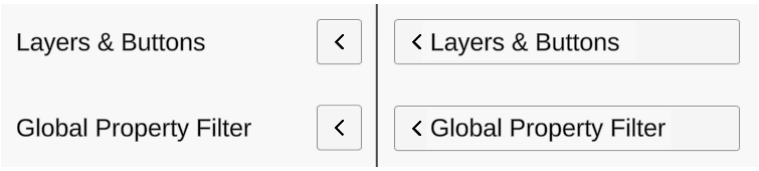


Figure 5.15: Before and after view of the subwindow buttons in the sidebar.

Upload Name

Two participants (no. 3 & 4) were unable to identify the project name input in the upload dialog right away. They both expected it to be at the top of the dialog, but instead, there is the FTP folder's path, unfortunately, without any label. Since the editor's users should not need to change the FTP credentials regularly, this information can be moved to the end of the dialog and replaced with the project name input, which a user has to enter at every upload. Additionally, labels should be added in front of each input, as users have to guess which input is the FTP path and which one is the FTP username.

Tooltips

Two participants (no. 3 & 4) requested tooltips to better understand some UI elements and their functions. One could implement this across all buttons and inputs. Additionally, some text fields could benefit from a tooltip explanation upon demand. When hovering the cursor above the hierarchy, a short text could explain that all IFC building elements are listed in hierarchical order and be selected by clicking them and deleted with the button next to their name. Participant 4 mentioned a clear case where tooltips could be

beneficial as she was unsure what the checkbox next to the IFC attribute was supposed to do. A short tooltip indicating that deactivating the checkbox results in a disabled property that won't show up in the exported project could help users directly without them needing to look up the relevant section in a manual.

Transform Inputs

Two participants (no. 4 & 5) had issues with the transform inputs' current implementation. Both tried to select the number inside the input by click-dragging across the input, a standard interaction for most applications. The IFC editor reacts to this interaction by either adding or subtracting to the relevant value causing the object to move, rotate, or scale based on the direction and distance of the click-drag interaction. This issue probably caused participants to rate the task as more difficult than other tasks with an average rating of 1.9 (SD=1.1). The 3D editing application Blender includes similar behavior but indicates this by changing the cursor icon into an indicator of this functionality. As it is questionable whether adding this indicator to the IFC editor will solve the problem, removing the feature seems like a better course of action. Instead, a separate coordinate gizmo could be displayed inside the 3D view to allow more direct manipulation of the 3D model.

Axes Labels

The 3D view includes differently colored axes to indicate the coordinate center and the three dimensions. Two participants (no. 5 & 7) mentioned the lack of labels on these axes as they tried to move the spawned 3D model in a specific direction. This oversight is easily fixable by adding text billboards in the 3D scene, which always face the camera and are of the same color as the axes.

Annotation View

As one opens the annotation view, the sidebar fills with the necessary inputs, which caused two participants (no. 6 & 7) to add name and image to the annotation before placing it inside the scene view. Unfortunately, a bug causes the newly created annotation to spawn empty, leading to an inconsistent application state. The users then tried to enter the information again, which resulted in confusion and a higher-than-average difficulty score for this task: 1.7 (SD=0.8)

To alleviate this, one could hide the sidebar when the annotation mode is activated, which would probably lead users to interact with the scene view and notice faster that they can place annotations. After a user places a new annotation, the sidebar could appear, and the user could enter the required information, upon deselection the sidebar could disappear again.

In line with the previous recommendation to better indicate modifying functionality, one could add a save button to the sidebar's bottom section, which persists the changes and hides the sidebar again.

Also, concerning the annotation view, participant 4 tried to upload an image by clicking inside the image path input and was unsure what to do next as no upload dialog appeared. The input should allow users to enter a path to an image manually, but no participant tried this. To fix this issue, the path input should be changed to a non-editable text field.

Miscellaneous

The following usability problems were mentioned by one participant each but still represent some important IFC editor issues.

A right-click context menu could provide quick access to the delete functionality inside the scene view. One could add other features to the menu, like refocusing the camera on the selected object, a functionality that no participant discovered.

Moveable subwindows could help users to customize the application layout. Unfortunately, Unity's UI System has no subwindows concept, so this functionality would have to be custom made, accounting for limited developer resources. This feature is probably not in the future scope of this project.

Active and inactive tools inside the toolbar are not distinguished enough as they only change slightly in brightness. One could fix this by adopting a more widely spread approach of displaying active tools as a darker button.

The right-click triggered camera orbit was unexpected for some participants. One would have to look at the camera controls of traditional 3D tools, e.g., ArchiCAD, which uses the *Shift+Middle Mouse Button* key combination for the same feature. Blender uses only the middle mouse button, Unity 3D, the *Shift+Left Mouse Button* combination, and Maya 3D uses the *Alt + Left Mouse Button* combination. So no real best practice exists in this case. Instead, a short info text at the top of the screen could indicate the right-click orbit camera. When the user applies the right key combination, this hint could disappear not to clutter the screen. Similar to this problem, more camera controls were requested, e.g., the possibility of moving the orbit camera's origin. This is currently possible by selecting an object and pressing the *F* key to recenter the camera on it. Additionally, one could add the option to pan the camera and thereby moving the origin by pressing *Shift+Right Mouse Button* or another suitable key combination.

Lastly, some participants had issues with the readability of individual UI elements and thought that headings were too small across the application. Some icons should be enlarged, especially the three icons inside the sidebar's property section to add a comment, assign a layer, and bulk-disabling all properties. That could also be the cause of the high difficulty rating of task 10 & 12 (*Add the building element Wand-001 to the new "Layer 1". & Check if the Buttons control the visibility of the assigned Elements.*), on average 2.0 (SD=0.6) & 1.6 (SD=0.8) as users struggled to identify the layers and visibility icon. This should be fixed across the whole application to provide a uniform look in all the different sidebars and subwindows.

CHAPTER 6

Conclusion

Although both applications received good usability scores, the viewer approx. 89 and the editor 91 points, the qualitative methods' results uncovered usability problems in both of them. They share a common usability issue: the lack of feedback to users in cases where the application's state changes through inputs or buttons. The current auto-save functionalities seemed like an efficient solution but are not transparent and leave the users unsure whether these changes persist. By adding confirmation and abort buttons to both applications' dialogues, users would get more control over their edits.

Another issue that arose in both applications was the IFC details section. One has to scroll through an unsorted list of entries while searching for a specific IFC property and its attributes. Although editor users can fix this for the viewer application by disabling unnecessary properties inside the editor, the underlying issue remains. A first step could be the addition of a search feature that allows easy filtering. Future versions could extend this to enable filtering based on commonly needed attributes, like finding load-bearing building elements or materials.

Testing the viewer application showed that the current tutorial is too minimal and people feel lost in their first interactions with the AR scene. They didn't recognize flags as annotations and buttons as layers. A short introduction with added visuals would help first-time users and give them the required prior knowledge to find their way through the different UI elements. Furthermore, the ergonomics are also a big issue, revealed by the HARUS questionnaire results. One could alleviate this either by adding a hand-strap or grip to the device or adapting the input functionalities to allow for an operation where both hands can still hold the device.

The editor application needs an updated hierarchy as users had trouble understanding its behavior. Tooltips, labels, and section headings could allow users to explore the application's functionalities without opening a user manual. Additionally, participants suggested many small improvements and fixes during the sessions, which will probably be implemented in coming versions of the IFC Editor & AR Viewer framework.

CHAPTER 7

Summary and future work

During this master's thesis, an IFC editing and annotation framework consisting of a desktop editor and AR viewing application for Android tablet devices was developed utilizing the Unity game engine. This development was accomplished in cooperation with members of the Center of Digital Building Process (E234, Faculty of Civil Engineering, TU Wien), who provided requirements and feedback throughout the development process. Both applications' usability was evaluated in a mixed-method approach, combining and interpreting results from quantitative and qualitative methods. While quantitative methods resulted in good usability scores, qualitative methods revealed several usability problems that were discussed and possible solutions were proposed.

The initial question of whether students and lecturers of civil engineering could benefit from such a framework could not be answered fully, as a more long-term evaluation would be needed. This first evaluation of the prototypes indicated that participants from the relevant target groups found these applications easy to use and believed that it presents relevant information in an approachable way. Following this initial success, the applications will be updated according to the mentioned suggestions as part of a future research project while being evaluated further. Furthermore, future versions of the framework will aim to remove dependencies on closed-source components, replacing them with freely available open-source alternatives while also supporting newer standards like IFC4.

Appendix

IFC Viewer App Usability Questionnaire

Consent to participate in Usability Study

Background

The purpose of this study is to test the usability of the IFC AR Viewer Android Application Prototype. The gathered information will be used to further improve the application and its usability.

Procedure

After this short pre questionniare you will be tasked with working through a demo scenario which incorporates the features of the IFC AR Viewer Application which should take around 15 minutes. Afterwards a post study questionnaire needs to be filled out where we ask you to asses your experience of using the application and give additional feedback.

Confidentiality

The results of this questionnaire will be analyzed and discussed in the master's thesis "AR Training Application and Authoring Tool for 3D BIM Visualization", the participants of this user study will not be named.

Your Participation

There are no known risks and discomforts involved during this user study. You are free to stop the study at any time without giving a reason and your responses will be deleted. Please note that you will not be compensated for the participation in this study.

The Researchers

This user study is conducted as part of a master's thesis with the faculty of informatics (Institute of Visual Computing and Human-Centered Technology, Computer Graphics E193-02)

Author: Konstantin Höbart, B.Sc. k.hoebart@gmail.com
Advisor: Privatdoz. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistance:
Projektass. Dipl.-Ing. Dr.techn. Christian Schönauer
Assistant Prof. Dipl.-Ing. Dr.techn. Christian Schranz, M.Sc.
Projektass. Dipl.-Ing. Harald Urban

Demographics & Prior Knowlege

Age

☐ 18-24 ☐ 25-29 ☐ 30-34 ☐ 35-39 ☐ 40-44 ☐ 45-49 ☐ 50-54 ☐ 55-59 ☐ 60+

Gender

☐ male ☐ female ☐ other:

Do you have experience with Augmented Reality Applications?

	1	2	3	4	5	
No experience	☐	☐	☐	☐	☐	A lot of experience

Do you have experience with handling building information models?

	1	2	3	4	5	
No experience	☐	☐	☐	☐	☐	A lot of experience

Task Scenario

Starting Time:

The following tasks should be solved and their difficulty rated from 1 (Easy) to 5 (Hard).

1. Open the project "Attika-Demo-3".
2. Follow the instructions on the screen and lock the position.
3. Find & Select the building part with the title "Decke-001".
4. Find the width of the element "Decke-001".
5. Activate the layer "Rohdecke".
6. Find and Select the Annotation with the title "Abdichtung".
7. Open & Play the embedded video.
8. Close the video.

9. Now you can explore the other layers and annotations if you want, at the end add a comment with your name and a short text.

The Scenario is finished. Please note the time:

IFC Viewer App Post Questionnaire

Handheld Augmented Reality Usability Score Questionnaire

The following statements should be rated on a 7-point likert scale from 1 (Strongly disagree) to 7 (Strongly agree).

1. I think that interacting with this application requires a lot of body muscle effort.
2. I felt that using the application was comfortable for my arms and hands.
3. I found the device difficult to hold while operating the application.
4. I found it easy to input information through the application.
5. I felt that my arm or hand became tired after using the application.
6. I think the application is easy to control.
7. I felt that I was losing grip and dropping the device at some point.
8. I think the operation of this application is simple and uncomplicated.
9. I think that interacting with this application requires a lot of mental effort.
10. I thought the amount of information displayed on screen was appropriate.
11. I thought that the information displayed on screen was difficult to read.
12. I felt that the information display was responding fast enough.
13. I thought that the information displayed on screen was confusing.
14. I thought the words and symbols on screen were easy to read.
15. I felt that the display was flickering too much.
16. I thought that the information displayed on screen was consistent.

Feedback

How would you improve the IFC Viewer App? What did you like & didn't like?

The questionnaire is finished. Please note the time:

IFC Viewer Questionnaire Data

Participant	Age	Gender	AR experience	BIM experience
1	30-34	male	2	1
2	25-29	male	3	1
3	25-29	male	4	2
4	25-29	female	4	3
5	18-24	male	3	3
6	25-29	male	3	4
7	30-34	male	2	1

Table 1: IFC viewer questionnaire demographics & prior knowledge from 1 to 5

Participant	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9
1	1	1	2	1	1	1	1	1	1
2	1	1	2	1	1	1	1	1	1
3	1	1	2	1	2	1	1	1	1
4	1	2	3	4	2	1	1	1	1
5	1	1	3	1	1	1	1	1	1
6	1	2	1	2	1	2	1	1	1
7	1	2	4	1	1	1	1	1	1

Table 2: IFC viewer questionnaire task difficulties rated from 1 to 5

Participant	H 01	H 02	H 03	H 04	H 05	H 06	H 07	H 08
1	1	7	1	7	1	6	1	7
2	2	5	5	7	2	7	1	7
3	1	7	1	7	1	7	1	7
4	1	7	1	6	1	6	2	6
5	2	4	5	6	1	6	4	6
6	1	7	1	7	1	6	1	6
7	1	5	6	6	1	2	5	6

Table 3: HARUS questions 1-8

Participant	H 09	H 10	H 11	H 12	H 13	H 14	H 15	H 16
1	1	6	1	6	2	7	1	7
2	1	6	1	7	1	7	1	7
3	1	7	2	7	1	6	2	7
4	2	6	1	6	1	7	1	7
5	1	6	2	6	4	7	1	6
6	2	6	2	7	2	6	1	6
7	1	7	2	7	3	5	1	7

Table 4: HARUS questions 9-16

Participant	HARUS	Tasks Duration	Total Duration
1	95.8	04 min	11 min
2	90.6	07 min	11 min
3	96.9	06 min	14 min
4	92.7	10 min	17 min
5	78.1	06 min	12 min
6	91.7	07 min	17 min
7	76.0	12 min	19 min

Table 5: HARUS (rounded) and questionnaire durations

IFC Editor Usability Questionnaire

Consent to participate in Usability Study

Background

The purpose of this study is to test the usability of the IFC Editor Desktop Application Prototype. The gathered information will be used to further improve the application and its usability.

Procedure

After this short pre questionniare you will be tasked with working through a demo scenario which incorporates the features of the IFC Editor Application which should take around 30 minutes. Afterwards a post study questionnaire needs to be filled out where we ask you to asses your experience of using the application and give additional feedback.

Confidentiality

The results of this questionnaire will be analyzed and discussed in the master's thesis "AR Training Application and Authoring Tool for 3D BIM Visualization", the participants of this user study will not be named.

Your Participation

There are no known risks and discomforts involved during this user study. You are free to stop the study at any time without giving a reason and your responses will be deleted. Please note that you will not be compensated for the participation in this study.

The Researchers

This user study is conducted as part of a master's thesis with the faculty of informatics (Institute of Visual Computing and Human-Centered Technology, Computer Graphics E193-02)

Author: Konstantin Höbart, B.Sc. k.hoebart@gmail.com
Advisor: Privatdoz. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistance:
Projektass. Dipl.-Ing. Dr.techn. Christian Schönauer
Assistant Prof. Dipl.-Ing. Dr.techn. Christian Schranz, M.Sc.
Projektass. Dipl.-Ing. Harald Urban

Demographics & Prior Knowlege

Age

☐ 18-24 ☐ 25-29 ☐ 30-34 ☐ 35-39 ☐ 40-44 ☐ 45-49 ☐ 50-54 ☐ 55-59 ☐ 60+

Gender

☐ male ☐ female ☐ other:

Do you have experience with handling building information models?

	1	2	3	4	5	
No experience	☐	☐	☐	☐	☐	A lot of experience

Task Scenario

Starting Time:

The following tasks should be solved and their difficulty rated from 1 (Easy) to 5 (Hard).

1. Open the IFC file attika.ifc from the Desktop.
2. Find and select the building element with the title "Decke-001".
3. Delete the selected building element "Decke-001".
4. Find a way to select the group of building elements "Wand-001".
5. Disable the attributes with the title "ArchiCADProperties" on the building element "Wand-001".
6. Add the comment "test comment" to the building element "Wand-001".
7. Create two new Buttons called "Button 1" and "Button 2".
8. Create a new Layer called "Layer 1" and assign the "Button 1" to it.
9. Create a new Layer called "Layer 2" assign the "Button 1" and "Button 2" to it.
10. Add the building element Wand-001 to the new "Layer 1".
11. Add the building element "Wand-002" to the new "Layer 2".
12. Check if the Buttons control the visibility of the assigned Elements.
13. Move the whole model by 1 along the x-axis.
14. Add a new Annotation with the title "Test Title" at a random spot in the model.
15. Add the image "test.jpg" from the Desktop to the new Annotation.
16. Change the appearance of the Annotation to a ball.
17. Export the project as "Test Project".

The Scenario is finished. Please note the time:

Post Study Questionnaire

System Usability Score Questionnaire

The following statements should be rated on a 5-point Likert scale from 1 (Strongly disagree) to 5 (Strongly agree).

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very awkward to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Feedback

How would you improve the IFC Editor? What did you like & didn't like?

The questionnaire is finished. Please note the time:

IFC Editor Questionnaire Data

Participant	Age	Gender	BIM experience
2	25-29	male	1
3	25-29	male	2
4	25-29	female	3
5	18-24	male	3
6	25-29	male	4
7	30-34	male	1

Table 6: Demographics and prior knowledge

Participant	T01	T02	T03	T04	T05	T06	T07	T08
1	1	1	3	1	4	1	1	1
2	1	1	2	1	1	1	1	1
3	1	1	1	1	5	1	1	1
4	1	1	1	2	2	1	2	1
5	1	1	3	2	3	2	2	1
6	1	2	1	2	2	2	2	1
7	1	1	2	1	1	1	1	1

Table 7: IFC editor task 1-8 difficulties

Participant	T09	T10	T11	T12	T13	T14	T15	T16	T17
1	1	1	1	1	1	2	1	1	2
2	1	3	1	3	1	1	1	1	1
3	1	2	1	1	1	1	3	1	2
4	1	2	1	1	2	1	2	1	2
5	1	2	1	2	4	2	1	1	1
6	1	2	1	2	2	2	1	1	1
7	1	2	1	1	2	3	1	1	1

Table 8: IFC editor task 9-17 difficulties

Participant	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10
1	4	1	5	2	5	1	5	3	4	4
2	5	1	5	1	4	1	5	1	5	2
3	5	1	5	1	5	1	5	1	4	1
4	5	1	4	1	5	1	4	1	5	1
5	5	1	4	1	4	1	4	1	4	1
6	5	1	4	2	4	2	5	4	4	1
7	5	1	5	1	5	1	5	1	5	1

Table 9: SUS questionnaire ratings

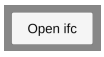
Participant	SUS	Task Duration	Total Duration
1	80	15 min	27 min
2	95	09 min	11 min
3	97.5	27 min	37 min
4	95	15 min	20 min
5	90	08 min	16 min
6	80	19 min	31 min
7	100	14 min	22 min

Table 10: SUS results and durations

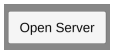
Quick-Start Guides

IFC Editor Quick-Start Guide

1 Starting a new project



Open an IFC File from your computer.



Open an existing project from the server.

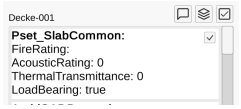
2 Edit the project

Select Tool

Create Layers & assign them to buttons.

Hide types of IFC properties for all objects.

Browse through the building elements and select them.

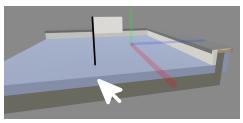


Disable IFC Elements, Add comments, Put the object on a Layer or bulk disable all its IFC Elements.

Move, Rotate & Scale Tool

Adjust the Position, Rotation and Scale of the selected object

Annotation Tool



First, place an annotation in the scene. Then enter a title, description or media path in the sidebar. You can also change the appearance to a ball.

3 Export the project to the Server

Open the export dialog, enter a project name and export the project

IFC AR Quick-Start Guide

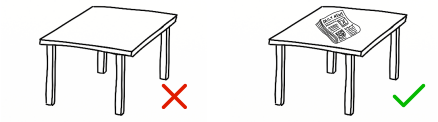
1 Choose a viewing mode

AR Core — Primary AR Viewing Mode.

Vuforia — Fallback mode. Use when AR Core is not available.

2 Open a project

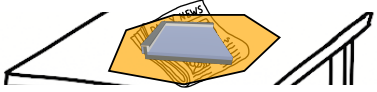
0. Prepare a work surface with a textured surface.



1. Select a project in the FTP browser.

2. Scan your environment until feature points appear.

3. When a complete surface is detected drag your finger on it to position the building model



4. Rotate the model with two fingers and Lock the Position

Lock Position

3 Explore

Layers

Layer Button

Layer Button

Layer Button

Annotations

Title

Selection Details

IFC Property 1
IFC Attribute 1
IFC Attribute 2

IFC Property 2
IFC Attribute 3
IFC Attribute 4

Bibliography

[Ama] Amazon. *Amazon.com: Galaxy Tab S4 Case*. URL: https://www.amazon.com/dp/B07K828V8R/ref=twister_B07KCW1ZT5 (visited on 10/15/2020).

[AMA16] Ayer Steven K., Messner John I., and Anumba Chimay J. „Augmented Reality Gaming in Sustainable Design Education". In: *Journal of Architectural Engineering* 22.1 (Mar. 2016), p. 04015012. DOI: 10.1061/(ASCE)AE.1943-5568.0000195. URL: https://ascelibrary.org/doi/full/10.1061/%28ASCE%29AE.1943-5568.0000195.

[App20] Apple Inc. *Apple Developer Documentation - ARKit*. 2020. URL: https://developer.apple.com/documentation/arkit (visited on 10/15/2020).

[Aut20a] Autodesk Inc. *3ds Max | 3D Modeling, Animation & Rendering Software | Autodesk*. en-US. 2020. URL: https://www.autodesk.com/products/3ds-max/overview (visited on 10/15/2020).

[Aut20b] Autodesk, Inc. *Revit | BIM Software | Autodesk*. en-US. 2020. URL: https://www.autodesk.com/products/revit/overview (visited on 10/15/2020).

[Azu+01] R. Azuma et al. „Recent advances in augmented reality". In: *IEEE Computer Graphics and Applications* 21.6 (2001), pp. 34–47.

[Azu19] Ronald T. Azuma. „The road to ubiquitous consumer augmented reality systems". In: *Human Behavior and Emerging Technologies* 1.1 (Jan. 2019). Publisher: John Wiley & Sons, Ltd, pp. 26–32. DOI: 10.1002/hbe2.113. URL: https://doi.org/10.1002/hbe2.113 (visited on 10/04/2020).

[Azu97] Ronald T. Azuma. „A Survey of Augmented Reality". In: *Presence: Teleoperators and Virtual Environments* 6.4 (1997). _eprint: https://doi.org/10.1162/pres.1997.6.4.355, pp. 355–385. DOI: 10.1162/pres.1997.6.4.355. URL: https://doi.org/10.1162/pres.1997.6.4.355.

[BBV13] David Bryde, Martí Broquetas, and Jürgen Marc Volm. „The project benefits of Building Information Modelling (BIM)“. In: *International Journal of Project Management* 31.7 (2013), pp. 971–980. ISSN: 0263-7863. DOI: https://doi.org/10.1016/j.ijproman.2012.12.001. URL: http://www.sciencedirect.com/science/article/pii/S0263786312001779.

[Ben20] Bentley Systems. *Bentley MicroStation*. 2020. URL: https://www.bentley.com/en/products/brands/microstation (visited on 10/15/2020).

[BKM08] Aaron Bangor, Philip T. Kortum, and James T. Miller. „An Empirical Evaluation of the System Usability Scale“. In: *International Journal of Human–Computer Interaction* 24.6 (July 2008), pp. 574–594. ISSN: 1044-7318. DOI: 10.1080/10447310802205776. URL: https://doi.org/10.1080/10447310802205776.

[Boe18] Stefan Boeykens. *Getting BIM data into Unity (Part 9 - using IfcConvert)*. Sept. 2018. URL: http://cad-3d.blogspot.com/2018/09/getting-bim-data-into-unity-part-9.html (visited on 10/15/2020).

[Bro96] John Brooke. „SUS-A quick and dirty usability scale (in "Usability Evaluation in Industry", PW Jordan, B Thomas, I McLelland, BA Weerdmeester (eds))“. In: *Usability evaluation in industry (P. W. Jordan, B. Thomas, B. A. Weerdmeester, & A. L. McClelland (Eds.))* June 1996. ISBN: 9780748404605.

[bui] buildingSMART. *IfcSlab Reference*. Reference. URL: https://standards.buildingsmart.org/IFC/RELEASE/IFC2x3/FINAL/HTML/ifcsharedbldgelements/lexical/ifcslab.htm (visited on 10/15/2020).

[bui20a] buildingSMART. *IFC Release Notes*. en-GB. Apr. 2020. URL: https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/ifc-release-notes/ (visited on 10/15/2020).

[bui20b] buildingSMART. *Industry Foundation Classes (IFC)*. en-GB. 2020. URL: https://technical.buildingsmart.org/standards/ifc/ (visited on 10/15/2020).

[bui20c] buildingSMART. *Ten principles for a future IFC*. en. Mar. 2020. URL: https://github.com/buildingSMART/NextGen-IFC (visited on 10/15/2020).

[Cro00] Douglas Crockford. *Introducing JSON*. 2000. URL: https://www.json.org/json-en.html (visited on 10/15/2020).

[Dav92] A. M. Davis. „Operational prototyping: a new development approach“. In: *IEEE Software* 9.5 (1992), pp. 70–78.

[Dic+17] Paul E. Dickson et al. „An Experience-based Comparison of Unity and Unreal for a Stand-alone 3D Game Development Course“. In: *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. ITiCSE '17. Bologna, Italy: Association for Computing Machinery, June 2017, pp. 70–75. ISBN: 9781450347044. DOI: 10.1145/3059009.3059013. URL: https://doi.org/10.1145/3059009.3059013.

[Din+17] Fábio Matoseiro Dinis et al. „Virtual and augmented reality game-based applications to civil engineering education“. In: *2017 IEEE Global Engineering Education Conference (EDUCON)*. ISSN: 2165-9567. Apr. 2017, pp. 1683–1688. DOI: 10.1109/EDUCON.2017.7943075.

[DP13] Ben Dalton and Maxwell Parfitt. „Immersive visualization of building information models“. In: *Design innovation research center working paper* 6.1.0 (2013).

[DS09] Phillip S. Dunston and Do Hyoung Shin. „Key Areas And Issues For Augmented Reality Applications On Construction Sites“. en. In: *Mixed Reality In Architecture, Design And Construction*. Ed. by Xiangyu Wang and Marc Aurel Schnabel. Dordrecht: Springer Netherlands, 2009, pp. 157–170. ISBN: 9781402090882. DOI: 10.1007/978-1-4020-9088-2_10. URL: https://doi.org/10.1007/978-1-4020-9088-2_10.

[Du+18] Jing Du et al. „Zero latency: Real-time synchronization of BIM data in virtual reality for collaborative decision-making“. en. In: *Automation in Construction* 85 (Jan. 2018), pp. 51–64. ISSN: 0926-5805. DOI: 10.1016/j.autcon.2017.10.009. URL: http://www.sciencedirect.com/science/article/pii/S0926580517309172.

[Eas+11] Charles M Eastman et al. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons, 2011.

[Gha+17] Ali Ghaffarianhoseini et al. „Building Information Modelling (BIM) uptake: Clear benefits, understanding its implementation, risks and challenges“. In: *Renewable and Sustainable Energy Reviews* 75 (2017), pp. 1046–1053. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2016.11.083. URL: http://www.sciencedirect.com/science/article/pii/S1364032116308413.

[Goo20] Google LLC. *Google Developers Blog - ARCore Updates*. en. June 2020. URL: https://developers.googleblog.com/search/label/ARCore (visited on 10/15/2020).

[Gra20] Graphisoft SE. *Archicad*. en-US. 2020. URL: https://graphisoft.com/solutions/products/archicad (visited on 10/15/2020).

[HFV01] M. A Hassanain, T. M Froese, and D. J Vanier. „Development of a maintenance management model based on IAI standards“. en. In: *Artificial Intelligence in Engineering* 15.2 (Apr. 2001), pp. 177–193. ISSN: 0954-1810. DOI: 10.1016/S0954-1810(01)00015-2. URL: http://www.sciencedirect.com/science/article/pii/S0954181001000152.

[Ifc20] IfcOpenShell. *IfcOpenShell - the open source ifc toolkit and geometry engine*. original-date: 2015-08-10T09:03:14Z. Dec. 2020. URL: http://ifcopenshell.org/ (visited on 10/15/2020).

[ISO04] ISO/TC 184/SC 4. *The EXPRESS language reference*. en. Nov. 2004. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/03/80/38047.html (visited on 10/15/2020).

[ISO16] ISO/TC 184/SC 4. *ISO 10303-21:2016 Clear text encoding of the exchange structure*. en. Mar. 2016. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/31/63141.html (visited on 10/15/2020).

[ISO18a] ISO/TC 59/SC 13. *ISO 16739-1:2018*. en. Nov. 2018. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/03/70303.html (visited on 10/15/2020).

[ISO18b] ISO/TC 59/SC 13. *ISO 19650-1:2018*. en. Dec. 2018. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/80/68078.html (visited on 10/15/2020).

[jso19] json-schema-org. *JSON Schema*. en. Sept. 2019. URL: https://json-schema.org/ (visited on 10/15/2020).

[LBČ17] Steve Lockley, Claudio Benghi, and Martin Černý. „Xbim.Essentials: a library for interoperable building information applications“. en. In: *Journal of Open Source Software* 2.20 (Dec. 2017), p. 473. ISSN: 2475-9066. DOI: 10.21105/joss.00473. URL: https://joss.theoj.org/papers/10.21105/joss.00473.

[LS09a] James R. Lewis and Jeff Sauro. „The Factor Structure of the System Usability Scale“. en. In: *Human Centered Design*. Ed. by Masaaki Kurosu. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 94–103. ISBN: 9783642028069. DOI: 10.1007/978-3-642-02806-9_12.

[LS09b] James R. Lewis and Jeff Sauro. „The Factor Structure of the System Usability Scale“. en. In: *Human Centered Design*. Ed. by Masaaki Kurosu. Vol. 5619. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 94–103. ISBN: 9783642028052. DOI: 10.1007/978-3-642-02806-9_12. URL: http://link.springer.com/10.1007/978-3-642-02806-9_12.

[LS17] James Jim R Lewis and Jeff Sauro. „Revisiting the Factor Structure of the System Usability Scale.“ In: *Journal of Usability Studies* 12.4 (2017).

[LS18] James R Lewis and Jeff Sauro. „Item benchmarks for the system usability scale.“ In: *Journal of Usability Studies* 13.3 (2018).

[MAX20] MAXON Computer. *Cinema 4D*. en. 2020. URL: https://www.maxon.net/en-us/products/cinema-4d/overview/ (visited on 10/15/2020).

[Mil+95] Paul Milgram et al. „Augmented reality: a class of displays on the reality-virtuality continuum“. In: *Telemanipulator and Telepresence Technologies*. Ed. by Hari Das. Vol. 2351. Backup Publisher: International Society for Optics and Photonics. SPIE, 1995, pp. 282–292. DOI: 10.1117/12.197321. URL: https://doi.org/10.1117/12.197321.

[MTD14] Sebastjan Meža, Žiga Turk, and Matevž Dolenc. „Component based engineering of a mobile BIM-based augmented reality system“. en. In: *Automation in Construction* 42 (2014), pp. 1–12. ISSN: 09265805. DOI: 10.1016/j.autcon.2014.02.011. URL: https://linkinghub.elsevier.com/retrieve/pii/S0926580514000363.

[Nan+18] Anirudh Nandavar et al. „Interactive Virtual Reality Tool for BIM Based on IFC - Development of OpenBIM and Game Engine Based Layout Planning Tool - A Novel Concept to Integrate BIM and VR with Bi-Directional Data Exchange“. In: *T. Fukuda, W. Huang, P. Janssen, K. Crolla, S. Alhadidi (eds.), Learning, Adapting and Prototyping - Proceedings of the 23rd CAADRIA Conference - Volume 1, Tsinghua University, Beijing, China, 17-19 May 2018, pp. 453-462*. CUMINCAD, 2018. URL: http://papers.cumincad.org/cgi-bin/works/Show?caadria2018_057.

[NLZ17] Esha Nerurkar, Simon Lynen, and Sheng Zhao. „System and method for concurrent odometry and mapping“. en. US20170336511A1. May 2017. URL: https://patents.google.com/patent/US20170336511A1/en.

[Noa+20] Francesca Noardo et al. „Reference study of IFC software support: the GeoBIM benchmark 2019 – Part I“. In: *arXiv:2007.10951 [cs]* (July 2020). arXiv: 2007.10951. URL: http://arxiv.org/abs/2007.10951.

[NS07] David Nistér and Henrik Stewénius. „A Minimal Solution to the Generalised 3-Point Pose Problem“. en. In: *Journal of Mathematical Imaging and Vision* 27.1 (Jan. 2007), pp. 67–79. ISSN: 1573-7683. DOI: 10.1007/s10851-006-0450-y. URL: https://doi.org/10.1007/s10851-006-0450-y.

[Nur+09] Nurzhan Nurseitov et al. „Comparison of JSON and XML data interchange formats: a case study.“ In: *Caine* 9 (2009), pp. 157–162.

[NW20] Pawel Nowacki and Marek Woda. „Capabilities of ARCore and ARKit Platforms for AR/VR Applications“. In: *Engineering in Dependability of Computer Systems and Networks*. Ed. by Wojciech Zamojski et al. Cham: Springer International Publishing, 2020, pp. 358–370. ISBN: 978-3-030-19501-4.

[Pet15] Josh Peterson. *Unity Blog - An introduction to IL2CPP internals*. en-US. Blog. May 2015. URL: https://blogs.unity3d.com/2015/05/06/an-introduction-to-ilcpp-internals/ (visited on 10/15/2020).

[Pol+16] Jarkko Polvi et al. „SlidAR: A 3D positioning method for SLAM-based handheld augmented reality". en. In: *Computers & Graphics* 55 (Apr. 2016), pp. 33–43. ISSN: 0097-8493. DOI: 10.1016/j.cag.2015.10.013. URL: http://www.sciencedirect.com/science/article/pii/S0097849315001806.

[PPP13] S. Camille Peres, Tri Pham, and Ronald Phillips. „Validation of the System Usability Scale (SUS): SUS in the Wild". en. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 57.1 (Sept. 2013), pp. 192–196. ISSN: 1541-9312. DOI: 10.1177/1541931213571043. URL: http://journals.sagepub.com/doi/10.1177/1541931213571043.

[Sam+10] Alcínia Z. Sampaio et al. „3D and VR models in Civil Engineering education: Construction, rehabilitation and maintenance". en. In: *Automation in Construction* 19.7 (Nov. 2010), pp. 819–828. ISSN: 0926-5805. DOI: 10.1016/j.autcon.2010.05.006. URL: http://www.sciencedirect.com/science/article/pii/S092658051000083X.

[San+14] Marc Ericson C. Santos et al. „A usability scale for handheld augmented reality". In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. VRST '14. Edinburgh, Scotland: Association for Computing Machinery, Nov. 2014, pp. 167–176. ISBN: 9781450332538. DOI: 10.1145/2671015.2671019. URL: https://doi.org/10.1145/2671015.2671019.

[SDT12] Jim Steel, Robin Drogemuller, and Bianca Toth. „Model interoperability in building information modelling". en. In: *Software & Systems Modeling* 11.1 (Feb. 2012), pp. 99–109. ISSN: 1619-1374. DOI: 10.1007/s10270-010-0178-4. URL: https://doi.org/10.1007/s10270-010-0178-4.

[TS04] Thomas S Tullis and Jacqueline N Stetson. „A comparison of questionnaires for assessing website usability". In: *Usability professional association conference*. Vol. 1. Minneapolis, USA, 2004.

[TUI17] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. „Visual SLAM algorithms: a survey from 2010 to 2016". en. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (June 2017), p. 16. ISSN: 1882-6695. DOI: 10.1186/s41074-017-0027-2. URL: https://doi.org/10.1186/s41074-017-0027-2.

[Tur+17] Yelda Turkan et al. „Mobile augmented reality for teaching structural analysis". en. In: *Advanced Engineering Informatics* 34 (Oct. 2017), pp. 90–100. ISSN: 1474-0346. DOI: 10.1016/j.aei.2017.09.005. URL: http://www.sciencedirect.com/science/article/pii/S1474034616302944.

[Vir92] Robert A. Virzi. „Refining the Test Phase of Usability Evaluation: How Many Subjects Is Enough?“ en. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 34.4 (Aug. 1992), pp. 457–468. ISSN: 0018-7208, 1547-8181. DOI: 10.1177/001872089203400407. URL: http://journals.sagepub.com/doi/10.1177/001872089203400407.

[WS03] Daniel Wagner and Dieter Schmalstieg. „First steps towards handheld augmented reality“. In: *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.* IEEE, 2003, pp. 127–135.

[Zho+19] Xiaoping Zhou et al. „Cross-platform online visualization system for open BIM based on WebGL“. en. In: *Multimedia Tools and Applications* 78.20 (Oct. 2019), pp. 28575–28590. ISSN: 1573-7721. DOI: 10.1007/s11042-018-5820-0. URL: https://doi.org/10.1007/s11042-018-5820-0.