Designing Another Open Source BIM Environment

A collection of ideas for BIM software with focus on ease of use, from a small scale (historic) architecture background, applied to existing open source 3D Modeling software

M2.2 , chair of design computation, RWTH Aachen University
Julian Hüther
julian.huether@rwth-aachen.de

February 10, 2026

Preface

Despite the format of this document suggesting otherwise, this is not a strictly scientific project. Instead, it is the documentation of a project that - quite naively - attempts to build a practical bridge between highly digitized and specialized BIM architects, developers, consultants, academics and thinkers on one side and "average", "low-tech", generalist architects, architecture students and architectural heritage preservers on the other. This work can arguably be considered to be below the scientific standards of this university and chair, though I still think it is work that can be beneficial to my field. I want to thank the chair of design computation for letting me work on this - what is essentially a software development project approached by someone who is not a trained software developer and inspired by lots of anecdotal evidence about how architects use software tools - and giving me the opportunity to think and learn about these things.

Table of Contents

1 Introduction

This document is a documentation of a software extension that was worked on in the past term with the working title "Goliath". It was written for Blender (Blender Foundation, 2026a), an open source vertex/mesh based 3D modeling software. The document consists of this section, which aims to explain why this project was approached in the way it was and refers to some potentially interesting literature on the broader topic. The following section contains the actual documentation of the extension. Finally, in the last section, the project is briefly critically reviewed. It is assumed that readers of this document are somewhat familiar with BIM, CAD and 3D modeling software, including Blender. For readers less familiar with this, a glossary (B) was attached to this document.

1.1 Background and Motivation

The motivation for this project arose from about 5 years of work experience in one of Germany's many small, 1-4 employee architecture firms which account for 73% of architecture firms here (BAK, 2024) working almost exclusively on small buildings (average-sized single family homes to houses with up to 3 flats). In this firm, proprietary planning software is used to create 2D drawings, transporting all non-geometry information on those drawings as texts/notes/comments, in emails, or on the phone. While the reasons for this have not or only partly been scientifically determined, here are 3 hypotheses/assumptions based on experience and anecdotal evidence:

1. High flexibility in the face of changing circumstances - sometimes an important piece of information is found out in the middle of tearing down part of a building. Other times clients alter their decisions right before the concerned part is scheduled to be built. The quickest way to communicate these changes is a phone call or a few pen strokes on a piece of paper.
2. 2D is easy/intuitive - it is always easier to 2D-draw or photograph highly individualized building elements than to make 3D models of them.
3. This is the way it has always been - changing the workflow and/or learning new things requires time and effort.

The previous term - technically the first part of this project - was spent using Bonsai (Bonsai, 2026) making BIMs of two (small) historic buildings (a chapel and a windmill) based on point clouds (Hüther, 2025). During this process, using Bonsai proved challenging for these highly individual architectures. Especially the high complexity in the UI and the parametric modeling functions which are at the core of the software's workflow were experienced as lacking flexibility and intuitiveness. The cause of this is multi-faceted and surely includes lack of experience using the software, but the experience lead to the hypothesis that one central element

is Bonsai being unsuited for these kinds of projects - likely because it was developed for the kinds of projects that are still traditionally associated with a BIM workflow.[1]

[1] This is not intended to be a jab at Bonsai. The Bonsai project deserves the recognition it has gotten for being a promising, free (not in the monetary sense) alternative to proprietary BIM software.

As a consequence, the "Goliath" project was conceived to fill this gap. Blender was chosen as the base for multiple reasons, including: it is already well documented and established (for instance, using it is being taught at RWTH Aachen University); it is open source and has an advanced and well documented application-programmer interface (API) (Blender Foundation, 2026b); it is the base for arguably the most established open source BIM platform Bonsai so that ideas and functions developed in "Goliath" could theoretically be easily transferred to Bonsai; it uses a vertex or mesh based modeling workflow which can be beneficial when working with highly specialized and individualized elements.

1.2 Project Goals

In summary, the purpose of this work was:

1. To learn how to modify tools for one's specific needs, on a theoretical level by outlining concepts and on a practical level by actually making modifications (in this case to Blender).
2. To propose changes in how architectural software works, from a small scale and "Bauen im Bestand" (B) background.
3. To make a usable prototype of a useful tool.

1.3 Related Literature

A characterization of the role of "Bauen im Bestand" in Germany and the usage of BIM in it is given by Petzold and Reichenberg (2021). Ewert (2024) explores how BIM is used and perceived by stakeholders of architectural projects. Liu et al. (2022) gives an overview over technologies and tools that are applied in HBIM. Brookes (2017) presents two case studies for HBIM along with some recommendations, Bruno and Roncella (2019) present two further HBIM case studies. Lombardi and Rizzi (2024) demonstrate how Blender can be modified and used for archaeological modeling purposes, proposing a way of dealing with different information/time layers. Törmä (2014) discusses BIM interoperability and the idea of type and instance level interoperability. Finally, loosely related to the motivation of this project, Akbar et al. (2023) reports on a course that aimed to teach architecture students how to make their own software tools in an effort to "democratize" design tools.

2 Modifications and Additions

In this section, all modifications and additions made to the software are compiled along with ideas on how to (practically) advance them. Some of

them are in and for themselves very specific to Blender, others are more general. However, all of them should ideally be understood on a general, conceptual level if some background knowledge about some of the programs core concepts is given. Readers unfamiliar with these concepts are recommended to consult the glossary (B).

2.1 Changes to Native Blender Operations

2.1.1 Object and Mesh Logic

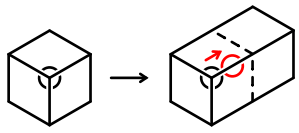

Figure 1: Margin figure caption.

To avoid confusion, the general idea was to not have two different parameters that work on the same thing (geometry or other). Object Scale and Rotation are automatically applied in Edit Mode and Object Origins are automatically moved to center of geometry when the Mesh is edited. This is to ensure that all edit mode lengths are real lengths, that there is only one method of rotating geometry and to avoid confusion with offset object origins. (Figures 1 & 2)

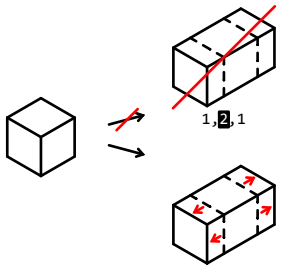

Figure 2: Margin figure caption.

Alternatives

- All object parameters are always 0/0/0 etc. and all geometry changes are always applied only in Edit Mode, that is on the Mesh level instead of the Object level. This might be nice for Meshes and Curves, but would obviously not work with other Object Types.

Ideas

- Rotation could be made to set automatically according to geometry in a similar way Location does. Scale might be problematic, since changes to it always clash with the dimensions of the actual mesh.

2.1.2 Object Separation

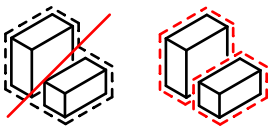

Figure 3: Margin figure caption.

When done editing Meshes (on leaving Edit Mode), any unconnected geometry is separated into its own Object. This is done to force users into having a dedicated Object and thereby Properties for any separate piece of geometry. (Figure 3)

Ideas

- The new, separated Objects are supposed to be grouped automatically on separation, however this is not yet implemented.

Issues

- The necessity of this concept should be questioned. There are probably other ways of achieving a similar effect.

2.1.3 Hierarchy Handling

When working on Objects with Child Objects attached to it, using Duplicate duplicates the whole Parent/Child structure; using Remove removes the whole Parent/Child Structure. (Figure 4)

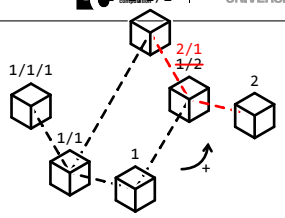

Figure 4: Margin figure caption.

2.1.4 Face Editing

When editing a Mesh in Face mode, using Grab automatically uses the normal axis of the face. (Figure 5)

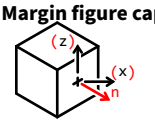

Figure 5: Margin figure caption.

Ideas

- Think about whether default settings like this may be useful for Edge or Vertex mode as well.

2.2 Added Functions

2.2.1 Element Types

Every Object has an Element Type. The integrated types were chosen based on IfcElement types, but several types were omitted to not overwhelm and slow down users, focusing on typically relevant types for smaller scale architecture. The selection is close to the IfcBuildingElement types, though slightly more simplified (Figure 6)

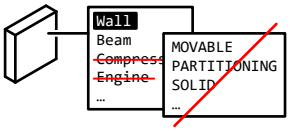

Figure 6: Margin figure caption.

Ideas

- Evaluate the choice of types.

2.2.2 Primitive Type Detection

An Object's type is automatically determined based on its geometry, after editing said geometry. This automatic determination can be overridden. (Figure 7)

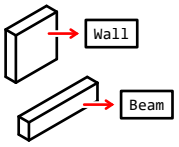

Figure 7: Margin figure caption.

Ideas

- The rules/heuristics of determining an object type can and probably should become more robust, while not increasing complexity to an unreasonable level. It does not need to be perfect, the override function exists for a reason, failure is expected.
- The rules could become customizeable.

2.2.3 Storeys

Objects of Type Slab can be made to mark Storeys. Storey height is then set to the topmost (Global Z) Face of the Mesh. Every Object is put in one of the Storeys present in the Project based on the height. The goal was to avoid having extra Storey Objects unless absolutely necessary. (Figure 8)

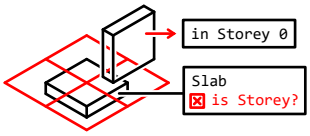

Figure 8: Margin figure caption.

Ideas

- Storeys need properties like Name, Offset etc.
- Objects can only be in one Storey; should it be possible for them to be in multiple, in none?
- Non-Slab Storeys should be possible (that is similarly set Storeys for Objects that are not Slabs *and* completely Object independent Storeys for edge cases)

2.2.4 (Custom) Property Handling

Similarly to native Blender, (Mesh) Objects and Materials can have custom properties, with respective property types. Additionally, each property has a certainty value attached to it, informing about the source and reliability of the property. These Properties represent what would be in Psets in traditional BIM software, which is why "Goliath" automatically suggests the Pset*Type*Common property names when adding a property. The goal was to remove the "hurdle" of creating and managing Psets, again simplifying and accelerating the working process. (Figure 9)

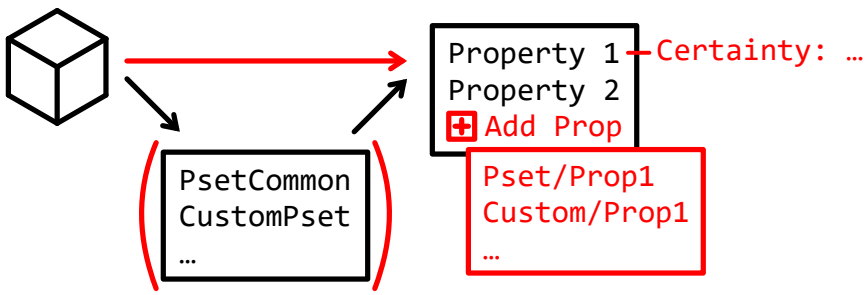

Figure 9: Margin figure caption.

Ideas

- Psets can of course be useful and they should be integrated. But handling them should remain as simple and quick as described here, as far as possible.[2]
- Properties could have an Element Type setting that determines which Objects can/should have the property and which should not.

Issues

- When using a Pset*Type*Common property, the property type needs to be set accordingly.
- Previously added custom properties are not included in the name search pool.

[2] Psets are (presumably) especially useful in high complexity projects, less so in small projects. Maybe having them could be optional.

2.2.5 Oriented Bounding Boxes

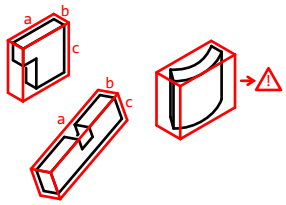

Figure 10: Margin figure caption.

Every Mesh Object is automatically put in an Object Oriented Bounding Box. This Bounding Box is used to determine type and dimensions/proportions. (The current algorithm for this is simple: it groups faces according to orientation, the largest area sum is used to draw the base of the box and it is extruded until it encloses the furthest vertex. See code for more.) This approach of course does not work with all geometries. When the difference between the Bounding Box and the Mesh is too great, calculating the quantities will generate a warning in the Object Panel (2.2.7). (Figure 10)

Ideas

- The Bounding Box algorithm can be improved and made more robust. Unlike with Element Type Detection, there is no override for this, so it is desirable to have a near perfectly robust algorithm. The only limiting factor is software performance.

Issues

- The idea of dimensions of course only works for at least roughly cuboid shapes. There will always be objects that are incompatible to this approach.

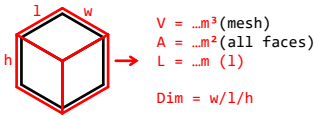

Figure 11: Margin figure caption.

2.2.6 Quantities and Dimensions

Using the Mesh and aforementioned Bounding Box geometry, four quantities can be calculated: length; area (of all Mesh faces); volume; Object count. As mentioned, dimensions are also determined from the Bounding Box. (Figure 11)

Ideas

- Different configurations for quantities could be added, similar to Quantity Takeoff. This is probably especially interesting when calculating areas - for this it would be nice to see which Faces are counted and which are not. This should however never replace current Area calculation, since it can already quite useful when just calculating the total surface area.

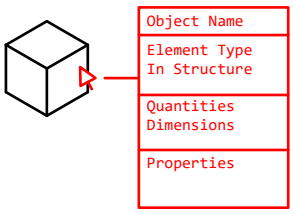

Figure 12: Margin figure caption.

2.2.7 Object Panel

This Panel is - like other aspects of this project - inspired by existing architectural/CAD software. It delivers all aforementioned information and properties to the user the moment an Object is selected. Quantities are updated live and summed up across the selection. Properties can be named, added and removed with one click respectively. The whole Panel is intended to be visible without scrolling. The aim is for users to be able to understand everything "at first glance". Descriptions are displayed when hovering, just as with Blender's native properties. The Panel

changes with the type of Object selected (e.g. Group Objects, see Grouping). (Figure 12)

Ideas

- For the object count, show how many of each type.

Issues

- All (interactive) panel elements need descriptions.

2.2.8 Object Notes

On Object selection, a note Text is generated for the Object, if there is not yet one. It is automatically displayed in the bottom right corner of the window in the "Goliath" workspace. In it, users can write plain text about anything concerning the object. (Figure 13)

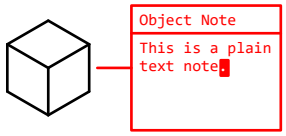

Figure 13: Margin figure caption.

Ideas

- It could be useful to have Notes that are not only linked to Objects.
- Ideally, Notes could contain links (like a hyperlink, to other Objects, file paths etc.).

2.2.9 Grouping

Objects can be grouped using a shortcut, generating a Group Object (Empty) that all grouped Objects are Parented to. A Group Object is useful because: it can be moved, controlling the whole Group's transformation; it can have its own Properties. Objects can be taken out of the Group or the Group can be completely disbanded. (Figure 14)

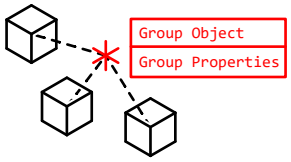

Figure 14: Margin figure caption.

Ideas

- Think about how Group Editing works; should it be possible to select grouped Objects individually, or should the group serve as a barrier, getting its own "Edit Mode"? (The latter is, of course, how most design software handles groups).
- Find a good way to integrate Group properties into an Object's properties.

2.2.10 Construction Lines

Objects can be grouped using a shortcut, generating a Group Object (Empty) that all grouped Objects are Parented to. A Group Object is useful because: it can be moved, controlling the whole Group's transformation; it can have its own Properties. Objects can be taken out of the Group or the Group can be completely disbanded. (Figure 15)

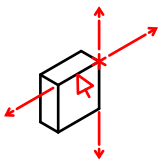

Figure 15: Margin figure caption.

Ideas

- This concept could and should be expanded to Faces and Vertices as well, generating Face/Edge intersections, or a closest point on an Edge to a Vertex, to mention two examples.
- Evaluate automatic orientation.

2.2.11 Point Clouds

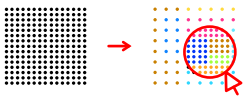

Figure 16: Margin figure caption.

Point Clouds can be processed after import. In addition to coloring the points, the Point Cloud is filtered as follows: since loading the complete Point Cloud can lead to performance issues, points are removed by distance, reducing the number of points in exchange for less accuracy. The full resolution Point Cloud is only loaded inside of a "Magnifier" sphere that can be moved and resized by the user. This way full accuracy is present where it is needed while increasing performance. Point Clouds are non-selectable and can be moved with a Parent Object, similar to groups. (Figure 16)

Ideas

- It should be possible to use multiple Magnifiers.
- It should eventually be possible to use Magnifiers that are non-spherical, like other primitives.

2.2.12 Cameras (Drawing Objects)

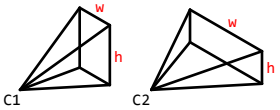

Figure 17: Margin figure caption.

Cameras contain a dimensions property, which is automatically used when the camera is active. This enables using cameras with specific image sizes. This, too, is nothing new, but was "low hanging fruit" and therefore quickly integrated into the Extension. (Figure 17)

Ideas

- Camera dimensions should be available in different units (pixels, cm, in etc.), additionally there should be a scale property.

2.2.13 Workspace (User Interface)

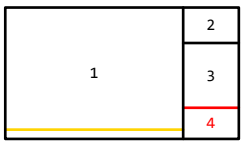

Figure 18: Margin figure caption.

The default workspace layout closely resembles that of native Blender. The central element is the 3D Viewport (1), on the side there is an Outliner that gives an overview over the projects data (2), below it the Object Panel (3), below that a space for Object Notes (4) (Figure 18).

2.3 Future Work and Other Ideas

2.3.1 Translate Operation

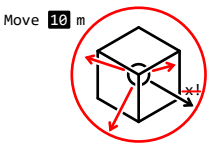

Figure 19: Margin figure caption.

When translating Objects, inputting a distance should not automatically lock the translation to an axis. Instead, the total distance of the translation should be limited. (Figure 19)

2.3.2 Modifiers/Special Geometry Operations

Especially useful for modeling architecture are Blender's Modifiers (like Boolean, Array etc.), which exist in other forms in most existing architectural software. Optimally, using Modifiers should be made to feel like using operators, meaning quicker and less complicated than the current process of adding a Modifier, setting parameters etc.. Moreover, the quality of modifiers being non-destructive is beneficial. Since many Blender Extensions have already approached this problem, this was not worked on in this project. (Figure 20)

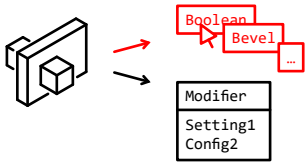

Figure 20: Margin figure caption.

2.3.3 Drawing Output

The very elementary topic of drawing output was explicitly not a part of this project, mainly because 1. another group of students initially worked on this and 2. the domain of this topic is large enough to fill an entire term by itself. However, of course it would not only be great but necessary to integrate this functionality into an Extension like "Goliath". (Figure 21)

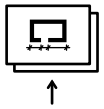

Figure 21: Margin figure caption.

2.3.4 Texture/UV Based Damage Marking

Another important topic when doing "Bauen im Bestand" is mapping damage. Like the Drawing Output, this is nothing new, but it would be helpful for a way to deal with this to be implemented in this project. One idea for how this could be done is using UV Maps and interactive texturing (like Texture Paint in Blender or similar functions in other CG software). In contrast to Object or Vertex based methods, this would allow for an appropriate mapping level of detail. Groups or types of damage could be distinguished using separate texture maps or color coding. Since UV Maps are supported in the IFC format, this information could probably even be exported quite easily. (Figure 22)

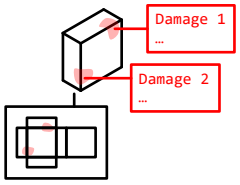

Figure 22: Margin figure caption.

2.3.5 Documenting Time and Building History

This is yet another old but important topic for "Bauen im Bestand", especially for historic or "heritage" architecture. Documentation and clear, readable display of building history should be approached for a tool like "Goliath". (Figure 23)

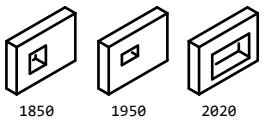

Figure 23: Margin figure caption.

2.3.6 IFC Export

The way "Goliath" is intended to work is partly inspired by the IFC standard and Bonsai specifically. Unlike Bonsai, the idea is not to use the .ifc file as a Project file. However, Projects are still intended to be exchanged as .ifc files (or at least the possibility of doing this is). For this, "Goliath" needs an .ifc exporter/translator. Bonsai could serve as this itself, or it could be the main reference for creating one. (Figure 24)

Figure 24: Margin figure caption.

2.3.7 References

Objects in a building model should be able to refer to other data with a type/category/reason of reference. Closely related to this is the previously explained ability to have (hyper-)links in Notes. This other data should not only encompass other objects in the project file but also external objects, like photos, old drawings, old texts etc.. (Figure 25)

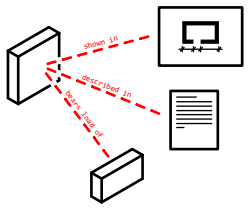

Figure 25: Margin figure caption.

2.3.8 Linking Data

In the Extension and in the proposed ideas, as well as in construction projects in general, we are dealing with many different types of data. Currently, in this prototype stage, the connecting "fulcrum" for all of this data is the (Blender/"Goliath") Project file. This should be questioned, because it is inevitable that there are going to be multiple project files and file types from different sources, be it software or other, throughout a projects working process, especially when this process includes long time storage an archiving of the BIM. To deal with this, there could be a meta-level structure (database, manifest, graph) coordinating project data and managing the links in between.[3] (Figure 26)

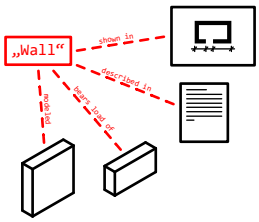

Figure 26: Margin figure caption.

[3] This is related to the concept of type and instance level interoperability mentioned in 1.3

3 Discussion

As stated in the beginning of this document, this is a prototype or concept of a software tool. In the process of this work, typical software development hurdles quickly became apparent, like bugs, compatibility and configurability. Finding a balance between "developing" (as in writing code) the tool and thinking about concepts in a more general way was attempted. Similarly, finding a balance between developing the tool as a whole and developing the individual functions was attempted. In consequence, to be usable outside its role as a prototype or demonstrator, it would need to be developed further. It cannot be guaranteed that this happens. Furthermore, the idea of simplifying the BIM process can be questioned. Present day construction projects are complicated and it can be argued that modeling them simply has to reflect this complexity in order to reach current (technical) standards. Finally, the feasibility of the whole workflow implicitly proposed by this project can be questioned. Modeling buildings using meshes and vertex based modeling is generally much less common than modeling them with parametric methods and most users might be in favor of sacrificing individualization for practicability, similarly to what is here presumed to hold back the usage of BIM itself (1.1). This would need to be investigated by studying user behaviour.

References

Akbar, Z., Ron, G., & Wortman, T. (2023, September 22). Democratizing the designer's toolbox. In *Digital design reconsidered - proceedings of the 41st conference on education and research in computer aided architectural design in europe (eCAADe 2023)* (pp. 41–50).

BAK. (2024). Ergebnisse der befragung der selbstständig tätigen kammermitglieder. https://bak.de/wp-content/uploads/2024/11/2024_BAK_Strukturbefragung_Bericht_Selbststaendige.pdf

Blender Foundation. (2026a). *Blender*. https://www.blender.org/

Blender Foundation. (2026b). *Blender 5.0 python API documentation*. https://docs.blender.org/api/current/index.html

Bonsai. (2026). *Bonsai*. https://bonsaibim.org/

Brookes, C. (2017). The application of building information modelling (BIM) within a heritage science context.

Bruno, N., & Roncella, R. (2019, July 25). HBIM for conservation: A new proposal for information modeling.

Ewert, E. (2024, June 7). Heritage-BIM - die nutzung digitaler modelle in der baudenkmalpflege.

Hüther, J. (2025, August 15). Vorläufige handlungsempfehlungen für die modellierung von bestandsgebäuden als BIM unter nutzung der LOIN-methodik.

Liu, J., Foreman, G., & Willkens, D. (2022). An introduction to technological tools and process of heritage building information modeling (HBIM). https://doi.org/10.4995/ege.2022.17723

Lombardi, M., & Rizzi, D. (2024). Semantic modelling and HBIM: A new multidisciplinary workflow for archaeological heritage. https://doi.org/10.1016/j.daach.2024.e00322

Petzold, F., & Reichenberg, B. (2021, December 3). BIM und bauen im bestand. In *Building information modeling* (pp. 507–532).

Törmä, S. (2014, August 21). Web of building data—integrating IFC with the web of data. In *eWork and eBusiness in architecture, engineering and construction* (pp. 141–148).

Appendices

A Links

Project GitHub Repository

OSArch Community Post/Discussion

User-oriented Video Introduction

B Glossary

BIM Building Information Model(-ing). Building models containing object oriented non-gemoetry information.

HBIM Heritage or Historic BIM. BIM in the context of architectural heritage or historic buildings. Mostly used when buildings of high historic value or significance are concerned, eg. cathedrals.

Bauen im Bestand German term for construction by (re)using existing built environment. This includes but is not limited to what is called renovation in English. It is also not limited to buildings of historic value, but applies to all existing built environment.

Bestands-BIM, BIM im Bestand An attempt to coin a handy term that describes BIM in the context of "Bauen im Bestand", removing the limitation to buildings of historic value.

CAD Computer Aided (Architectural) Design.

IFC Industry Foundation Classes. The main, universally established BIM exchange format and/or the schema behind it.

Pset Property set. Property sets are where object properties are stored in the IFC format.

UI User Interface. The front end or "face" of a software that users directly interact with when using it.

Mesh A way of describing three-dimensional shapes. Meshes consist of vertices (points in space, singular: vertex), edges (lines between vertices) and faces (surface between edges/vertices).

Edit Mode A mode in which a (mesh) geometry can be modified on the point (vertex) level.

Parent/Child Objects Child objects are subordinate to their parent object, a parent/child object relation is a form of data hierarchy. Often, changes to parent objects affect their child objects in some way.

Workspace The layout of windows/areas that is visible in the program window.

Bounding Box A box that encloses a geometry.

Camera A type of object that marks a view (position and orientation) to render drawings or images from.

Capitalization In this document, words that describe what happens inside the software are capitalized. For instance, an "Object" is an object specifically inside the (CAD, 3D modeling, BIM) software while "object" retains its general meaning.