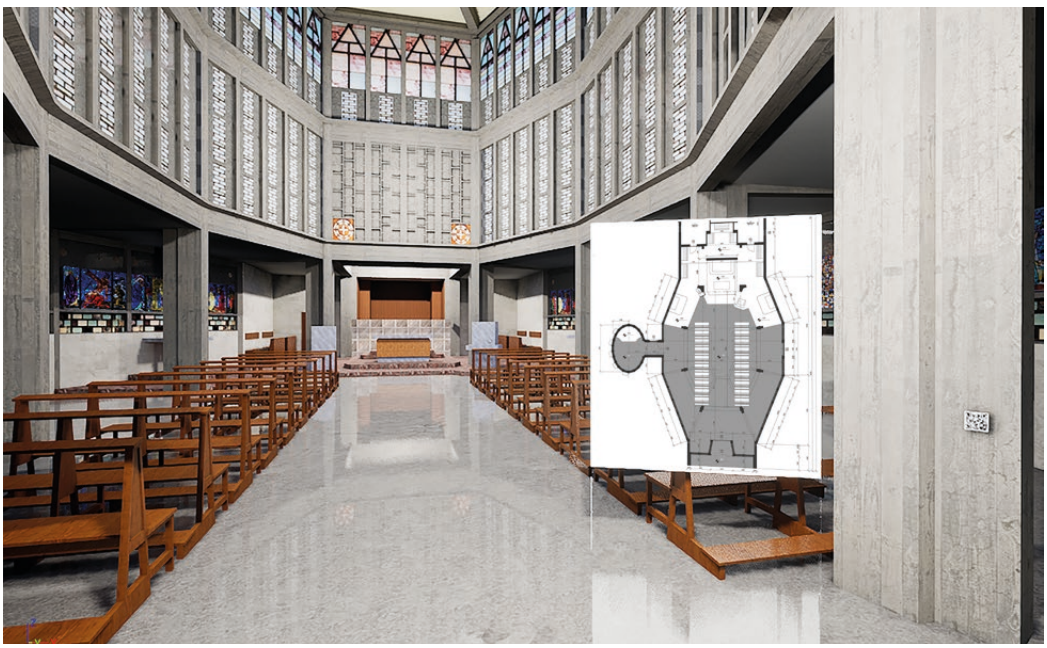
H–Bim to Virtual Reality: a New Tool for Historical Heritage

Carlo Biagini
Ylenia Ricci
Irene Villoresi

Abstract

In recent years, the application of Building Information Modelling to cultural heritage has led to the development of solid operating methods that have enabled a more efficient management of information. The use of other real time methodologies, such as Virtual Reality (VR), has also begun to be tested in the architectural context. The objective of this work is to test whether BIM models can be exploited to create immersive experiences in digitally simulated environments, with the aim of setting up new visualization and evaluation modalities for the built space. Giovanni Michelucci's Church provides an opportunity to test the use of Historic–BIM (H–BIM) models for the development of VR.

Keywords
H–BIM, virtual reality, revit, unreal, Giovanni Michelucci.



doi.org/10.3280/oa-686.55

Background

The use of Virtual Reality in the architectural field has recently become more and more widespread, opening up new opportunities for interaction between people and the built environment. In the case of historic buildings, it is becoming a powerful tool for the preservation and the enhancement of the historical heritage. This might also give to the professionals involved in the conservation process the chance to exploit it for decision making.
For many historic buildings, most of the available documentation consists of the original paper documents dating back to the time of construction. Lacking a pre–existing digital model, one often needs to be created specifically for VR applications. However, the increasing use of H–BIM has made digital models of these buildings available in some cases. To what extent BIM models can be re–purposed for VR use is, however, unclear, and this will be the objective of this work. The subject of the current study is the *Beata Maria Vergine* Church designed by *Giovanni Michelucci*. The church was built in 1957 for the industrial village of *Larderello*, near *Pisa* in Tuscany, Italy. Most of the original documentation has been provided by *Fondazione Giovanni Michelucci*, Florence, Italy.

Fig. 1. Left: 3D model of the church in Revit. Right: the same model imported in Unreal.

BIM–VR Interfacing

The BIM model for the church was built using Autodesk Revit. Unreal Engine is the game engine chosen for the implementation of the VR. The communication between Revit and Unreal is managed through Datasmith, a collection of tools and plug–ins that enables models, scenes and layouts built with a variety of 3D design applications to be imported into Unreal. With the 'Export Datasmith' Plug–in, elements from Revit are exported as '.udatasmiht' files. Through the Datasmith importer, the content of these files is transformed into a set of "actors" (the objects of a scene). Datasmith enables the export of geometries, materials and textures and their assignment (both Autodesk default materials and customised ones), lights and cameras. Visibility settings need to be carefully selected in Revit prior to export, since Datasmith will only export objects set as visible in the Revit 3D view.
Every solid geometry inside a Revit project is imported into Unreal as a Static Mesh Actor. A Static Mesh is a piece of geometry made of polygonal faces connected to create the object. Therefore, the Static Mesh Actors are created as a collection of surface faces, not tied to a unique solid volume. Every Static Mesh Actor in Unreal is generated based on families defined inside Revit. If there is more than one solid inside a family, these parts will be imported as a single actor, and cannot be selected separately in the main environment.

During the import process, some identification codes and tags from Revit are imported along with each geometry: 1) the name of the family; 2) two identification numbers automatically assigned to the object by Revit; 3) the level the object was linked to in Revit, and, if present, the identification number of the hosting element.
The identification data match between environments, maintaining the correspondence between the objects in Revit and the actors in Unreal. This makes it possible to carry out targeted substitutions of elements from Revit in Unreal.
Datasmith also imports materials and textures from Revit as Unreal assets. The static mesh actors maintain the material and texture assignment from Revit families (fig. 1), but with some limitations. For example, every face of the geometries is mapped, meaning that 2D coordinates are assigned to the vertexes of every surface. This coordinate system is used to define how the texture of the materials is applied to the faces of the solid geometry. In Revit every surface of a 3D object has the same default coordinate system, so that textures are applied with the same orientation on every surface. However, Revit does not allow a flexible coordinate system to manage the orientation of textures assigned to the materials. This might result in unrealistic texture visualisation in VR. When this occurs, it is necessary to import the element into another 3D design application where it is possible to modify the texture mapping data. Then, the element can be imported again into Unreal, and the faces of the mesh will show the texture with the proper orientation.

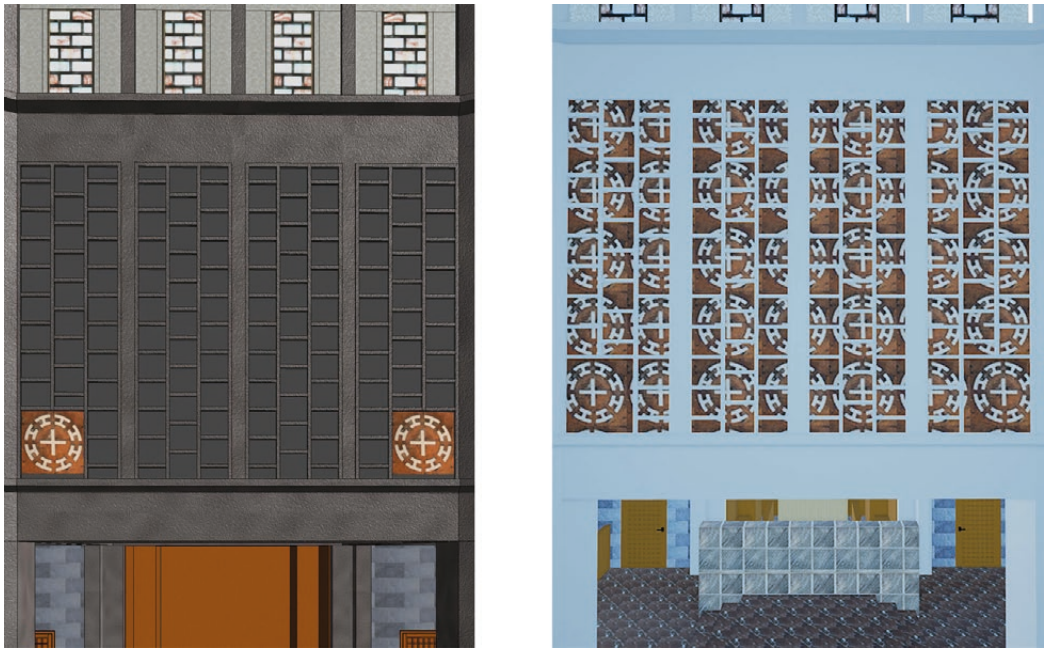


Fig. 2. Left: wall material assignment using Paint tool in Revit. Right: same wall material assignment after import process in Unreal.

Another issue about the import process regards the loss of information caused by the use of the Paint tool in Revit. In Revit, the material of the external faces of a volume can be modified separately from the main material assigned to the volume of that same object. This change, applied using the Paint tool, only affects the external appearance and is not transferred to the data about the main material of the volume. Individual sub–regions of a surface can also be modified with the Paint tool. However, the export procedure assigns only one material to the entire object. Also, the information about the shapes of existing sub–regions is not transferred during the export to Unreal. For example, in our case study, some sub–regions have been created to add pieces of decorations on some of the walls and floors. In the example reported here (fig. 2), the decorations were created as square patches on the surface of the wall. Once imported in Unreal, this surface information was completely lost: the patches disappeared, and the appearance of the entire wall matched the material assigned to the decorative patches in Revit.

Management of the VR Model

As previously mentioned, information about materials and textures is maintained during the transfer process. However, the final result in Unreal is rather rough and incomplete, probably due to the different encoding of material properties between the two applications. In Revit, materials are managed through a pane where the appearance properties are listed together with their assigned values. This list is fixed based on the class of the material and it is not possible to add or delete them. Properties can only be changed by modifying the values of their parameters.
In Unreal, materials are one of the most critical and complex aspects to handle, because of the wide variety of options for material characterisation managed through a network of visual scripting nodes (called Material Expressions). Every node contains a piece of code that is responsible for one aspect of the physical behaviour of the material. Different nodes are connected to achieve optimal visual results, especially when it comes to the interaction of surfaces with light.
However, in Revit, some material properties are either not modelled or managed in background without being explicitly reported in the data for the material. Instead, Unreal needs specific handling of these properties to achieve optimal visualisation. Nevertheless, during the import process, the Material Expressions are automatically populated with nodes, many of which are not necessary for the specific material or contain incorrect values. Therefore, these Revit–derived material structures might be very difficult to interpret and handle in the Material Editor. In our specific case, we often resorted to re–programming the network of nodes entirely, retaining only the properties relevant for the visual restitution in VR.

Addition and Display of Informative Resources

One of the most important features of BIM models is that all kind of information can be included directly inside the project. Unfortunately, this information is not exported in Unreal and cannot be automatically recalled in VR. In this case study, we tested the use of info–points as a method to include some pieces of information manually for real time visualisation of the resources. This was limited to a small portion of the information, but the same methodology can be extended to the rest of the data. The info–points are specific spots inside the scene where it is possible to bring up multimedia panels presenting the information. These panels consist of screens, realized with a blank plane rectangular actor, showing manually inserted media content, both in the form of videos and images.
These panels are normally hidden and can be made visible through QR codes linked to them. These QR codes mark the location of the info–points and can be activated by the user with the controllers by proximity.
When the QR code is activated, the panel pops up displaying its content. The info–points were placed in meaningful locations inside the scene, next to points of interest, for which they provide information when queried. For example, a panel describing a stained–glass window was located just below this object. The user could visualise this additional material while standing in front of the window in the VR environment.
However, when it comes to generic information, such as blueprints, this methodology showed some limitations, because the panels could only be recalled at the specific locations where the info–point had been placed. Therefore, optimal solutions will need to be developed for this kind of data.

Conclusions

Using BIM models as a base for building VR is possible and it is effective in the transfer of 3D geometries. However, the communication between the two applications used in this study is still not optimised for seamless transfer of data. Of notice, these are two standard applications used in their respective fields. It is therefore expected that the challenges described in this work will be faced by other users and designers interested in BIM–VR

Bottlenecks	Suggested developments
Neither platform allows modification of surface mapping to adjust texture orientation when applied to the objects	Developing a plug–in to add this specific feature to Unreal
Material properties assigned with the Paint tool in Revit are not transferred to Unreal	Implementing automatic separation of surfaces with different materials to preserve the correct assignment in Unreal
The encoding of material properties is different in the two environments	Optimizing the translation of Revit material properties to the visual scripting environment used in Unreal
It is not possible for BIM metadata to be automatically transferred to Unreal models	Implementing transfer protocols within the Datasmith plug–in to allow efficient integration

Tab. 1. Problems met in the transition from Revit to Unreal and possible developements to address them.

interfacing. Hence, it is necessary to streamline the communication between the people and teams responsible for these two aspects of project design. It is also important to note that BIM models are often insufficient to convey all the necessary information for optimal material modelling and rendering for VR applications. Therefore, external references are of paramount importance for the correct modelling of the materials in VR. Finally, the passage of metadata between the two modelling environments is quite limited and the rich BIM information often has to be manually transferred in the VR model. This could further increase the workload in case of revisions and updates. The table below summirizes the bottlenecks in the process, suggesting possible developments to address these issues.

References

Antonopoulou Sofia (2017). *BIM for Heritage: Developing a Historic Building Information Model*. Swindon: Historic England.

Biagini Carlo (2020). Oltre la modellazione informativa: "componibilità come composizione". In *Firenze Architettura Quaderni 2020*, Anno XXIV, pp. 96-101.

Biagini Carlo, Donato Vincenzo (2014). Building Object Models (BOMS) for the documentation of Historical Building Heritage. In *EGraFIA 2014: Revisiones del futuro, Previsiones del pasado*. Rosario: CUES, pp. 142-149.

Epic Games (2021). Unreal Engine 4 Documentation, https://docs.unrealengine.com (15 January 2021).

Fondazione Giovanni Michelucci (2011). *Michelucci a Larderello: il piano urbanistico e le architetture*. Firenze: Alinea.

Pellegrin Luigi (1959). La Chiesa di Larderello, con commento di Giovanni Michelucci. In *L'Architettura. Cronache e storia*, 46, pp. 226-232.

Quattrini Ramona., Clini Paolo, Nespeca Romina., Ruggeri Ludovico. (2016). Measurement and Historical Information Building: chalenges and opportunities in the representation of semantically structured 3D content. In *DisegnareCon*, 9 (16), pp. 1-11.

Tagliaventi Ivo (1959). L'ossatura organica della Chiesa di Larderello. In *L'Architettura. Cronache e storia*, 46, pp. 280-281.

Zevi Bruno (1957). Una chiesa per Lardrello, Architetto Giovanni Michelucci. In *L'Architettura. Cronache e storia*, 16, pp. 714-715.

Authors
Carlo Biagini, Dept. of Architecture, University of Florence,carlo.biagini@unifi.it
Ylenia Ricci, Dept. of Architecture, University of Florence, ylenia.ricci@unifi.it
Irene Villoresi, Dept. of Architecture, University of Florence, ire.villoresi@gmail.com

Isbn 9788835125280