A VIEW TO THE FUTURE: GAMING ENGINES AND VR VIEWPOINT AUTOMATION FOR DESIGN ANALYSIS

In what ways can new actions in gaming engines improve opportunities for design analysis in the VR space?

B. COSIC,
University of New South Wales, Sydney, Australia
branko.cosic@unsw.edu

Virtual reality (VR) technologies in the practice of architecture are typically afforded a specialised status that has limited the extent of their use in the design process. The architecture industry has incorporated VR for visualisation, but through this it has highlighted the disconnections that exist between the design and visualisation stages. The 3D file formats that are exported by designers for visualisation are densely packed with data which can be used to build automation in gaming engines. This data can be used to spawn viewpoints for use in VR by extracting the locations input by designers and architects. This can be used to build a template for a design analysis tool, where the designers determine where they visualise in VR. While current methods of utilizing VR offer significant potential as a visualisation tool, it is impeded by its viewpoint rigidity. This process would facilitate the generation of numerous viewpoints for analysis in VR. Enabling more seamless interoperability between design and visualisation software while also helping dissolve boundaries between the design and visualisation stages. Such a process could extend the functionality and usability of VR for viewpoint analysis, promoting the designer's knowledge to the gaming engine consolidating the two separate stages. It aims to facilitate a step towards a more inclusive process that capitalises on 3D file formats within the design process for better VR use, facilitating better design outcomes.

Keywords: Virtual Reality, Design Analysis, Gaming Engines, Viewpoints, Automation, 3D file formats, FBX

1. Introduction: Research Aims and Motivations

Virtual reality (VR) is a well established technology in the architectural industry and widely researched subject in various academic fields. While numerous scholars have demonstrated clear benefits of adopting VR in the design process of architectural projects Achten et al. (1999); Liu & Bai (1998); Hoon & Kehoe (2003), the architecture industry has typically afforded VR a specialised software and skill status that has contributed to limiting its exploration as a productive design space. Equally, using VR as a design exploration space and not simply a visualisation tool, is significantly hindered by the usability of the software and its lack of function that might correspond to what designers would like to explore at early design stages. Miller (2015) argues that, fundamentally, technologies likes these do not account for the complexity of architectural projects and the variety of tasks and toolsets required. In short, interoperability remains a significant barrier to the usability of VR as a design space.

2. Research Observations and Objectives

The utilisation of VR in the contemporary practice has identified a disconnection that exists between designing and visualising a project. As a project departs from the design process for visualisation in VR, it exits spheres of common interoperability in design software, and enters the field of gaming engines. The workflows that surround design software are optimised for design and cannot account for the complexities associated with VR and gaming engines. As a result, VR is overlooked as an analysis tool that runs concurrently with designing due to the lack of accommodating workflows and interoperability that enable the design process. It becomes necessary for members of the design team to demonstrate an understanding of the view impact of a design change or element, and they are often limited to static two or three dimensional renders.

This research will outline the exploration of new processes in gaming engines and workflows that surround the implementation of VR in the design process. It will be structured across 3 prototypes that produce relevant technical results from an exploratory lens and aims to extend the usability of VR for design analysis.

The overarching research objectives of VR viewpoint generation are to improve opportunities for design analysis in the VR space. This system involves exploring new actions in Unreal engine, via the generation of viewpoints for VR use. It intends to streamline the process of visualising in VR concurrently with the design process. A subsequent objective would also be to resolve the disconnection between designing and visualising in architecture. The objectives would form an evaluation system that advise the iterative action research process.

The contemporary architecture office is structured so that many of these objectives would benefit them. This is because many of the technologies and software used to visualise design work do not account for the complexities associated with the various trades and toolsets within a practice. There is an apparent disconnection when the design process nears visualisation, and even more so when attempting to visualise using VR. This is most likely due to the nature of gaming engines, required for VR. Most design orientated software can communicate well with various other software of its genre, but gaming engines descend from a different branch of software and application. Regardless, the value of gaming engines and VR have motivated many architecture practices to explore the technology, with varying levels of success. This research intends to further explore this interoperability to create a more streamlined process of VR analysis. It intends to empower all members of the design process who can export their designs in an FBX format. This way, members of the design process can determine the locations for visualisation in VR.

3. Research Questions

In what ways can new actions in Unreal Engine improve opportunities for design analysis in the VR space?

The outcomes of this research aim to explore existing friction points in the workflows and interoperability of FBX file formats and unreal engine to enhance and extend the functionality, performance and usability of the VR space.

4. Methodology

This research follows a change-oriented action research (AR) framework, guided by this overarching research question. It aligns closely with the categorical AR process outlined by Hopkins (Hopkins 1985), where a plan is created from an exploratory stance, executed via new actions and revised through observation and reflection of the action, the outcomes then determine a revision of the initial action. However, this framework is augmented by the processes outlined by Purao et al. (Purao et al. 2005), focusing more on the practical aspect of the research question, producing relevant technical results which inform the research theory and potential revisions of the initial action plan. It intends to be an iterative research process informed by three prototypes which will be produced, tested and evaluated. The knowledge gained from these prototypes will advise potential changes in the research plan, and consequent prototypes, but their overarching aim is to explore the research question through technical means and advise any revisions in the research itself. An overview of this technical exploration:

1. Location recognition in Unreal Engine via FBX/OBJ file format:
    a. Import/Export settings
    b. Recognition via material or name
    c. Storing locations
2. Generate location data for Unreal Engine
    a. Setting teleport locations
    b. Explore teleport parameters/collisions
    c. Test different marker scenarios
3. VR Headset interface
    a. Accommodating interface

- Test/Evaluate each prototype that advises the research theory, question and methodology.
- Review of outcomes.

Any new knowledge gained from observations and reflections of the prototypes will advise the research theory and any subsequent changes in technical process. The motivation surrounding this AR is to gain versatile theoretical and practical knowledge through a technical exploration of the research question through the steps listed above.

5. Background Research

The outcomes of this research aim to resolve and explore existing friction points in the workflows and interoperability between the software platforms of Rhino and Unreal Engine to enhance and extend the functionality / performance / usability of the VR space. A review of literature surrounding this topic has revealed that there has been little consideration of the interoperability between these two software packages, and even more-so for design analysis in VR.

Research on the use of gaming engines and virtual reality (VR) in the field of architecture has shown positive benefits in terms of how visualisations can enhance an understanding of spatial quality (Whyte 2003). In a study of earlier incarnations of VR technologies in architecture, Liu & Bai (1998) observed that VR was a powerful tool for visual impact analysis, although admitting subjects with professional training in VR should be included to draw any real empirical evidence. Achten et al. (1999) similarly says that VR is an enabling technology for architecture for visualisation, interaction and analysis, although continuing to tell us that the full potential of VR hasn't been realised due to shortcomings in interface, real-time and multi-user technologies. These early explorations outline the complexity of using VR systems practices, but also generally afford VR a 'specialty' status, used by experts in the gaming and architectural industries (Liu and Bai, 2001; Achten et al., 1999; Hoon and Kehoe, 2003). In the succeeding two decades VR technology had greatly developed to become more accessible to wider range of users. The reasons for this include, free open source gaming engines, extensively documented online tutorials, and greater research and exploration of the wide range of applications VR affords in architecture. Yet, the sense of VR being a specialty skill remains prevalent and to some degree contributes to hindering its more extensive use and applicability (Craig, Sherman and Will, 2009). Equally, the use of gaming engines and VR visualization has also been limited in architecture by the ways such practices are typically employed for marketing purposes or end-stage renderings. Consequently, a large body of research around the topic of VR and architecture explores visualisation quality, less attention is given to research that explores the interoperability of design software and visualization software, in this case gaming engines. Recent research exploring how we can utilize gaming engines and VR to access our increasingly data-rich models has also fallen short in the areas of interoperability. Suggesting that the transitions from design software to gaming engine still need to mature to conduce a good, interactive design environment within VR (Dokonal et al. 2015). The only information that is carried into the game engine is the structure itself, departing from a range of understanding in the designer. Can interoperability be established with Unreal Engine to carry over the designers understanding to benefit design analysis?

Research reporting on workflows and the establishment of interoperability between software has identified the benefit of customised workflows in the design process, coordinating different communication techniques, facilitating opportunities for seamless collaboration (Miller, 2010). The ideas put forth by Miller review the benefits of proprietary software packages, suggesting that the establishment of thorough interoperability between software would better serve design outcomes. Given that design analysis in a clear majority of applications using VR involves two or more software packages, it would be sensible to assume that considerations of interoperability are established, but the process still relies heavily on the game engine itself. In similar concurrent research by Hawton et al. (2018) & Coppens et al. (2018) who investigate the power of workflows to harness software interoperability relative to VR. This research explored the challenge of real-time parametric modelling within VR, reimagining the virtual world as a design space, and evaluating its benefit in sustaining more collaborative communication of design concepts. Much like the ideas set forth by Miller, these research projects have accentuated the advantages of workflow design in prospect of better collaboration and communication. Although these reports focus on real-time communication and modelling within VR, they reiterate the question whether the development of workflows between Rhino and Unreal for design analysis in VR could also share the same expansion of functionality and usability. It is interesting that research regarding the benefits of architectural visualization in VR are relatively prevalent, but the same research interest is not afforded to the actual process of design analysis using VR and the workflow underlying it. The steps required to visualise a prospective design in VR are widespread across various software platforms, and the formation of viewpoints for analysis only extends this process. It seems the workflow that encompasses design analysis is underutilised.

Many of these explorations into workflows and software interoperability share similar objectives with this research project, albeit with a different process. They explore the interoperability between software to enhance and extend VR communication and collaboration. Their successes suggest that much has been left unexplored in areas of interoperability for the benefit of VR and more specifically outline the need for further research to resolve workflows for viewpoint design analysis.

6. Case Study

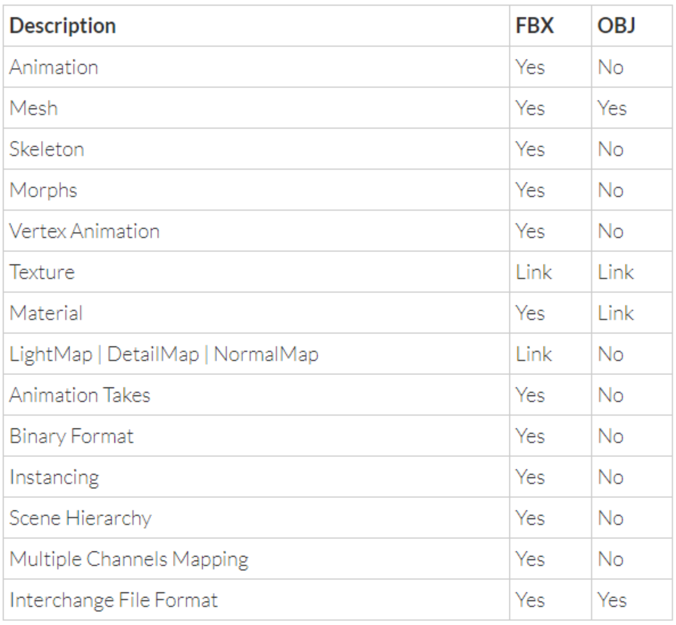
This process will explain the processes which are unique to this research and will not explain the typical processes that are taken to set up a VR world that are shared by all projects of their type.

6.1 File Format

The basis for visualising any 3D object in Virtual Reality is the FBX file format. It is a proprietary format owned by Autodesk, but since it's introduction in 2006 it has become a widely supported and industry dominant format for building object based models. It can be represented as either binary or ASCII data, but this research will approach this file format from an organizational lens. Instead of reading into the code of the FBX it will used the data readily available to form automation.

Other file formats that exist and can also be used for similar purposes, such as OBJ, which are a dated form of 3D file format. Figure 1 below quantifies the extent of their difference. In short, FBX extends more variety and feature to the user, as well as having a higher data fidelity, and being the industry dominant file type for use in gaming engines.

The primary feature used to build automation in this research is the material data stored within the FBX. This data is embedded when it is applied to an object and exported as an FBX.



Description	FBX	OBJ
Animation	Yes	No
Mesh	Yes	Yes
Skeleton	Yes	No
Morphs	Yes	No
Vertex Animation	Yes	No
Texture	Link	Link
Material	Yes	Link
LightMap | DetailMap | NormalMap	Link	No
Animation Takes	Yes	No
Binary Format	Yes	No
Instancing	Yes	No
Scene Hierarchy	Yes	No
Multiple Channels Mapping	Yes	No
Interchange File Format	Yes	Yes

Figure 1. FBX & OBJ Comparison

6.2 Exporting/Importing FBX

All 3D design software can export FBX files apart from a rare few which are not relevant to the architecture industry. All Autodesk software can produce FBX files, with the addition of Rhinoceros 3D, Blender, Cinema 4D, Google SketchUp, among others. The process in this will test Rhinoceros 3D, Revit, 3DS Max and a combination of each of them.

In order to store material data in an FBX it must be applied to the appropriate object that will be used to identify the location. It is important that the material is bespoke in the context of the 3D object being visualised. It cannot be shared by any other elements in the design. A simplified instance of this process within Rhinoceros 3D is outline below in figure 2. The markers material (red) is created and applied to the cubes, while 'all other objects' has a different material. Later this marker material will be called for in Unreal Engine and it will ignore all other material instances. The process is ultimately the same with all other software types.

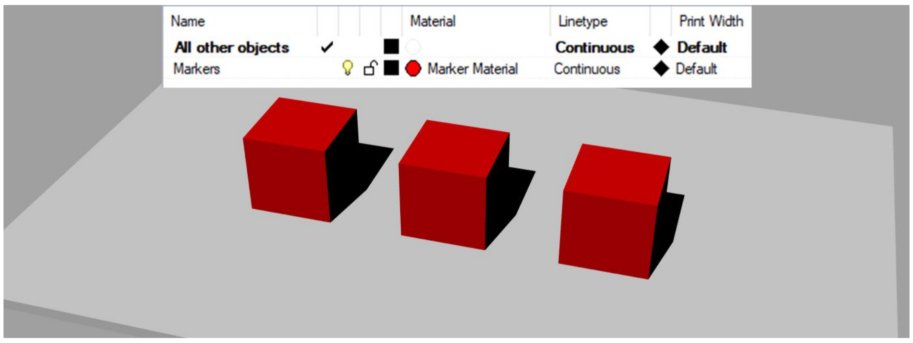
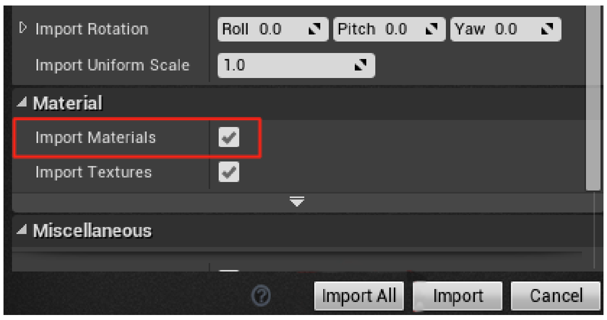


Figure 2. Example of material application to markers

When importing into Unreal Engine, the import options panel appears and extends a variety of features relative to the data in the FBX. The only option that is required for this process is that the materials are imported along with the objects. The remaining options can be used and will not impact the function of the process.



Import Rotation: Roll 0.0, Pitch 0.0, Yaw 0.0
Import Uniform Scale: 1.0
Material
Import Materials: ✓
Import Textures: ✓
Miscellaneous

Figure 3. Import options

Importing a simple model with floors and walls will display information similar to this in the content panel. The markers (yellow), and their associated material (markers_2) are the driving elements of the process.

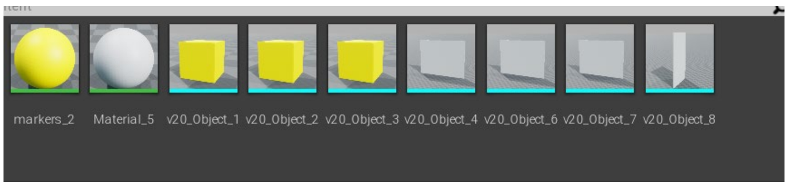


Figure 4. Import scenario

6.3 'Markers'

The markers are simply 3D objects that represent virtual positions in the software. In the instance of this process they are cubes, but in theory they can be any object the user is willing to create in order to determine the position they want in VR, for example, a sphere, cone, or rectangle. The nature of the object is not relevant to the process, rather its position and the material applied to it. However, the maximum size of the object should not exceed the scale of a human being. This is because the process later computes the objects centre of mass in order to extract a location. It is a placeholder that simply identifies to the user that it is the position they will later be able to travel to, and visualise in VR, for this reason they should be arbitrary.

6.4 Location recognition

This process utilises the markers we import in a script that searches all actors in the Unreal Engine landscape. 'Actors' is the terminology given to 3D objects in Unreal Engine. When objects are imported into Unreal Engine, they become Static Mesh Actors. The script within the level blueprint of the engine gets all static mesh actors in the world and executes a foreach loop that cycles through each of them, seen in figure 5 below. Material recognition was used because the naming system would depart some levels of interoperability. The different software that export FBX often group, and name objects in different methods and in order to preserve application across all software, this process was used.

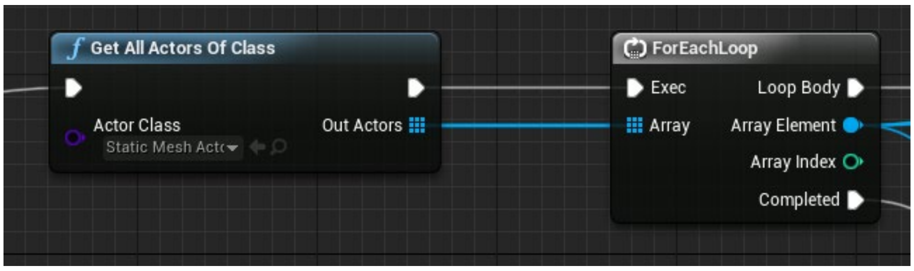


Figure 5. Static mesh actor process

The script then uses a 'get material' function which references all the materials present in the project at that time. This is where the material set prior to input is referenced. The script then filters each static mesh actor in the world and will only return actors with that specific material. Once it has done this, it sets the mobility of the static actor to 'moveable'. This is because Unreal Engine cannot run many of its functions on actors which are static or stationary and requires them to be totally dynamic. The actors are then added to their respective arrays (lists), ‘static meshes with desired materials’ for the markers (figure 6).

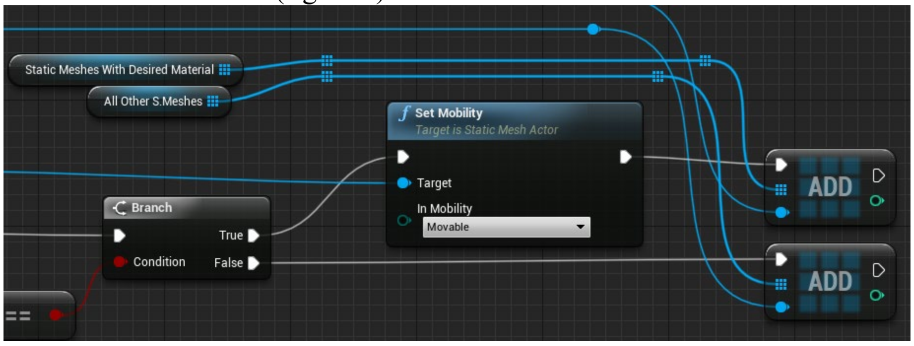


Figure 6. Assigning mobility and array

6.4.1 Extracting and storing world location

There are numerous components that can extract the location of the actor/s in question. For these applications ‘get centre of mass’ worked best and was preferred seeing as it would return the centre of the object. However, ‘get actor location’ or ‘get world location’ would also serve a similar purpose. Following this, the locations gathered are added to another vector array.

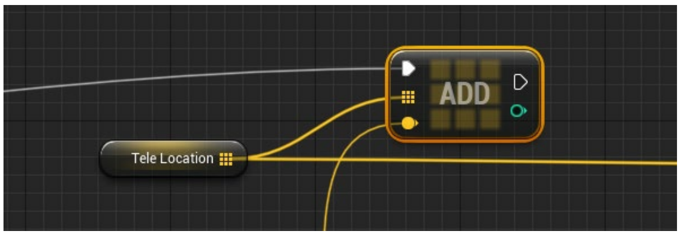


Figure 7. Vector array

6.4.2 Interoperability

Each FBX could be utilised in this process regardless of which software it was exported from. The only input that needed to be changed, provided the import and export settings were correct, were that the material that was filtered needed to be changed. The material was constructed as a input for the custom function for finding the locations (figure 8), and would tie into the branch that filtered the static mesh actors within the world seen in figure 6.

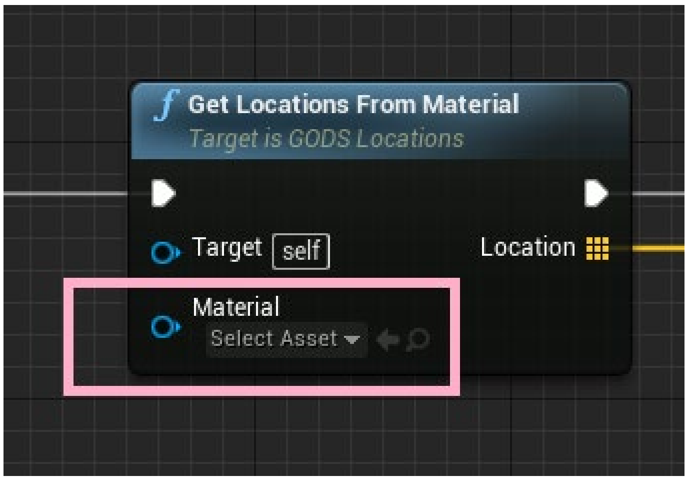


Figure 8. Material input

6.5 Interface

A variety of interface systems were tested for this application. A combo-box interface as explored, which functions via a dropdown method. However, at the time of this research, the combo-box was subject to a software bug which prevented it from opening in VR templates. Any future research could possibly capitalise on this interface type. The type of interface was dependent on the quantity of locations needed to be visualised, or the features that would be added. If the locations ranged up to 5, it would be fitting for a wrist interface for example, seen in figure 9 below. However, anything exceeding, and the number of buttons spawned would not fit in the setting. A floating widget at each of the locations is a good option because it will always be in a familiar position and would help to minimize motion sickness. The system uses each of the vector locations that are gathered in an array to spawn a child button for each. Each child button references the index of the location at the list and binds itself to that specific location. Although the child button needed to spawn all the other buttons would still need to be manually created. In short, each location would spawn a button that corresponded to it.

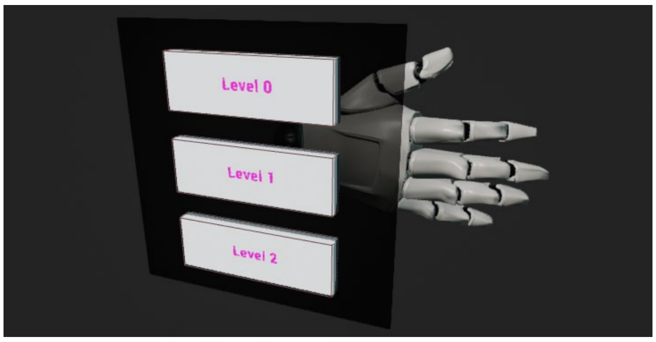


Figure 9. Wrist interface

The final interface (figure 10) used a widget switcher that allowed the user to move between 3 tabs. One of which was the viewpoints for teleporting to, the others was a simple feature that was built into the view analysis for solar analysis and the first was to switch between a different building design for the purpose of the final presentation. The locations gathered can spawn buttons or options in any given interface. The style of interface depends on the context of its use and the number of locations. This research will avoid labeling which is best because the process can utilise most styles available within Unreal Engine. However, for building any additional features such air flow, solar pathways or similar that are concise and easily navigated by first time VR users it is recommended to use large interfaces with clear functionality.

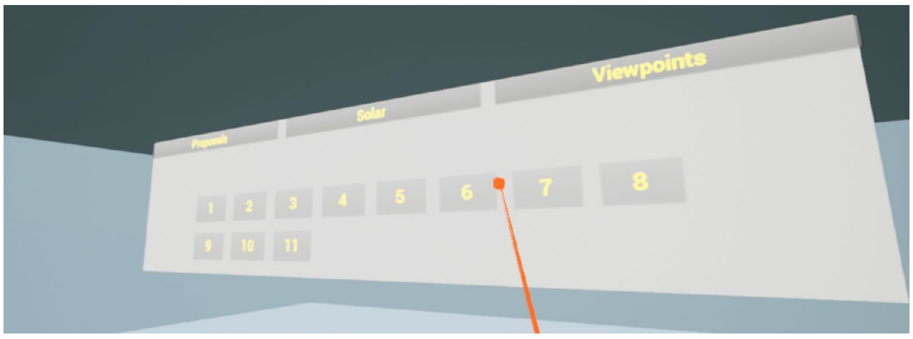


Figure 10. Presentation interface

6.6 Additional Features

Solar analysis was added to the viewpoint analysis scenario because it reflected the power of enabling designers to determine the location, they wanted to visualise. Referencing the directional light component, it was bound to a slider in the interface, when the sliders value was changed it would correspond with the sun rotation position. This feature is symbolic of many features that can be added.

6.7 Collisions

For VR to be utilised all actors in the world that need to be travelled on need to have collisions. Each object that is imported from a FBX automatically develops collisions unless otherwise stated in the import options. The problem that arises is that the collisions can vary in success depending on the nature of the model that is being imported, particularly how it is constructed. This requires manual input to check, and manage, if the need arises. In short, the objects that will be travelled upon need to be within a ‘navmesh bounds’ volume. This volume computes each planar surface as a teleportable surface for the player pawn. Further research could explore automatic ways of encompassing all the actors within the model with a navmesh volume.

6.8 Context model

Adding a context model surrounding the particular design can be beneficial, but the meshes need to be relatively simple. VR places a heavy load on the graphics and other processing systems relative to Unreal Engine. Keeping the simple with as little surfaces as possible was beneficial to the smoothness of the gameplay. The context should be added just like the model, import the fbx. However, the navigation meshes should not encompass it or anything that doesn’t need to be travelled upon.

7. Significance of Research

This research reveals many of the complexities and limitations which surround virtual reality and its implementation across design software. The prototyping during this research was indicative of new processes for interpreting data in FBX’s and how that data can be used to build automation which enables a new process of design analysis. This process removes some of the barriers that exist between designing and visualising stages, enabling architects and designers to have an input into the VR and visualisation process. It is indicative of the benefits that exist in bridging the two stages in a technical method, which positively benefit the design process.

The process using existing methods, with readily available data to build a new process of automation. While the method still requires manual building of many aspects in the VR world, it still relieves a great deal of workload from the visualisation members. It also provides a more efficient method of determining the positions for VR visualisation, reducing the verbal exchange that typically plagues the division of designing and visualising.

The process is demonstrating the power that technical adaptations to the process of utilizing VR can greatly enable and benefit the design process. Further development is needed to truly solidify this process and accumulate a complete tool for VR design analysis. It can be applied to urban settings to gain better understanding of the contextual impact of proposals. It may also provide a floor to floor, interior to exterior analysis tool, as displayed in this research. It is relevant to fields of development, architecture and urban planning.

8. Evaluation of research project

The objectives of this research were to explore new processes of automation and interoperability for VR viewpoint analysis. The VR process is synonymous with gaming engines and is reliant on 3D file formats (FBX) for building environments. The automation and interoperability in this research project were built predominantly by script in Unreal Engine using the data in FBX file formats. It revealed that the data stored in FBX could be used for material recognition and location generation. Most existing research papers fail to outline the value of data within such files, more specifically for building automation within VR.

The outcome of this research reveals a process that can extend VR users hundreds of automatically created locations for fast first-person analysis, without the need for cumbersome travel through the VR environment. Given that these positions are determined outside of the gaming engine by any software that can export an FBX file format, it empowers people outside of VR or gaming engine expertise.

The automation that has been built involves location recognition with markers positioned outside of the gaming engine. These markers are categorised by material, where the process would search the virtual environment for each instance of the material and replace it with a VR location position. Initially, this process was intended to be used in a high-rise building analysis scenario where each floor, apartment, or point of interest could have a VR teleport location created for it. In retrospect, this process is not constrained by this application. It could be used to understand the impact of proposals at an urban scale from various positions in a context or to understand view-impact of neighbouring buildings at various levels. The testing conducted in this research was merely a vessel for understanding the process of automating viewpoints; it could serve many purposes and the testing scenarios in this research paper only intended to highlight its benefit. More testing in various scenarios will serve to strengthen the research.

The limitations in this process emerge when naming and labelling each VR teleport position. Given the time-frame available of this research the process names each position via the sequence of its creation. Further building the script to recognise spaces and apply more specific names to the list of locations generated would help avoid confusion when teleporting between locations.

The process of building environments for VR remains relatively unchanged from its first incarnations, despite the technology's improvements and improvements in software. The process still hasn't matured fully Hoon & Kehoe (2003), despite the evolution of data available. Through this research it is apparent that new utilization of the data available in FBX file formats to build automation demonstrated a higher capacity of functionality, performance and usability in the VR space.

9. Conclusion

Interoperability and accommodating workflows are decisive in producing an effective and efficient design process. Current processes of utilising VR have lost their grasp on the aggressive technical progression and requirements of architecture. This research has revealed the benefits of modifying existing processes and data to improve the utilisation of VR for design analysis. It has demonstrated the potential of such an analysis tool which enables members of the design process through interoperability and establishes a new method of input into virtual reality from outside its field of expertise. Looking toward the future, further exploration into VR interoperability and workflows will continually benefit technological integration and the creation of architecture.

Acknowledgements

This project was done in collaboration with UNSW and BIM Consulting.

References

Achten, H., Roelen, W., Boekholt, J., Turksma, A. and Jessurun, J. (1999). Virtual Reality in the Design Studio: The Eindhoven Perspective. Architectural Computing from Turing to 2000 [eCAADe Conference Proceedings / ISBN 0-9523687-5-7] Liverpool (UK), [online] pp.169-177. Available at: http://papers.cumincad.org/cgibin/works/paper/e336.

Craig, A., Sherman, W. and Will, J. (2009). Developing virtual reality applications. Amsterdam: Morgan Kaufmann/Elsevier.

Coppens, A., Mens, T. and Gallas, M. (2018). Parametric Modelling Within Immersive Environments - Building a Bridge Between Existing Tools and Virtual Reality Headsets. Computing for a better tomorrow – Proceedings of the 36th eCAADe Conference, [online] 2, pp.721-726. Available at: http://papers.cumincad.org/cgibin/works/paper/ecaade2018_188 [Accessed 12 Aug. 2018].

Dokonal, W., Knight, M. and Dengg, E. (2015). New Interfaces - Old Models. Martens, B, Wurzer, G, Grasl T, Lorenz,

WE and Schaffranek, R (eds.), Real Time - Proceedings of the 33rd eCAADe Conference, [online] 1, pp.101-106. Available at: http://papers.cumincad.org/cgi-bin/works/paper/ecaade2015_119.

Hawton, D., Cooper-Wooley, B., Odolphi, J., Doherty, B., Fabbri, A., Gardner, N. and Haeusler, M. (2018). Shared

Immersive Environments for Parametric Model Manipulation - Evaluating a Workflow for Parametric Model

Manipulation from Within Immersive Virtual Environments. T. Fukuda, W. Huang, P. Janssen, K. Crolla, S.

Alhadidi (eds.), Learning, Adapting and Prototyping - Proceedings of the 23rd CAADRIA Conference, [online] 1, pp.483-492. Available at: http://papers.cumincad.org/cgi-bin/works/paper/caadria2018_162.

Hoon, M. and Kehoe, M. (2003). Enhancing Architectural Communication with Gaming Engines. ACADIA, [online] pp.349-355. Available at: http://papers.cumincad.org/cgi-bin/works/paper/acadia03_045 [Accessed 9 Aug.2018].

Hopkins, D. (1985). A teacher's guide to classroom research. Milton Keynes [Buckinghamshire]: Open University Press.

Liu, Y. and Bai, R. (2001). The Hsinchu experience: a computerized procedure for visual impact analysis and assessment. Automation in Construction, 10(3), pp.337-343.

Miller, N. (2010). [make]SHIFT: Information Exchange and Collaborative Design Workflows. ACADIA, [online] 10, pp.139-144. Available at: http://papers.cumincad.org/data/works/att/acadia10_139.content.pdf [Accessed 10 Aug. 2018].

Purao, S., Cole, R., Rossi, M. and Sein, M. (2005). The overlaps between Action Research and Design Research. [online] Slideshare.net. Available at: https://www.slideshare.net/SandeepPurao/draricis2005 [Accessed 20 Aug. 2018].

Whyte, J. (2003). Virtual Reality and the Built Environment. Presence: Teleoperators and Virtual Environments, 12(5), pp.550-552.

Appendix

To view a presentation video of the end process, visit:
https://youtu.be/o6fEjWzUUd0