SHARED IMMERSIVE ENVIRONMENTS FOR PARAMETRIC MODEL MANIPULATION

Evaluating a workflow for parametric model manipulation from within immersive virtual environments

D. HAWTON

University of New South Wales, Australia

d.hawton@unsw.edu.au

Abstract. Virtual reality (VR) and augmented reality (AR) provide designers with new visual mediums through which to communicate their designs. There is great potential for these mediums to positively augment current visual communication methods by improving visual communication between remote collaborators. Enabling designers to interact with familiar computational tools through external virtual interfaces would allow designers to both tweak design parameters and visualise parametric outcomes from within the same immersive virtual environment. This paper outlines a workflow for parametric manipulation and mesh replication between immersive applications developed in the Unity game engine and McNeel's Grasshopper plugin. This paper serves as a foundation for future research into integrating design tools with external VR and AR applications in an effort to enhance remote collaborative design.

Keywords. Augmented Reality; Virtual Reality; Grasshopper; Parametric; Procedural.

1. Research Aims and Motivation

Communication can be defined as an external expression of human cognition. The notion of "cognitive artifacts", initially coined in reference to human-computer interactions (Norman 1992), translates to design artifacts used to communicate design intent. Effective design representation and communication are determinative of the cognitive process of the designer and the functional process of what is being designed, and is at the crux of successful collaborative design thinking (Abdelmohsen 2012). Shared immersive environments extend this notion of the cognitive artifact and provide an alternative form of visual communication for remote design collaborators that aid in reducing miscommunication when conveying design intent. This project aims to enable designers to both visualise and change their designs from within immersive environments.

Virtual Reality (VR) and Augmented Reality (AR) technologies are already enabling architects, engineers and designers to visualise their designs in unparalleled immersive visual mediums (Trimble 2017; IrisVR 2017). Both these technologies have the potential to transform the way designers conceptualise and communicate design intent when collaborating remotely, and are currently being used in this manner in the aerospace, medical and maintenance industries (Microsoft HoloLens 2017). Exploring how to exchange data between familiar computational design tools and these new visual mediums opens up the possibilities for how designers can integrate them into their existing workflows and extend collaborative design capabilities.

This research aims to determine how immersive virtual environments are currently being used in the built environment industries and to examine potential use cases for the future. It also outlines a prototype for visualisation and model manipulation in VR and AR applications. This prototype demonstrates how these technologies can be integrated with existing parametric tools to create a workflow that enables remote collaboration and model interaction from within these immersive visual mediums. It should serve as a foundation for further research into the benefits of using immersive virtual environments for remote collaboration in real-world parametric design workflows.

2. Research Observations and Objectives

Effective collaboration is dependent on effective communication; this is pertinent for collaborative designers, as design intent is often miscommunicated, leading to time and money wasted (Slater et al. 2012). Research into ways for improving visual design communication is therefore especially valuable for designers, and through a combination of different visual communication methods it is expected that fewer mistakes will be made in built environment projects due to the enhanced understanding provided by VR and AR.

This paper outlines a basic prototype for sharing mesh geometry between Grasshopper and other connected AR and VR Unity applications via a cloud server. Cloud servers have previously been used to transmit BIM data

(Afsari et al. 2016), and exchange data between BIM and VR (Kieferle et al. 2014), however at the time of writing exchanging mesh data between Grasshopper and VR and AR applications has been relatively unexplored. This workflow will allow designers to expose and manipulate parameters on a virtual interface and visualise the resulting mesh within the same immersive environment. By using a web server as an intermediary between the Unity and Grasshopper environments, data can be transferred bidirectionally, meaning that parametric updates in the Unity environment mirror those of the Grasshopper environment. These virtual environments can be shared by collaborators in remote locations, which will enhance visual design communication on parametric projects.

3. Research Questions

VR and AR, as visual mediums, are unique in their ability to convey scale and perspective and improve design engagement through interaction. Direct interaction and manipulation has been shown to improve collaborative design communication and understanding, therefore it is important for designers begin to evaluate how immersive environments can be incorporated into their design workflows (Gauglitz et al. 2014). Virtual environments have also shown to encourage a more user-centric design approach in university design studios by allowing designers to inhabit the buildings they're designing (Moleta 2016). The following research therefore aims to answer the question of whether shared immersive environments can enhance parametric workflows.

Parametric design is widely used within architectural and design studios, therefore exploring how parametric design communication can be improved through immersive virtual environments is an important area of exploration. As immersive environments provide a unique platform for collaboration and shared visualisation, it is anticipated that parametric workflows would benefit from improvements in design communication.

4. Methodology

In order to evaluate the viability of collaborative virtual environments a prototype has been developed to demonstrate how users in different virtual or augmented environments can interact with the same parametric model. The process for exchanging data between these immersive mediums and Grasshopper can be broken down into several sections. First, the anatomy of a mesh will be explained, as this is crucial for understanding how the mesh is organised into a transmittable data structure in Grasshopper for procedurally reconstructed within the Unity engine. This paper will then outline how user inputs will be exposed in the virtual spaces to enable designers to manipulate values in the Grasshopper script. These values will be sent to Grasshopper via a web server to compute any model changes. The updated Grasshopper model data will then be sent to all connected virtual spaces and reconstructed for visualisation. An evaluation of this method and suggestions for further research will then be provided.

5. Background Research

Shared collaboration in VR and AR is relatively new, however several related precedents already exist. The projects below provide an overview of how different application platforms can exchange data or be used to visualise design thinking. They illustrate the motivation for developing this prototype and provide a foundation upon which the prototype was built.

5.1 Firefly Grasshopper Plugin

External inputs for Grasshopper are not new. The Firefly Grasshopper plugin allows designers to connect input/output devices and sensors to their Grasshopper. Designers can use sensor data or manipulate actuators such as motors from their scripting environment. They can manipulate parameters within their Grasshopper environment using physical inputs, such as physical slider, buttons or dials. This precedent demonstrates how external data can be sent to and captured in Grasshopper, which is a central part of the project outlined in this paper.

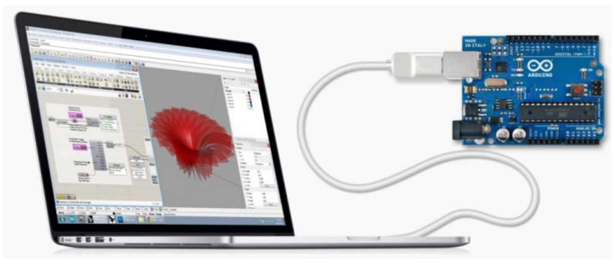


Figure 1: The Firefly plugin connects Grasshopper to external inputs and outputs, such as Arduino microcontrollers.

5.2 Trimble Sketchup Viewer

Though the HoloLens is relatively new there are already several applications targeting architectural visual communication. Trimble's Sketchup Viewer is one of these applications, allowing users to view their Sketchup models in the HoloLens through a cloud server that users can upload to, store and download their projects from (Trimble 2017). Users can then navigate around the model and perform various model manipulations such as scaling, rotating and isolating elements.

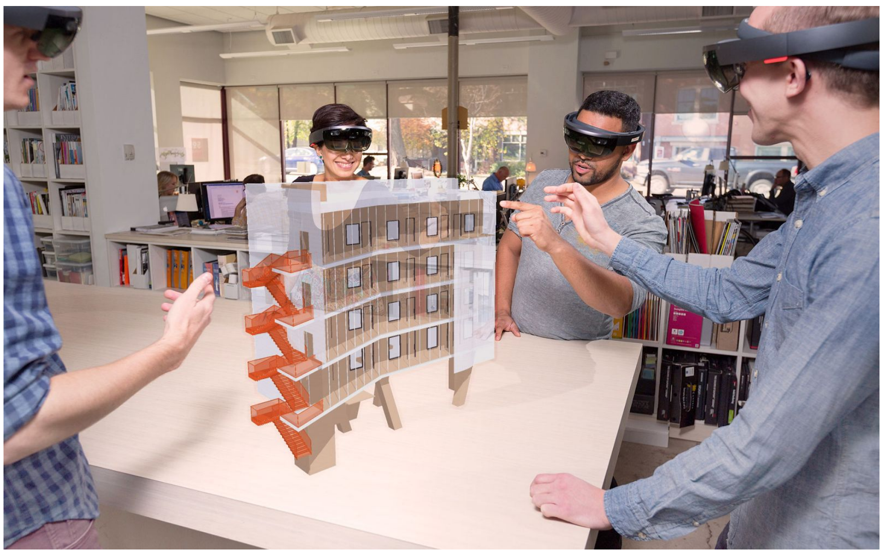


Figure 2: A visualisation of the shared holographic visualisations possible with the Trimble Sketchup viewer.

5.3 Iris VR

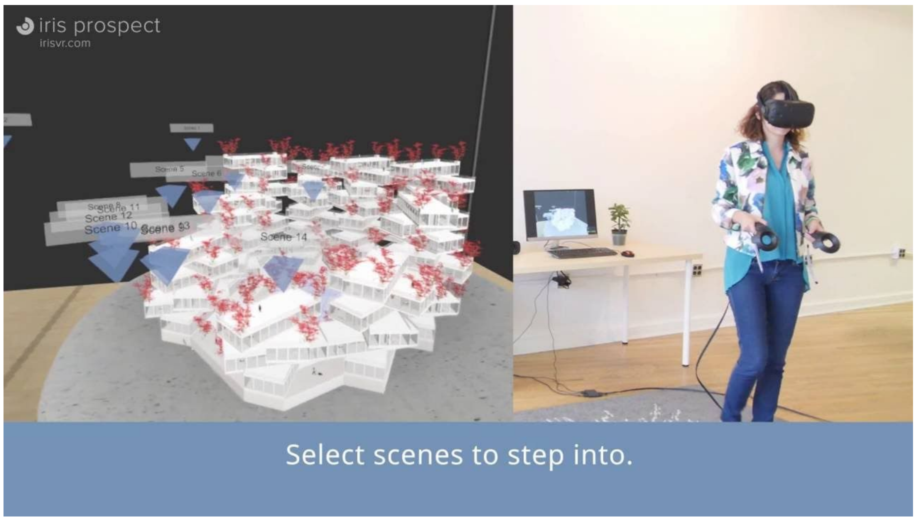


Figure 3: IrisVR is a streamlined solution that allows users to easily push their Revit and Sketchup models to VR.

VR technology has recently become more common and is now used as a visualisation tool within architecture offices. IrisVR is a VR platform developed specifically for architects who wish to easily and efficiently visualise their projects in VR (IrisVR 2017). The application allows users to push their Revit or Sketchup models to the IrisVR platform easily and efficiently, removing the technical and tedious processes commonly associated with visualising architecture in VR.

5.4 Google VR Blocks

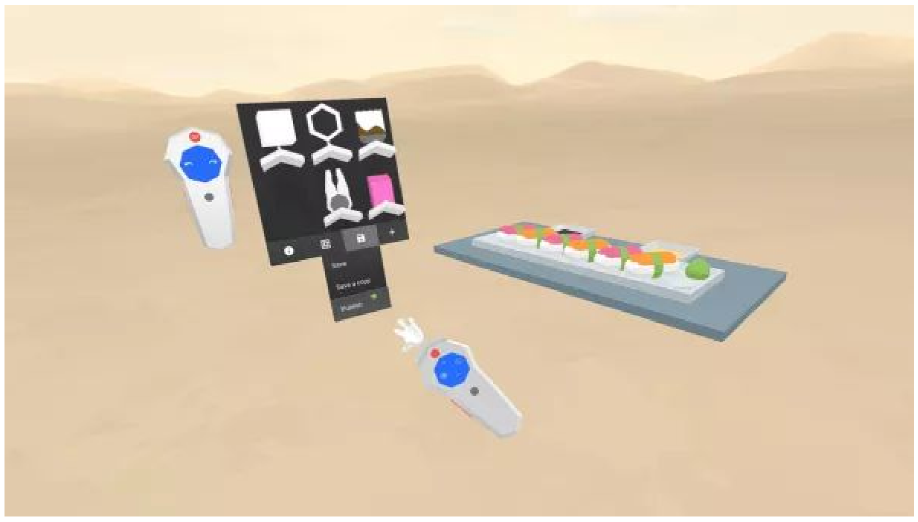


Figure 4: A 3D scene modelled in Google VR Blocks, a VR 3D modelling application.

Google VR Blocks is a VR 3D modelling application that provides an immersive environment for creating and experiencing 3D models (Google VR 2017). Artists can design and create their models in VR and export them out for rendering or use in other software. Modelling in VR provides users with a unique sense of scale and immersion, which allows artists and designers to more closely convey their original design intent.

5.5 Urban Pinboard Bi-Directional Pipeline

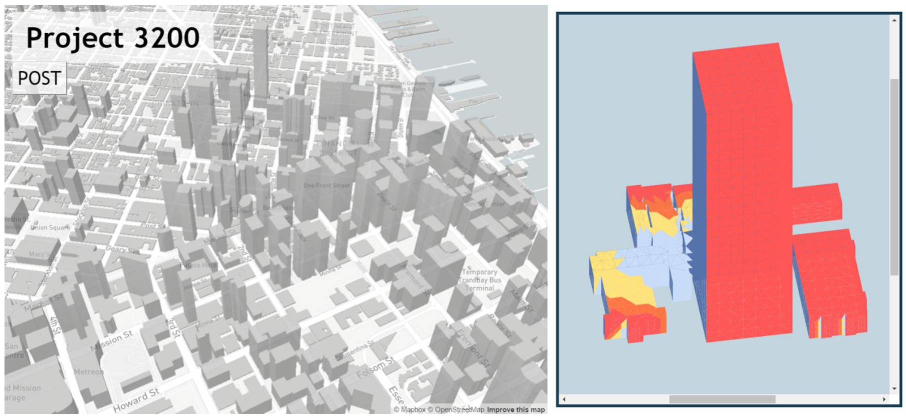


Figure 5: The Urban Pinboard Bi-Directional Pipeline demonstrates a way of accessing the computational power of Grasshopper from web interface for environmental analysis.

Urban Pinboard is an online urban design collaboration platform in development that aims to include the public, private and community members in a digital conversation for future smart city initiatives (Johansen et. al 2017). As part of this initiative a bidirectional pipeline between online data repositories and Grasshopper was developed to automate the computation of environmental analysis, in this case the amount of sunlight hours buildings are receiving in their urban context. Users can select geographical locations from a web interface and receive back an SVG representing the sunlight hours received for buildings in this area. Innovative tools such as this empower architects and designers by automating data exchange between different platforms, making them more accessible to clients and collaborators.

5.6 Platypus Grasshopper Plugin

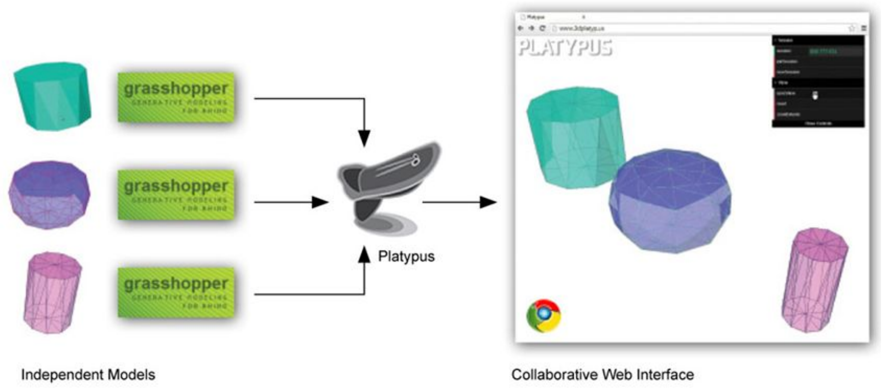


Figure 6: The Platypus Grasshopper plugin facilitates interactive multi-user parametric modelling via their web.

The Platypus Grasshopper plugin is an innovative web collaboration tool developed by CORE Studios. Multiple users can join a Platypus web session and both change paramater variables and view the Grasshopper model being streamed from the Grasshopper environment to each client's web browser (CORE Studio 2017). This plugin provides an invaluable tool for architects and designers looking to collaborate remotely, only requiring access to a web browser.

6. Case Study

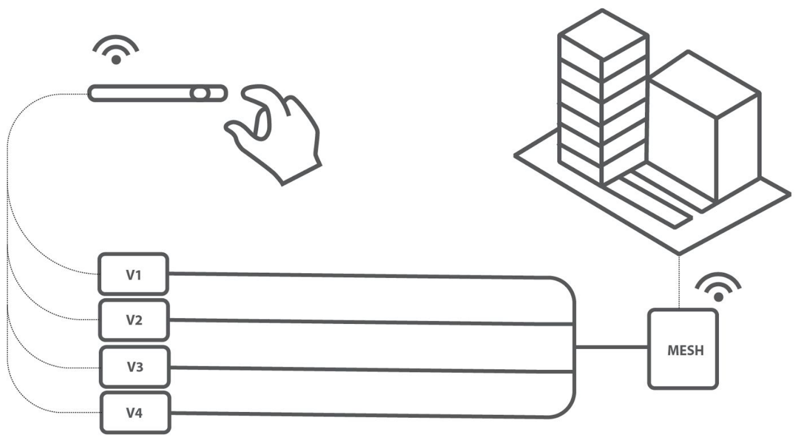


Figure 7: An early conceptual diagram of the prototype, demonstrating the flow of data from the immersive environment to Grasshopper and back again.

This case study outlines a prototype for multiple immersive virtual spaces visualising and interacting with the same parametric models through their respective interfaces. The prototype and study involves exposing the parameters for users to interact with in the Unity VR or AR application. Data from these inputs values are sent to Grasshopper via a web server to compute any changes. Changes are computed in Grasshopper and mesh data is sent back to the Unity VR or AR application. This mesh data is used to procedurally reconstruct the mesh from the data received from the Grasshopper environment in connected VR or AR applications. This will be concluded with an evaluation of use cases for this prototype and discuss current limitations and suggested further research and development.

6.1 UNDERSTANDING THE ANATOMY OF A MESH

This prototype will be exchanging mesh data between McNeel's Grasshopper plugin for Rhinoceros 3D and Unity, a popular gaming engine and currently the most accessible way for developers to develop applications for both VR and AR devices. It is beneficial to start by understanding the anatomy of a mesh and how mesh information from Grasshopper can be used to procedurally create mesh geometry in Unity.

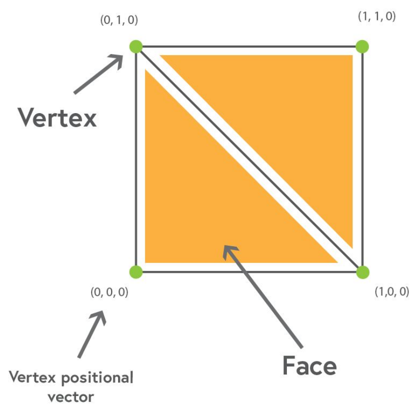


Figure 8: The anatomy of a mesh quad, highlighting the mesh characteristics that are needed to store and retrieve mesh data.

6.1.1 Meshes in Grasshopper and Unity

Meshes in Grasshopper are defined by a Face-Vertex data structure (Mode Lab 2017). This means that groups of vertices—or points in 3D space—are grouped together to define polygons or triangles. To create meshes using a list of vertices Grasshopper needs instructions that define the structure of the mesh in terms of its individual triangles and those triangle's individual vertices. Grouping the vertices' positional vectors into groups of threes provides Unity or Grasshopper with the information needed to recreate a complete mesh by procedurally creating its individual mesh triangles in a sequential manner.

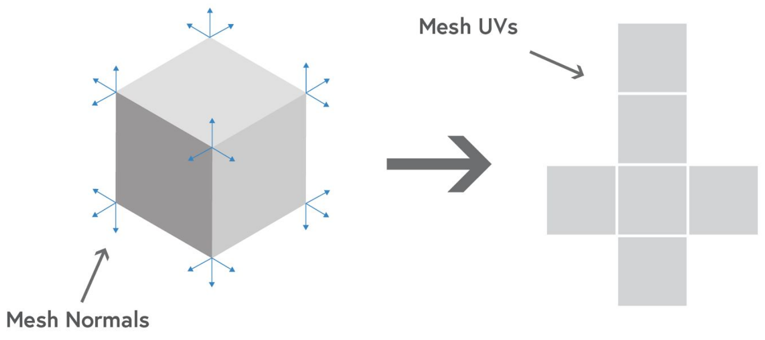


Figure 9: Normals and UVs are also required for procedural mesh generation.

The custom mesh structure used for the case study was developed by Junichiro Horikawa, whose plugin for communicating mesh data between Grasshopper and Unity was used as the foundation for this workflow. This custom mesh structure comprises of a lists of vertices, uv coordinates, normals and faces (Horikawa 2017). These mesh parameters are extracted from the Grasshopper mesh using methods in McNeel's Rhino.Geometry API and organised into Horikawa's custom mesh structure that is used to reconstruct the mesh in Unity. Serialisation involves converting an object into a stream of bytes that can be more easily transmitted and is therefore ideal for communicating this custom mesh object, and is how the mesh data is transmitted to our server and subsequently over to the Unity immersive application (Microsoft 2015).

6.2 CREATE AN APPLICATION IN UNITY TO SEND AND RECEIVE DATA FROM

6.2.1 Creating a Node.js web server for relaying data between Grasshopper and Unity

A Node.js web server was used to relay data between the two software environments (clients). This server receives the input values from the Unity application and sends them to the custom Grasshopper components, which are outlined in section 6.3. It also listens out for the mesh data message from

Grasshopper and relays it to Unity. This simple web server is responsible for listening for websocket events from Grasshopper and Unity and passing on the appropriate data.

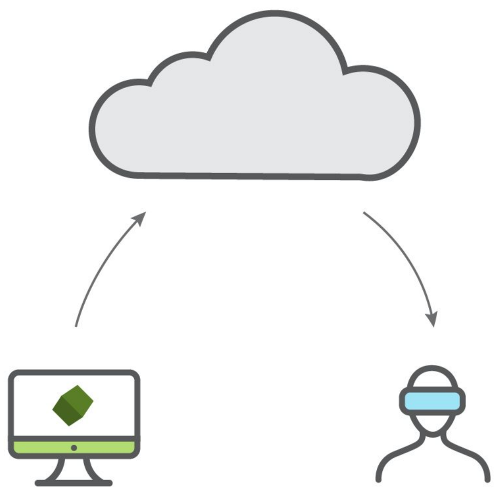


Figure 10: The Node server is hosted externally and relays information between Grasshopper and the VR/AR clients.

The server relies on several javascript libraries in order to route data between the two separate platforms. Express.js is a minimal web framework for Node.js applications, which is used for basic routing; Socket.IO is used for realtime websocket communication and this is how Grasshopper and Unity communicate with each other; Http.js is HTTP request library suggested by the Socket.IO documentation.

The core of the application revolves around the Socket.IO functions *on* and *emit*. The *on* function listens for Socket.IO messages that are published with the *emit* function. *Emit* functions send messages containing data, usually in a JSON format with keys and values. When a value on a slider in a Unity application is changed a message is sent that defines the message name (the key), which would be the name of the variable to send the data to, and the number to change the parameter to (the value).

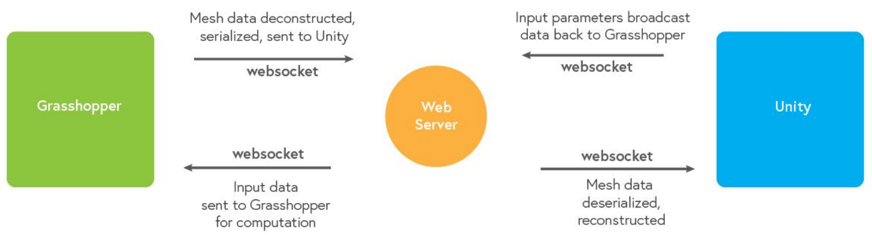

Mesh data deconstructed, serialized, sent to Unity
websocket
Input parameters broadcast data back to Grasshopper
websocket
Grasshopper
Web Server
Unity
websocket
Input data sent to Grasshopper for computation
websocket
Mesh data deserialized, reconstructed

Figure 11: Websockets listen out for input changes in Unity and relay new values to Grasshopper for computation before returning the updated mesh.

6.2.2 Creating an immersive application and enabling user input through sliders

The Unity project uses a few assets from the Asset Store, namely a Socket.IO plugin for connecting to our server, the Procedural Toolkit for procedurally reconstructing our mesh, and the VR Toolkit (VRTK) for VR functionality. It also uses the Windows Mixed-Reality Toolkit for its HoloLens libraries and prefabs (Microsoft 2017).

In order to allow users to manipulate values in Grasshopper from the Unity environment an input method must be provided. This example will use the standard Unity sliders as they suitably replicate sliders found in Grasshopper. This process would be similar in both VR and AR projects that require numerical user input. This slider can then be converted into a Unity Prefab that can be used in either a VR or AR Unity application.

Unity sliders provide several adjustable parameters out of the box, such as minimum, maximum and starting values. These value ranges should reflect the desired value ranges for the parameters within Grasshopper. Each time a slider value is changed a function called *SendData* is fired off, which packages the value to be sent in a C# Dictionary and is subsequently sent to the server in the form of a JSON object, which is the format used to parse the data on the server side.

6.2.3 Using Socket.IO to listen for changes within Unity and broadcast them back to the server

Socket.IO is a popular javascript framework for harnessing the power of websockets, which allow for interactive communication between clients and a server (Mozilla Developer Network 2017). The clients, in this case, are the custom Grasshopper components and the Unity immersive environments, and the server is the Node.js server setup in section 6.2.1. Using websockets the server can listen out for events fired off from either the Unity sliders or the Grasshopper *Send Mesh* components and relay that information to the appropriate client.

6.3 BUILDING THE GRASSHOPPER PLUGIN TO RELAY DATA TO THE HOLOLENS APPLICATION

6.3.1 Defining the Grasshopper components needed to send and receive data

The functionality of the Grasshopper plugin is divided into several components. This was an adaptation of an open-source mesh streaming library written by Junichiro Horikawa (Horikawa 2017), though many of the components have remained the same. These components use several external libraries for Socket.IO, JSON interpretation and mesh serialization, which are:

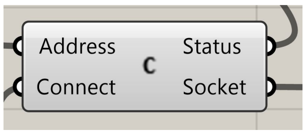
- **ZeroFormatter**
  This is used for serialization of the mesh into binary data that is sent from Grasshopper to Unity.

- **Json.NET**
  This is used to store the binary mesh data in a JSON format that can be parsed on the server.

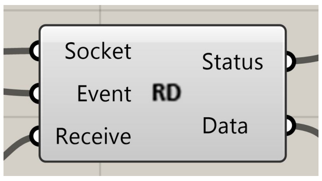
- **Socket.IO Client Library for .Net**
  This is used to communicate to the Socket.IO back end server.

The mesh streaming plugin contains the following components:

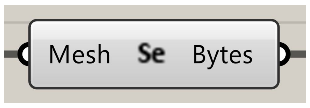
- ***Connect***
  - This component is used to connect to the server



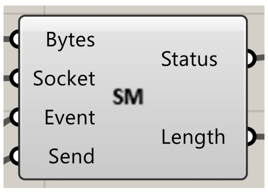
- ***Receive Data***
  - This component listens for a specific predefined message on the server and outputs the data value received



- ***Serialize Mesh***
  - This component takes a mesh as an input and serializes it into bytes to be sent to our server



- ***Send Mesh***
  - This component is used to send the serialised mesh data to the server



6.3.2 Connecting to the server

To connect to the server a URL must be provided to the *Address* input of the *Connect* node. A boolean toggle can be used to begin the connection to the websocket on the server. For this prototype, the Heroku cloud platform was used to host the server as it provides a free hosting tier for small servers. Cloud hosting is essential for accessibility, and as the server requires access from both AR and VR devices we cannot use a local server on each device.

6.3.3 Receiving numerical data from the Unity application

Every time a user changes the values of a slider a message is sent to the server to indicate that the values have been changed. This message is then relayed to the Grasshopper plugin outlined in Section 6.3.1, along with the value that needs to be changed. For example, if the script being manipulated contains a parameter called *NumberOfFloors*, and the *ReceiveData* Grasshopper node receives a value of 4 along with the event name *NumberOfFloors*, that value can be used to increase the number of floors in the building from the previous value to the new value of 4. This would then lead to a change in mesh output connected to the *SendMesh* component, which would fire off a separate event with a predefined name of *Grasshopper* (this could be anything, as long as it matches the event name defined in the server script), carrying the serialised custom mesh object.

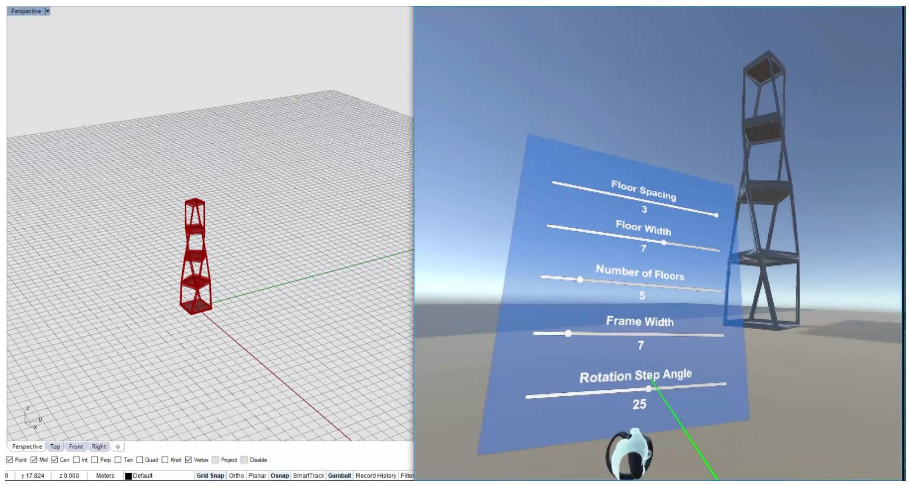

Floor Spacing
3
Floor Width
7
Number of Floors
5
Frame Width
7
Rotation Step Angle
25

Figure 12: A Grasshopper model being manipulated and visualised from within the VR environment.

6.4 REASSEMBLING THE GRASSHOPPER MESH IN UNITY

6.4.1 Using the Procedural Toolkit to reconstruct the mesh

Using the information from the custom mesh object passed into Unity from Grasshopper, the mesh can now be procedurally reconstructed. This used the Mesh Streaming components from Horikawa. he script iterates over the vertex, uv, normal and face information in the Custom Mesh object and procedurally rebuilds the mesh face-by-face. The Procedural Toolkit, an open source library available on the Unity Asset Store and developed by Daniil Basmanov, is used to interpret the mesh data received from Grasshopper and procedurally generate a mesh that mirrors that of our Grasshopper mesh (Basmanov 2017).

6.5 EVALUATION AND LIMITATIONS

6.5.1 Potential use cases and limitations

This paper outlines an experimental workflow to test the viability of real-time data transfer for collaborative design in VR and AR applications. It was anticipated that this would be useful for collaborators who are in different physical locations but wish to visualise and manipulate Grasshopper models in either VR or AR mediums, or a combination of both. For this tool to be viable for use in the industry it needed to be reliable, easy to use and provide distinct advantages over other collaborative communication tools.

Through developmental testing, and comparisons with alternative methods for parametric design communication, this workflow, in its current form, has demonstrated limited viability for real-world usage due to technical limitations. These include not being able to dynamically add or remove variables on the server or within the Unity applications and application crashes with large mesh transfers, leading to the conclusion that a native VR/AR parametric application that removes the need for bidirectional communication would likely be much more beneficial to designers collaborating remotely on parametric design. Procedural mesh generation is perhaps the biggest bottleneck within the workflow as larger meshes take too long to be reconstructed. Multiple parameter changes in quick succession can easily overwhelm the server, leading to lengthy delays and application crashes.

6.5.2 Suggested further development and research

Due to time limitations on this project some prototype functionality was not realised, the most important of which was deploying the application to the HoloLens device. Microsoft's Universal Windows Platform (UWP) does not necessarily reflect what is displayed in Unity's inbuilt HoloLens emulator, which was used for the majority of testing and development of this project. Specifically, the Socket.IO implementation that worked in the Unity HoloLens emulator and the Unity VR application did not work when deployed to the HoloLens. This is because the UWP uses its own proprietary websocket class to communicate with websockets on a server and does not seamlessly integrate with the Socket.IO library. This was realised late in the project and it is anticipated that this could be resolved with more time.

Additionally, further quantitative and qualitative research could be conducted to determine how advantageous design collaborators would find this tool in a real-world scenario. This would give a better indication of how practical and beneficial a cross-platform immersive collaborative design workflow would be in practice.

7. Significance of Research

This research serves as the basis for future research relating to emerging mediums for visual design communication. Understanding how data and geometry can be passed between applications provides designers with the tools they need to experiment with cross-application data manipulation. This has the potential to reduce the time taken to accurately communicate original design intent through the efficient exchange of data from design environments to visualisation environments. The prototype developed for this paper has several key limitations that prevent it from being used in a commercial setting, as discussed in Section 6.5. With more time these issues could likely be resolved, however user testing is still required to determine its viability and usefulness.

8. Evaluation of Work

Virtual environments show potential for improving design communication, however quantitative research must be undertaken to determine the degree to which this medium is advantageous over conventional representational methods. This paper demonstrates an overview of how emerging immersive environments are beginning to assist designers in design communication and provides a practical example of how these immersive mediums could be integrated with a popular computational tool. Due to time limitations this practical example was presented in an unfinished and unrefined state and is impractical to use as it currently stands. It does however demonstrate how current technologies can be integrated to create a shared data platform that could lead to improved communication between designers, especially those collaborating remotely and unable to share the same physical space as their colleagues.

8. Conclusion

It is expected that architects and designers would benefit from workflows that integrate currently used parametric design tools with emerging immersive technologies, such as VR and AR. What is not clear is how large projects with vast datasets can efficiently exchange data without significant delays between interactions. The prototype presented in this paper enables designers to interact with and visualise their Grasshopper designs from the same immersive medium from which they are situated. While this worked adequately for smaller designs, it became slow and unresponsive when meshes began to grow beyond a certain size. This is compounded if multiple users are making changes to the model simultaneously, which led to lengthy delays and applications crashes. This is likely due to the procedural mesh creation method used to assemble the mesh each time a new mesh is sent over from Grasshopper into the Unity application.

Perhaps this algorithm could be refined to improve performance, however it is likely that a native VR or AR parametric application that does not have to compute design algorithms externally would produce the most optimum results. With the recent release of Microsoft's Windows Mixed Reality VR Headsets (Windows Mixed Reality 2017), which are part of the same ecosystem as the Microsoft HoloLens, developing a parametric design platform accessible from both VR and AR environments seems the most sensible path forward.

Acknowledgements

This research was done in conjunction with ARUP Consulting and the University of New South Wales Built Environment Faculty. Thank you to Junichiro Horikawa whose open-source library for communicating between Grasshopper to Unity was essential to this project.

References

Abdelmohsen, S. (2012) ‘Genres of communication interfaces in bim-enabled architectural practice’, Ascaad CAAD | INNOVATION | PRACTICE [6th International Conference Proceedings of the Arab Society for Computer Aided Architectural Design, pp. 81–91.

Afsari, K., Eastman, C. M. and Shelden, D. R. (2016) ‘Data Transmission Opportunities for Collaborative Cloud-Based Building Information Modeling’, Blucher Design Proceedings, pp. 907–913. doi: 10.5151/despro-sigradi2016-448.

Basmanov, D. (2017). *Syomus/ProceduralToolkit*. [online] Available at: https://github.com/Syomus/ProceduralToolkit [Accessed 4 Nov. 2017].

CORE Studio (2017). *Platypus | CORE studio*. [online] Available at: http://core.thorntontomasetti.com/platypus/ [Accessed 30 Oct. 2017].

Gauglitz, S., Nuernberger, B., Turk, M. and Hollerer, T. (2014) ‘In touch with the remote world: Remote collaboration with augmented reality drawings and virtual navigation’, Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, pp. 197–205. doi: 10.1145/2671015.2671016.

Google VR (2017). *Blocks - Create 3D models in VR - Google VR*. [online] Available at: https://vr.google.com/blocks/ [Accessed 8 Nov. 2017].

Horikawa, J. (2017). *Mesh Streaming Grasshopper*. [online] Available at: https://github.com/jhorikawa/MeshStreamingGrasshopper [Accessed 1 Oct. 2017].

IrisVR (2017). *IrisVR - Virtual Reality for Architecture, Engineering, and Construction* . [online] Available at: https://irisvr.com/ [Accessed 8 Nov. 2017].

Johanson, M., Khan, N., Asher, R., Butler, A., Haeusler, M. H., (2017) ‘Urban pinboard’, CAADRIA 2017 - 22nd International Conference on Computer-Aided Architectural Design Research in Asia: Protocols, Flows and Glitches, (Brynjolfsson 2014), pp. 715–724.

Kawai, Y. (2017). *Zero Formatter*. [online] Available at: https://github.com/neuecc/ZeroFormatter [Accessed 21 Oct. 2017].

Kieferle, J. and Woessner, U. (2014) ‘BIM Interactive - About Combining BIM and Virtual Reality A Bidirectional Interaction Method for BIM Models in Different’, eCAADe, 1, pp. 69–75.

Lee, J., Gu, N. and Williams, A. P. (2014) ‘Parametric Design Strategies for the Generation of Creative Designs’, International Journal of Architectural Computing, 12(3), pp. 263–282. doi: 10.1260/1478-0771.12.3.263.

Mode Lab (2017). *What is a Mesh? | The Grasshopper Primer (EN)*. [online] Available at: http://grasshopperprimer.com/en/1-foundations/1-6/1_What%20is%20a%20Mesh.html [Accessed 1 Oct. 2017].

Mozilla Developer Network. (2017). WebSockets. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API [Accessed 1 Oct. 2017].

Microsoft (2015). *Serialization (C#)*. [online] Available at: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/ [Accessed 7 Nov. 2017].

Microsoft (2017). *Microsoft/MixedRealityToolkit-Unity*. [online] Available at: https://github.com/Microsoft/MixedRealityToolkit-Unity [Accessed 4 Nov. 2017].

Microsoft HoloLens. (2017). *Microsoft HoloLens*. [online] Available at: https://www.microsoft.com/en-au/hololens/partner-program [Accessed 5 Nov. 2017].

Moleta, T. (2016) ‘Game on: Exploring constructive design behaviors through the use of real-time virtual engines in architectural education’, International Journal of Architectural Computing, 14(3), pp. 212–218. doi: 10.1177/1478077116663341.

Norman, D. (1992). Design principles for cognitive artifacts. *Research in Engineering Design*, 4(1), pp.43-50.

Quobject. (2017). *Quobject/SocketIoClientDotNet*. [online] Available at: https://github.com/Quobject/SocketIoClientDotNet [Accessed 21 Oct. 2017].
Slater, R. and Radford, A. (2012) ‘Perceptions in the Australian building industry of deficiencies in architects ’ design documentation and the effects on project procurement’, Australasian Journal of Construction Economics and Building, 8(1), pp. 23–33.

Schumacher, P. (2009) ‘Parametricism: A new global style for architecture and urban design’, Architectural Design, 79(4), pp. 14–23. doi: 10.1002/ad.912.

Trimble (2017). *Sketchup Viewer* [online] Available at: https://www.sketchup.com/products/sketchup-viewer [Accessed 7 Nov. 2017].

Windows Mixed Reality (2017). *Windows Mixed Reality | AR Mixed with VR Gaming, Travel & Streaming in Windows 10*. [online] Microsoft.com. Available at: https://www.microsoft.com/en-au/windows/windows-mixed-reality [Accessed 7 Nov. 2017].

Image References

Figure 1:
Eric J. Forman (2013). *Firefly framework connects Grasshopper to Arduino*. [online] Available at: https://ericjformanteaching.wordpress.com/2013/09/11/firefly-framework-connects-grasshopper-to-arduino/ [Accessed 8 Nov. 2017].

Figure 2:
Trimble (2017). *Sketchup Viewer* [online] Available at: https://www.sketchup.com/products/sketchup-viewer [Accessed 7 Nov. 2017].

Figure 3:
IrisVR (2017). *IrisVR - Virtual Reality for Architecture, Engineering, and Construction* . [online] Available at: https://irisvr.com/ [Accessed 8 Nov. 2017].

Figure 4:
Google VR (2017). *Blocks - Create 3D models in VR - Google VR*. [online] Available at: https://vr.google.com/blocks/ [Accessed 8 Nov. 2017].

Figure 5:
Johanson, M., Khan, N., Asher, R., Butler, A., Haeusler, M. H., (2017) 'Urban pinboard', CAADRIA 2017 - 22nd International Conference on Computer-Aided Architectural Design Research in Asia: Protocols, Flows and Glitches, (Brynjolfsson 2014), pp. 715–724.

Figure 6:
CORE Studio (2017). *Platypus | CORE studio*. [online] Available at: http://core.thorntontomasetti.com/platypus/ [Accessed 30 Oct. 2017].

Figures 7, 8, 9, 10, 11, 12:
The author.