A User Interface Configurable via JSON for Augmented and Virtual Reality Applications

Laurence Ruberl
lcfruberl@gmail.com
Earlham College Department of Computer Science
Richmond, Indiana

ABSTRACT

A user interface that can be defined using JSON does not currently appear to exist for Augmented Reality applications. This project lays the groundwork to correct that and allow for all kinds of applications to leverage Augmented Reality as a system however people see fit. We used mostly open source hardware and software to achieve that goal, using the Project NorthStar headset, Leap Motion sensor, and associated software packages to make an interface that can be defined in JSON and instantly used in an application.

ACM Reference Format:
Laurence Ruberl. 2020. A User Interface Configurable via JSON for Augmented and Virtual Reality Applications. In *Proceedings of Earlham College Computer Science Senior Capstone*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

This paper will cover the background and design of a user interface for Virtual and Augmented Reality to fill in a gap in existing work on User Interfaces in those fields. We will discuss a selection of what has already been done in the field of Augmented and Virtual reality before discussing our implementation of such an interface.

The primary contribution of this paper is the groundwork for a User Interface that is configurable exclusively in JSON and is purpose-built for Augmented and Virtual Reality applications[13]. This groundwork overcomes some of the hurdles that might prevent a developer from making their own automatically generated interface by automating the steps necessary to make an interface manually.

2 BACKGROUND AND RELATED WORK

Most interfaces for any system or program are designed specifically for that singular purpose. For example, the Heads Up Display (HUD) for a video game is designed only for that specific game. This concept also extends to Augmented Reality. As an example, the interface for the Windows Home menu in Windows Mixed Reality headsets is only used and accessible by that one program. It would make designing some applications, as well as porting over existing non-Augmented Reality applications, much easier if there was a unified interface that could be quickly configured by a developer.

2.1 Previous Research

There is a litany of work related to designing interactions for Augmented Reality devices. Most of the earliest work utilizes application-specific hardware with tracking markers to keep track of positioning. This includes the work of Billinghurst and Grasset in 2005[2] in which they used a purpose-built tracking mat and physical "handle" as their interface and tracking mechanism. More recent work deals with hand tracking in relation to objects that appear to be in the same space as the user, similar to this project. An example is the work of Lee et al in 2018[8] in which they endeavored to determine the best method for a user to interact with digital windows. Some work deals with how to classify Augmented Reality systems, an idea that is useful to keep in mind as one works on such systems[4]. There is also work to catalogue and address the challenges of Augmented Reality applications, such as security concerns like those raised by Kiron Lebeck et al[7].

There is also a body of work dedicated to how to interact with Augmented Reality interfaces. Some research works with similar technology to the Leap Motion Sensor, specifically interacting with two hands[12][8].

2.2 Applications of Augmented and Virtual Reality in Other Fields

One of the fields that has attempted to integrate Augmented Reality into itself is that of Theatre. Most research has been devoted to how Augmented Reality can be used to enrich the audience experience or the artistic vision[1][5][14][3]. There is very little research to be found regarding using Augmented Reality in the management of Theatre, which is a gap this project hopes to inspire the filling of. One way in which something like this could be used is for Stage Managers during a show. Normally, they would have to look away from the stage to follow along in their script, but with an AR headset powered by an interface like the one detailed in this paper, they would only need to move their eyes to one side rather than their whole head.

3 DESIGN AND IMPLEMENTATION

3.1 Hardware

This project was designed and implemented for the Project NorthStar open source Augmented Reality headset started by the company formerly known as Leap Motion (now UltraLeap). The headset is primarily 3d-printed and uses custom electronics for the display driver. It projects objects into the user's vision by rendering them

Earlham College Computer Science Senior Capstone, Richmond, IN,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

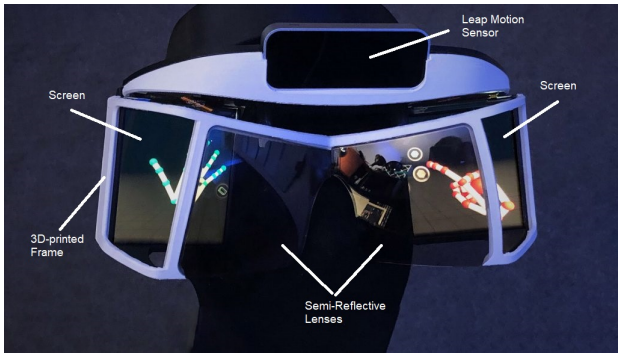

Figure 1: A rendering of the Project NorthStar Headset[11]

on two commercially available screens and then bouncing those images off of a pair of custom semi-reflective lenses into the wearer's eyes. This allows the user to see their real-world surrounds while having the digital objects overlaid into their field of vision. The headset displays are powered directly off of the user's computer and graphics card, utilizing DisplayPort for a graphics adapter and USB 3.0 for power. Hand tracking for the headset is provided by the proprietary Leap Motion sensor, which uses USB 2.0, mounted on the front of the headset[10].

Due to tracking restrictions with the NorthStar and COVID-19 preventing access to materials to rectify them, most testing for this project was done on a Lenovo Explorer Virtual Reality headset. The Lenovo Explorer is a Head Mounted Display (HMD) developed by Lenovo in cooperation with Microsoft as part of their Windows Mixed Reality program[9]. When using the Explorer, the Leap Motion Sensor from the NorthStar was mounted to the front using a developer mount sold by UltraLeap.

3.2 Software

The majority of this project was created in the Unity game engine using packages provided by the Project NorthStar development community and UltraLeap [16]. Most pregenerated assets, such as models for the user's hands used during testing, also came from those teams. We also used scripts from some of the examples provided with the Interaction Engine module. We used LitJson[6] for JSON parsing.

3.3 Architecture

The proof of concept interface that accompanies this paper utilizes a few types of interactables. When designing anything interactive there is a concept called *interactables*, meaning anything that the user can manipulate or interact with within their digital environment[17]. For example, a static object, like a table or a safe, that cannot be moved would not be an interactable, while something like a coffee cup or a door would be an interactable because the user can interact with and manipulate them. In our context, buttons and *draggables* are all interactables. *Panels*, on the other hand, are not interactables and are semi-static objects that hold other objects, such as buttons or sliders, in a single place. They all have an associated *draggable* that allows the user to move the panel to another location and pin it there. If the user tries to grab

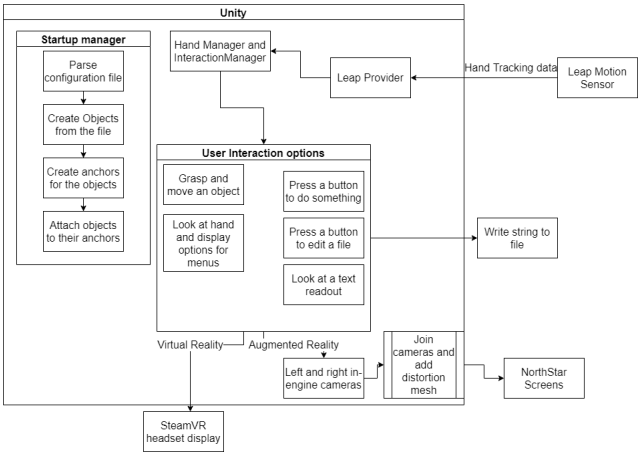

Figure 2: Architecture diagram

any other part of the panel, it will not respond to them in the same way. *Anchorables* are objects that can be picked up by the user and attached to an *anchor* which holds it in a specific place and enables it to be interacted with like a button or slider.

Some UI elements can also be bound to the user's hands. For example, by default, a pallet with interactables that the user can drag into the space can be displayed by taking their non-dominant hand and facing it toward them, conjuring the objects they have configured into the space so they can use their opposite hand to move them into the space. The pallet is grouped into columns of three anchors that each hold one object. More columns are added depending on the number of objects being generated. This design is directly based off of the work of the Leap Motion team's examples but was modified and expanded to include support for more than 3 objects

To customize the functionality of the different elements, the users can place definitions in a JSON file that the program reads from and configures the interface objects accordingly. For example, if the user wants a panel that displays the information recorded by a thermometer attached to their computer, they would write in the configuration file a label for the display and where the program should get the readout from. As another example, if the user wanted to have a button that spawns several readouts, they would put in the file that they want it to be a button and they'd configure the screens in the same way as the aforementioned display. In both these cases, once the interface program has loaded, they can interact with them exactly as they configured and intended.

3.4 Scenes

In the Unity engine, scenes are collections of objects existing at once alongside their associated scripts. In the words of the Unity documentation, "Think of each unique Scene file as a unique level"[15]. We used three Unity scenes: one as an initial scene to load in to, one dedicated to a Virtual Reality setup, and one dedicated to an Augmented Reality setup. The first scene exists as a place for the initial loading and parsing to occur and is not seen by the player.

The second and third scenes are very similiar but exist separately due to the differences required by the NorthStar. The Northstar

scene contains three main cameras[1], one for the left eye, one for the right, and one to combine the two images to display in the headset. The VR scene only has one camera . Both scenes contain a root LeapRig object that manages everything in regards to the Leap Motion sensor, including several sets of child objects. The Attachment Hands objects tracks and organizes objects attached to the User's hands, while the Interaction Manager keeps track of what those hands interact with. Attachment hands also contains the anchors that the User Interface objects attach to when they are not deployed. There are also objects to display the outlines of user's hands in the scene.

3.5 Interface Generation

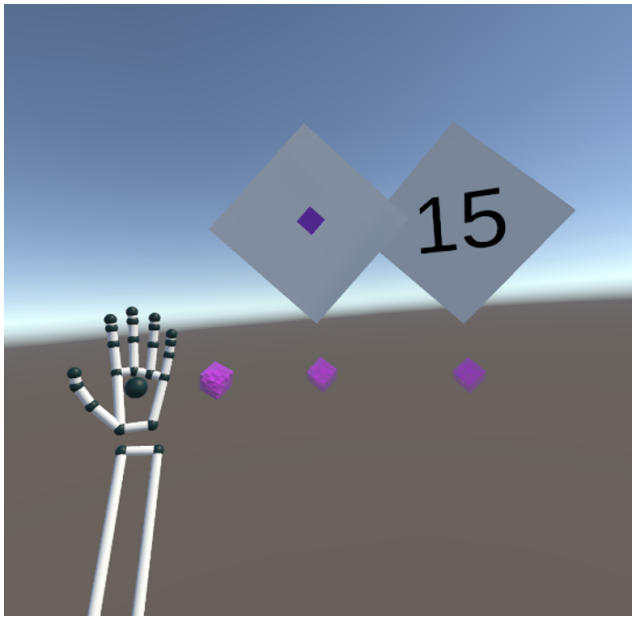

Figure 3: A development screenshot containing (from left to right) a user's hand with one collapsed UI object attached to their hand, a deployed button that adds to a text file when pressed, and a text readout counting the number of lines in a text file in real time.

The user interface generation is powered by a single class called RuntimeManager. When the program is started, it reads from the defined configuration file and loads it into an object using LitJSON. This PreferenceObject contains information such as which scene to load into and a list of the objects to generate into the scene. Then, depending on what the user defines in the configuration, the class loads into the scene built for the given type of headset. From there, the class determines which objects are which, such as where the Interaction Manager is and where the necessary attachment points are before iterating through the list of objects to see what needs to be generated.

[1]In this context, cameras are what a user would see looking from that specific point in the scene.

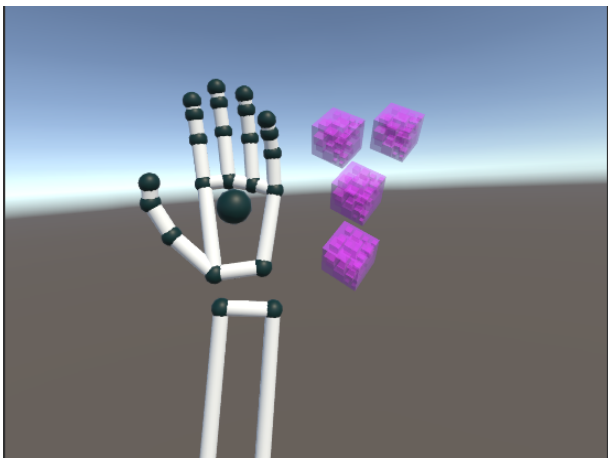

Figure 4: A development screenshot containing a User's hand with a pallet of 4 objects.

For each object, the class first determines the anchor it needs to be initially attached to. If there are no more free anchors remaining, it will generate a new pallet of three anchors as children of the left palm under the Attachment Hands, and shift it over a few units so the user can see all the objects they want. Then, the class creates a wrapper draggable object that user will use to move it, then associates it with an anchor before determining what kind of object should be generated. Finally, the class generates the actual object as a child of the wrapper object, tells it what object contains the interaction manager, and associates to the object any events that the user defined. For example, if a user defines a button that writes to a file when pressed, the class will create said button and add a listener that writes the user-defined string to the specified file. Once it is done, it keeps following the list to see if there are anymore objects to generate and then repeats the process.

A similar process is followed for data readouts. For data readouts, such as the readout of a thermometer attached to the computer, the program currently assumes that it is stored in a file on the user's filesystem. The class creates a text object in much the same vein as for the interactables above, however it attaches a script that will check the given file every frame update to see if it has changed in size. If it has, it reads the last line of the file and updates the text object accordingly. We used the FileInfo class in C# to get file information in order to limit the amount of input-output operations that would need to be done to get that information otherwise.

The above processes allow the user to look at their left hand and see a pallet of objects, pick one, drop into the space near them, and then either get data readouts from it or interact with it as they need.

4 RESULTS

The completed proof of concept demonstrates the feasibility of an interface generated from a user-defined text file for VR and AR. It automates much of the work necessary to built a UI within the framework of Project Northstar and the Leap Motion Sensor. Our

proof-of-concept contains two kinds of objects with the potential for more to be added in the future.

We hope that this project inspires in some way the creation of more applications for Augmented Reality by lowering the bar necessary to create software for those applications. While some work has been done on additional applications of Augmented and Virtual reality, the field is still relatively new has has many ways it can be expanded and we hope this paper contributes towards that.

5 FUTURE WORK

There are many ways the interface can be improved, some of which were postponed due to the impact of the COVID-19 Pandemic separating us from some of our equipment. Additional interactables can be added to the interface, such as sliders or toggle switches. There is the also the possibility of more control for the user when writing the configuration file, such as user defined variables within events. The ability to put multiple interactables on one panel is possible as well, allowing for clusters of buttons or buttons below their indicator. Another major improvement is the generalization of many of the methods such that the class can be used in any application, not just one custom made for it. This will allow future applications that need an interface to get started relatively quickly and easily. Another improvement that was put on hold due to COVID-19 was the addition of a tracking mechanism to the NorthStar headset. One can hook up a tracking system using IR LEDs attached to the headset and tracking data from a laptop camera or webcam.

ACKNOWLEDGMENTS

Dr. David Barbella, Dr. Charles Peck, The Earlham College Department of Computer Science, and The Project NorthStar Development Community and Discord server.

REFERENCES

[1] Dimitrios Batras and Thomas Morisset. 2015. DOLMENS: Presence and Autonomy in Digital Stages. *Proceedings of the 2015 Virtual Reality International Conference* (2015), 1–4. https://doi.org/10.1145/2806173.2806180

[2] Mark Billinghurst and Raphael Grasset. 2005. Designing Augmented Reality Interfaces Physical Elements Input Interaction Metaphor Display Elements Output. *Interface* (2005), 17–22. https://doi.org/10.1145/1057792.1057803

[3] Adrian David Cheok, Wang Weihua, Xubo Yang, Simon Prince, and Fong Siew Wan. 2002. Interactive Theatre Experience in Embodied + Wearable Mixed Reality Space. (2002), 1–10.

[4] Emmanuel Dubois and Laurence Nigay. 2000. Augmented Reality: Which Augmentation for Which Reality? *Dare 2000* April 2000, 1 (2000), 165–166. https://doi.org/10.1145/354666.354695

[5] Georges Gagneré, Cédric Plessiet, Andy Lavander, and Tim White. 2018. Challenges of movement quality using motion capture in theatre. *Moco* (2018), 1–6. https://doi.org/10.1145/3212721.3212883

[6] Mattias Karlsson. 2020. LitJSON/litjson. https://github.com/LitJSON/litjson original-date: 2013-04-05T17:05:02Z.

[7] Kiron Lebeck, Tadayoshi Kohno, and Franziska Roesner. 2016. How to Safely Augment Reality : Challenges and Directions. *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications* (2016), 45–50. https://doi.org/10.1145/2873587.2873595

[8] Joon Hyub Lee, Sang-Gyun An, Yongkwan Kim, and Seok-Hyung Bae. 2018. Projective Windows. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (2018), 1–8. https://doi.org/10.1145/3173574.3173792

[9] Lenovo. [n.d.]. Lenovo Explorer | Mixed Reality Headset | Lenovo US. https://www.lenovo.com/us/en/virtual-reality-and-smart-devices/virtual-and-augmented-reality/lenovo-explorer/Lenovo-Explorer/p/G10NREAG0A2 Library Catalog: www.lenovo.com.

[10] Leap Motion. 2020. leapmotion/ProjectNorthStar. https://github.com/leapmotion/ProjectNorthStar original-date: 2018-06-05T01:00:43Z.

[11] Smart Prototyping. [n.d.]. First Project North Star Kit now available for the masses - Polaris AR. https://www.smart-prototyping.com/blog/Polaris-AR-releases-Project-North-Star-Kit-for-the-masses Library Catalog: www.smart-prototyping.com.

[12] Eun Joo Rhee, Seiheui Han, Junyeong Choi, and Jong-il Park. 2011. An Interface between Users and Virtual Objects Using Two Hands. (2011), 441–442.

[13] Laurence Ruberl. [n.d.]. senior-capstones-2020 / Laurence Ruberl Capstone. https://gitlab.cluster.earlham.edu/senior-capstones-2020/laurence-ruberl-capstone Library Catalog: gitlab.cluster.earlham.edu.

[14] Eric Sauda and Chris Beorkrem. 2006. Theatre of embedded intelligence. *ACM SIGGRAPH 2006 Sketches on - SIGGRAPH '06* (2006), 150. https://doi.org/10.1145/1179849.1180037

[15] Unity Technologies. [n.d.]. Unity - Manual: Scenes. https://docs.unity3d.com/Manual/CreatingScenes.html Library Catalog: docs.unity3d.com.

[16] UltraLeap and Leap Motion. [n.d.]. Unity. http://developer.leapmotion.com/unity Library Catalog: developer.leapmotion.com.

[17] Mikael Wiberg. 2017. From interactables to architectonic interaction. *interactions* 24, 2 (Feb. 2017), 62–65. https://doi.org/10.1145/3036203