

Immersive Visualization in Biomedical Computational Fluid Dynamics and Didactic Teaching and Learning

John Thomas Venn
Marquette University

Recommended Citation

Venn, John Thomas, "Immersive Visualization in Biomedical Computational Fluid Dynamics and Didactic Teaching and Learning" (2018). *Master's Theses (2009 -)*. 459.
https://epublications.marquette.edu/theses_open/459

IMMERSIVE VISUALIZATION IN BIOMEDICAL COMPUTATIONAL FLUID
DYNAMICS AND DIDACTIC TEACHING AND LEARNING

By

John T. Venn

A Thesis Submitted to the Faculty of the Graduate School,
Marquette University,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science

Milwaukee, Wisconsin

May 2018

ABSTRACT

IMMERSIVE VISUALIZATION IN BIOMEDICAL COMPUTATIONAL FLUID DYNAMICS AND DIDACTIC TEACHING AND LEARNING

John T. Venn

Marquette University, 2018

Virtual reality (VR) can stimulate active learning, critical thinking, decision making and improved performance. It requires a medium to show virtual content, which is called a virtual environment (VE). The MARquette Visualization Lab (MARVL) is an example of a VE. Robust processes and workflows that allow for the creation of content for use within MARVL further increases the userbase for this valuable resource.

A workflow was created to display biomedical computational fluid dynamics (CFD) and complementary data in a wide range of VE's. This allows a researcher to study the simulation in its natural three-dimensional (3D) morphology. In addition, it is an exciting way to extract more information from CFD results by taking advantage of improved depth cues, a larger display canvas, custom interactivity, and an immersive approach that surrounds the researcher.

The CFD to VR workflow was designed to be basic enough for a novice user. It is also used as a tool to foster collaboration between engineers and clinicians. The workflow aimed to support results from common CFD software packages and across clinical research areas. ParaView, Blender and Unity were used in the workflow to take standard CFD files and process them for viewing in VR. Designated scripts were written to automate the steps implemented in each software package. The workflow was successfully completed across multiple biomedical vessels, scales and applications including: the aorta with application to congenital cardiovascular disease, the Circle of Willis with respect to cerebral aneurysms, and the airway for surgical treatment planning. The workflow was completed by novice users in approximately an hour.

Bringing VR further into didactic teaching within academia allows students to be fully immersed in their respective subject matter, thereby increasing the students' sense of presence, understanding and enthusiasm. MARVL is a space for collaborative learning that also offers an immersive, virtual experience. A workflow was created to view PowerPoint presentations in 3D using MARVL. A resulting Immersive PowerPoint workflow used PowerPoint, Unity and other open-source software packages to display the PowerPoint presentations in 3D. The Immersive PowerPoint workflow can be completed in under thirty minutes.

ACKNOWLEDGMENTS

John T. Venn

I would like to gratefully acknowledge my advisor, Dr. John LaDisa who has a passion for research and knowledge. I often wondered how he could be involved in such a vast array of disciplines and have intimate knowledge in each subject area. He underscores the importance of professionalism and work ethic. The opportunities Dr. LaDisa presented me will forever be appreciated. I would also like to thank him for funding me throughout my time at Marquette, via a MARVL research assistantship and Marquette's Lafferty project. Also, I would like to thank the many individuals behind Marquette's Lafferty project for suggesting additional involvement by Marquette's engineering professors and MARVL. I truly believe that an engineering professor's research would benefit greatly by the use of MARVL.

Secondly, I would like to thank Marquette's Visualization Specialist, Chris Larkee. I came to Marquette with minimal knowledge of VR and nominal experience in coding. Chris took me under his wing and taught me most everything there is to know about VR. He was happy to help whether I was lost, frustrated, or needed his feedback. Chris's help will never be forgotten. I truly appreciate his efforts!

A special thanks to Dr. Garcia and Dr. Rayz for not only serving on my committee but also for providing me with their CFD research. I would have not been able to advance the CFD to VR workflow without their efforts. Dr. Vanhille and Dr. Garcia were instrumental in presenting some of MARVL's CFD to VR work at the Advances in Rhinoplasty Conference in Chicago, IL., thank you very much. Dr. Choi and Dr. Sachdeva provided their clinical feedback on select biomedical CFD VR models. Their assistance is much appreciated. I would also like to recognize Jesse Gerringer, Evelyn Granados Centeno, and the Marquette Engineering professors who volunteered to test and or provide feedback for both of my workflows.

Lastly, I would like to thank my friends and family for their continuous support. Most notably my mom and dad, Ann Marie and Tom Venn. Whenever I needed a pick-me-up or motivation, they were there for me.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	i
LIST OF ABBREVIATIONS & ACRONYMS	vi
CHAPTER 1: INTRODUCTION.....	1
1.1 Specific Aims.....	1
1.2 Virtual Reality	4
1.2.1 Virtual and Augmented Reality Defined	4
1.2.2 CFD in Immersive VR.....	5
1.2.2.1 Visualization as a Solution.....	9
1.2.3 Academe in VR	9
1.2.4 Components Recommended for a Quality Virtual Experience	14
1.3 Visualization Theories and Methods	15
1.3.1 Depth Perception	15
1.3.2 Field of View	20
1.3.3 Stereoscopy.....	21
1.3.3.1 Color Multiplexed Approach	22
1.3.3.2 Polarization Multiplexed Approach.....	23
1.3.3.3 Time Multiplexed Approach.....	24
1.4 VR Visualization Tools	24
1.4.1 CAVE®	25
1.4.2 Standalone 3D Projectors	27
1.4.3 Head Mounted Displays	27
1.4.3.1 Samsung Gear VR.....	28
1.4.3.2 Oculus Rift.....	29

1.4.3.3 Microsoft HoloLens	30
1.5 Challenges of VR	31
1.6 Biomedical Computational Fluid Dynamics	33
1.6.1 Pre-Processing	34
1.6.2 Solving Governing Equations.....	36
1.6.3 Post-Processing.....	39
CHAPTER 2: CFD TO VR.....	41
2.1 Previous Work Done at MARVL.....	41
2.2 Materials and Methods	43
2.2.1 Workflow Requirements.....	45
2.2.2 Format of CFD Results.....	47
2.2.3 Convert CFD Results Into a 3D Format	49
2.2.4 Reorganizing of CFD Results.....	52
2.2.5 Add Supplementary Data.....	55
2.2.6 Customize for a Given VR Environment	56
2.2.6.1 Unity Template	57
2.2.6.2 Load .fbx File.....	58
2.2.6.3 Scale, Rotate, and or Reposition CFD simulation	60
2.2.6.4 Run Editor Script	61
2.2.6.5 Manually Add Supplementary Data.....	63
2.2.7 Arrange VR Environment.....	66
2.3 Results	69
2.3.1 Cases	69

2.3.1.1 Thoracic Aorta Simulation.....	71
2.3.1.2 Brain Aneurysm Simulation	75
2.3.1.3 Airway Simulation.....	79
2.4 Discussion.....	83
2.4.1 Novice Users CFD to VR Workflow Experience.....	83
2.4.2 Clinical Feedback from Derivative Project	86
2.4.2.1 Virtual Nasal Surgery	86
2.4.2.2 Summary	90
2.5 Limitations and Future Directions	91
CHAPTER 3: IMMERSIVE POWERPOINT	94
3.1 Previous Work Done at MARVL.....	94
3.2 Materials and Methods	95
3.2.1 Data Acquisition	95
3.2.2 Converting Slides	96
3.2.3 Transform to Meet Virtual Environment.....	97
3.3 Results	100
3.3.1 Case Study	100
3.4 Discussion.....	101
3.5 Limitations and Future Directions	101
CHAPTER 4: CONCLUSION.....	103
BIBLIOGRAPHY	105
Appendix A: Convert CFD Results into Streamlines Using ParaView	113
Appendix B: Convert CFD Results into Glyphs Using ParaView.....	120

Appendix C: Create/Turn Off Flow Waveform in Unity	126
Appendix D: Label Scales in Unity	130

LIST OF ABBREVIATIONS & ACRONYMS

2D – Two dimensional	IVE – Immersive virtual environment
3D – Three dimensional	MRA – Magnetic resonance angiography
AR – Augmented reality	MRI – Magnetic resonance imaging
CAD – Computer-aided design	MARVL – MARquette Visualization Lab
CAVE® – Computer- automatic virtual environment	NS – Navier-Stokes
CFD – Computational fluid dynamics	OCT – Optical coherence tomography
CoA – Coarctation of the aorta	OCOE – Opus College of Engineering
CT – Computer tomography	OSI – Oscillatory shear index
DICOM – Digital imaging and communication in medicine	PC-MRI – Phase contrast-magnetic resonance imaging
EVL – Electronic Visualization Lab	SDK – Software development kit
FSI – Fluid-structure interaction	TAWSS – Time-averaged wall shear stress
FOV – Field of view	VE – Virtual environment
GUI – Graphical user interface	VR – Virtual reality
HMD – Head mounted display	VVG – Verhoeff-Van Gieson stain
HUD – Head-up display	WSS – Wall shear stress
IHC – Immunohistochemical stain	

CHAPTER 1: INTRODUCTION

1.1 Specific Aims

Experiences that allow for motion within a realistic environment promote active learning, critical thinking, improved decision making and better performance [1, 2, 3].

This is the basis for the Opus College of Engineering's (OCOE) MARquette Visualization Lab (MARVL), which opened within Marquette's Engineering Hall on January 16, 2014 as a facility to (1) demonstrate how advanced visualization technology can be used in learning, research, and industry applications, (2) teach the theory rooted in this technology, and (3) create technologically advantageous visualization content [4].

The ability to have robust processes and workflows that allow for the creation of content for use within MARVL and its available head-mounted-displays (HMD) would further increase the userbase for these important resources. As seen in Figure 1, the objective of this thesis was to create two semi-automatic workflows for this purpose. In doing so, the current work aims to demonstrate (1) the practicality of using advanced visualization via virtual and augmented reality in biomedical computational fluid dynamic (CFD) research and (2) presentation of didactic lectures and related content for use in teaching.

Additional details are included below.

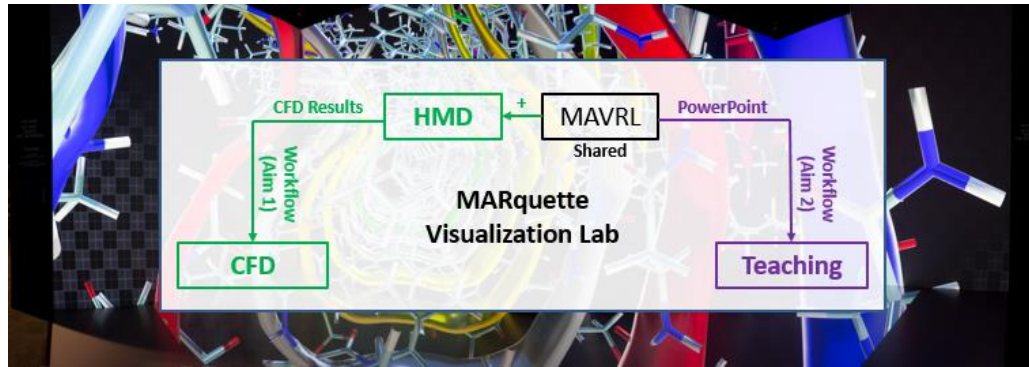


Figure 1: Two workflows were created for the current thesis in order to increase and extend usage of the MARquette Visualization Lab. Green indicates the Aim 1 workflow. Purple indicates the Aim 2 workflow.

Aim 1: Develop a semi-automatic workflow that combines biomedical computational fluid dynamic results and subsequent complementary data to be viewed in large-scale and head-mounted virtual environments.

CFD results are typically viewed on a standard computer screen and presented slowly as sequential time steps. In addition, CFD results are seldom viewed with complementary data that could be beneficial when understanding pathology. Virtual environments (VE) can attenuate many of these issues by taking advantage of improved depth cues, a larger display canvas, custom interactivity, and immersive approach that surrounds the researcher [13,14]. The application of this workflow is used across multiple biomedical vessels, scales and applications including: the aorta with application to congenital cardiovascular disease, the Circle of Willis with respect to cerebral aneurysms, and the airway for surgical treatment planning.

Aim 2: Create an easy to use, interactive and rapid workflow that converts a standard PowerPoint slideshow, for viewing in 3D using Marquette's Visualization Lab.

MARVL within Marquette University's OCOE contains a large-scale computer-automatic virtual environment (CAVE®). VE hold tremendous potential as an educational tool. Multiple studies have shown the advantages of using VR in didactic teaching and learning [18-38]. As a starting point, the current workflow was created to allow faculty, staff and instructors within the OCOE to present lectures and presentations with added depth cues using the CAVE® within MARVL.

1.2 Virtual Reality

1.2.1 Virtual and Augmented Reality Defined

Virtual reality (VR) according to McMenemy et al. is defined as “a computer-generated 3D environment within which users can participate in real time and experience a sensation of being there” [5]. VR can be shown in a variety of VE, defined as a computer-generated space where a user can interact with a virtual element [6]. VE include large-scale immersive virtual environments (IVE) such as the CAVE®’s, HMD, and select three-dimensional (3D) projectors. Many IVE are customizable, immersive, semi-enclosed rooms that display virtual, 3D content when a person wears site-specific glasses [5]. HMD are devices that are worn on a person’s head to display virtual content in front of his or her eyes [5]. Select 3D projectors display virtual content onto a standard projector screen that appears in 3D when a person wears projector-specific glasses. In recent years, the price of VE’s have decreased, while the quality has increased, leading to more people using VR [7, 8].

Augmented reality (AR) is another term commonly associated with VR [5]. AR takes place in the real, physical world where the user can interact with a superimposed virtual element [9]. An example of AR, illustrated in Figure 2, is the popular smartphone app, Pokémon Go. The app works by using a smartphone’s camera and GPS. In the application, GPS is used to find AR cartoon avatars placed throughout the world. When the user finds the general area of the cartoon avatar using GPS, the smartphone camera is used to precisely locate the avatar cartoon. For example, in Figure 2, a cartoon avatar is

highlighted in the red square, the user does not see the cartoon avatar in the real world, but when the user looks at the real world through his or her smartphone camera, the cartoon avatar appears.



Figure 2: Augmented reality, Pokémon Go application [10]

1.2.2 CFD in Immersive VR

An area where VR could have a positive impact is in biomedical CFD research [13-16]. CFD is a method to generate computer simulations using numerical analysis and algorithms to replicate or solve problems that involve fluid flow. It is also used to illustrate spatial and temporal patterns of fluid flow. In the biomedical field, CFD allows for highly repeatable simulations throughout the body to replicate complex situations that are difficult to measure experimentally due to high costs, non-existent methods, or the complexity of the physiologic situation [11].

CFD researchers often spend weeks determining boundary conditions, running simulations and solving governing equations [12]. However, CFD results are typically viewed on a two-dimensional (2D) display, at one point in time, or very slowly at sequential time steps without the aid of complementary data. This means that only a fraction of the CFD results are being studied in detail, at an extremely slow rate, resulting in associations from related data possibly going unnoticed. In addition, individuals who did not perform the CFD analysis, for example clinicians who would like to review CFD results, may not appreciate the utility of CFD results when operating currently available visualization tools [13]. VR alleviates many of these issues by taking advantage of improved depth cues, a larger display canvas, custom interactivity, and an immersive approach that surrounds the researcher [14].

The first attempt to bring biomedical CFD research into VR was in 2000 by Andrew Forseberg and colleagues [13]. Their immersive simulation explored CFD results from a coronary artery graft and aimed to better uncover approaches to reduce failure rates in artery grafts. The VR-based simulation allowed the viewer to study changes in a vessel's flow patterns and geometry in addition to exploring possible lesion sources [13]. The virtual simulation was built for a 4-walled CAVE® (3 walls and a floor). The users manipulated and moved simulation results around using a widget, voice, and hand commands. The feedback was mostly positive, but the paper noted that “significant time” was put toward reformatting the simulation results before being viewed in VR [13].

Quam et al. developed a semi-automatic workflow for the “import, processing, rendering, and stereoscopic visualization of high resolution, patient-specific imaging data, and CFD results in an IVE” [14]. In turn, the workflow expedited the process of

reformatting select CFD results to be viewed in a CAVE®, which was an issue in Forseberg et al's. work. The workflow imitated a pulsating cardiovascular vessel by using velocity vectors to represent blood flow. Each velocity vector for each time step had the same coordinates, but different magnitudes (dependent on the speed in each particular time step and vector location). When each time step was rapidly played in succession, blood flow within the featured arteries appeared as though they were pulsating. The workflow could also display select hemodynamic parameters and patient-specific imaging data used to create the model. The workflow used MATLAB (The MathWorks; Natick, MA) as an intermediary tool to convert the CFD results into files that were compatible with VR. The converted files were then imported into the IVE using custom Visual Basic scripts [14]. EON STUDIO 7 (EON Reality; Irvine, CA) was used to display the content in Discovery World Science and Technology Museum's IVE (Milwaukee, WI), where the cardiovascular CFD results were viewed in VR. The workflow was successfully tested on two CFD data sets. The first was a left common carotid artery. The second was a left-circumflex coronary artery after stenting and again after a 9-month follow-up period [14, 15].

Though Forseberg et al, Quam et. al, and likely other researchers, advanced the application of CFD to the field of VR, the applications lacked accessibility and functionality. Preparation of CFD results for use in VE was conducted on in-house computer software packages where only the individuals who built, tested, ran and operated the simulations could advance them to the point of use within the VE [13]. In 2015, Berg et al. developed a pipeline that visualized architectural CFD research used for urban planning via an open source software called Unity (Unity Technologies; San

Francisco, CA). Unity is a multipurpose game engine used to display both 2D and 3D graphics. Unity is also compatible with almost every HMD and is used as the primary gaming engine in many CAVE®'s throughout the country, including MARVL. The workflow was designed to “bridge the gap between architects and engineers” [16].

Despite this potentially notable advancement with the use of Unity, many architects are unfamiliar with post-processing CFD software and would prefer software to be more user friendly and allow for “real-time exploration of results” [16]. The paper noted that Unity could fix many of these shortcomings. The workflow's input requires the numerical CFD results from Fluent (ANSYS Inc; Canonsburg, PA) and the 3D computer-aided design (CAD) geometry of the architecture. The numerical CFD results were either pre-processed or directly imported into the Unity game engine as either .txt, .csv, .dat, or .xml files. The 3D CAD files were imported into the Unity game engine as a .fbx or .obj file. After the Unity scene was properly set up, the user could easily visualize and interact with the results [16]. Though the workflow used Unity, it did not take the extra step of viewing the results in a VE. The paper did not note whether the workflow could be completed by any user, or if the workflow needed to be completed by the engineer before being presented to the architect. In addition, the paper neglected to mention whether the workflow was compatible with a variety of CFD software packages, or if it was only for use with CFD results from Fluent.

1.2.2.1 Visualization as a Solution

CFD results can be analyzed using tabulated results in spreadsheets or in common visualization programs such as ParaView (Kitware, Inc; Clifton Park, NY). Visualization is the basis of any virtual or augmented reality device [5]. Visualization was originally defined in a paper written by DeFanti et al in 1987 stating: “Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen... it is a tool for both interpreting image data fed into a computer, and generating images from complex multi-dimensional data sets” [17]. Based on this definition, visualization through the use of a VR or AR device can be used to illustrate trends or anomalies in experimental or theoretical data, which would otherwise be difficult to recognize in tabular form [5]. VR offers the user an alternative to view and/or study data by taking advantage of improved depth cues, custom interactivity, and an immersive approach that surrounds his or her self [14].

1.2.3 Academe in VR

In recent years, there has been an increased investment in academia using VR [8]. For example, in 2015, Case Western Reserve and Cleveland Clinic announced a partnership with Microsoft HoloLens, an AR-based HMD. As a result of this partnership, beginning in the summer of 2019, nursing, dental, and medical students will no longer learn anatomy using traditional cadaver-filled laboratories. Instead, as shown in Figure 3,

the students will use AR to learn human anatomy [18, 19]. One of the benefits of VR is that students are fully immersed in their respective subject matter, thereby increasing their sense of presence, understanding and enthusiasm [21-38].

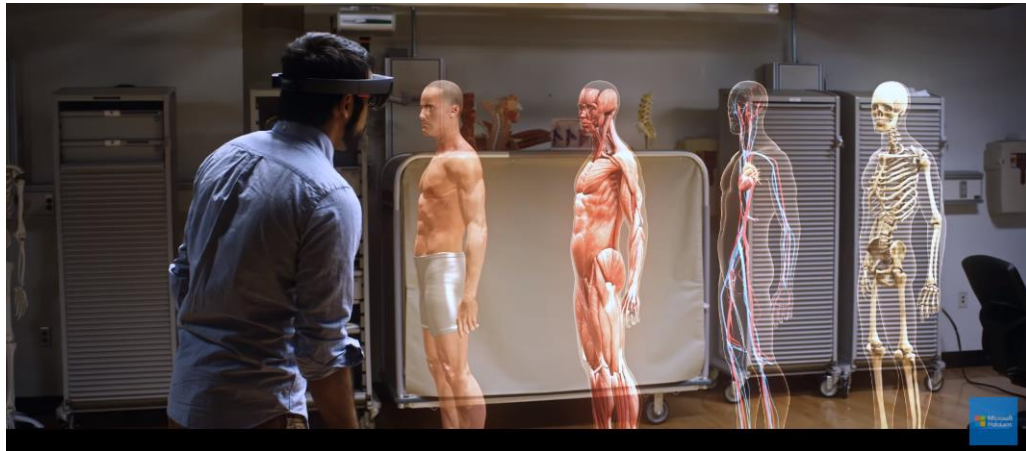


Figure 3: HoloLens based AR application that will be implemented at Case Western Reserve and Cleveland Clinic [20]

Compared to a standard classroom, VR provides a higher sense of presence (i.e. “being there”) by using visual, haptic (touch), auditory and/or sensory feedback. As more senses are stimulated, the student feels immersed and present in the VE, which results in triggering semantic and physiological associations from the student’s previous experiences and or assumptions [21]. A study by Everson et al. tested nursing students’ apathy for patients in third world clinical settings [22]. The rationale for this work was that prior studies have shown that health professionals had less empathy for culturally and linguistically diverse patients. The lack of empathy could have grave consequences leading to ignored or under-treated patients from different racial or ethnic backgrounds. The simulation positioned nursing students in a third world community hospital where

they were exposed to a different language, clinical practice, and unfamiliar smells and tactile stimuli. After the simulation, nursing students showed a significant improvement in empathy towards racially and culturally diverse patients [22].

VR is also an excellent tool for “learning by doing.” For years, medical professionals have operated on cadavers or mannequins to learn, practice and hone their curative skills [19]. The importance of “learning by doing” via simulations allows the user to construct personal knowledge and refine his or her understanding of the task at hand in a way consistent with cognitive constructional learning theories [23, 24]. Real life simulations though, are difficult to replicate, are not cost effective, and or consume a large amount of space [27-29]. VR can mitigate these issues. An example of a real-world simulation are physical simulation centers used by nursing students at multiple universities throughout the world. These simulation centers are designed to replicate clinical hospital settings where life-like mannequins are used to practice specific skills, like a woman giving birth. Simulation centers require a significant amount of space and are expensive to maintain. Marquette and Purdue Universities were able to recreate the simulation environments using CAVE® [25, 26]. VR simulation environments reduce the amount of space needed by altering the VE to replicate a wide variety of clinical settings [27].

It is also sometimes very difficult to replicate scenarios from the real world [29]. A study done by Lee et al. created an immersive, post-disaster virtual world. The simulation used visual graphics, as well as sounds and smoke to recreate what disaster relief personnel would see after a catastrophic event. The simulation allowed disaster

relief personal to practice how to treat the wounded and mark the dead. In this scenario, it would be almost impossible to replicate this simulation without the aid of VR [28, 29].

The medical field has begun creating virtual simulations for training of certain surgeries/procedures. For example, Rhienmora et al. created an AR, force and kinematic feedback, dental simulator [30]. The simulator was designed for dental students to practice dental surgery. As seen in Figure 4, when the student puts on the AR-HMD, a “virtual” tooth and mirror appear. When the student moves a dental drill and starts operating (changes geometry) on the tooth, force and kinematic feedback are provided to the user [30].

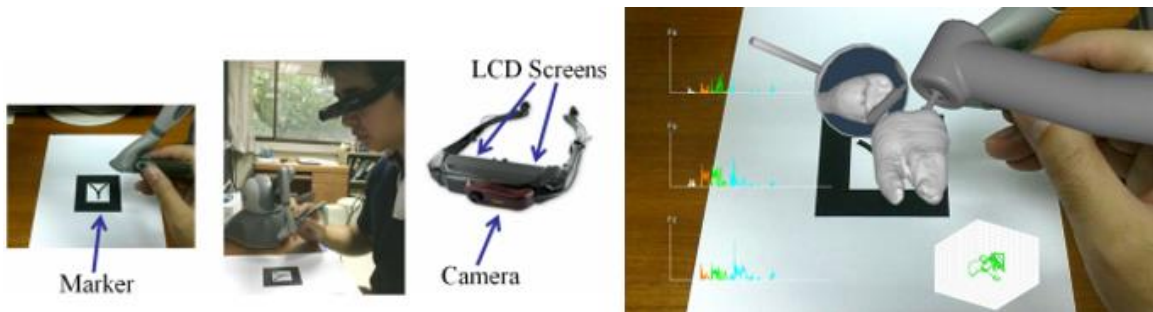


Figure 4: AR-based dental training as discussed in Rhienmora et al. [30]

In contrast, arguments can be made that computer-based learning does not accurately represent what is going on in a clinical setting. Moule et al. investigated whether computer-based learning in conjunction with practical instruction was comparable to common classroom methods when studying the use of an automated external defibrillator. The study determined that there was no significant difference

between e-learning and classroom-based learning, suggesting that computer-based learning was a viable substitute for classroom-based learning [31].

VE are also used to enhance visualization and allow users to view content from a range of perspectives [32]. For example, when studying architecture, it is advantageous for the student to view the structure from every angle to understand how different lines of sight affect a person's perception of a structure. Thorndyke et al. investigated whether spatial knowledge would result in different conclusions. The study investigated the knowledge acquired via a birds-eye-view vs. a first-person view when determining how to get from one place to another. When the person uses a map (birds-eye-view) he or she has a better idea of location and straight-line distances between where they would start and the target. When the person was on the ground (first-person view), he or she had a better idea on how long it would take to reach the target [32, 33]. If those two perspectives were viewed together, it would help the person better navigate through traffic. VR allows the user to view virtual objects from a variety of perspectives, hence more information is presented to make decisions.

One important aspect of any educational setting is collaborative learning. Studies have shown that students who are in social learning environments outperform students in individualistic settings [34]. In social learning environments, students can bounce ideas off one another and walk through a specific problem and or task. Collaborative learning environments that also incorporate some type of game, offer the users a competitive setting, motivating them to learn [35, 36]. VR can achieve a collaborative learning via avatars or a CAVE®. Avatars are computer-generated graphics that represent the user. In the VE, each student and teacher have their own avatar that represents his or her self.

When the student feels comfortable and safe with his or her character, there is an increase in creativity and imagination [37]. More advanced avatar based virtual systems track the user's head and hands. Those actions are then implemented to the user's avatar at a rate of 60 Hz [32]. One advantage of avatar-based learning is that users can be anywhere in the world and "meet" in the VE by using a HMD. A CAVE® is another avenue to incorporate VR in a collaborative learning environment. The students can see and interact with virtual elements by wearing site specific glasses but are still able to collaborate with fellow students and the professor [38].

VR can also be used to increase student's enthusiasm about a topic. In a study done by Limniou et al. students previously viewed chemical reactions using standard 2D animations. When the class was brought into a CAVE®, the students were more engaged in the lecture because they could interact with the objects and feel like they were inside the chemical reaction. The paper noted multiple times how enthusiastic the students were when viewing the reactions in VR [38].

1.2.4 Components Recommended for a Quality Virtual Experience

To develop VR applications for CFD or teaching/training experiences, one of the goals for the developer is to 'trick' the user's state of awareness into believing he or she is in a virtual world [5]. According to Sherman et al., there are 4 key components for a quality virtual experience: virtual world, immersion, sensory feedback, and interactivity [9]. The virtual world is the VE, which can be displayed in an IVE, HMD or 3D projectors. Immersion is defined by Stuart et al. as "presentation of sensory cues that

convey perceptually to the users that they're surrounded by a computer-generated environment" [6]. Sensory feedback is the sensory information provided by the sensory system including visual, auditory and haptic feedback. For the VE to accommodate the user's sensory and auditory feedback, the VE must track the user's position. This can be done by head or hand tracking [9]. More advanced VE also incorporate artificially created haptic feedback [9]. Interaction in VR is the communication/sync between the user and the VE. The most important interaction in VR, is that the VE changes based on the user's viewpoint [9]. The user should be able to walk around an object, while the virtual scene adjusts accordingly. More advanced virtual systems allow the user to physically interact with game objects by picking them up, rotating, and throwing the object(s). This typically results in higher immersion [9].

1.3 Visualization Theories and Methods

For a user to understand how he or she interprets a virtual element, he or she must first understand how they construe 3D objects in the real world. There are three main theories behind vision including depth perception, field of view, and stereoscopy.

1.3.1 Depth Perception

Depth perception provides humans with the ability to see objects in 3D. The main principle behind depth perception is the Cue Theory [39]. The theory states that humans interpret objects in 3D by using three cue groups: oculomotor cues, monocular cues, and

binocular depth perception. Oculomotor cues are created by the convergence and accommodation of an individual's eye and eye lens that move and flex when focusing on nearby objects. The concept behind this is that an individual can feel his or her eye move, and eye lens tighten [39]. As seen in Figure 5, oculomotor cues are effective only at very short distances, and are therefore ineffective in CAVE settings. When using HMD, oculomotor cues can result in some discomfort [40].

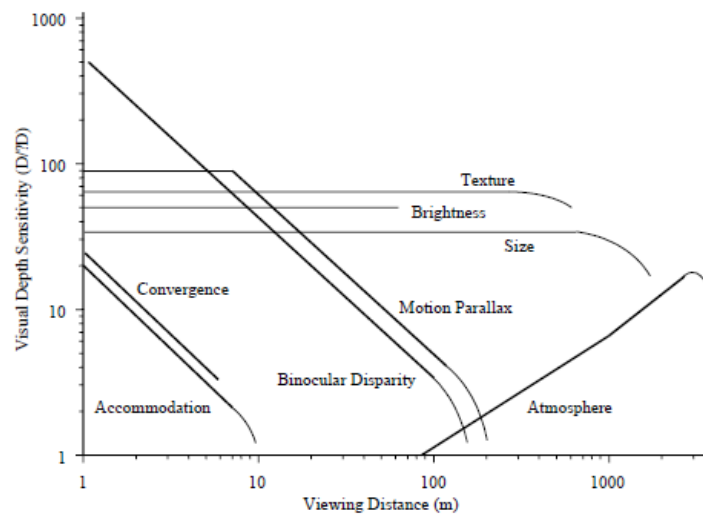


Figure 5: Importance of cues relative to viewing distance [40]

Monocular cues work for one eye and include the following subgroups: occlusion, size and position cues, aerial perspective, and motion cues [41]. Occlusion is a cue that interpolates whether an object is in front or behind another object. As shown in Figure 6, an individual will decipher that the circle is the most forward object and the triangle is the object that is furthest back. This interpolation is found almost everywhere within the physical world, and it is considered to be the most reliable depth cue [41]. Size and position cues take advantage of an individual's pre-programed visual system. A person

infers that if there are two of the same objects but one of the objects is closer, the closer object should be larger. Texture gradients and relative heights also play a large role in size and position cues. Texture gradients are used when observing a scene, if all the objects in the scene are evenly spread out, the objects furthest away appear closer together. An example of texture gradients can be found in Figure 6 where marathon runners appear more tightly packed further away [39]. Relative height is the observation when objects near to the ground appear closer, and objects at the top appear further away. Aerial perspective refers to the depth cue where further away objects appear less defined. This is the result of light being scattered through the atmosphere. Further away objects have more time for the light to become scattered, therefore making the object appear less distinct [41]. Motion cues refer to a geometric relationship when a person moves their eyes, objects closest to the person will move the furthest away. Motion parallax is the theory behind this phenomenon. Motion parallax defined by Wolfe et al. is “The geometric information obtained from an eye in two different positions at two different times is similar to the information from two eyes in different positions in the head at the same time” [41]. Although some monocular cues in the virtual world can be easy to achieve, many can be computationally expensive. For example, shaders are used to make objects appear more realistic by altering an object’s lighting, darkness, and or color. Typically, computationally expensive shaders look more realistic but can result in a slower frame rate [40].



Figure 6: Monocular depth cues. Left- occlusion. Right- texture gradient [41]

Binocular depth perception is the angular information taken from both eyes to determine the depth of an object. The term vergence, is exemplified when an individual looks at an object that is close to their face, and their eyes converge to focus on the object. When an individual adjusts their focus from a close object to an object that is further away, the eyes diverge [41]. Vergence works in conjunction with stereovision. Stereovision describes the relationship between the location of an object in each eye's retina to the distance of an object. An example of stereovision can be found in Figure 7 [42]. On the table, the apple and pear are the same distance from the person's face. Therefore, the apple and pear are spaced the same distance apart, in the same location, in both the right and left retina, this term is referred to corresponding points [42]. Meanwhile, on the table, the flowers are behind the apple. When looking at the individual's retinas, the right retina has the flowers aligned behind the apple, while the left retina has the flowers next to the apple. The difference in distance causes the images in the left and right eye to have different spacing, this term is called non-corresponding

points [42]. Stereovision allows an individual to infer the different distances from different objects and can be used to artificially create 3D images [42]. Sir Charles Wheatstone was the first person to use stereovision to create 3D displays using 2D images when he invented the stereoscope in the 1830's [41]. As seen in Figure 8, stereoscopes work by taking two pictures of the same image at slightly different angles. When the individual views the stereo photo through the stereoscope, the individual sees a 3D image. Currently, stereovision is the primary driving force for many HMD [51-58].

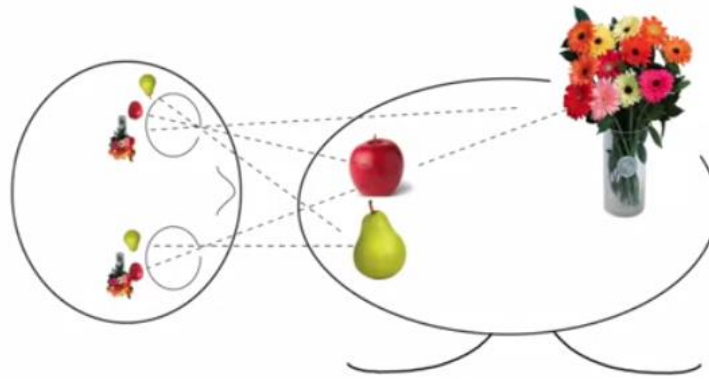


Figure 7: Stereovision. A person observing an apple, pear, and flowers on a table. With the corresponding alignment of the objects in the persons left and right retina [42]



Figure 8: Stereoscope using stereovision [42]

1.3.2 Field of View

A major component that affects the quality of an immersive display is the field of view (FOV) [6]. It is defined as the part in space that a person can see without moving their head [6]. As seen in Figure 9, the FOV of the human eyes, represented by the dark thick lines, is quite large covering roughly 150° horizontally and 120° vertically, and includes an overlap where stereoscopy occurs called binocular field of view [40]. A large FOV makes a person feel fully immersed in the real world [6]. Studies have shown that when an individual's FOV is decreased in the real world, the subject's sense of balance and presence is reduced [43, 44]. When developing for a quality, immersive, virtual experience the FOV should be heavily considered. Lin et al. investigated whether a user's FOV in a VE altered the user's sense of presence using the "Real Drive" driving simulator. The simulator included a full-sized car placed in a 3 walled CAVE®. The study established a positive correlation between the user's sense of presence with an increased FOV [45]. As seen in Figure 9, the FOV of HMD, represented by the shaded rectangular box, is much smaller than the FOV of normal human eyes.

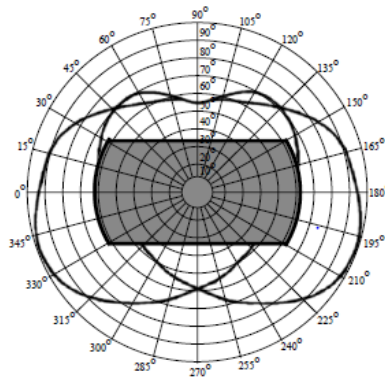


Figure 9: FOV the normal human eyes (represented by the thick black lines) and a typical stereoscopic HMD (grey square) [40]

1.3.3 Stereoscopy

As previously mentioned, in reviewing human depth perception, stereoscopy is one of the primary driving forces in VE [51-58]. The key theory behind stereoscopy is there are two sets of images or videos taken at two slightly different angles. Lenses are used to ensure that the proper image or video is being displayed in the individual's correct retina. Many HMD, including the Gear VR and Oculus Rift use wide angle lenses that take in an input image/video like the one shown in Figure 10 [46]. When the input image/video is viewed through the HMD-wide angle lenses, the user achieves the stereoscopic effect. These devices also allow for movement of the head, which then extends the FOV. Stereoscopy is also used in many TV's, projectors, cinema, and CAVE® settings [47]. Some of the most common stereoscopic multiplexed (multiple inputs combined into one signal) approaches include color multiplexed, polarization multiplexed, and time multiplexed [47-49].



Figure 10: Left- Oculus Rift, with its wide-angle lenses. Right- input for HMD [46]

1.3.3.1 Color Multiplexed Approach

The simplest stereoscopic method is the color multiplexed approach, where different colored lenses are used to filter color multiplex images [48]. The most common color multiplex glasses are the anaglyph glasses shown in Figure 11 [40]. They work by having the left lens encode for red channel frequencies and the right lens encode for cyan channel frequencies. The anaglyph images are typically created with a binocular camera [48]. When a person is wearing anaglyph glasses, and observing an anaglyph image, the left lens blocks out all cyan channel frequencies in the anaglyph image, meaning that the left eye is observing the image that has red in it. The right eye blocks out all red channel frequencies in the anaglyph image, meaning that the right eye is observing any image that has cyan in it. This process results in one of the two images taken by the binocular camera being blocked in each eye, creating a stereoscopic effect and resulting in the user viewing a 3D image. Anaglyph glasses are inexpensive to make and can be used without a special display, but, are not as commonly used in VE due to a loss in color information and an increased degree of crosstalk inherent in their use [48].



Figure 11: Anaglyph glasses [42]

1.3.3.2 Polarization Multiplexed Approach

Polarization glasses work by selectively polarizing the light that is projected from the VE [48]. As seen in Figure 12, alternating horizontal rows are assigned to the left or right eye, with odd and even rows being orthogonal to one another. There are two types of polarization filters, circular and linear. Circular filters alternate each row between clockwise and counter-clockwise orientations. Linear filters alternate each row perpendicular to the previous row. Circular filters are more commonly used because they allow the person to tilt his or her head, where linear filters do not [48]. Depending on the system, a polarization image can come from one or two projectors. One of the downsides of the polarization approach is that some systems require special screens to preserve that state of polarization. The special screens increase the cost, but there is no loss in color information, making it ideal for cinemas [46]. A less expensive polarization system uses a passive approach, where half of the pixels are assigned to one eye, and the other half to the other eye. The downside of this approach is that there is a 50% reduction in the image resolution [49].

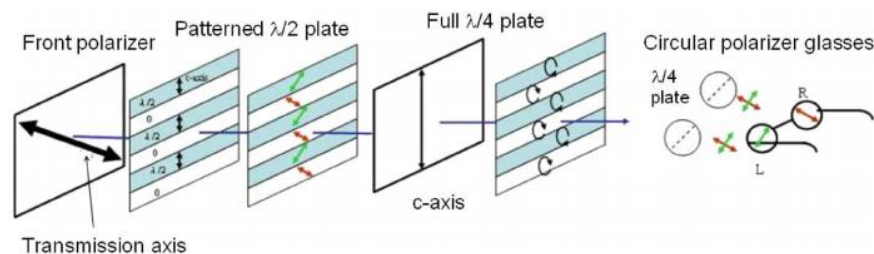


Figure 12: Stereoscopic Polarization method [48]

1.3.3.3 Time Multiplexed Approach

The theory behind the time multiplexed approach is that a projector flickers between images for the right eye and left eye at around 120 frames per second. Battery powered, shutter glasses use liquid crystal lenses to block a specific image [49]. To achieve the stereoscopic effect, the glasses flicker at the same rate as the projectors, to block the left-eye image from being viewed from the right-eye, and block the right-eye image from being viewed from the left-eye [48]. Without shutter glasses, the time multiplexed image appears to have a ghost image. This is the result of human's eyes/brain not able to process a single image being flickered 120 times in one second [48]. The synchronization between the shutter glasses and the projectors is achieved by either an infrared emitter, radio frequency emitter, or DLP-link [48]. Time multiplex approach is common in many TV's and IVE. The downside of time multiplexed approach is that brightness is noticeably decreased. Benefits over the color and polarization multiplexed approaches includes preservation of native resolution [48].

1.4 VR Visualization Tools

To achieve the stereoscopic effect, the above stated visualization methods are then applied to VE using approaches such as a CAVE®, standalone 3D projector, and or HMD such as the GearVR, Oculus Rift and HoloLens.

1.4.1 CAVE®

CAVE®, also defined as an IVE, are customizable rooms used to immersively view virtual content. Many CAVE®'s operate by having a series of rear-screen projectors projecting content on the environments walls, however newer CAVE®'s are beginning to use LCD screens, which create more vibrant graphics [50]. One of the downsides of using a series of LCD screens are the small but visible bezels between screens positioned together to cover a large FOV. Bezels are the frames that encapsulate the LCD screens. When viewed in VR, the bezels are still visible to the user.

CAVE®'s are often run using multiple servers, therefore intermediary programs are required to cluster the servers together to get a seamless transition between each projector or LCD screen. CAVE®'s range in shape (cylinder, square, rectangle, etc.), sizes, and different accessories (i.e. head tracking) with each environment designed for a unique purpose [9]. Unlike some HMD, CAVE®'s are nonintrusive spaces, where the viewers are free to move around while having awareness of both the real and virtual world [51]. The first CAVE was invented in 1992, at the University of Illinois' Electronic Visualization Lab (EVL) and has been recently upgraded [51]. The original EVL CAVE® had four sides, three walls and a floor, and used projector technology. The EVL opened CAVE2® in October 2012. CAVE2®, as seen in Figure 13, uses a nearly circular footprint, 24 feet in diameter, and has 18 columns of four 45-inch LCD screens to display the content. When the user puts on stereoscopic glasses, the 3D effect is revealed by passive stereoscopic technology [50].



Figure 13: CAVE2®, located at University of Illinois' Electronic Visualization Lab [52]

Another example of a more recently built CAVE is MARVL which, as mentioned in the introduction, opened in January 2014, at the cost \$1.2 million [4]. The environment has four walls (front, two sides, and floor), a front wall which is twice as long as the side walls, with the length, height, and depth dimensions being 18'6" x 9'3" x 9'3". Ten rear-screen projectors run on six servers clustered together by a program called MiddleVR (MiddleVR; Paris, France) [4]. Like the EVL, MARVL also uses stereoscopic glasses to create a 3D effect. MARVL designed the environment to have an extra-long front wall because a survey of users indicated this preference could better accommodate a larger number of potential users across a wider range of applications, as compared to a traditional cubic CAVE® VE. The environment comfortably seats up to 30 people, which permits a collaborative learning environment. MARVL has the capability to use head tracking, though it is typically not used because MARVL is primarily used in a group setting. When engaging head tracking, the person using the head tracking device has an optimum experience, but other individuals in the CAVE® experience the unpleasant feeling of the virtual world randomly moving around.

1.4.2 Standalone 3D Projectors

Standalone 3D projectors are a less expensive alternative to a CAVE® while still offering a collaborative viewing environment [53]. For example, the rear-screen projectors used in a CAVE® could also serve as standalone projectors if used in isolation. While the projector is on, and the user is wearing projector specific glasses, he or she sees 3D virtual content. An issue with standalone stereoscopic projectors is that they typically are not as immersive as a large-scale multi-surface IVE. The virtual world does not surround the user, instead it is displayed on a smaller screen. Due to humans wide FOV, even if the user is looking directly at the screen from some pronounced distance, the user can conclude they are in the real world [40, 43].

1.4.3 Head Mounted Displays

HMD are devices that are worn on a person's head to display virtual content directly in front of his or her eyes [5]. Recently designed HMD show quality virtual content at a relatively inexpensive cost [53-60]. Ivan Sutherland made the first conceptual idea for HMD in 1965. He wrote a paper titled "The Ultimate Display" which discussed a force feedback device, where the user could view interactive graphs, as well as hear, smell and taste [54]. Later in 1968, Sutherland built the first HMD which updated a graphic display when the user moved their head to appropriately replicate the new viewing position. The device worked by having two half-slivered mirrors that superimposed computer-graphics into the real world [54]. One of the greatest contributors

to the development of HMD was the US Air Force [51]. In the 1970's and 1980's the US Air Force built a flight simulator called the Visual Coupled Airborne System, where the pilot could view optimal flight paths, identify the enemy, and target threat information. While the simulator created an accurate environment, the project cost millions of dollars to build and operate [54]. Updates to HMD continued for the next four plus decades, but the devices either lacked functionality and or the costs were excessive [51, 55]. In 2010's, through companies like Samsung, Oculus Rift, and Microsoft, costs were driven down, and HMD's became less expensive, more accessible and provided high quality content. Some examples of HMD's include Gear VR, Oculus Rift and HoloLens.

1.4.3.1 Samsung Gear VR

Gear VR is a HMD released on November 27th, 2015. The HMD works in conjunction with select Samsung smartphones. According to amazon.com, the Gear VR costs approximately \$30 with an optional trackable controller for an additional \$18. To display virtual content, Gear VR uses the same stereoscopic technique previously discussed in the Stereoscopy section, where stereoscopic images are observed through two wide angle lenses. Gear VR is a passive device. It does not need to be charged, although its associated smartphone requires power. The Gear VR design contains a focus adjustment wheel, home key, back key, touchpad, volume key device holder, cushion foam, and two wide angle lenses. To run any Gear VR application, the game is first uploaded onto the Samsung smartphone and depending what generation the smartphone is, connected to the Gear VR's MicroUSB or USB type-c. Once the smartphone is in

place, the user can put on the HMD and view virtual content. To account for rotational head tracking, inside the device, there is a sensor, accelerometer, and three-axis gyroscope that measures angular velocity and external specific force [56]. For interaction, the user can either use the built-in touchpad located on the right side of the device, the Gear VR trackable controller, or a variety of external controllers that connect via Bluetooth.

1.4.3.2 Oculus Rift

The Oculus Rift, bought by Facebook in 2014, is a HMD released on March 28th, 2016. It is tethered to a computer or a high-end laptop and provides the user an immersive, virtual experience [7, 57]. In addition to the headset, the Oculus Rift has a Tracker v2 and two trackable controllers [58]. According to amazon.com, the entire bundle costs approximately \$399. Oculus Rift displays virtual content the same way as the Gear VR, through the stereoscopy technique. Oculus facilitates head tracking by the use of a gyroscope, accelerometer, and magnetometer [57]. The gyroscope tracks the angular velocity around the x, y and z axes. The Oculus software development kit (SDK) uses the angular velocity measurements to determine the direction of the rift relative to the starting point. Head tracking coordinates are sent to the Oculus Tracker v2 at a sampling rate of 1000 Hz, resulting in roughly a 2-millisecond lag between the player's head movement in the real and virtual world [57]. To combat the issue of drift, the accelerometer estimates the “down” vector and the magnetometer measures the magnetic fields strength and direction [58]. Oculus also has 6-degrees-of-freedom position tracking

which is traced by an infrared camera inside the Oculus Tracker v2. Position tracking allows the user to move closer or peak around in-game-objects [58]. Oculus reduces motion sickness and dizziness by synchronizing the user's virtual viewpoint in real-time. Movement is pivoted around a point near the base of the user's neck. This location in conjunction with position tracking creates a near motion parallax [57]. The Oculus SDK can support Linux, MacOS, and Windows. No specific hardware is required; however, specific relatively high-end graphics card is recommended for the best gaming experience. Interaction with content is achieved via the Oculus tracking controllers or select gaming controllers.

1.4.3.3 Microsoft HoloLens

Unlike Gear VR and Oculus Rift, the Microsoft HoloLens uses AR instead of VR. HoloLens was initially available to developers in early 2016 and is currently being sold for \$3,999 on amazon.com. The device is a fully untethered holographic computer that contains an inertial measurement unit, Intel Cherry Trail chip and, a holographic processing unit that runs on Windows 10 [59]. The holographic images are projected from two HD 16:9 "light engines" comprising of miniscule crystal on silicon displays. These "light engines" are mounted on the bridge of the HoloLens and projections are passed through a "combiner" that projects the virtual element onto the real world [59]. The user is able to view, place, record, interact, and walk around virtual elements via multiple "environmental understanding cameras", time of flight depth cameras, ambient light sensors, and a video camera. "Environmental understanding cameras" are used for

head tracking. The time of flight depth cameras place objects in the real world and track the user's hand movements [59]. An ambient light projector facilitates how bright the projected virtual element needs to be. The video camera is used to record what the user is seeing while using the device. An issue with the current design is that the FOV is narrow. Interaction with the HoloLens is achieved by either voice controls, gaze tracking, custom gestures, the HoloLens clicker, or select gaming controllers [60].

1.5 Challenges of VR

Many of the issues that were previously associated with VR, such as cost and accessibility, have been mitigated over the last decade by the introduction of less expensive, higher quality HMD such as the Gear VR. One of the challenges facing VR is that between 20-40% of individuals during and after viewing virtual content experience some sort of simulator sickness [61]. Simulator sickness was initially documented in 1957, where individuals were in helicopter training and experienced symptoms similar to motion sickness including: general discomfort, apathy, drowsiness, headache, disorientation, fatigue, pallor, sweating, salivation, stomach awareness, nausea and vomiting [61]. Simulator sickness is the result of a lack of consistency between information being received by the visual and vestibular systems. The user is visually seeing him or herself moving, but the vestibular system is interpreting something else [61]. The vestibular system is located in a person's inner ear and is responsible for the sense of balance, spatial orientation, head position, and motion [62]. Vestibular labyrinth is the main compartment in the vestibular system which includes three semicircular

canals filled with fluid that are aligned in three different planes. When a person moves their head, the fluid movement is detected, and the information is sent to the brain [62]. When the user is in a VE, what the user is visualizing often times does not match what the vestibular system is detecting, resulting in simulator sickness [61]. Although developers have created solutions to combat simulator sickness, it remains an important issue facing the virtual community.

Another challenge in VR is the lack of effective gestural and voice interaction. Many virtual devices execute commands using a gaming controller, some of which can be tracked and interpreted as a hand in the virtual world [57]. While these controls are effective, there is a lack of full immersion because the user is pressing a button in the real world while being in the virtual world. However, with the advanced development of in-hand tracking and machine learning, games in the near future will adapt to the user's hand and voice gestures, resulting in higher immersion [63].

A limitation for VR developers is the lack of collaboration between HMD companies. As will be discussed in the Methods section, the CFD to VR workflow targeted GearVR and Oculus Rift because they are one of the few HMD's that collaborated and used the same SDK [64]. SDK are libraries of code with examples that help aid the developers build an application for a specific HMD. The VIVE is another popular HMD. Regarding the CFD to VR workflow, the VIVE was not selected because it uses an entirely different SDK, as compared to the Oculus Rift and GearVR. The Oculus Rift and GearVR use Oculus SDK while the VIVE uses OpenVR SDK [65]. The vastly dissimilar SDK's between different HMD's makes building a generic VR application difficult for developers. With additional collaboration between HMD

companies, where developers use a common SDK, more VR applications could be shared to a wider population.

1.6 Biomedical Computational Fluid Dynamics

Before a user builds and or observes a VR CFD simulation, he or she must first understand how the simulation was created and solved. As shown in Figure 14, there are multiple steps to create biomedical CFD simulations [66]. The CFD process can be segmented into three main phases: pre-processing, solving governing equations, and post-processing [11]. The pre-processing phase requires the user to obtain medical images for some portion of the cardiovascular system of interest, followed by segmenting the medical images to create a surface model. The surface model is then discretized into millions of finite elements that represent the volume available for blood flow. Solving the governing equations of fluid flow requires a user to apply boundary and initial conditions. This typically consists of an inflow waveform and profile, and outflow representations of the downstream vasculature not captured by imaging data, such as a 3-element Windkessel approximation. The Navier-Stokes (NS) equations are then iteratively solved at each node in the volume mesh. Finally, the post-processing phase requires the user to analyze the simulation using velocity profiles, pressure distribution, and related indices such as wall shear stress (WSS), and/or oscillatory shear index (OSI).

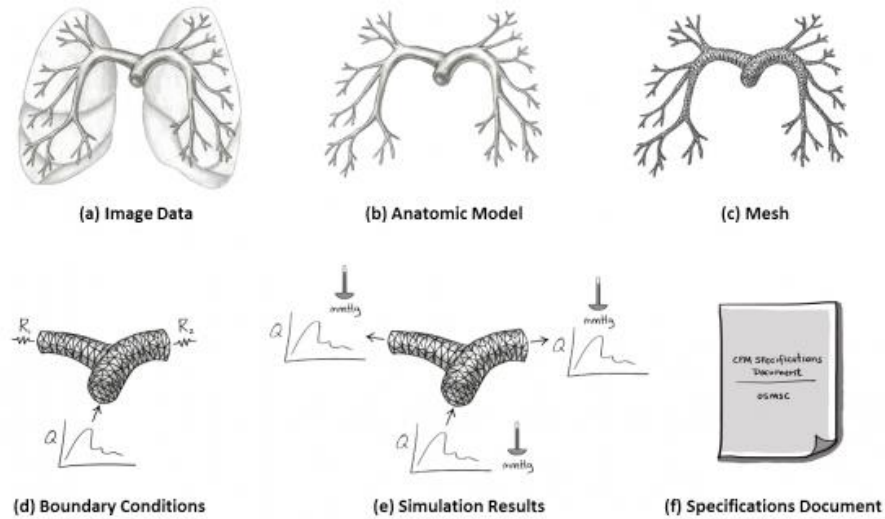


Figure 14: Steps to conduct a CFD simulation. Steps a-c represent the pre-processing phase, b represents the solving governing equations phase, and e and f represent the post processing phase [66].

1.6.1 Pre-Processing

The first step to create a CFD simulation is to obtain imaging data from the region of interest. Four forms of medical images are most often used to create a volumetric model for biomedical applications: magnetic resonance imaging (MRI), computed tomography (CT), ultrasound, and intravascular imaging modalities such as optical coherence tomography (OCT). MRI obtains medical imaging by using magnetic spin relaxation properties to discern between different tissue types [67]. For vascular CFD purposes, an advantage of MRI is its ability to create phase contrast magnetic resonance images (PC-MRI). PC-MRI has the velocity encoded into its phase of the complex MRI signal and may also be used to obtain flow waveforms and elastic moduli [67]. A variety of software platforms use PC-MRI to measure the cross-sectional area of the vessel to obtain velocity information that can be used as inflow boundary condition information.

Being that MRI can create high quality, cross sectional images and be used to calculate a flow waveform, it is the most common imaging technique used to create CFD models [67]. CT uses a radiation beam and X-ray detectors to measure the amount of radiation absorbed by a person's body. The radiation beam and X-ray detector are measured from multiple angles. This information is then sent to a computer to reconstruct the area of interest [68]. Ultrasound is a widely available imaging modality that uses high frequency sound pressure waves. An ultrasound machine includes a transducer that uses an array of piezoelectric crystals. When an electrical signal is applied, the crystals create high frequency sound pressure waves. The sound waves reflect off different tissue types, creating an echo that is then picked up by the piezoelectric crystals. The echo is then converted into an electrical signal and eventually an image. Ultrasound is not typically used to create the volumetric model because it does not have references to a fixed coordinate system and, resolution is lower than MRI or CT. However, Doppler ultrasound is an often-used tool to provide real-time velocity, vessel dimensions, and flow measurements [67]. OCT is a light-based image modality that can be used with vessels of the cardiovascular system. A catheter is inserted into an artery where an optical fiber emits infrared light, and the same fiber measures the intensity of the infrared light after its reflection to generate a local cross-sectional image [69].

Once medical images are obtained, the volumetric model can be constructed by segmentation. Depending on the images taken and the complexity of the model, the user can either outline the lumen of the vessel by direct 3D segmentation, or determine a centerline for each vessel of interest, travel along that centerline to segment the vessel at corresponding 2D imaging slices, and then loft these segments to create a representation

of the flow domain. Many software packages, such as SimVascular (simtk.org), can do this step to create the model. For more complex models, such as models with multiple bifurcations, the vessel junctions are blended together to create a smooth transition between the vessels [67].

Finally, the surface model needs to be discretized into a finite element mesh for use in solving the NS equations. Best practices dictate that the mesh needs to be denser in areas of complex flow. However, there is a tradeoff, the more dense the mesh, the higher the computational cost. The user must therefore balance solution accuracy arising from a denser mesh with computational cost. For complex CFD simulations, the volumetric mesh is typically discretized into millions of elements. To expedite solving the NS equations at each element, CFD simulations are generally submitted to a high-performance computing resource.

1.6.2 Solving Governing Equations

Once the volume mesh is obtained, the next step is to define fluid properties and apply boundary conditions to the wall, inlet and outlet. Fluid properties generally consist of density and viscosity. The density of blood is often taken from literature and viscosity is typically assumed to be constant (i.e. Newtonian fluid). In reality, blood is a non-Newtonian fluid with a non-linear relationship between shear stress and strain rate [70]. However, in large arteries, blood can be assumed to be a Newtonian fluid because the large arteries have very high strain rates making the blood's viscosity relatively constant (~4 cP for humans) [70]. Boundary conditions applied at the walls of a model typically

include either a rigid (i.e. no-slip) or deformable wall. A no-slip boundary condition means that the fluid layer closest to the surface model is moving at the same rate as the surface model. If the walls are rigid, the particle velocity at the wall is zero. If the wall is deformable, governing equations are needed to solve for displacement at the fluid-structure interface that allows for the replication of vessel wall movement [71]. Inflow boundary conditions require prescribing the velocity of the fluid at the inlet. The flow rate can be defined as either pulsatile or steady-state. The analytic shape of the inlet can take the form of several well-known shapes including Womersley, plug, parabolic or patient-specific profiles. PC-MRI, Doppler ultrasound, literature, or an invasive process are typically used to calculate time-resolved volumetric blood flow for an inlet [72]. The last step is to apply outflow boundary conditions to account for the influence of the vessels distal to the CFD simulation's branches [72]. There are many different methods of selecting outlet boundary conditions, some of which include zero pressure, resistance, and 3-element Windkessel (i.e. RCR) [73]. Zero pressure is sometimes used for its simplicity, but causes inaccurate simulation results. Zero pressure ignores the fluctuation in pressure as the result of wave reflection from distal vessels. In addition, there is typically a residual pressure at the end of the vessel being modeled, which zero pressure outlet boundary conditions ignores [73]. Local resistances in the model may therefore play a larger role in the distribution of flow to branches than occurs in reality. Resistance models define the distal branches as a single variable, resistance [73]. The 3-Element Windkessel model, as shown in Figure 15, adjusts the local resistance/local impedance (R_c), distal resistance (R_d), and total capacitance (C) values until the computed pressure matches a desired pressure range [74].

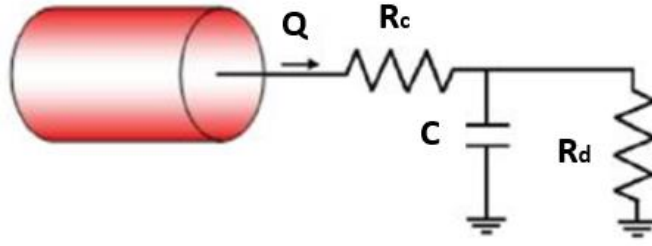


Figure 15: 3-Element Windkessel Model [74]

For every CFD simulation, the NS equation must be solved at each location in the mesh. NS equations are sets of non-linear partial differential equations that solve for two laws (1) conservation of mass and (2) balance of fluid momentum. Regarding biomedical CFD applications, the energy equation is generally ignored because human body temperature is considered constant.

Conservation of mass, Equation 1, means that the mass within a control volume must be constant over time, mass cannot be added or removed.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \quad (1)$$

In this equation, ρ represents density, and u represents velocity. Human blood has a constant density of 1.06 g/cm^3 [75]. A constant density eliminates the density and partial derivative terms resulting in Equation 2 or in simple Cartesian coordinate form, Equation 3.

$$\nabla \cdot u = 0 \quad (2)$$

$$\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} + \frac{\partial u_3}{\partial z} = 0 \quad (3)$$

Balance of fluid momentum, Equation 4, means that the rate of momentum

entering the system must be the same as that leaving the system. Momentum can enter and or leave the system via convection (i.e. bulk fluid flow) or molecular transfer (i.e. velocity gradients or viscous contributions).

$$\rho \left(\frac{\partial u}{\partial t} + u \cdot \nabla u \right) = -\nabla p + \mu \nabla^2 u + F \quad (4)$$

Where $\frac{\partial u}{\partial t}$ is temporal acceleration, $u \cdot \nabla u$ is convective acceleration, ∇p is the pressure gradient, $\mu \nabla^2 u$ represents viscous contributions, and F includes body forces. For biomedical purposes, the body force term is ignored because the images used for the CFD simulation are taken in a system where the patient is typically lying down, therefore gravitational body force is often assumed to be zero.

In simpler terms, Equation 4 is Newton's second law, Equation 5. On the left side of the equation are mass and acceleration and on the right side of the equation are forces.

$$m \cdot a = \sum F \quad (5)$$

The NS equations are applied at each location in the mesh where velocity, u , and pressure, P , are typically solved using a high-performance computing resource.

1.6.3 Post-Processing

After a simulation converges, the researcher uses its associated velocity and pressure information to analyze indices of interest. Some common indices used frequently include: velocity profiles, pressure distributions, WSS, and OSI. WSS is a measure of the tangential force exerted by flowing blood on a vessel wall [75]. As shown

in Equation 6, in its simplest form, WSS is the product of viscosity and strain rate. Strain rate is a representation of the near-wall velocity gradient. In areas where there is low WSS, there is increased risk of neointimal thickening [76].

$$\tau = \mu\dot{\gamma} = \mu[\nabla u + \nabla u^t] \quad (6)$$

As shown in Equation 7, OSI is derived from WSS. In areas where there is high OSI, blood is bi-directional, moving back and forth over the same area the vessel. In areas where there is low OSI, blood is flowing unidirectional, in one uniform direction. Increased OSI is typically correlated to the organization of atherosclerotic lesions [77]. Areas where there are low WSS and high OSI result in the inhibition of nitric oxide-synthesis, endothelial dysfunction, increased apoptosis and many other adverse outcomes [78].

$$OSI = 0.5 \left(1.0 - \frac{\left| \int_0^T \tau dT \right|}{\int_0^T |\tau| dT} \right) \quad (7)$$

CHAPTER 2: CFD TO VR

2.1 Previous Work Done at MARVL

As seen in Figure 16, prior Marquette University graduate students and faculty members have used VR to view and study CFD data since 2010. Previous researchers created a pipeline that took CFD simulation results from the first generation of SimVascular and converted those results to be viewed as pulsating velocity glyphs in VR using MATLAB, Visual Basic (Microsoft; Redmond Washington), and EON STUDIO 7 [14].

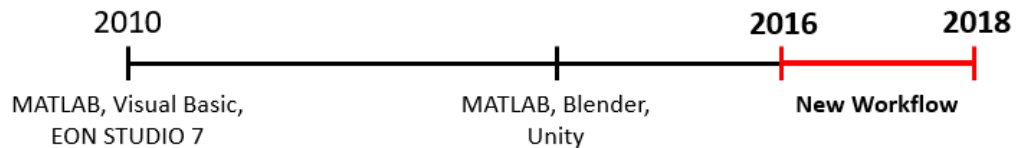


Figure 16: Timeline of viewing CFD results in VR at Marquette University

In the most recent version of the related workflow, before the creation of the current work, MATLAB was used to convert the first generation of SimVascular result files (.vis and .vtk) into formats that store 3D geometry and vertex magnitude.

In total, thirteen MATLAB scripts were used for the prior conversion process. One of the thirteen MATLAB scripts was primary and had an executable script with a graphical user interface (GUI). The GUI instructed the user which file or folder to input in order to execute file conversion. The GUI, along with each of the scripts, had concise

comments and file names allowing the user to determine what each script, command or function was designed to do. In addition, it allowed the user to copy a specific section of code if the user only needed a certain portion to be completed.

The first half of the single executable MATLAB script was designed to convert .vis files, representing the blood flow velocity of each time step into .ply files. The MATLAB scripts were used to extract spatial locations of roughly 0.1% of the original CFD blood flow velocity results. Spatial locations were uniformly distributed with a set amount along the outer edge of the wall to satisfy the no-slip boundary condition from the CFD simulation. If the user only wanted to create .ply files, the user would run the first half of the single executable script.

The second half of the single executable MATLAB script was designed to convert .vis files (containing pressure data of each time step) and .vtk files (containing TAWSS and OSI) into PNG images and create a .x3d file representing the wall mesh geometry. If the user decided to run the entire script, it took approximately one minute to create the .ply file and PNG pressure image per time step.

The .x3d (geometry) and .ply (glyphs for each time step) were then reorganized in Blender (Blender Foundation; Amsterdam, Netherlands), an open source computer graphics program. Lastly, the Blender project and PNG images were brought into Unity where custom C# (Microsoft; Redmond Washington) scripts were written to view and interact with the simulation in MARVL.

While this process was reliable, there were many issues including:

- Inability to run with data from other CFD software packages
- Only able to view the velocity information via glyphs
- Additional software packages were required to display the pressure results in VR
- SimVascular updated its software; in the new update users were no longer able to save simulation results as .vis files, only .vtu or .vtp
- The Unity step was labor intensive, where novice users would struggle to correctly set up the Unity scene to work in MARVL

As a result, a new workflow was created, noted in red in Figure 16, which works on a variety of CFD software packages, views CFD results as glyphs or streamlines, uses fewer software packages, works in a diverse set of VE, and is almost fully automated. In addition, the new workflow significantly reduces the time it takes to convert the CFD results into files that store 3D geometry.

2.2 Materials and Methods

The new workflow, as shown in Figure 17, has 5 steps to convert CFD simulation results into VR: (1) format of CFD results, (2) convert CFD results into a 3D format, (3) reorganize CFD results, (4) customize for a given VR environment, and (5) arrange VR environment. The workflow is compatible on a variety of CFD software packages and uses open source packages once the CFD results are obtained. For a standard multi-time

step CFD simulation, meaning the user wants to view some combination of wall pressure, TAWSS, OSI, and animated vector information (streamlines/glyphs), the workflow is practically fully automated. The user only needs to complete small tasks and run scripts for each step.

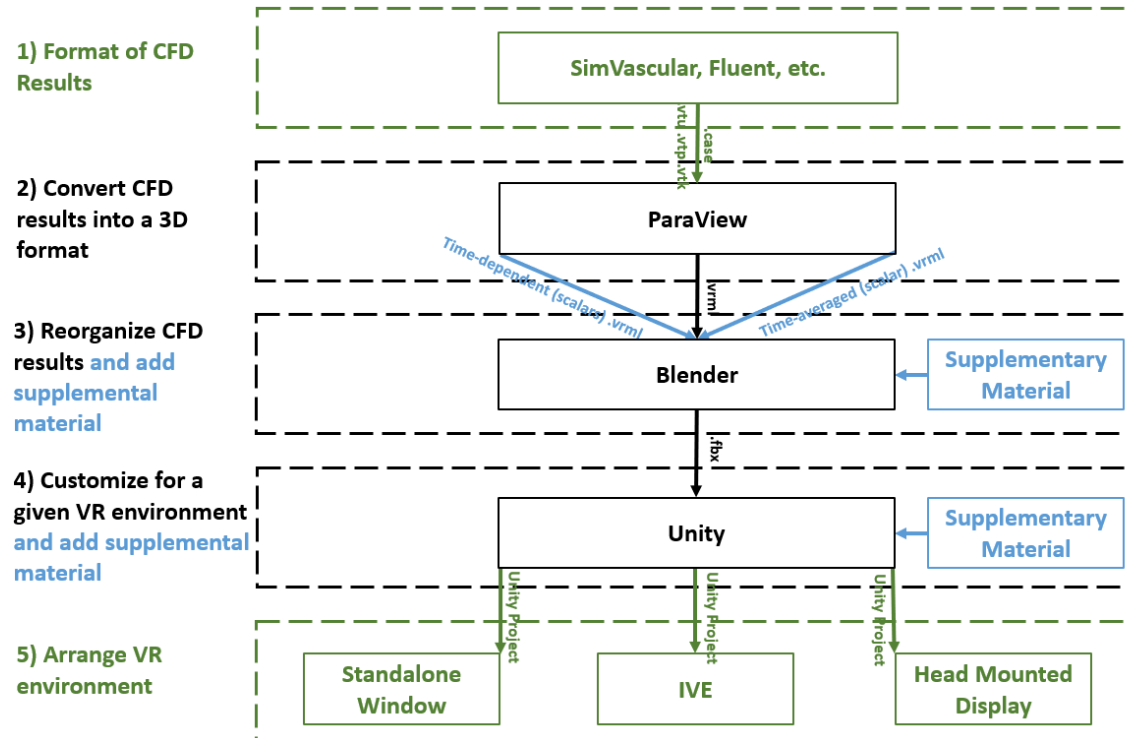


Figure 17: Flowchart describing the immersive visualization workflow after CFD is complete. Steps 1-5 are the focus of the current work to create a pipeline for biomedical CFD results within virtual and augmented reality environments. Colors denoted in black = necessary steps, blue = optional steps, green = pick one of the options. CFD results (1) are acquired and converted into .vrml files (type of 3D geometry) (2) .vrml files are reorganized in a graphics program called Blender where supplementary material can be added (3) simulation is transferred into Unity (4) more supplementary material can be added and 3D, animated content is generated from the CFD results (5) simulation is arranged for VR environment by using a proprietary plug-ins where the Unity project can be presented in VR.

2.2.1 Workflow Requirements

Requirements for the workflow presented above are determined based on the simulation and visualization needs of investigators within Marquette's OCOE and collaborating clinical divisions, while keeping potential derivative projects in mind. Based on input from these researchers, the workflow was designed to facilitate viewing of CFD results using a diverse set of VE including standalone stereoscopic projectors, HMD, and IVE. The workflow has been successfully tested on a stereoscopic standalone projector, MARVL, Samsung Gear VR, Oculus Rift, and Microsoft HoloLens. For automation purposes, the main template used in part 5 of the workflow, customized for a given VR environment, is geared to work on standalone stereoscopic projectors, Gear VR, and Oculus Rift, although modest alterations can be made to the template if the CFD simulation is viewed using the Microsoft HoloLens or IVE.

If the user is planning to purchase a HMD to view his or her CFD data, the user should consider the information listed in Table 1. The first consideration is whether to view CFD results in VR or AR. When viewing content in VR, the user is fully immersed in the virtual world [9]. For CFD applications, the virtual world is the computational model, where the user can have a sense of presence both inside and outside the model. This is exemplified in Figure 18. When viewing content in AR, the user understands that they are still in the real world and lacks full immersion [79]. Therefore, in AR, it is difficult to get the feeling of being inside the computational model [79]. For this reason, as shown in Figure 19, in AR, the computational model is observed from outside the model.

The Gear VR and Oculus Rift are fully immersive, but collaboration is minimal. The HoloLens allows fellow users who are also wearing the HoloLens to collaborate and share their experience, and can permit external feedback by an audience using a live-streaming option. When comparing the Gear VR and Oculus Rift, the user should weigh the importance of budget, quality of experience, and mobility.

	Samsung Gear VR	Oculus Rift	Microsoft HoloLens
<i>Cost*</i>	\$30 - headset \$18 - optional controller	\$399	\$3,999
<i>Platform</i>	Virtual reality	Virtual reality	Augmented reality
<i>Medium</i>	Samsung Galaxy Smartphone	Desktop or high-power laptop	Built-in untethered holographic computer
<i>Graphics Quality</i>	Lower	High	High
<i>FOV</i>	Standard for HMD	Standard for HMD	Small
<i>Advantages</i>	Portable, less expensive	Relatively inexpensive, high quality virtual content, quality trackable controllers	Collaborative, uses hand and voice gestures
<i>Disadvantages</i>	Lower quality graphics, application may need alteration for peak performance	Tethered to a computer or laptop	Expensive, small FOV, difficult to develop

*Table 1: Facts/Recommendations for Gear VR, Oculus Rift, and HoloLens - *all costs were the lowest values found on amazon.com*



Figure 18: Left- Birds-eye-view of CFD model using VR. Right – inside the CFD model using VR

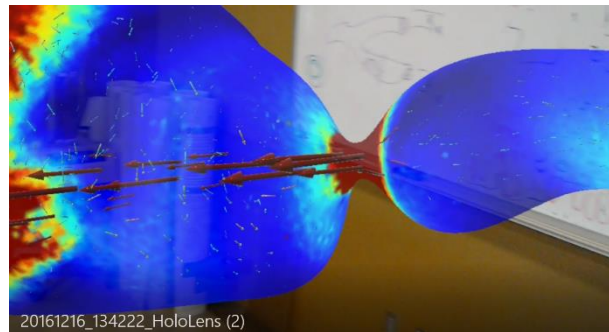


Figure 19: Screenshot of what a CFD model looks like in AR

2.2.2 Format of CFD Results

As stated above, the workflow was aimed to support several common CFD software packages used by members of Marquette University's OCOE and their respective collaborators. The software packages are fairly common and include Fluent and SimVascular. Execution of these software packages yields result files that can be saved as either .vtk, .vtu, .vtm or .case file formats. Given the interchangeability and capability of geometry, mesh and simulation results, it is possible the workflow could

also work for other software packages that can be saved in one of these formats. The aforementioned files facilitate viewing CFD results on 2D display but are not natively compatible with VR. The provided workflow allows the user to convert CFD results into formats that are compatible for VE. The steps within the workflow are designed to use supporting software packages that are common within academic, engineering and digital design facilities. Table 2 lists the supporting packages that are needed to use the workflow in its current form, along with the function being performed by each supporting package.

	Function
<i>ParaView</i>	Convert CFD results into a 3D format
<i>Blender</i>	Reorganize 3D format files and add supplementary data
<i>Unity</i>	Animation, custom interaction, add supplementary data and application used to view CFD results in VR

Table 2: Necessary programs for workflow

All the following figures that include a CFD model in the methods section come from a CFD simulation run on a PE-240 cylinder phantom. Images of PE-240 tubing were obtained from a micro CT scan as discussed in detail elsewhere [80]. The imaging data was segmented, lofted into a 3D model, meshed, and an associated simulation was run using SimVascular. The inflow waveform was pulsatile, the walls were considered rigid, and a 3-element-Windkessel model was applied to the outlet that allowed pressure within the tubing to range between 80-120 mmHg, which represents normal diastolic and systolic blood pressure values.

2.2.3 Convert CFD Results Into a 3D Format

The majority of CFD simulations display flow patterns over a set time. These results are typically shown using streamlines or glyphs. To view the streamlines or glyphs in a VE, the CFD results need to be converted into files that store 3D geometry, vertex magnitude (e.g., color coded), and a compatible file format for the selected 3D gaming engine. For the workflow's purpose, streamlines and glyphs for each time step are converted into .vrml files using ParaView.

ParaView is an open source software package that is used for both scientific and interactive visualization. It was chosen because it is an excellent file conversion tool and also allows the user to visualize the monoscopic CFD simulation before it is brought into VR. ParaView supports stereoscopic viewing, but with several limitations. ParaView is incompatible with HMD, there is a lag between viewing consecutive time steps, the viewer is unable to view complementary data sources, and when displayed in stereoscopic 3D, the graphic quality is poor.

As seen in Figure 20, there are 5 steps to create streamlines that are compatible with a VE. First, the CFD project is loaded, if the user was only given scalar velocity information, a vector velocity field measurement is made using ParaView's calculator tool. The vector measurement is made by taking the sum of each unit vectors (i.e. \hat{i} , \hat{j} , \hat{k}) and subsequent scalar value (i.e. x-directed velocity, y-directed velocity, z-directed velocity). If vector velocity information is already provided, step two can be skipped. Next, streamlines are added using ParaView's streamline tracer tool. Volume is then added to the streamlines so that each line will appear as tubes in a VE. Lastly, a pre-

written Python script is run to save each time step as a .vrml file. Step by step instructions can be found in Appendix A.

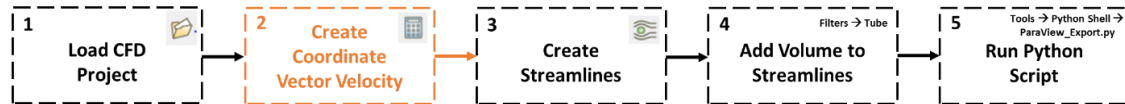


Figure 20: Flowchart to create 3D geometry velocity streamlines files (.vrml). Icons on the top right hand corner of steps 1-3 represent: (1) open, (2) calculator, (3) streamline tracer. Step 2 may be skipped, depending on the format of velocity in the CFD data.

As seen in Figure 21, there are 4 steps to create glyphs that are compatible with a VE. Steps 1, 2, and 4 (Step 5 for streamline workflow) are identical to creating streamlines. To create glyphs that are compatible with a VE, first, the CFD project is loaded. If the user was only given scalar velocity information, a vector velocity measurement is made using ParaView's calculator tool. If vector velocity information is already provided, step two can be skipped. Next, glyphs are added using ParaView's glyph tool. Finally, a pre-written Python script is run to save each time step as a .vrml file. The step by step instructions can be found in Appendix B.



Figure 21: Flowchart to create 3D geometry velocity glyph files (.vrml). Icons on the top right corner of steps 1-3 represent: (1) open, (2) calculator, (3) glyph. Step 2 can be skipped depending on the CFD file

For both the streamline and glyph workflow, being that only portions of the available velocity information will be used, the user should be aware of the vertices where the streamline or glyphs are being generated. If the user would like to display streamlines or glyphs in precise locations (i.e. important model features), specific ParaView properties can be adjusted to match the desired output.

If the user would like to represent their CFD simulation results using velocity glyphs, ParaView should appear similar to Figure 22 before the script is run.

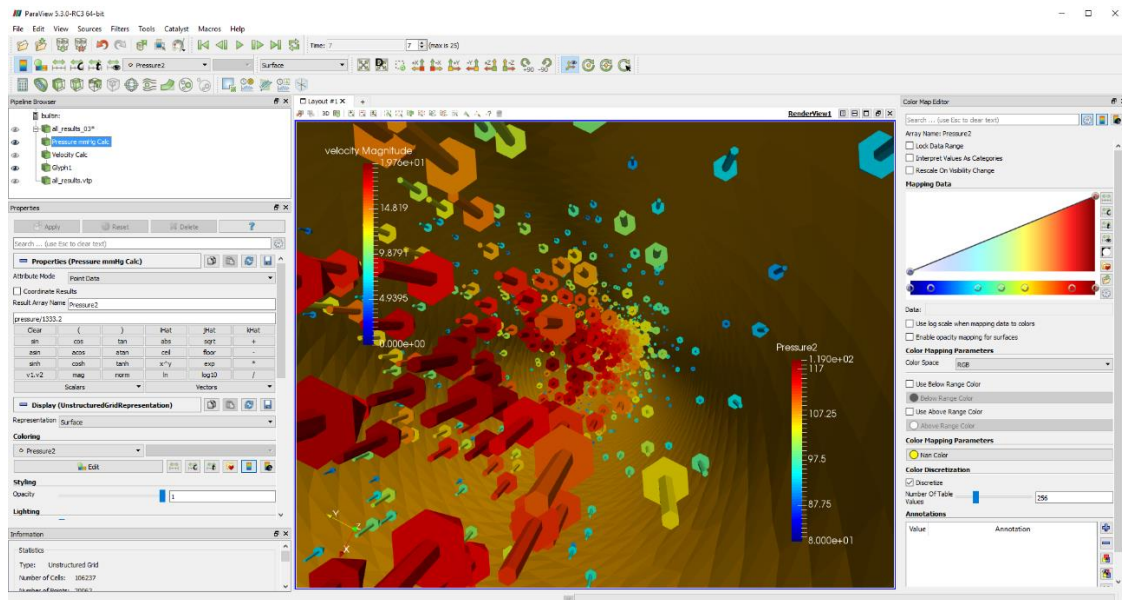


Figure 22: Results from a phantom tube CFD simulation with pulsatile, rigid, RCR boundary conditions as visualized using the ParaView glyphs workflow

The same Python (python.org) script is used for the streamline and glyph workflow. Before the script is run, the user must open the script, change the end time step to match the last time step of the CFD simulation, and map the location of the folder that will store each .vrml file. The execution order for the script is:

1. Define the time step variable
2. Re-check that the simulation is on the first time step
3. Loop through, and save each time step as a .vrmf file

2.2.4 Reorganizing of CFD Results

Blender is a graphics program that is used to reorganize the .vrmf files created in ParaView. As to the workflow, Blender is where all the objects are created. Blender is user friendly software with numerous tutorials and websites to help answer questions. It takes each 3D geometry file and re-creates the CFD simulation. For the workflow's purpose, Blender does not create an animation, but, as shown in Figure 23 Part B, all the objects and vertex colors to make the animation are created in one Blender project.

When using CFD simulations that have multiple time steps, it is imperative that each time step is uploaded in the correct 3D space. To ensure proper object positioning, a Python script was written that can be implemented in Blender's text editor. The script re-creates the original CFD simulation, sequencing the 3D-geometry-time-steps inside the wall mesh and creates proper parent-child relationships that are necessary when the Blender project is loaded into Unity. As shown in Figure 23 Part C, each streamline or glyph is placed under an empty "parent object." The streamlines or glyphs are now considered "child objects." The same parent-child relationship is done for a mesh representing the vascular wall. Another important aspect of the Python script is that it creates a wire-frame wall-mesh game object. The wire-frame wall-mesh game object is used as a monocular cue while in the VE, giving the user another sense of depth.

Importantly, this wire-frame wall-mesh is a representation of the mesh on the surface of the model that was used with the CFD simulation. The Python script also UV maps the wall mesh (technique used to wrap a 2D texture on a 3D model), and deletes unnecessary cameras and lamps that are rooted in the .vrmf files, thereby increasing the efficiency of execution of the Blender project.

After the user has followed the instructions in the Python script, as shown in Figure 23 Part A, it can be run. For each time step being imported, it should take approximately seven seconds to load (time based off a 8,500 KB .vrmf file - larger files will take a longer time to load). Progress of the script can be tracked by Blender's System Console as shown in Figure 24. After the script is completed, the CFD simulation appears re-created in Blender's 3D view (Figure 23 Part B). In addition, each object in Blender's 3D view appears in Blender's Outliner with proper parent-child relationships (Figure 23 Part C).

The next step is dependent on what the model looks like after the first script is completed. In the Blender template, there is also a secondary script that removes duplicate structures (i.e. doubles) and reduces the number of vertices. If after the first script is completed, and in the 3D view, the glyphs or streamlines are not visible, only the wall mesh, this means that the model's mesh has duplicate structures (double-sided mesh). For this scenario, the secondary script would need to be run. This ensures, if the user is looking at the model in the VE, that the glyphs or streamlines are visible. If the user would like to reduce the number of vertices, the secondary script should be run regardless of whether the wall mesh is double-sided. While in VR, too many vertices can slow down the frame rate of the Unity project, causing simulator sickness. If the model is

not double-sided and there is a reasonable number of vertices, the secondary script can be skipped. Lastly, the Blender file is exported as an .fbx file and loaded into Unity.

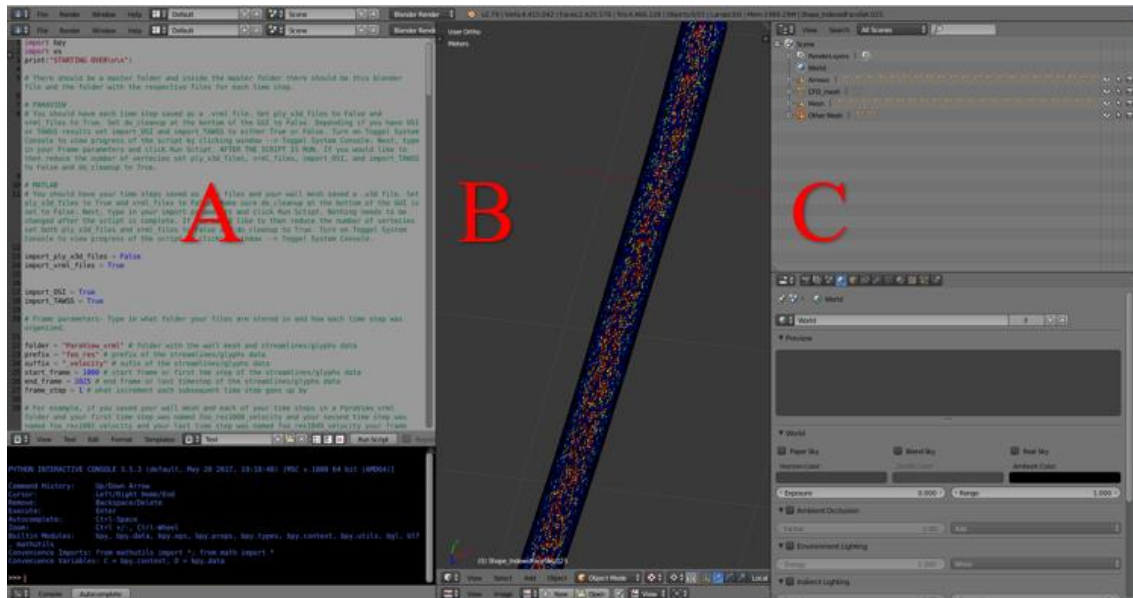


Figure 23: Screenshot of Blender's interface depicting a CFD simulation (same simulation that is used in Figure 22). (A) Blender's Text Editor where a Python script is used to reorganize the simulation and reduce the number of vertices. (B) Blender's 3D View which displays every single object of the simulation. In Unity, the objects will be turned on or off. (C) Blender's Outliner of parent-child relationships where each object may be edited. It is important the objects are properly named and organized so Unity's C# scripts do not have to be altered.

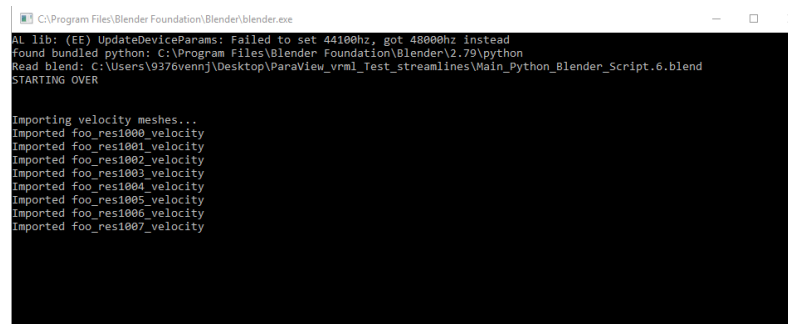


Figure 24: Blenders System Console in the middle of running the Python script

2.2.5 Add Supplementary Data

Depending on the type of supplementary material available, the material can be added in either Blender or the Unity template. MARVL amassed three workflows/C# code from various CFD projects that requested specific tasks and custom interaction. The first workflow allowed the user to toggle through volumetric imaging data used to create the CFD model. The second workflow recognized and displayed scientific imaging data (i.e. histology) in specific portions of the CFD simulation. The third workflow allowed the user to “fly through” the CFD simulation on a set track. The ability to add related data sources to a CFD simulation is one reason why it is beneficial to view a simulation in VR.

To toggle through the volumetric imaging data, each image slice is added in the Blender project. After the CFD simulation is reorganized, a blank array is added into the Blender scene and scaled to the CFD simulation. The image slices are then added to the corresponding array's plane. Like each glyph/streamline and wall mesh game object, an empty parent object is created where each image slice is placed underneath, becoming a child object. When the .fbx file and each image slice is imported into the Unity template, the template recognizes the parent object, adds a script, tags, and assigns a custom shader to each image plane.

A more project specific workflow can recognize where the user is at in the virtual simulation and display a corresponding site-specific image. The execution behind this function comes from the “Histology” C# script that is available in the Unity template. However, it is likely that the script would need to be slightly altered depending on the

user's intent. In Blender, after the CFD simulation is reorganized, wire-framed cubes are added to break up the simulation into specific sections. The wire-framed cubes are placed under an empty parent object. After the .fbx file is imported into Unity, a box collider and the "Histology" script are added to each cube. Lastly, a plane is added to the head-up display (HUD). When in play mode, and the user is moving around the simulation, the head node collides with the box collider. Upon collision, the corresponding site-specific images are shown. This approach can be used for any scenario where the user wants designated sections to show specific content.

Sometimes, the CFD model is too narrow to manually fly directly down the middle without going through the wall mesh. Other times, the user might want to have a pre-animated track where they can stop at certain sections for observation purposes. There are two alternatives to create the set pathway. One is by using Blender's path tool where a physical line is created. In Unity, a C# code is written where the camera is attached to the physical path. The second option uses Unity's animation tool where key frames are used to interpolate between points.

2.2.6 Customize for a Given VR Environment

The Unity game engine was selected for the workflow because of its speed, flexibility and ability to integrate data from multiple sources. Through multiple C# scripts, Unity is able to generate animations and interact with objects that were created in Blender or added into the Unity scene. As shown in Figure 25, to expedite the CFD to VR workflow, a pre-programmed Unity template was created where the initial number of

steps changed from over fifty to only four necessary steps for the user to view the CFD simulation in VR.

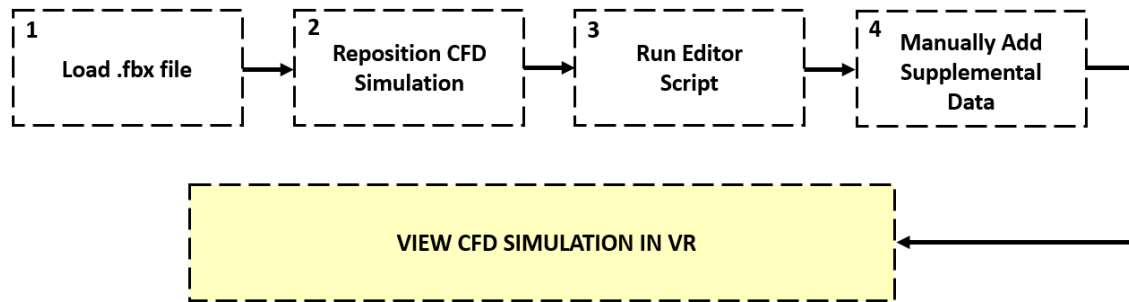


Figure 25: Required Unity template steps to view CFD simulation in VR

2.2.6.1 Unity Template

The Unity template was created for a standalone stereoscopic projector, Gear VR, and Oculus Rift. If the user prefers to view content using Microsoft HoloLens or IVE, changes to the Unity scene, C# code, and custom packages are required. As seen in Table 3, the Unity automated template has a minimum and maximum output. Changes to the template are required for more advanced or specific tasks. The Unity template, via automated code, animates the streamlines/glyphs, animates/changes the wall mesh texture, and toggles through medical imaging data (i.e. CT, MRI). Importing/animating a flow waveform and assigning proper labels to the scales (velocity, pressure, TAWSS, OSI) are not automated. However, the instructions are straightforward. These instructions can be accomplished relatively quickly and do not require alteration to any code. If the user does not have a full set of complementary data such as volumetric imaging data, or wall mesh scalar information like pressure, the Unity template still works. If the user is

interested in adding supplemental medical imaging data such as histology, adjustment to the Unity template and prewritten code would be required.

Minimum Output	Maximum Output	With change of code, Additional Output
Animated Streamlines/glyphs	Flip through wall indices including one or more of pressure, TAWSS, or OSI	<i>View supplemental medical imaging data such as histology/myograph</i>
Fly throughout simulation	<i>Animated flow waveform</i>	<i>View multiple Unity scenes</i>
<i>Scales</i>	Toggle through volumetric imaging data	<i>Fly through model on a set track</i>
Overhead map of simulation		

Table 3: Minimum output, maximum output, and possible alterations of the Unity template. Italicized text indicates the user must manually alter the Unity prefab

2.2.6.2 Load .fbx File

Once the Unity template is opened, the user copies his or her .fbx file created in Blender, and replaces the .fbx file in the Unity template “Asset folder.” After the .fbx file is overridden, the Unity scene is refreshed. This step can take up to 15 minutes to complete and eventually creates a prefab called ParaView_CFD. After the “prefab” is created, the user drags and drops the CFD prefab into the hierarchy.

In addition to the newly imported CFD prefab in the hierarchy as seen in Figure 26, the user should also see Scene Objects and OVRCameraRig. The Scene Objects includes Map Camera, Audio Source, Background Cylinder, and Directional Light. The Map Camera records a birds-eye view of the CFD simulation. It is also used to show

where the user is at in relation to the simulation. The Audio Source plays a heartbeat noise that is synced to the animated streamlines/glyphs script. The Background Cylinder encapsulates the CFD simulation. It is best practice in VR to have a distant point of reference when there is a moving camera to avoid simulator sickness. The Directional Light makes the simulation look more enhanced in the VE and is also used to cast shadows. OVRCameraRig includes TrackingSpace and the HUD. The OVRCameraRig is used for camera and Oculus Rift controls. The rig also allows for a HUD to be attached to the head. When the user moves around the simulation, the HUD moves with it. TrackingSpace is part of the ORVCameraRig prefab provided by Oculus. The HUD is where all of the supplemental data is viewed including the canvas (buttons), scales, map, and flow waveform. The Unity template also includes pre-imported textures, shaders, and materials that make the simulation look realistic in a VE.



Figure 26: List of objects in the Unity Template Hierarchy

2.2.6.3 Scale, Rotate, and or Reposition CFD simulation

The next step is to scale, rotate, and or reposition ParaView_CFD prefab so it is visible in the HUD's Map View. This can be done by selecting on the ParaView_CFD prefab in the hierarchy, then in the inspector transform, adjust the Position, Rotation, and or Scale until the simulation appears properly aligned in the Map View. For example,

Figure 27 shows the simulation before realignment and Figure 28 shows the simulation after realignment. Note that the prefab needed to be moved -14.7 in the x-direction, -5.4 in the z-direction, rotated 90 degrees in the y-direction, and scaled up by 1.5 to properly appear in the Map View.

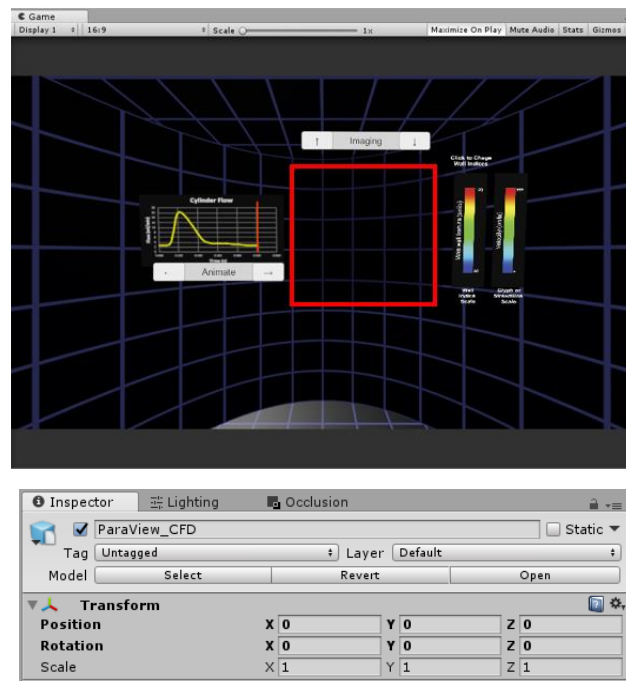


Figure 27: ParaView_CFD prefab before realignment

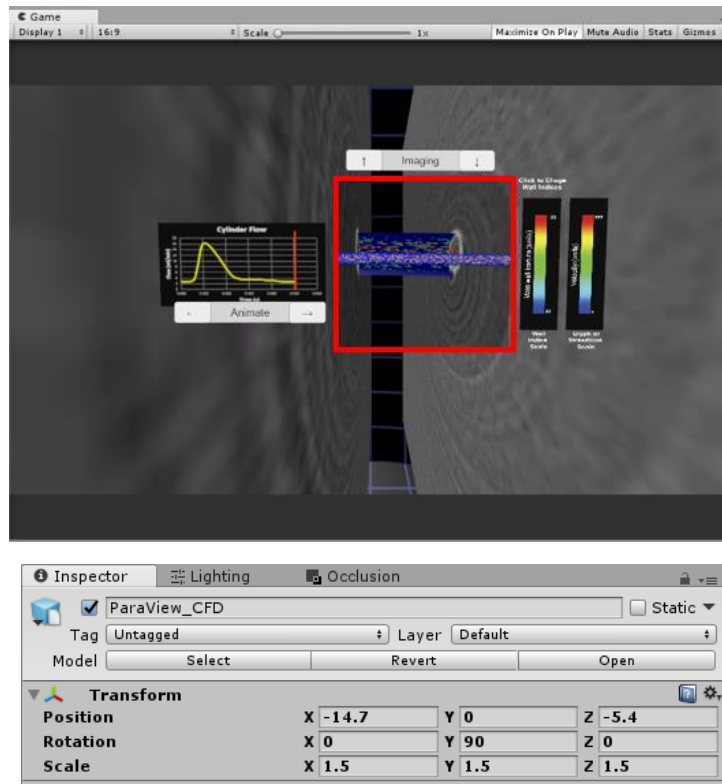


Figure 28: ParaView_CFD prefab after realignment. Note that the cylinder phantom is now viewable in the Map View. The virtual model also includes the volumetric imaging data used to create the phantom cylinder.

2.2.6.4 Run Editor Script

The last step is to run the “Set up CFD Scene” editor script by selecting Assets → Set up CFD Scene. The editor script recognizes the type of input in the .fbx file, tags child game objects, assigns materials/shaders, and assigns scripts to parent game objects.

As seen in Table 4, the Unity template has multiple C# scripts that were either provided by Oculus SDK or made from scratch. The scripts made from scratch include over 1,500 lines of code. The scripts highlighted in orange indicate that they are already assigned to game objects before the “Set up CFD Scene” editor script is run. The scripts

highlighted in purple indicate that after the “Set up CFD Scene” editor script is run, depending on the type of input, these scripts are assigned to specific game objects. The scripts highlighted in blue indicate that these scripts are highly customizable, where the script and or Unity template would need to be modified to work in the scene. Oculus SDK provides a multitude of other scripts that are also included in the Unity template but, for this specific project, they are not used. Therefore, non-used Oculus SDK scripts are not shown in Table 4.

Script Name	Function
animateGlyphs	Attaches to ParaView_CFD parent object. Creates an animation loop that makes the velocity glyphs appear as though they are pulsating OR loops through each streamline.
attachToHead	Used for multiple scene. After the start and menu scene, script finds the Head node of the camera rig and attaches this script.
blinkingAction	When initialize movement, scene replicates a “blink” (screen goes black) - done to combat simulator sickness.
BlinkingMovement	Move around simulation with mouse or joystick.
BlinkPointer	Makes the mouse icon blink in Map View.
dontDestroy	Used if user has multiple scenes. Brings the same OVRCameraRig to each scene.
GearVRControls	Play/Pause/Execute scripts once a button is clicked on Oculus or GearVR.
GlobalControls	Used if user has multiple scenes. Switch between scenes.
Histology	Display scientific imaging data dependent on location within the simulation.
MasterControl	Attaches to ParaView_CFD parent object. Universal state control. Toggles things on and off.
mouseMode	Recognizes if using the Oculus or GearVR, then moves the HUD and changes mode of navigation accordingly.

moveIndicatorLine	Moves line over flow waveform, making the waveform animated. Used so the user knows which time step they are observing.
MouseLook	Controls for mouse rotation in pitch and tilt axis.
MriHiding	If user is close to volumetric image slice, it disappears.
MRSlices	Attaches to MRI parent object. Toggle through each volumetric image slice one at a time.
OVRCameraRig	Built in Oculus script. A head tracked-stereoscopic VR camera rig.
OVRManager	Built in Oculus script. Configuration data for Oculus VR.
OVRGearVRController	Built in Oculus script. Enables rendering of trackable controller if it is connected.
rotateHud	When user rotates head, the HUD also moves.
VREyeRaycaster	Built in Oculus script. Casts a ray into scene to find VRInteractive item.
VRInput	Built in Oculus script. Input required for most VR games.
VRInteractiveItem	Built in Oculus script. Script is attached to interactive game objects. Is recognized by VREyeRaycaster.

Table 4: C# scripts in the Unity Template. Orange- already in scene. Purple- assigned to game object after “Set up Scene” editor script is run. Blue- highly customized scripts, likely would need to be altered depending on what the user would like to display.

2.2.6.5 Manually Add Supplementary Data

There are two steps that the user must complete manually: add the CFD simulation’s flow waveform and label the scales for velocity (represented as either glyphs or streamlines) and wall indices. There is a prefab waveform (child object of the HUD) initially placed in the Hierarchy. It is synced to animateGlyphs script. When the streamlines or glyphs are animated, the flow waveform also is animated at the same rate. The animation appears as a moving orange line as shown in Figure 29. It is recommended

to have a flow waveform in the scene because it provides the user with a sense of time when viewing the content in VR. However, if the user does not want to display the waveform, it can be turned off. The instructions on how to program or turn off the flow waveform can be found in Appendix C.

Scales (i.e. legends) are necessary because the user needs an idea what they are looking at, otherwise they would just see colorful streamlines or glyphs animate back and forth. The legends provide values for context. Through a C# script, depending on what wall index the user is looking at, the correct corresponding scale is displayed. As seen in Figure 29, the user needs to correctly label the Scale Title with its exact units, in addition to the minimum and maximum values. The step-by-step instructions, to properly label the scales, can be found in Appendix D.

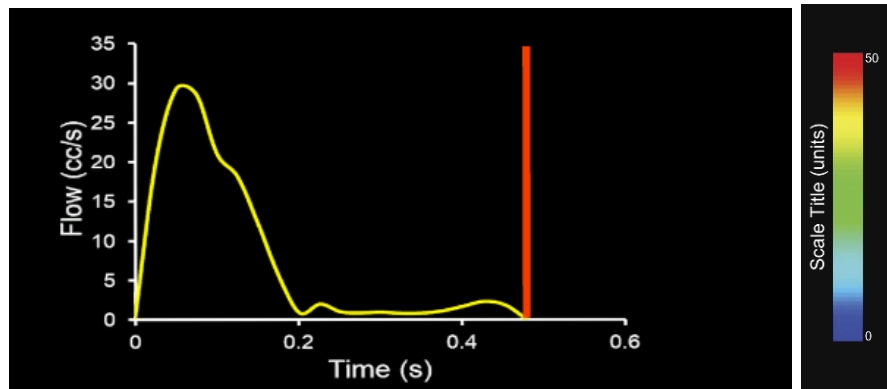


Figure 29: Left- flow waveform with orange animated line that is synced to the animated streamlines/glyphs. Right- sample scale used for the streamlines/glyphs and wall indices

When everything is correctly set up, the Unity scene should appear similar to Figure 30. Figure 31, is a screenshot of the Unity scene when Figure 30 is in “Play mode,” inside the CFD simulation.

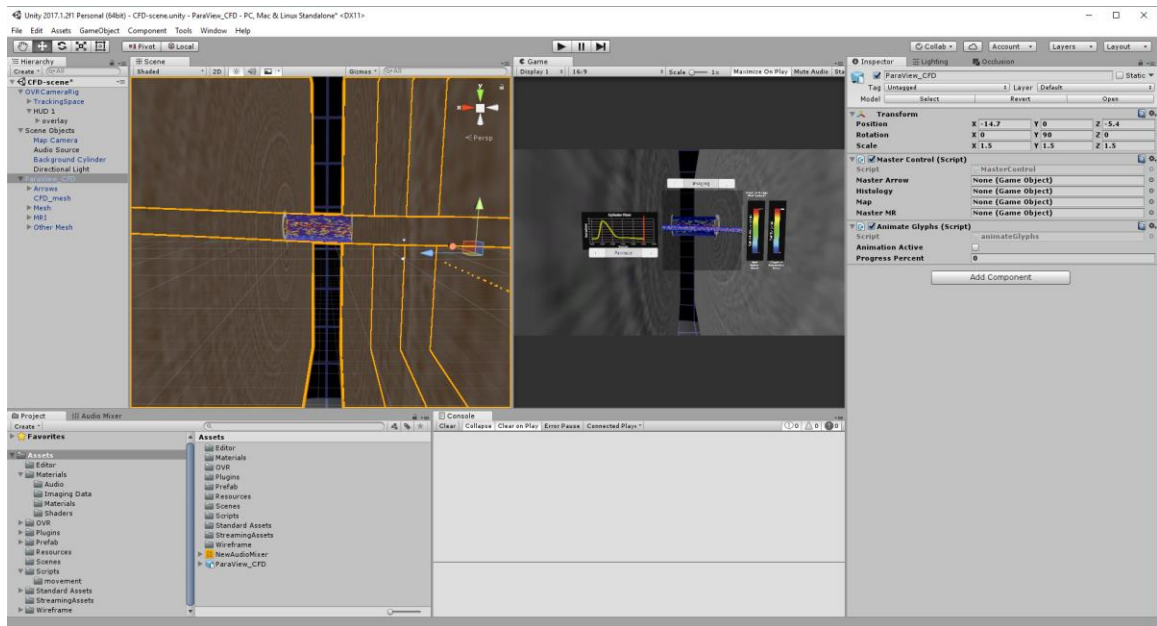


Figure 30: Screenshot of Unity's interface after upon completing the steps described above (same CFD simulation from the cylinder phantom used in Figure 22).

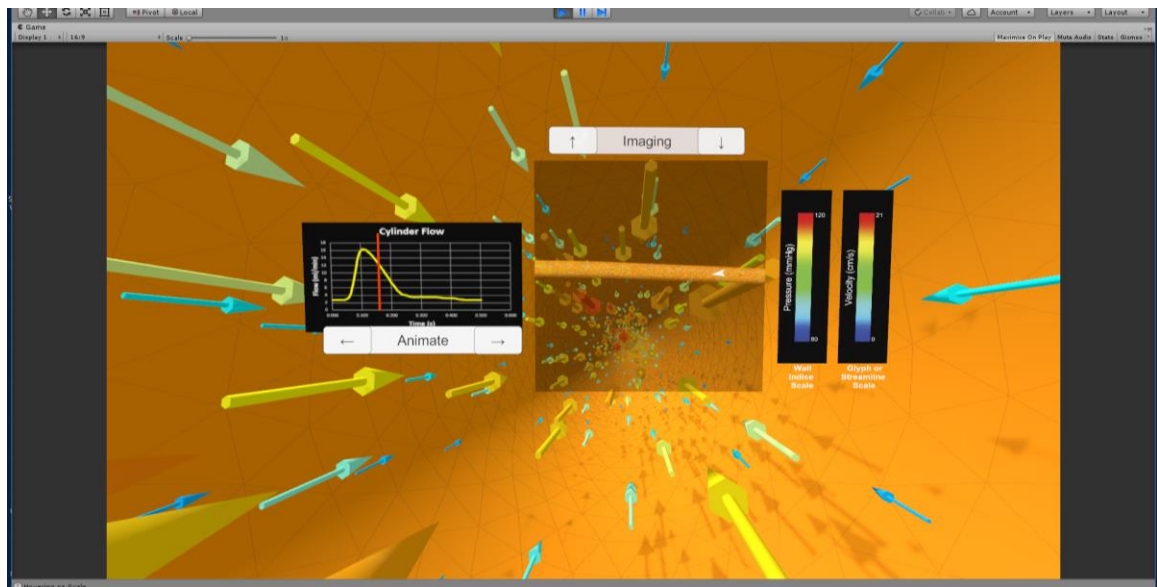


Figure 31: Unity scene in play mode (same CFD simulation from the cylinder phantom used in Figure 22).

2.2.7 Arrange VR Environment

After the Unity template has been properly setup, and if the workflow is targeted toward the Gear VR, Oculus Rift, or a standalone stereoscopic projector, no changes are required for these platforms, the CFD simulation can then be viewed in VR. Interaction is done via select buttons in the Unity Scene. As seen in Figure 32, if the user has the maximum input for the Unity template (waveform, imaging, and hemodynamic data), there are seven buttons.

1. Play/Pause animation
2. While in Pause mode, move forward one time step
3. While in Pause mode, move back one time step
4. Switch wall indices
5. Turn on/off volumetric imaging data
6. While in Image mode, move up one image slice
7. While in Image mode, move down one image slice

For the Oculus Rift and Gear VR, the trackable controller is viewed as an orange “wand” in the VR simulation. When the user positions the orange “wand” over one of the buttons, as shown in Figure 32, the button turns yellow. When the user selects the button, the corresponding action is applied to the simulation. As seen in Figure 33, for movement, the Gear VR uses the gaming pad and the Oculus Rift uses the right analog stick. For both the Gear VR and Oculus Rift, steering is achieved by the user’s head rotation. For a standalone stereoscopic projector, interaction, movement, and steering is

achieved by mouse and keyboard controls.

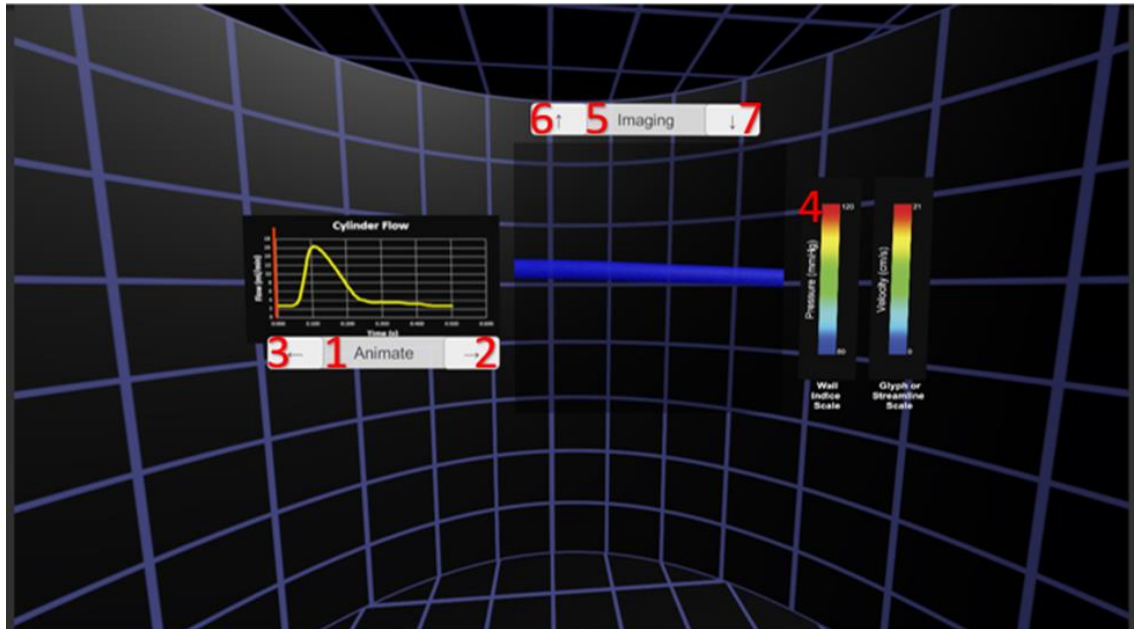


Figure 32: Controls to interact with CFD simulation (same CFD simulation from the cylinder phantom used in Figure 22).

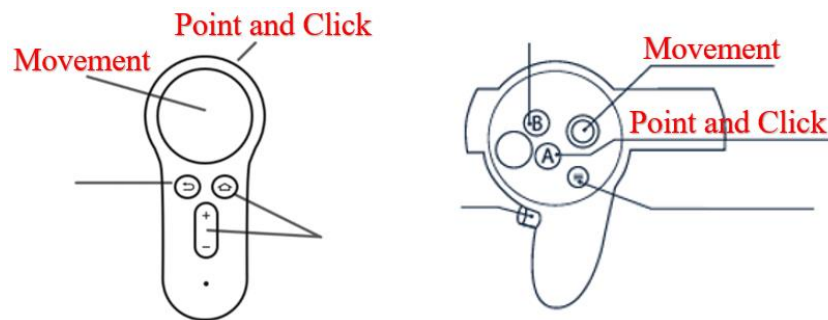


Figure 33: Left- Gear VR controls. Right- Oculus Rift controls. Both trackable controllers are viewed as an orange “wand” in the Unity Scene. The user moves the orange “wand” to point and select on buttons shown in Figure 32

For Microsoft HoloLens, the C# script must be altered so the commands can be executed via a user’s finger or voice commands. Microsoft’s API has many frameworks

and scripts to assist the user to program the commands. Or, like the C# script used for Gear VR and Oculus Rift, the HoloLens C# script can be altered to use a gaming controller as the interactive device. The Unity scene needs more modest adjustments. The supplementary material must be moved to a convenient location. In addition, the scene's background has to be removed and replaced with a solid black color so that the objects appear as semi-transparent holograms on the HoloLens' display.

For IVE, Unity currently lacks features such as clustered rendering, clustered input, nonplanar camera alignment, and infrared tracking of an interactive device, therefore a proprietary plugin is required to add these features to Unity. Special site-specific configuration files are also necessary to create an array of cameras that correctly align to the specific IVE. An example of a proprietary plugin is MiddleVR used by MARVL. MARVL executes custom interaction via a "wand" that has four buttons, a trigger and a joystick with each having its own unique task when interacting with the simulation. The interaction can be achieved by having specific commands within the C# scripts.

2.3 Results

2.3.1 Cases

The workflow was tested on three different CFD simulations: hemodynamics of the thoracic aorta with respect to congenital cardiovascular disease, vertebral vascular with application to aneurysm progression, and nasal airflow with application to virtual treatment planning.

The thoracic aorta simulation was based on a previous dissertation and journal article written by Menon et al [81]. The research centered on coarctation of the aorta (CoA), a congenital cardiovascular defect where the aorta is narrowed. This narrowing typically occurs just distal (i.e. beyond) the left subclavian artery. CoA is associated with decreased life expectancy, increased hypertension and an early risk of coronary disease [81]. The prior study examined rabbits that had either untreated CoA, corrected CoA, or a normal aorta. One of the end goals of the study was to examine hemodynamic and biomechanical alterations that persisted as a result of CoA. CFD simulations were performed for 7 rabbits in each of the experimental groups, and one data set for each group was extracted for use in the current work. Available data included blood flow velocity, pressure, TAWSS, OSI, magnetic resonance angiography (MRA), and blood flow waveforms from performing a CFD analysis. In addition, there was Verhoeff-Van Gieson stains (VVG), immunohistochemical (IHC) stains, and myograph data of vascular function in response to chemicals that induce contraction or relaxation during previously-conducted scientific experiments. The objective with this data was to use VR to compare

the CFD and experimental results between the different thoracic aortas of each group. Each of the three thoracic aorta examples (CoA, corrected, and control) used the same methods to create the VR simulations. All animal data for use in the current work was obtained from studies that had obtained IACUC approval.

In addition to the brain aneurysm pulsatile CFD simulation, there was also 4D flow data from the same aneurysm [82, 84]. While CFD and 4D flow results are excellent tools to non-invasively study the complex flow patterns of the aneurysms, both have disadvantages regarding the accuracy when trying to replicate the pathophysiologic condition. CFD uses boundary conditions that are not always physiologically accurate. 4D flow (i.e. 3D phase-contrast MRI over time) sometimes introduces noise due to the image resolution and or the velocity encoding features implemented with the phase contrast MRI sequence used. A biomedical engineering professor from Purdue University sought to have a route to compare a CFD simulation and 4D flow data from the same brain aneurysm using VR. The end goal of this case study was to view the brain aneurysm in its natural 3D shape and observe the complex flow patterns that are caused by the aneurysm. The human data to create the CFD simulation and the 4D flow data came from IRB-approved studies and were provided to MARVL in a de-identified format.

The airway simulation was based on a steady CFD simulation where different aspects of the associated model were taken from different studies by the researching team that provided a merged model for the current work. The nasal airway and pharynx came from two separate patients while the bronchioles were created using CAD [83]. The collaborating researcher wanted to replicate the path an air molecule would take from the

left nasal cavity to the bronchioles, somewhat representing an endoscopy. The results would be used with application to virtual treatment planning. The collaborating researcher was also interested in viewing the air flow patterns near the intersection between the nasal cavity and the pharynx. The human data to create the CFD simulation came from IRB-approved studies and were provided to MARVL in a de-identified format.

2.3.1.1 Thoracic Aorta Simulation

Format of CFD results

The MRA imaging data from each thoracic aorta (control, CoA, and corrected) were obtained using GE Healthcare's 3-T Sigma Excite MRI scanner at the Medical College of Wisconsin [76]. Using an earlier generation of SimVascular, each simulation was segmented and created into a 3D model as discussed in detail elsewhere [72, 76]. Each of the three CFD simulations were presented as twenty-five .vtu files, representing each time step, and two .vtk files, representing TAWSS and OSI, to MARVL in early 2016. Each simulation used the same workflow.

Convert CFD results into 3D format

For the thoracic aorta CFD simulation, it was decided to view velocity as vectors. Therefore, the glyph workflow was used. The following was performed for each of the three cases (CoA, control, corrected). First, each .vtu file was loaded into ParaView. Next, pressure was assigned to display on the aorta wall for each time step. Glyphs were

then added and properly scaled. The Python script was then run, where each .vtu file was saved as a .vrml file, containing velocity (glyphs) and pressure (wall) information in a select folder. Each case also included OSI and TAWSS results (.vtk). After each time step was saved as a .vrml file, both .vtk files were loaded into ParaView, and then in the same folder, exported as .vrml files.

Reorganize CFD results

The following was done for each of the three cases (CoA, control, corrected). The provided Blender template and the folder that contained the .vrml files from ParaView were copied into a master folder. The Blender template was opened where the instructions were completed, and the Python script was run. For each time step, it took roughly fifteen seconds to load. After supplementary data was added into the Blender scene, it was saved as a .fbx file.

Add supplementary data, customize for a given VR environment, and arrange for VR environment

Each thoracic aorta simulation had three sources of complementary data: MRA image slices used to create the CFD model, VVG, IHC, and myograph (functional) data from proximal and distal ends of the aorta, and an inflow waveform.

After the CFD simulation was reorganized in Blender, the MRA slices were added to the Blender scene using a scaled array. To recognize what section the user was at in the simulation, two wire-framed cubes were added into the Blender scene. One wire-

framed-cube covered the entire proximal portion of the aorta upstream of the narrowing, while the other wire-framed-cube covered the entire distal end of the aorta downstream of the narrowing. These regions generally represent the locations from which histological and myograph functional sections were obtained. After the supplemental data was added into the Blender scene, the scene was exported as an .fbx file. Using GIMP (gimp.org), the VVG, IHC, and myograph data taken from the proximal end of the aorta was saved as one PNG image. The same was done for the distal end. A flow waveform was created in Microsoft Excel (Microsoft; Redmond Washington) and saved as a PNG image. Finally, each MRA image slice, scientific data PNG images, and flow waveform PNG image was copied into the Unity template.

To dynamically compare each CFD simulation and its corresponding functional data, the following was done for each of the three groups (CoA, control, corrected). The Unity template was opened and then refreshed, causing each PNG image and the .fbx file to appear (.fbx file was then a prefab). Next, the CFD prefab was dropped into the hierarchy where it was properly scaled and repositioned. The “Set up CFD Scene” editor script was run where the glyphs, wall mesh, and volumetric image slices were properly tagged, and numerous C# scripts were assigned to various game objects. Next, each scale was properly labeled, and the flow waveform was set up.

To display the scientific data, slight alterations to the Unity template were required. First, in the HUD, a secondary plane was added right in front of the Map View. Next, a box collider and the “Histology” C# script with corresponding scientific data were added to each wire-framed cube. The script recognizes the user’s location within the simulation, and then displays the corresponding scientific image. For example, if the user

was originally in the proximal end of the CoA and then flew through the CoA, into the distal end, the image would switch from the proximal VVG, IHC, and myograph data set to the distal VVG, IHC, and myograph data set. This was designed so the user would be aware of the endothelial changes that result from CoA. The final iteration of the thoracic aorta virtual simulation, using the Oculus Rift, is shown in Figure 34.

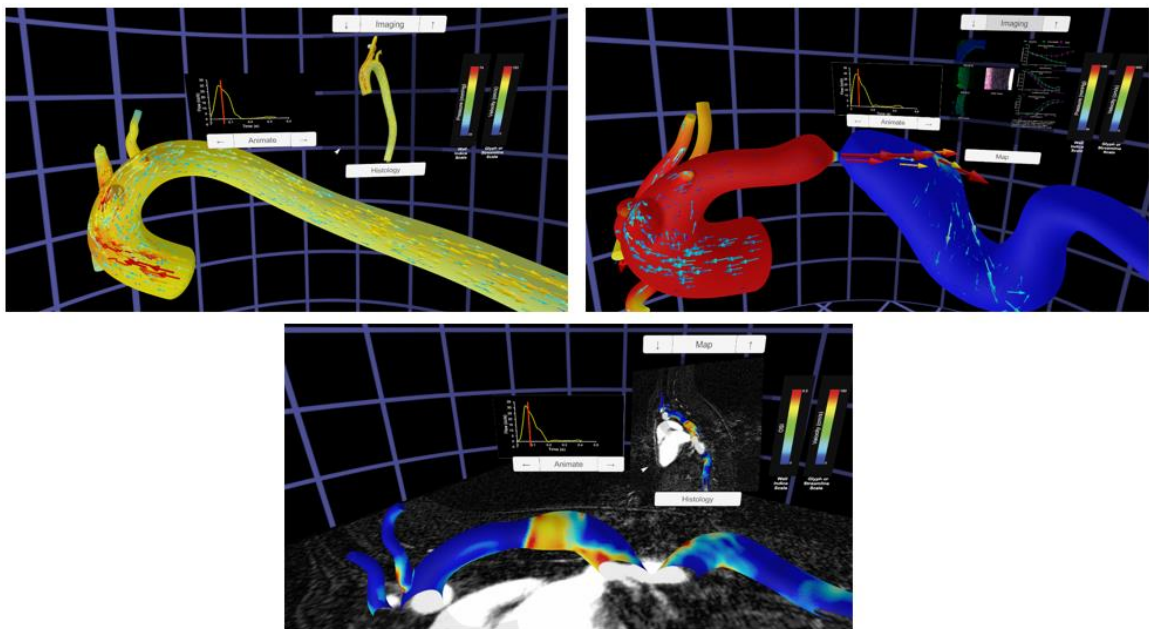


Figure 34: Thoracic Aorta CFD VR simulations. Upper Left- user observing control aorta simulation using Oculus Rift. Upper Right- user observing CoA aorta simulation at peak systole while also observing scientific data using Oculus Rift. Bottom – user observing corrected aorta simulation while viewing MRA and OSI results using Oculus Rift

Clinical Feedback

A pediatric cardiologist observed the thoracic aorta VR simulations and had the following to say:

“I think this would be a valuable education tool for our patients with coarctation of the aorta. This condition can present either very early in the first couple weeks of life, but it also commonly presents in older kids and teenagers when they come in with hypertension... this tool could help the patient population understand this condition better. I think this would be a valuable tool and of course to engage the parents... the virtual simulations are a unique, innovative sharing tool and I think it would only increase patient and family engagement.”

2.3.1.2 Brain Aneurysm Simulation

Format of CFD results

The medical imaging data used to create the pulsatile CFD simulation and 4D flow data at the request of the Purdue faculty members were previously obtained at the University of California San Francisco under an IRB-approved study. Both contrast-enhanced MRA and phase-contrast MR velocimetry (4D Flow MRI) datasets were obtained from the same patient. The CFD model was created using Hypermesh (Altair; Troy, Michigan) and the incompressible NS equations were solved numerically using Fluent [82]. The 4D flow data was processed using ParaView, Geomagic Design software (3D systems; Rock Hill, South Carolina), and EnSight (ANSYS Inc; Canonsburg, PA). Using ParaView’s Python counsel, the contrast-enhanced MRA images were segmented and streamlined [84]. Using Geomagic Design software the region of interest was defined, noise was reduced, and the data was converted into IGES format. The data was then converted into VTK format using an in-house software where it was imported into EnSight for data visualization.

Both the CFD simulation and 4D flow data were presented to MARVL as a .case file, only including velocity information. The CFD simulation had sixty-six time steps

and the 4D flow data had twenty time steps. The disparity in the number of time steps is expected. The 4D flow data depends on velocity encoding parameters implemented during the MRI scan, while the CFD simulation time step can be set to whatever the researcher determines appropriate to achieve simulation convergence and resolve details of the flow field during viewing.

Convert CFD results into 3D format

The collaborator was interested in viewing blood flow patterns that were caused by the morphology of a brain aneurysm. Therefore, the streamline workflow was used. First, the CFD .case file was loaded into ParaView. Next, with collaboration from the collaborator, the ratio between the number of streamlines and the radius for each “tubed” streamline was determined. Then each of the sixty-six time steps were saved as .vrml files using the Python script and placed in a specific folder.

The 4D flow data did not include the brain aneurysms wall mesh, only velocity encoded streamlines. However, the collaborator wanted to use the wall mesh data from the CFD simulation. Therefore, in Unity, the CFD wall mesh was manually overlaid on the 4D flow data. As regard to ParaView, the same exact process was done for the 4D flow .case file where each of the twenty time steps were saved as .vrml files (without the wall mesh for each time step) and placed in a specific folder.

Reorganize CFD results

The following was done for both cases (CFD and 4D flow). The provided Blender template and the folder that contained the .vrml files from ParaView were copied into a

master folder. The Blender template was then opened where the instructions were completed, and the script was run. For each time step, it took roughly twelve seconds to load. For the CFD simulation, after it was created, it was discovered that the wall mesh of the brain aneurysm was double-sided, and each time step had too many vertices. Therefore, the secondary Blender Python script was run to remove duplicate mesh structures and decrease the number of vertices. For the 4D flow data, after it was created, slight alterations were made to the Blender scene due to the model not having a wall mesh (was later added in Unity). Finally, the Blender project was exported as a .fbx file.

Add supplementary data, customize for a given VR environment, and arrange for VR environment

Both the CFD simulation and 4D flow data did not include any supplementary data. It only included the velocity information. The following was done for both cases (CFD and 4D flow). The Unity template was opened and then refreshed, causing the .fbx file (was then a prefab) to appear in the Unity template. Next, the CFD prefab was dropped into the hierarchy where it was properly scaled and repositioned. For the brain aneurysm CFD simulation, the wall mesh was originally solid white. While in VR, a solid white mesh could disorientate the user. Instead, a red material was created and then applied to the wall mesh. To add a wall mesh to the 4D flow data, the wall mesh of the CFD simulation was added into the 4D flow Unity scene, where it was then manually overlaid on the 4D flow data. Next, for both cases, the “Set up CFD Scene” editor script was run where the streamlines were tagged, and numerous C# scripts were assigned to various game objects. Next, the velocity scale was properly labeled while the other labels

(wall indices) and flow waveform were turned off. The final iteration of the CFD simulation and 4D flow data, using the Oculus Rift, are shown in Figure 35.

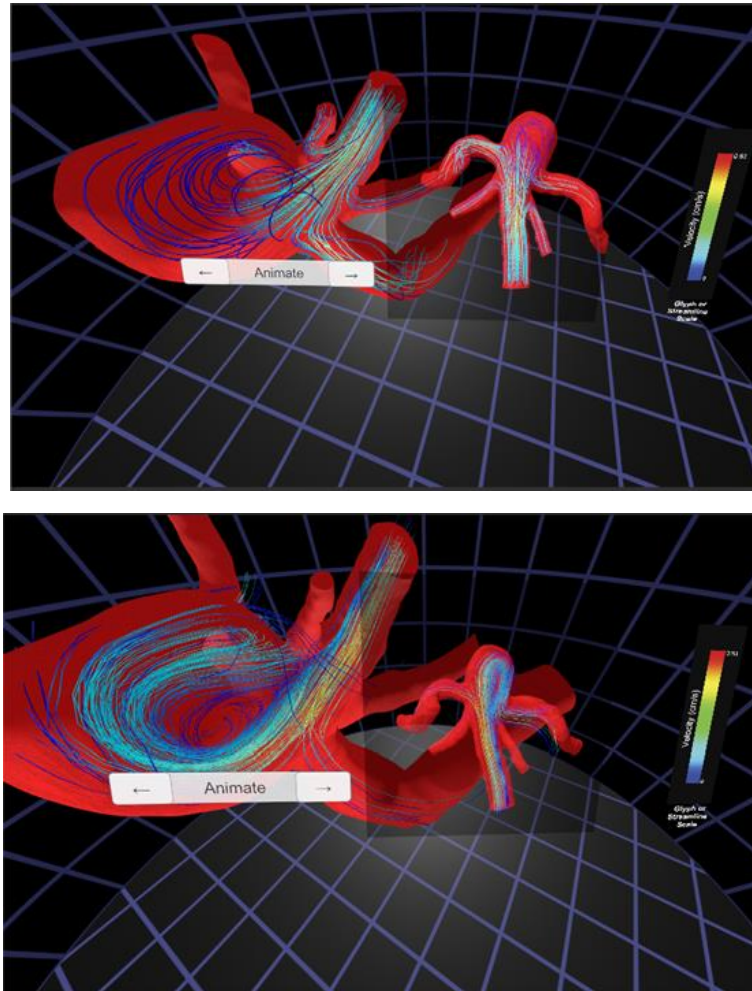


Figure 35: Upper- CFD Brain Aneurysm simulation shown using Oculus Rift. Lower- 4D flow data Brain Aneurysm shown using Oculus Rift

Clinical Feedback

A neurosurgeon observed the brain aneurysm CFD simulation and 4D flow data in VR and had the following to say:

“I see this as a first step in using flow dynamics to predict which aneurysms are more likely to rupture and therefore which ones need to be surgically treated. ... First we need to visualize the problem which can be done using the 3D virtual model to objectify what is bad and what is good.”

2.3.1.3 Airway Simulation

Format of CFD results

The airway steady CFD simulation comprised of three different data sets: nasal cavity, pharynx, and bronchioles. As mentioned above, the nasal airway and pharynx data came from CT scans of two different subjects as provided by the collaborating researcher [83]. The bronchioles were created using CAD. The CT scans were segmented in Mimics (Materialise; Leuven, Belgium) where then the nasal cavity, pharynx and bronchioles were assembled together. The complete airway model was meshed using ICEM-CFD (ANSYS Inc; Canonsburg, PA) [83]. The NS equations were solved numerically using Fluent and visualized in Fieldview (Intelligent Light; Rutherford, New Jersey). The CFD results were presented to MARVL as a .case file.

Convert CFD results into 3D format

The collaborating researcher was interested in viewing air flow patterns throughout the nasal airway down through the bronchioles. It was determined that streamlines would best represent the airway simulation CFD results. The .case file was loaded into ParaView and like the brain aneurysm example, with collaboration of the researcher, the ratio between the number of streamlines and the radius for each “tubed” streamline was determined. Two sets of “tube” streamlines were added because there were two inlets, the nasal and oral cavity. Being that the simulation was steady, the Python script was not required since temporal progress along a flow or other input waveform was not applicable. Instead, the ParaView scene was exported as one .vrml file.

Reorganize CFD results

Unlike the previous two examples, the airway simulation only had one .vrml file, therefore, the Python script was not needed in Blenders text editor. A blank Blender project was opened and the single .vrml file was imported into the Blender scene and scaled. Unnecessary lamps and cameras that were rooted in the .vrml file were deleted. The Blender project was then saved as a .fbx file.

Add supplementary data, customize for a given VR environment, and arrange for VR environment

The airway simulation did not include any supplementary data. The CFD simulation was steady, initially lacking animation. Therefore, slight alterations to the Unity template were required. The Unity template was opened and refreshed, causing the .fbx file (was then a prefab) to appear in the Unity template. Next, the CFD prefab was dropped into the hierarchy where it was properly scaled and repositioned. Labels were assigned to the scales. Being that the airway simulation was steady, the “Set up CFD Scene” editor script was not required because there were no successive streamlines to animate through. Instead, to create animation, a camera was placed at the tip of the nasal airway. Using Unity’s animation tool and subsequent C# scripts, a camera fly-through starting from the nasal cavity ending at one of the bronchioles was created. To make the simulation more realistic, a texture was created that mimicked the look of epithelium within the airway. The texture was then overlaid on the wall mesh. The final airway Unity project, displayed in MARVL, is shown in Figure 36, and details pertaining to clinical feedback from a derivative project are presented in more detail within the discussion.

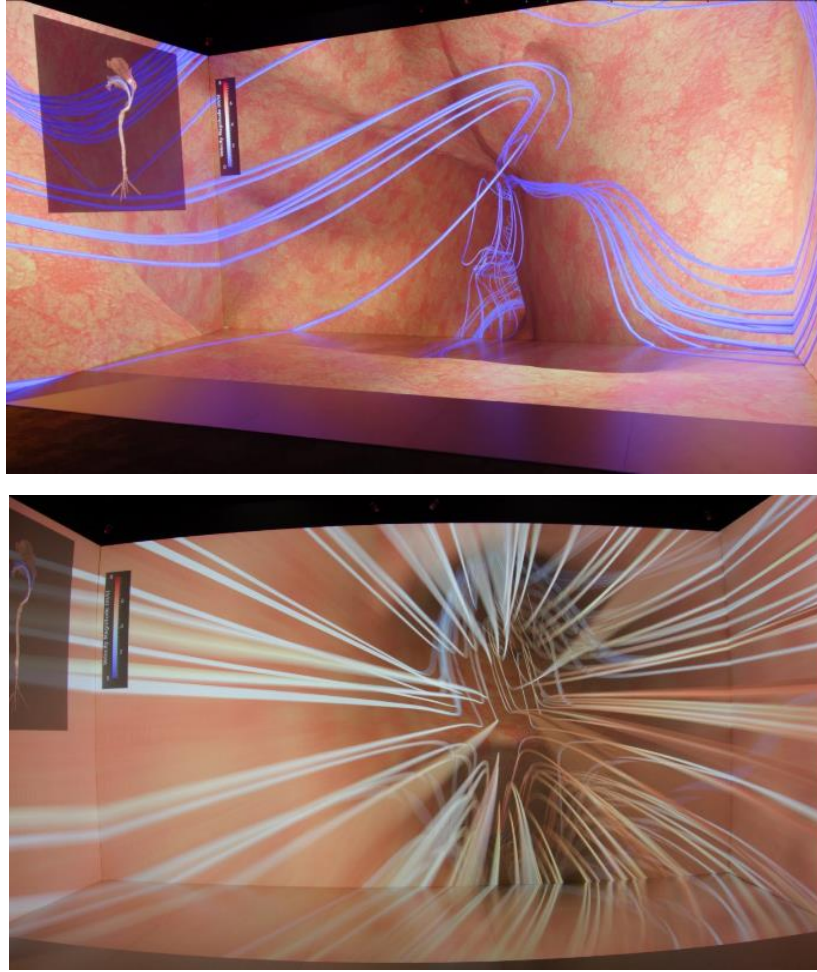


Figure 36: Airway VR simulation being displayed at MARVL

2.4 Discussion

2.4.1 Novice Users CFD to VR Workflow Experience

Application of the CFD to VR workflow was tested on two users, a graduate and an undergraduate student studying Biomedical Engineering at Marquette University. As seen in Table 5, the experience levels between the two users were slightly different. The graduate student was chosen because it was under the assumption that the individuals who would use the CFD to VR workflow would have prior knowledge in CFD but, little knowledge in VR, specifically in Blender and Unity. The undergraduate student was chosen as the baseline, showing that a novice person in CFD and VR could complete the workflow.

<i>Prior Experience?</i>					
	CFD	VR	ParaView	Blender	Unity
<i>Graduate student</i>	Yes	No	Yes	No	No
<i>Undergrad Student</i>	No	No	No	No	No

Table 5: Experience levels of tested users

The graduate student was the first user to test the CFD to VR workflow. The graduate student was presented with a PNG flow waveform and 20 .vtu files that were generated using SimVascular. The CFD to VR workflow was successfully completed in one hour and seven minutes. As seen in Table 6, the minimum output along with an animated flow waveform were shown in VR. The completion time also included

processing time (running the Python script in ParaView, running the Python script in Blender, Opening Unity, and reloading the .fbx file into the Unity scene). For the workflow, specifically to convert CFD files into 3D format, the subject used an older version of ParaView. This resulted in confusion for the graduate student because the older version of ParaView had different syntax to complete certain steps than the provided workflow made using ParaView version 5.4.2. After the test, the workflow was updated to include step-by-step instructions for ParaView version 5.2.0 to 5.4.2 (newest ParaView version at the time of the paper). Overall, MARVL personnel were satisfied with the result. Before the CFD to VR workflow, a person who had no prior knowledge of Blender or Unity could not have easily viewed their simulation results in VR.

The undergraduate student used the workflow after the graduate student, and was presented with the same data. The undergraduate user had a difficult time initially understanding ParaView. After the undergraduate was trained how to use ParaView, the workflow was successfully completed in fifty-nine minutes, including the processing time for each step. After successfully completing the workflow, the undergraduate mentioned that if they used the step-by-step workflow videos from the start, they would have finished much sooner. As seen in Table 6, the minimum output along with an animated flow waveform were shown in VR. Figure 37 depicts what the final Unity scene looked like for both the graduate and undergraduate student. As a frame of reference, a user who was proficient in ParaView, Blender and Unity was able to convert the same data set into VR in eighteen minutes, including the processing time for each step.

Minimum Output	Maximum Output	With change of code, Additional Output
Animated Streamlines/glyphs	Flip through wall indices including one or more of pressure, TAWSS, or OSI	View supplemental medical imaging data such as histology/myograph
Fly throughout simulation	Animated flow waveform	View multiple Unity scenes
Scales	Toggle through volumetric imaging data	Fly through model on a set track
Overhead map of simulation		

Table 6: Output from the CFD to VR workflow- area highlighted in red represents what both users presented in their final Unity scene

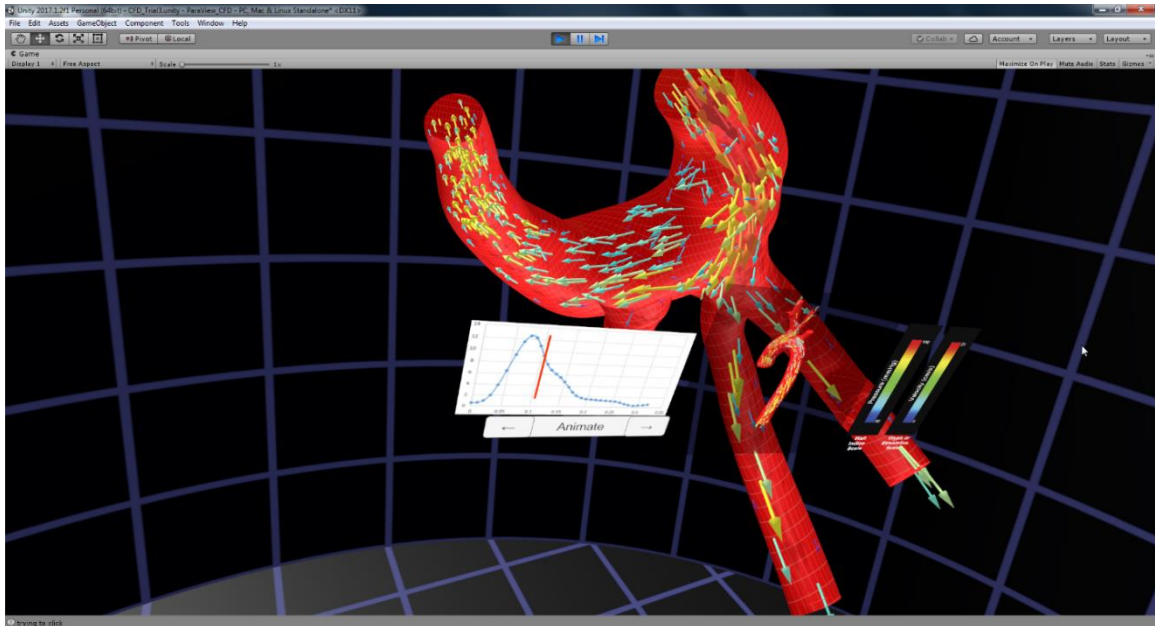


Figure 37: Final Unity scene created by both users

2.4.2 Clinical Feedback from Derivative Project

Feedback from clinicians on the practicality of using VR CFD simulations in a clinical setting, whether it be research, surgical planning, or patient education/communication was obtained anecdotally. The majority of the clinical feedback came from a derivative project of the Airway VR simulation discussed earlier.

2.4.2.1 Virtual Nasal Surgery

In the summer of 2016, a professor and clinician from the Medical College of Wisconsin came to MARVL asking to create a “Virtual Nasal Surgery.” The goal of the project was to use an Oculus Rift to effectively communicate virtual surgery modeling predictions (CFD results) to assist surgeons in determining what type of nasal surgery was optimal for patients with nasal airway obstruction [85]. The professor and clinician provided MARVL with two sets of steady-state CFD simulations saved as .case files. The first simulation was from a patient who had a nasal obstruction. The second simulation was from the same patient, but after a virtual “post-surgery” to correct the nasal obstruction [85]. Both CFD simulations had steady pressure and velocity information. The human data to create both CFD simulations came from IRB-approved studies and were provided in a de-identified format.

Portions of the CFD to VR workflow were used to create the “Virtual Nasal Surgery” application. Additional steps were required beyond the scope of the current thesis. Using ParaView, seven .vrml files were created to display the pre and post surgery

wall mesh and corresponding pressure and velocity streamline information. The .vrmf files were reorganized in the same Blender scene. Using one of MARVL's supplementary workflows, two set tracts (for the right and left nasal cavity) with subsequent cameras were added to the Blender scene and then exported as a .fbx file. Slight modifications to the Unity template were required to fly down the set tracks created in Blender and toggle between the pre and post-surgery CFD simulations. Figure 38 provides the final Unity application, after it was initially launched. As shown in Figure 39, when inside one of the nasal cavities, the user can toggle between pre and post-surgery CFD simulations, observing the changes in geometry, velocity, and pressure information.

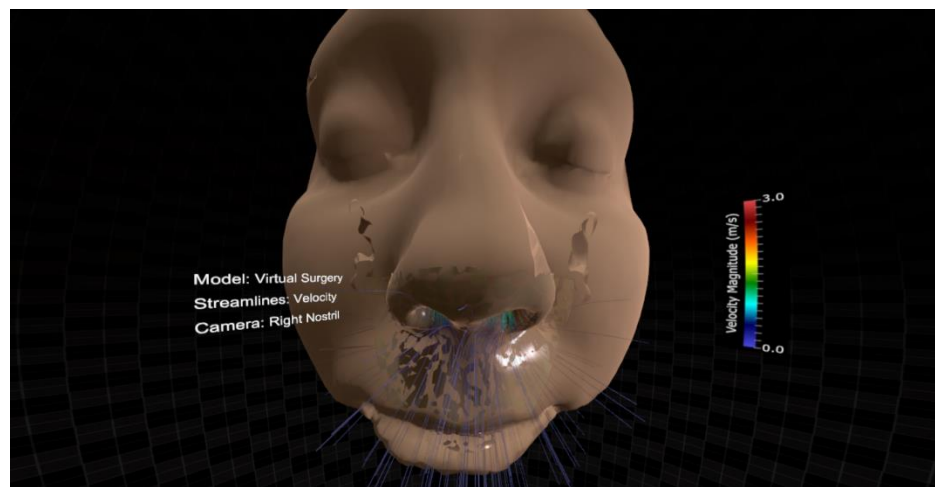


Figure 38: Initial view of the Virtual Nasal Surgery Project

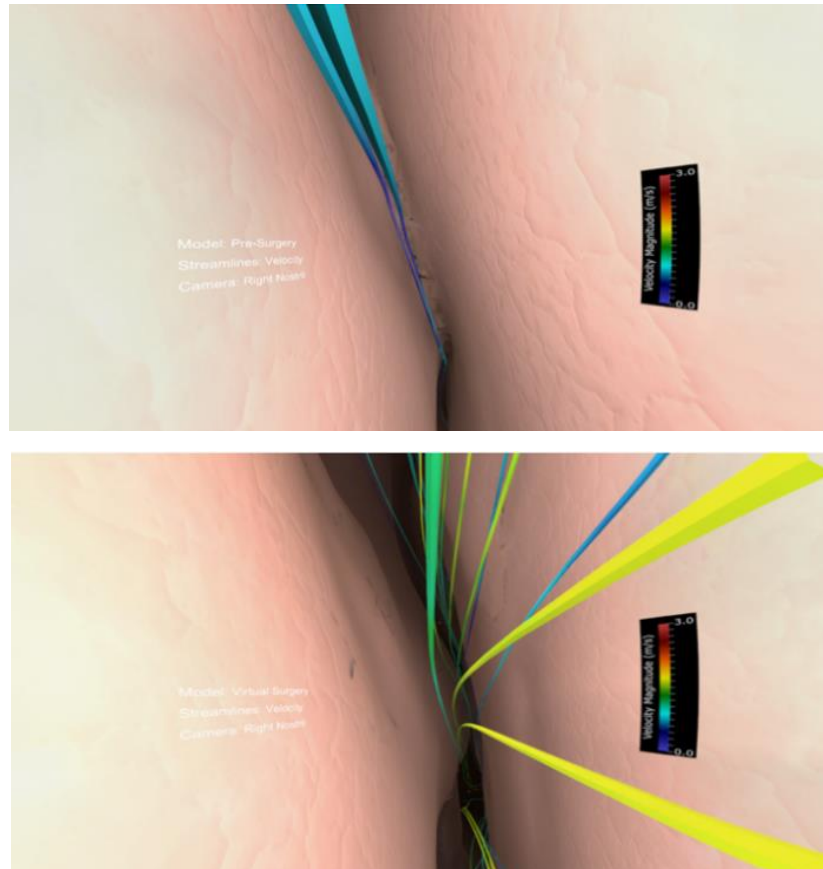


Figure 39: Upper -Inside the right nasal cavity pre-surgery observing velocity information. Lower- Inside the right nasal cavity post-surgery observing velocity information (note that there was an increase in velocity post-surgery because the patient had an unobstructed nasal cavity resulting in an increase in flow)

The Virtual Nasal Surgery application was presented at the Advances in Rhinoplasty Conference in Chicago, IL in May 2017. At the conference, surgeons from the United States and Brazil viewed the simulation using the Oculus Rift. Comments obtained by our clinical colleagues from the conference are shown in Table 7.

Surgeons Feedback from Advances in Rhinoplasty Conference	
<i>Surgeon 1</i>	<i>"...is an interesting tool to possibly teach or for patient discussions."</i>
<i>Surgeon 2</i>	<i>"I think it would be HUGE in patient education, resident teaching and research. It would be very useful in the cleft palate/VPI world, rhinoplasty and nasal applications, as well as sleep apnea evaluation. Combining with video-based data would also be great to evaluate mucosal changes and have better detail."</i>
<i>Surgeon 3</i>	<i>"This would be great to show patients and demonstrate nasal airway changes"</i>
<i>Surgeon 4</i>	<i>"I think a big positive to its use would not just be for the surgeon but for education of the patient. Patient benefit may far outweigh surgeon benefit and further strengthen the patient surgeon relationship and confidence in what the surgeon is suggesting to treat their problem."</i>
<i>Surgeon 5</i>	<i>"Great program. Really useful"</i>
<i>Surgeon 6</i>	<i>"Improves 3-D sense of surgery. Makes sure I am aware of total anatomy Frontal/Ventral; cranial, caudal"</i>
<i>Surgeon 7</i>	<i>"Great experience"</i>
<i>Surgeon 8</i>	<i>"As surgeons, this can help us to share the decision with our patients"</i>

Table 7: Surgeons feedback of the Virtual Nasal Surgery Unity App from the Advances in Rhinoplasty Conference

After the conference, the Medical College of Wisconsin clinician who spearheaded the “Virtual Nasal Surgery” application had the following to say about the simulation:

“The 3D model was very impressive. The general feedback from everyone was that this would be a very useful educational tool in this context and many other surgical contexts as a way to see the 3D relationships between structures. The ability to use the VR goggles to see the nose in a full 3D way was very cool, very visual, and very helpful. The addition of the two simulations on top of each other was a great way to visually see surgical changes. Surgery is a very visual field, and this was very helpful. As I was more familiar with the CFD information, having this information to visually see in the model was more impressive to surgeons than just seeing the simulations in 2D or the numbers.”

A few months after the Advances in Rhinoplasty Conference, the same neurosurgeon who observed the CFD simulation and 4D flow data from the same brain aneurysm viewed the “Virtual Nasal Surgery” application, and had the following to say:

(how he currently goes over a surgery with the patient) “... I use a spinal model, and then I use a real model MRI X-ray, then I show the patient this is what I'm going to do... I wish I could show them this is what it looks like before surgery, and this is what it will look like after surgery. But, you are basically asking them to visualize in an abstract term of what the post-surgery would look like. If you could show them what it looks like, using 3D geometry, that is very helpful for them to understand.”

2.4.2.2 Summary

The clinicians above concurred that viewing medical simulations in VR was impressive and had many advantages including patient education, resident teaching and research. It was their belief that visualization of a procedure gives the patient insight as to what the issue is and how surgery would rectify the problem. Finally, the clinician from the Medical College of Wisconsin, who had more experience with CFD, commented that

visualizing CFD results as streamlines in VR were “more impressive” than viewing the CFD results using spreadsheets or on a 2D monitor.

2.5 Limitations and Future Directions

The first limitation to this study is that MARVL has yet to do a scientific test to determine whether viewing CFD data in VR can add more value to CFD research when compared to traditional ways of studying CFD results. The study could also examine whether viewing CFD results in VR may be better used as a complementary tool to convey his or her results. As noted in Table 7, the surgeons at the Advances in Rhinoplasty Conference, believe that viewing simulations in VR could have a significant impact on patient education. That alone suggests an advantage to viewing simulations in VR, but it still does not prove that viewing CFD results in VR could add value to CFD research. MARVL has significantly reduced the time it takes to view CFD results in VR. But, converting the results to be shown in VR is an extra step that takes at least eighteen minutes. A future study would be helpful in determining the validity/practicality of viewing CFD results in VR.

A second limitation to the study is related to the boundary conditions used to create the CFD simulations. Boundary conditions are not always physiologically correct. For example, assuming the pressure at the outlet is zero is not realistic for CFD simulations involving arterial vasculature, and the results could be questioned by fellow colleagues. Moreover, if the VR CFD simulation is used as a patient educational tool, and

the CFD results are inaccurate, the VR simulation could mislead the patient, potentially resulting in unrealistic expectations or outcomes.

An area where the CFD to VR workflow can improve is the adding volumetric imaging data in Blender step. After the .vrml files are created, the workflow is almost fully automated with the exception of the volumetric imaging step. This step is very tedious and can be time consuming. A future direction could be to find a way to script: align the medical image to the CFD model, add a properly scaled array, add the medical imaging data for each corresponding plane, and properly naming each plane's object, texture, and material. If this step were scripted, it would automate everything needed to be done in Blender.

Another area where the CFD to VR workflow could improve would be to allow for deformable walls which can be done using fluid-structure interaction (FSI) simulations. These are simulations where the fluid flow deforms in a physical structure. For the current workflow, the walls must be rigid. Potential changes in the workflow to allow for a deformable wall would be to first load the FSI simulation results into ParaView. The displacement and wall mesh values would be exported as a CSV file. Then using a programming language for each time step, the wall mesh and displacement values would be used to create a new mesh, representing the displacement of the mesh for that particular time step. The new wall mesh for each time step would then be exported as a file that stores 3D geometry. Using Blender's text editor, a Python code would be written that would import the first time step 3D geometry file, then store the remaining time steps position values. Using Blender's shape keys, the first time step 3D geometry would be modified sequentially to the position values of the next sequential time step,

creating a seamless transition. The Blender project would then be exported and loaded into Unity as a .fbx file, creating a prefab. Using the prefab's animation component and subsequent scripts, the wall deformation would be synced to the pulsating glyphs or streamlines.

The most drastic alteration to the current workflow would be to eliminate the entire Blender step. By doing this, it could potentially expedite the total workflow. The reason Blender is used is because the import and export formats between ParaView and Unity do not align. ParaView's only suitable exportable 3D formats are .x3d or .vrmf. Unity's only suitable importable 3D formats are .fbx, .dae, .3ds, .dxf, .obj and .skp. For the current workflow, Blender takes .vrmf files from ParaView, then exports the entire CFD simulation as a .fbx file, where it can then be imported into Unity. To bypass Blender, there are three alternative methods (1) add a plugin to ParaView to export .fbx or .obj files (2) add a plugin to Unity to import .vtk or .x3d files or (3) write a custom format on both ends.

CHAPTER 3: IMMERSIVE POWERPOINT

3.1 Previous Work Done at MARVL

MARVL has been used as an integral part of select engineering lectures since it first opened in 2014. One lecture in particular was a fluid dynamics lecture where the professor viewed a VR, CFD carotid artery simulation. The professor used the CFD simulation to convey the fluid dynamic topics that the students learned in the classroom. For this specific lecture, at the touch of a button, the mode went from viewing the CFD simulation to viewing a 2D PowerPoint presentation. Though this process was very reliable, there were many issues including:

- The PowerPoint was in 2D
 - It did not take full advantage of MARVL's many benefits related to didactic teaching and learning [21-36]
 - i.e. students are fully immersed in their respective subject matter, increasing a sense of presence, understanding and enthusiasm
 - If there was not a relevant VR model for a professor's lecture, there was no reason to use MARVL
- A new Unity project had to be built for each presentation
 - Professors had to send their PowerPoint slide to MARVL days in advance
- Lacked interactivity

- Lacked the ability to annotate/draw on the slides

The Director of MARVL, Dr. LaDisa, spearheaded a project in 2017 as part of his role as the Lafferty Professor in the OCOE at Marquette University to make MARVL more accessible to Marquette's engineering professors. As discussed in the introduction, VR allows students to be fully immersed in their respective subject matter thereby increasing their sense of presence, understanding and enthusiasm [21-36]. The way of viewing PowerPoints in MARVL above was not conducive to many of professors in the OCOE. The end goal of the Lafferty Project was to have an easy, expedited process where a professor could view his or her PowerPoint presentation and have an exciting, interactive lecture that the students would enjoy, learn and remember. At completion, the project was able to take a professor's PowerPoint slide, setup the 3D presentation under 30 minutes and include the ability to view supplementary movie files and draw on each slide using a virtual wand.

3.2 Materials and Methods

3.2.1 Data Acquisition

PowerPoint, initially launched in 1990, is the world's most popular presentation software, accounting for an estimated 95% share of presentation software. The workflow that was created for the current thesis takes a standard PowerPoint deck and converts each slide for 3D viewing using MARVL.

3.2.2 Converting Slides

The workflow for creating 3D lectures was designed to make it easy for a professor to implement. The only input the professor needs to provide MARVL staff is a standard PowerPoint deck and any supplementary movie files they would like included. To create a 3D effect when viewed in MARVL, a standard PowerPoint slide as shown in Figure 40, is separated into three planes. The first plane is the background plane, the second plane is the text plane, and the third plane is the figures, graphs and or key words plane. As seen in Figure 40, the plane that is protruding most toward the user is the figure/graphs/key word plane, while the background plane is the furthest back. To create these three planes, the PowerPoint deck is saved as three distinct sets of PNG images. One generic single background slide that is used for the entire PowerPoint presentation, a text image for each slide, and a figure/graphs/key word for each slide.

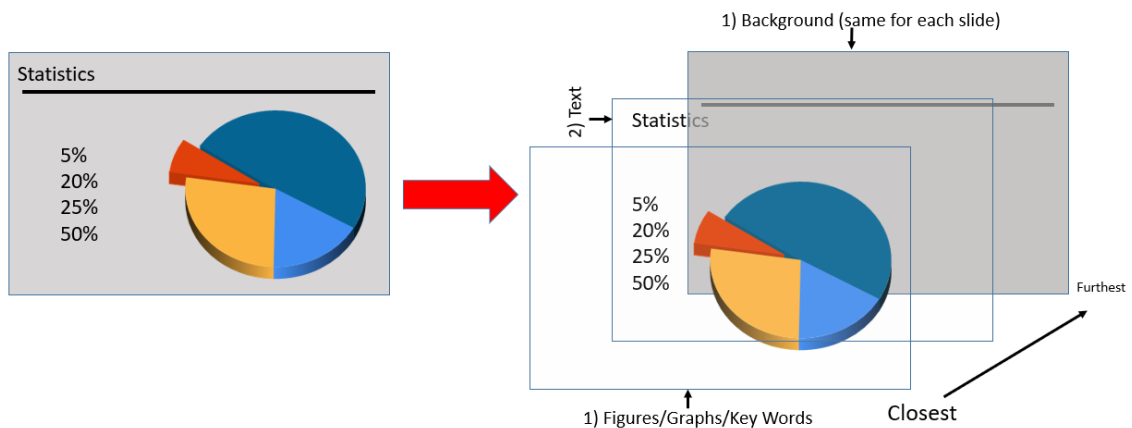


Figure 40: Left- normal slide. Right- normal slide sperarated into three PNG images: figure/graphs/key words, text, and background

To create each of these planes, the MARVL staff slightly alters the PowerPoint deck provided by the professor. First, a standard background slide is created and placed as the first slide in the deck. The background style is best matched to whatever style the professor used for his or her presentation. The background format is set to 100% transparent for the remaining slides. Next, two PDF files are created. One PDF includes the background slide and every figure/graph/key word deleted, leaving only the text. The second PDF includes a blank first slide (background slide is deleted and set to 100% transparent), and all of the text deleted, leaving only the figures/graphs/key words. The process of creating these two PDF files is fast. A standard thirteen slide PowerPoint deck that had at least one figure per frame, took under 115 seconds to create the two necessary PDF files.

Finally, using Cygwin 64 (includes ImageMagick and GhostScript) and a pre-written Bash script, each PDF is converted into a series of PNG images. The Bash script saves each slide as a PNG image, increases its resolution, resizes, renames, and places each PNG in a respective text or image folder. The text and image folder can then be easily swapped in the Immersive PowerPoint Unity project right before class.

3.2.3 Transform to Meet Virtual Environment

A Unity project was created to dynamically load new PowerPoint slides at runtime. For each presentation, the only task required by the MARVL staff is to convert the slides and put the image, text, and movie folders in the Unity project's StreamingAssets folder. As seen in Figure 41, interaction with the project is achieved by

a wand that includes four buttons, an analog stick and a trigger. Like every Unity project, build-set-up and interaction is achieved by a series of C# scripts. Some of the most important for this project include: instantiate a prefab for each slide, advance to the next/previous slide, draw, and play movies.

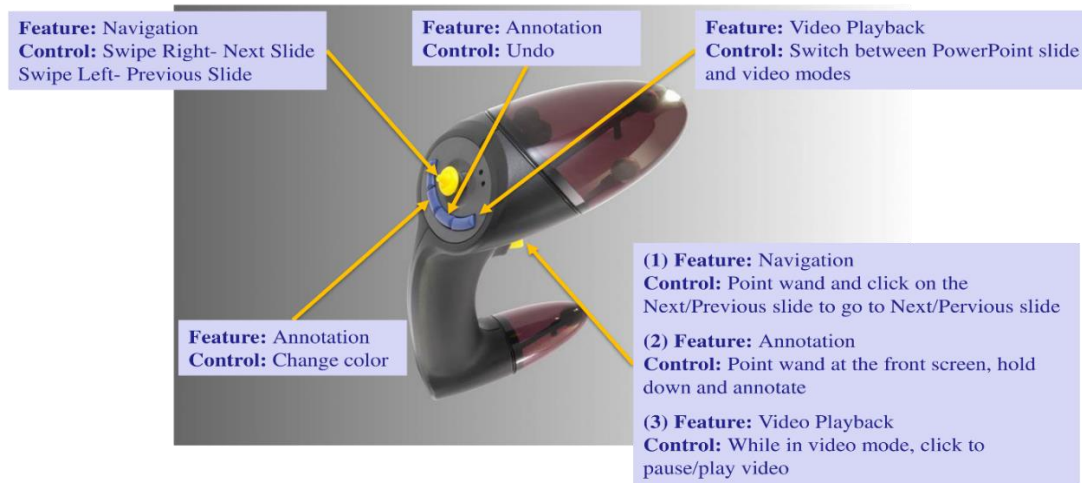


Figure 41: Wand used for Immersive PowerPoint presentation, with features and controls for each button

The program works by reading how many PNG images are in the text folder. That integer is then used to instantiate the correct number of 3D slide prefabs. The respective background, text, and figure/graph/key word PNG images are then overlaid on each prefab as a texture. The user is able to advance to the next/previous slide because after each slide is instantiated, it is placed in a game object array, neatly organizing the project. If the user would like to go to the next slide, he or she advances one slot in the array. If the user would like to go back a slide, he or she returns to the previous slot in the array. The user is able to draw on each slide by using Unity's raycast and line renderer. In Unity, raycast is one of the global physics properties. It creates a vector from the origin,

in this case, the wand, to the figure/graph/key word plane. This ray, looks like a virtual pen, and extends from the user's hand to the front wall of MARVL. When the user presses down and holds the trigger button, the user is able to draw using Unity's built in component, line renderer. The line renderer takes an array of X number of sampling points in 3D space and draws a line between each one. Movies are played using a plug-in called AVPro Video. The movie is dynamically loaded at runtime because, like the image and text folders, they are also called from the StreamingAssets folder.

Another important feature, before the Unity project was built, was properly scaling each of the three planes. As previously discussed in the Introduction, if there are two of the same objects, but one is closer, the closer object appears larger than the further away object. For this project, to create the 3D effect for each slide, the same object (plane) was placed at different depths relative to the user. However, when viewed, each plane needed to be properly scaled so when the user looks at the slide from straight on, each plane was properly aligned just like it would appear in a standard slide. The proper scaling ratio for each plane was measured using a grid. The scale of the plane was adjusted until each of the three grids were directly aligned. After alignment, the background plane was the largest plane, followed by the text plane, and the figure/graph/key word plane was the smallest. When viewed from straight on in MARVL, each slide appeared 3D with the same alignment as the standard PowerPoint presentation.

3.3 Results

3.3.1 Case Study

In the fall of 2017, two professors for undergraduate level classes at Marquette University used MARVL's Immersive PowerPoint during one of their lectures. The feedback was positive. The professors used the "virtual pen" to highlight and draw providing emphasis during the lecture. After the class was over, students were queried as to how they enjoyed MARVL's Immersive PowerPoint lecture compared to a standard lecture. The response was nearly unanimous, where students thought the Immersive PowerPoint lecture was "more exciting." It was noted that some students did not wear the stereoscopic glasses because of simulator sickness during their previous VR experiences. The students who did not wear the glasses revealed that they were able to easily track the lecture because the text was readable. An image of one of the classes using MARVL's Immersive PowerPoint can be found in Figure 42.

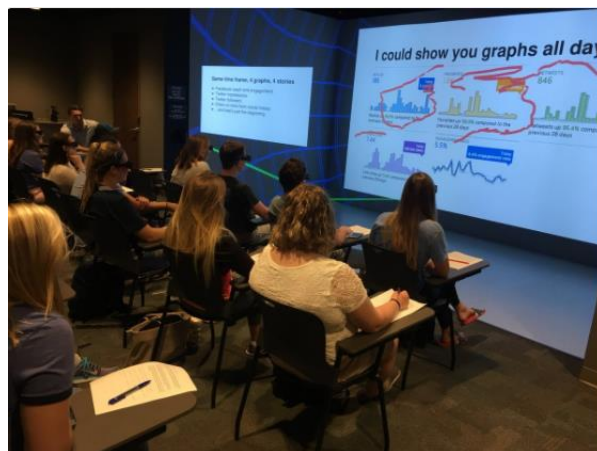


Figure 42: Marquette class using MARVL's Immersive PowerPoint

3.4 Discussion

An “Open House” was held for Marquette’s engineering professors to provide an informative/practice session using MARVL’s Immersive PowerPoint. In total, four engineering professors participated in the “Open House.” The feedback was mostly positive where three out of the four professors mentioned using MARVL’s Immersive PowerPoint for some of their important lectures during the upcoming semesters. Two of the professors stated that their students would likely be more excited and attentive when using the Immersive PowerPoint. When the professors were trained how to use MARVL’s interactive wand, they thought the controls were easy and straightforward. The professors felt confident that they could operate the Immersive PowerPoint on their own.

3.5 Limitations and Future Directions

Future work for MARVL’s Immersive PowerPoint lecture would be to scientifically test whether having lectures in an IVE can have a positive effect (i.e. increase in retention/ knowledge) on students. As discussed in the Introduction, there are a plethora of advantages using VR in didactic teaching and learning [21-36]. However, there has not been a study that scientifically tested whether viewing a standard PowerPoint lecture in VR would have any effect on the student. A future case study could have the same course be split into two classes. Each class would be taught by the same professor and have the same PowerPoint slides. The only difference between the two

classes would be that one class would be taught in MARVL and the other class would be taught in a standard classroom. Before the study, to account for student intelligence, the students would be separated so the average class GPA would be the same. After the semester, the grades between the two classes would be compared as well as a questionnaire, polling the overall experience of the class. Although the sample size would be small and could have a large error (i.e. student understanding the course), it would be a start in determining the validity of viewing PowerPoint in 3D.

CHAPTER 4: CONCLUSION

VE such as MARVL provide a collaborative learning environment to display virtual content. To increase MARVL's userbase, two robust processes and workflows (Aim 1 and 2) were created that permit increased content and involvement within MARVL and its additional VE.

Aim 1 was designed to take biomedical CFD simulations and display it in a variety of VE. The workflow was intended to take advantage of the tremendous potential of viewing CFD simulation results and complementary data in VR [13, 14]. The workflow was successfully implemented on a variety of VE and tested across multiple clinical applications. As shown in Table 8, each clinical application had a slightly different input and output, exemplifying the flexibility of the workflow.

	Thoracic Aorta	Brain Aneurysm	Airway
<i>CFD simulation Software</i>	SimVascular	Fluent	Fluent
<i>Visualized results as:</i>	Glyphs	Streamlines	Streamlines
<i>Portion of Unity template used</i>	Maximum (CFD results, scales, hemodynamic data, flow waveform, volumetric images) + slightly altered	Minimum (CFD results, and scale)	Minimum (CFD results, and scale) + slightly altered

Table 8: Flexibility of CFD to VR workflow exemplified by the three CFD clinical cases

To encourage widespread usage of medical simulations in VR, it was imperative to decrease and simplify the steps to complete the workflow. In under one hour, Aim 1's

workflow was completed by a novice user with no prior knowledge in either CFD or VR, demonstrating the ease of the workflow resulting in the potential increase in userbase. With the changes that have been implemented and outlined in this thesis, the use of medical simulations in VR shows great promise as an educational, surgical planning, and research tool for clinicians. This project was a new, collaborative effort between engineers and clinicians.

Aim 2 was designed to take advantage of the great potential VR has as an educational tool. Multiple studies have shown the advantages of using VR in didactic teaching which increases the students' sense of presence, understanding and enthusiasm [21-36]. Aim 2's workflow was successfully tested on college level lectures and completed in a time effective manner. With positive feedback from both professors and students regarding immersive PowerPoint, Marquette University classes have an opportunity to use MARVL on a more consistent basis, potentially increasing its userbase.

BIBLIOGRAPHY

-
- ¹ Roussou, Maria. "Learning by doing and learning through play: an exploration of interactivity in virtual environments for children." *Computers in Entertainment (CIE)* 2.1 (2004): 10-10.
- ² Hansen, Margaret M. "Versatile, immersive, creative and dynamic virtual 3-D healthcare learning environments: a review of the literature." *Journal of medical Internet research* 10.3 (2008).
- ³ Mikropoulos, Tassos A., and Antonis Natsis. "Educational virtual environments: A ten-year review of empirical research (1999–2009)." *Computers & Education* 56.3 (2011): 769-780.
- ⁴ MARVL - the MARquette Viz Lab - Home, Dec. 2017, www.eng.mu.edu/vizlab/.
- ⁵ McMenemy, Karen, and Stuart Ferguson. *A Hitchhiker's Guide to Virtual Reality*. AK Peters, 2007.
- ⁶ Stuart, Rory. *The Design of Virtual Environments*. McGraw-Hill, 1996.
- ⁷ Isaac, Mike. "Facebook's Virtual Reality Business Gets a New Leader." *The New York Times*, The New York Times, 26 Jan. 2017, www.nytimes.com/2017/01/26/technology/facebook-virtual-reality-hugo-barra.html. Web. 8 May 2011.
- ⁸ "Record over \$3B AR/VR Investment in 2017 (\$1.5B+ in Q4)." Digi-Capital, Jan. 2018, www.digi-capital.com/news/2018/01/record-over-3b-ar-vr-investment-in-2017-1-5b-in-q4/#.WmT7sqinHct.
- ⁹ Sherman, William R. *Understanding Virtual Reality: Interface, Application, and Design*. Morgan Kaufmann, 2013.
- ¹⁰ Turnr, J. (Artist). (2018) *Screenshot of the video game Pokémon Go with Treecko and a Poké Ball* [Digital image]. Retrieved from Wikimedia Commons website: <http://upload.wikimedia.org/wikipedia/commons/2/2e/Kamloops%E2%80%94Thompson%E2%80%94Cariboo.png>
- ¹¹ Lee, Byoung-Kwon. "Computational fluid dynamics in cardiovascular disease." *Korean circulation journal* 41.8 (2011): 423-430.

-
- ¹² LaDisa Jr, John F., et al. "Time-efficient patient-specific quantification of regional carotid artery fluid dynamics and spatial correlation with plaque burden." *Medical physics* 37.2 (2010): 784- 792.
- ¹³ Forsberg, Andrew S., et al. "Immersive virtual reality for visualizing flow through an artery." *Proceedings of the conference on Visualization'00*. IEEE Computer Society Press, 2000.
- ¹⁴ Quam, David J., et al. "Immersive visualization for enhanced computational fluid dynamics analysis." *Journal of Biomechanical Engineering* 137.3 (2015): 031004.
- ¹⁵ Quam, David J. *Advanced Visualization and Intuitive User Interface Systems for Biomedical Applications*. Marquette University, 2012.
- ¹⁶ Berger, Matthias, and Verina Cristie. "CFD Post-processing in Unity3D." *Procedia Computer Science* 51 (2015): 2913-2922.
- ¹⁷ T.A. DeFanti, M.D. Brown, Visualization in Scientific Computing, Computer Graphics. 21 (1987).
- ¹⁸ "Case Western Reserve, Cleveland Clinic Collaborate with Microsoft on Earth-Shattering Mixed-Reality Technology for Education." Case Western Reserve University - Hololens, 2015, case.edu/hololens/.
- ¹⁹ "Microsoft Highlights CWRU, Cleveland Clinic Partnership in New Video." Case School of Engineering, 12 July 2015, engineering.case.edu/HoloLens-video.
- ²⁰ "Microsoft HoloLens: Partner Spotlight with Case Western Reserve University." YouTube, YouTube, 8 July 2015, www.youtube.com/watch?time_continue=37&v=SKpKlh1-en0.
- ²¹ Dede, Chris. "Immersive interfaces for engagement and learning." *science* 323.5910 (2009): 66-69.
- ²² Everson, Naleya, et al. "Measuring the impact of a 3D simulation experience on nursing students' cultural empathy using a modified version of the Kiersma-Chen Empathy Scale." *Journal of clinical nursing* 24.19-20 (2015): 2849-2858.
- ²³ Dalgarno, Barney, and Mark JW Lee. "What are the learning affordances of 3-D virtual environments?." *British Journal of Educational Technology* 41.1 (2010): 10-32.

-
- ²⁴ Jonassen, David H. "Objectivism versus constructivism: Do we need a new philosophical paradigm?." *Educational technology research and development* 39.3 (1991): 5-14.
- ²⁵ Conover , Emily. "3D Technology Adds New Dimension to Marquette University Teaching." *Journal Sentinel*, 14 Aug. 2014, archive.jsonline.com/news/education/3d-technology-adds-new-dimension-to-marquette-university-teaching-b99313821z1-271331381.html/.
- ²⁶ Dunston, Phillip S., et al. "An immersive virtual reality mock-up for design review of hospital patient rooms." *Collaborative design in virtual environments*. Springer Netherlands, 2011. 167-176.
- ²⁷ Dunston P.S., Arns L.L., Mcglothlin J.D., Lasker G.C., Kushner A.G. (2011) An Immersive Virtual Reality Mock-Up for Design Review of Hospital Patient Rooms. In: Wang X., Tsai J.JH. (eds) *Collaborative Design in Virtual Environments. Intelligent Systems, Control and Automation: Science and Engineering*, vol 48. Springer, Dordrecht
- ²⁸ Lee, C. H., Liu, A., Del Castillo, S., Bowyer, M., Alverson, D., Muniz, G., et al. (2007). Towards an immersive virtual environment for medical team training. *Studies in Health Technology and Informatics*, 125, 274-279.
- ²⁹ Kilmon, Carol A., et al. "Immersive virtual reality simulations in nursing education." *Nursing education perspectives* 31.5 (2010): 314-317.
- ³⁰ Rhienmora, Phattanapon, et al. "Haptic augmented reality dental trainer with automatic performance assessment." *Proceedings of the 15th international conference on Intelligent user interfaces*. ACM, 2010.
- ³¹ Moule, Pam, et al. "A comparison of e-learning and classroom delivery of basic life support with automated external defibrillator use: A pilot study." *UWE, Bristol* (2006)
- ³² Bailenson, Jeremy N., et al. "The use of immersive virtual reality in the learning sciences: Digital transformations of teachers, students, and social context." *The Journal of the Learning Sciences* 17.1 (2008): 102-141.
- ³³ Thorndyke, Perry W., and Barbara Hayes-Roth. "Differences in spatial knowledge acquired from maps and navigation." *Cognitive psychology* 14.4 (1982): 560-589.

-
- ³⁴ Johnson, D., Johnson, R., & Skon, L. (1979). Student achievement on different types of tasks under cooperative, competitive, and individualistic conditions. *Contemporary Educational Psychology*, 4, 99-106.
- ³⁵ Huang, Hsiu-Mei, Ulrich Rauch, and Shu-Sheng Liaw. "Investigating learners' attitudes toward virtual reality learning environments: Based on a constructivist approach." *Computers & Education* 55.3 (2010): 1171-1182.
- ³⁶ Shih, Ya-Chun, and Mau-Tsuen Yang. "A collaborative virtual environment for situated language learning using VEC3D." *Journal of Educational Technology & Society* 11.1 (2008).
- ³⁷ Pan, Zhigeng, et al. "Virtual reality and mixed reality for virtual learning environments." *Computers & Graphics* 30.1 (2006): 20-28.
- ³⁸ Limniou, Maria, David Roberts, and Nikos Papadopoulos. "Full immersive virtual environment CAVE TM in chemistry education." *Computers & Education* 51.2 (2008): 584-593.
- ³⁹ Goldstein, E. Bruce, and James Brockmole. *Sensation and perception*. Cengage Learning, 2016.
- ⁴⁰ Pfautz, Jonathan David. *Depth perception in computer graphics*. No. UCAM-CL-TR-546. University of Cambridge, Computer Laboratory, 2002.
- ⁴¹ Wolfe, M. Jeremy, Kluender R. Keith, and Levi, M, Dennis. *Sensation & Perception*. Sinauer Associates, Inc., 2009.
- ⁴² Groh, Jennifer G. "Vision: Binocular Cues for Depth Perception." Duke University.
- ⁴³ Alfano, Patricia L., and George F. Michel. "Restricting the field of view: Perceptual and performance effects." *Perceptual and motor skills* 70.1 (1990): 35-45.
- ⁴⁴ Dolezal, Hubert. *Living in a world transformed: Perceptual and performatory adaptation to visual distortion*. Academic Press, 1982.
- ⁴⁵ Lin, JJ-W., et al. "Effects of field of view on presence, enjoyment, memory, and simulator sickness in a virtual environment." *Virtual Reality, 2002. Proceedings. IEEE*. IEEE, 2002.

-
- ⁴⁶ LaValle, Steven M., et al. "Head tracking for the Oculus Rift." *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014.
- ⁴⁷ Arthur, Kevin Wayne, and Frederick P. Brooks Jr. *Effects of field of view on performance with head-mounted displays*. Diss. University of North Carolina at Chapel Hill, 2000.
- ⁴⁸ Urey, Hakan, et al. "State of the art in stereoscopic and autostereoscopic displays." *Proceedings of the IEEE* 99.4 (2011): 540-555.
- ⁴⁹ Sexton, Ian, and Phil Surman. "Stereoscopic and autostereoscopic display systems." *IEEE Signal Processing Magazine* 16.3 (1999): 85-99.
- ⁵⁰ Grimes, Brad. "University of Illinois at Chicago: Virtual Reality's CAVE Pioneer." EdTech, 30 Jan. 2013, edtechmagazine.com/higher/article/2013/01/university-illinois-chicago-virtual-realitys-cave-pioneer.
- ⁵¹ Cruz-Neira, Carolina, et al. "The CAVE: audio visual experience automatic virtual environment." *Communications of the ACM* 35.6 (1992): 64-73.
- ⁵² "UIC / EVL." Evl | Electronic Visualization Laboratory, 18 June 2014, www.evl.uic.edu/entry.php?id=1165.
- ⁵³ Sharples, Sarah, et al. "Virtual reality induced symptoms and effects (VRISE): Comparison of head mounted display (HMD), desktop and projection display systems." *Displays* 29.2 (2008): 58-69.
- ⁵⁴ Earnshaw, Rae A., ed. *Virtual reality systems*. Academic press, 2014.
- ⁵⁵ Patterson, Robert, Marc D. Winterbottom, and Byron J. Pierce. "Perceptual issues in the use of head-mounted visual displays." *Human factors* 48.3 (2006): 555-573.
- ⁵⁶ Kok, Manon, Jeroen D. Hol, and Thomas B. Schön. "Using inertial sensors for position and orientation estimation." *arXiv preprint arXiv:1704.06053* (2017).
- ⁵⁷ Goradia, Ishan, Jheel Doshi, and Lakshmi Kurup. "A review paper on oculus rift & project morpheus." *International Journal of Current Engineering and Technology* 4.5 (2014): 3196-3200.

-
- ⁵⁸ Desai, Parth Rajesh, et al. "A review paper on oculus rift-a virtual reality headset." *arXiv preprint arXiv:1408.1173* (2014).
- ⁵⁹ Colaner, Seth. "What's Inside Microsoft's HoloLens And How It Works." Tom's Hardware, 23 Aug. 2016, www.tomshardware.com/news/microsoft-hololens-components-hpu-28nm,32546.html.
- ⁶⁰ Avila, Lisa, and Mike Bailey. "Augment your reality." *IEEE computer graphics and applications* 36.1 (2016): 6-7.
- ⁶¹ Kolasinski, Eugenia M. Simulator Sickness in Virtual Environments. No. ARI-TR-1027. ARMY RESEARCH INST FOR THE BEHAVIORAL AND SOCIAL SCIENCES ALEXANDRIA VA, 1995.
- ⁶² Goldberg, Jay M. The Vestibular System: a Sixth Sense. Oxford University Press, 2012.
- ⁶³ Olshannikova, Ekaterina, et al. "Visualizing Big Data with augmented and virtual reality: challenges and research agenda." *Journal of Big Data* 2.1 (2015): 22.
- ⁶⁴ "Best Virtual Reality SDK for VR Development in 2017." Thinkmobiles, 12 Feb. 2018, thinkmobiles.com/blog/best-vr-sdk/.
- ⁶⁵ Hahn, James F. "Virtual Reality Library Environments." American Library Association, 2017.
- ⁶⁶ Wilson, Nathan M. "Open Source Medical Software Corporation." *Cardiovascular and Pulmonary Model Repository*. Open Source Medical Software Corporation, 2013. Web. 06 Dec. 2016.
- ⁶⁷ Steinman, David A. "Image-based computational fluid dynamics modeling in realistic arterial geometries." *Annals of biomedical engineering* 30.4 (2002): 483-497.
- ⁶⁸ Seeram, Euclid. *Computed Tomography-E-Book: Physical Principles, Clinical Applications, and Quality Control*. Elsevier Health Sciences, 2015.
- ⁶⁹ Geraci, Salvatore, et al. "Optical Coherence Tomography for Coronary Imaging." *European Society of Cardiology*, vol. 9, 13 Dec. 2010.
- ⁷⁰ Bronzino, Joseph D., and Donald R. Peterson. *Biomedical Signals, Imaging, and Informatics*. Boca Raton: CRC/Taylor & Francis Group, 2015. Print.

-
- ⁷¹ Torii, Ryo, et al. "Computer modeling of cardiovascular fluid–structure interactions with the deforming-spatial-domain/stabilized space–time formulation." *Computer Methods in Applied Mechanics and Engineering* 195.13-16 (2006): 1885-1895.
- ⁷² LaDisa, John F., et al. "Computational simulations for aortic coarctation: representative results from a sampling of patients." *Journal of biomechanical engineering* 133.9 (2011): 091008.
- ⁷³ Moon, Ji Young, et al. "Considerations of blood properties, outlet boundary conditions and energy loss approaches in computational fluid dynamics modeling." *Neurointervention* 9.1 (2014): 1-8.
- ⁷⁴ "SimVascular." SimVascular, 2017, simvascular.github.io/.
- ⁷⁵ Les, Andrea S., et al. "Quantification of hemodynamics in abdominal aortic aneurysms during rest and exercise using magnetic resonance imaging and computational fluid dynamics." *Annals of biomedical engineering* 38.4 (2010): 1288-1313.
- ⁷⁶ Menon, Arjun, et al. "Altered hemodynamics, endothelial function, and protein expression occur with aortic coarctation and persist after repair." *American Journal of Physiology-Heart and Circulatory Physiology* 303.11 (2012): H1304-H1318.
- ⁷⁷ Markl, Michael, et al. "In vivo wall shear stress distribution in the carotid artery effect of bifurcation geometry, internal carotid artery stenosis, and recanalization therapy." *Circulation: Cardiovascular Imaging* 3.6 (2010): 647-655.
- ⁷⁸ Westerhof, N., Nikos Stergiopoulos, and Mark I. M. Noble. *Snapshots of Hemodynamics: An Aid for Clinical Research and Graduate Education*. New York, NY: Springer, 2005. Print.
- ⁷⁹ Milgram, Paul, et al. "Augmented reality: A class of displays on the reality-virtuality continuum." *Telemanipulator and telepresence technologies*. Vol. 2351. International Society for Optics and Photonics, 1995.
- ⁸⁰ J.F. LaDisa, Jr., L.E. Olson, K.M. Ropella, R.C. Molthen, S.T. Haworth, J.R. Kersten, D.C. Warltier, P.S. Pagel. Microfocal x-ray computed tomography post-processing operations for optimizing reconstruction volumes of stented arteries during 3D computational fluid dynamics modeling. *Comput Methods Programs Biomed.* 2005 Aug; 79(2):121-34.
- ⁸¹ Menon, Arjun, et al. "A coupled experimental and computational approach to quantify deleterious hemodynamics, vascular alterations, and mechanisms of long-term morbidity

in response to aortic coarctation." *Journal of pharmacological and toxicological methods* 65.1 (2012): 18-28.

⁸² Rayz, V. L., et al. "Computational modeling of flow-altering surgeries in basilar aneurysms." *Annals of biomedical engineering* 43.5 (2015): 1210-1222.

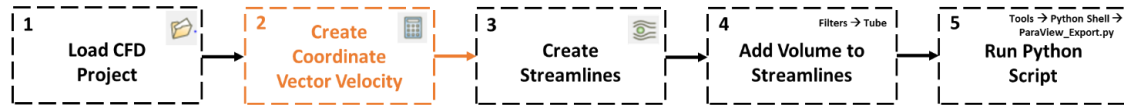
⁸³ Lin, Emily L., et al. "Relationship between degree of obstruction and airflow limitation in subglottic stenosis." *The Laryngoscope* (2017).

⁸⁴ Liu, Jing, et al. "Highly accelerated intracranial 4D flow MRI: evaluation of healthy volunteers and patients with intracranial aneurysms." *Magnetic Resonance Materials in Physics, Biology and Medicine* (2017): 1-13.



⁸⁵ Vanhille, Derek L., et al. "Virtual Surgery for the Nasal Airway: A Preliminary Report on Decision Support and Technology Acceptance." *JAMA facial plastic surgery* 20.1 (2018): 63-69.

Appendix A: Convert CFD Results into Streamlines Using ParaView

Below, are the specific commands to create velocity streamlines. Icons are identified with **bolded** font, keyboard commands are underlined., and keyboard entries are *italicized step number* (i.e. (1)).



1. Load CFD Project

- a. Open ParaView
- b. If your CFD simulation has multiple cardiac cycles create a new folder that only includes the last cardiac cycle
 - i. For example: If you have 3500 Time Steps going over 7 cardiac cycles (500 time steps per cycle) with the number of time steps between restarts as 20 you would select all of the files that are 3020 and greater into a separate folder
- c. Load the CFD results created in step 1.b (**.vtu, or .vtp**) or a **.case** files into ParaView by selecting **File → Open**.
 - i. A new window should pop up, select the CFD simulation and Press **OK**
- d. In the properties window press **Apply** (1) 
 - i. The model should now appear in the Layout Window
- e. Select the Coloring as whatever texture gradient the wall mesh should be (Example: Pressure, wss, or Solid Color) (Figure 43 a)
 - i. Select **Choose preset icon** () A new window should pop up, in Preset select jet and press **Apply** then press **Close** (Figure 43 b).

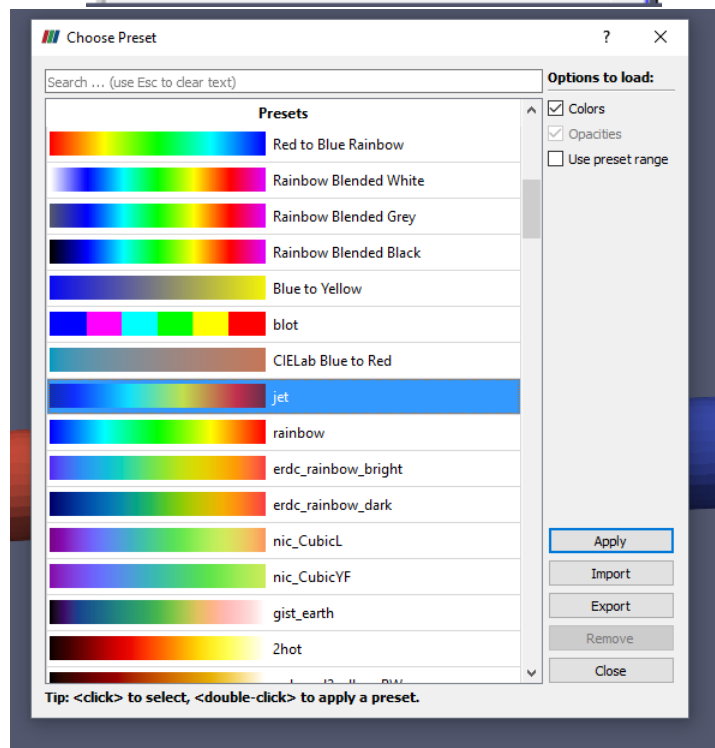
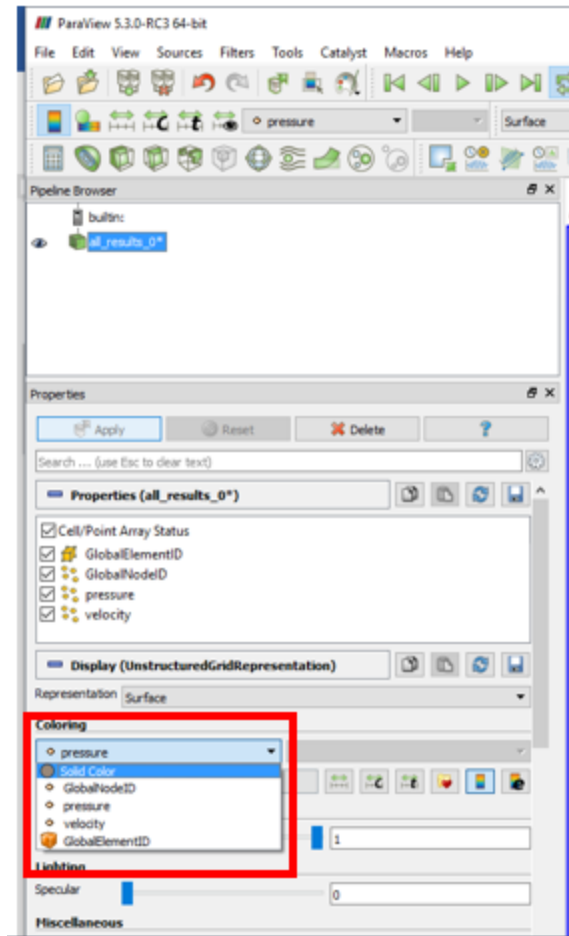



Figure 43: a) select the wall value you would like to show for each time step. b) Choose Present jet scale that should be used for all of your scaled

- ii. If you are going to select **pressure** for each time step the scale needs to be set/adjusted
 1. Select the **Rescale to data range over all timesteps** () in the Color Map Editor (Figure 44 a)
 - a. A new Window should pop up where then the user should then select **Rescale and disable automatic rescaling** (Figure 44 b) or in an older version of ParaView click **Yes**

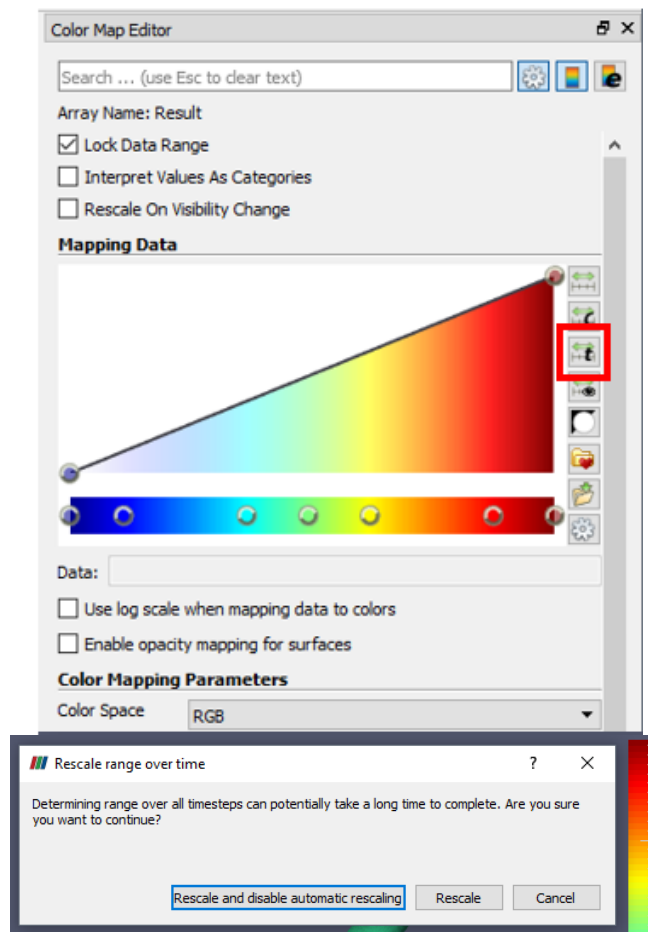



Figure 44: a) Color Map Editor b) setting range of scale

2. Create Coordinate Vector Velocity- Not always necessary
 - a. If Velocity is not given but, $x_velocity$, $y_velocity$, and $z_velocity$ where (Figure 45 a), scalar velocities must be converted to a coordinate vector velocity.

- b. Click the **calculator icon** () (Figure 45 b); type in $i\hat{H}at*velocity_X + j\hat{H}at*velocity_Y + k\hat{H}at*velocity_Z$.
- Note, if user's scalar velocity information was $x_Velocity$, $y_Velocity$, $z_Velocity$ the user would type in:
 $i\hat{H}at*x_Velocity + j\hat{H}at*y_Velocity + k\hat{H}at*z_Velocity$.
- c. In the **Result Array Name**, type in Velocity.2 and press **Apply** (2).
- An accurate representation of coordinate vector velocity is now depicted in the CFD model.
- d. If Velocity is already provided, the previous step should be skipped.

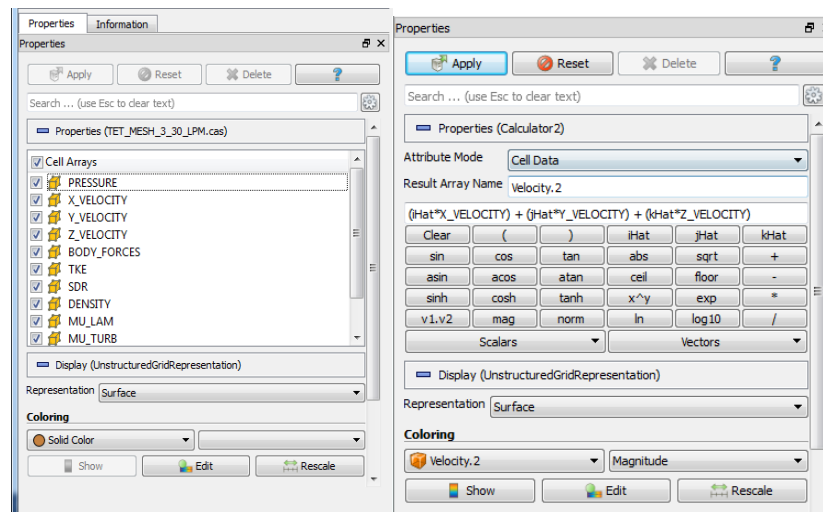



Figure 45: a) Example (without velocity) of CFD results after loading the CFD file into.
b) Calculator icons interface with the velocity equation and the Results Array Name.

3. Create Streamlines

- To create streamlines representing flow and velocity profile, click on the **streamline tracer icon** () and in the streamline tracer interface (Figure 46 a).
 - Select **Vectors** as Velocity and **Integration Direction** as FORWARD. Select **Seed Type** as Point Source. Select the desired **Number of Points** (streamlines).
 - In Figure 46 a, the streamlines are selected at 100, which is a low number of streamlines. Before pressing apply, there is a white wireframe tube with an asterisk in the middle. Move this asterisk into the center part of the models inflow (Figure 46 b), and set the radius of the wireframe circle (in **Sphere Parameters – Radius**) to roughly the radius of the inlet geometry then press **Apply** (3).

- ii. Select Solid Color as **Coloring** (Figure 46 a)

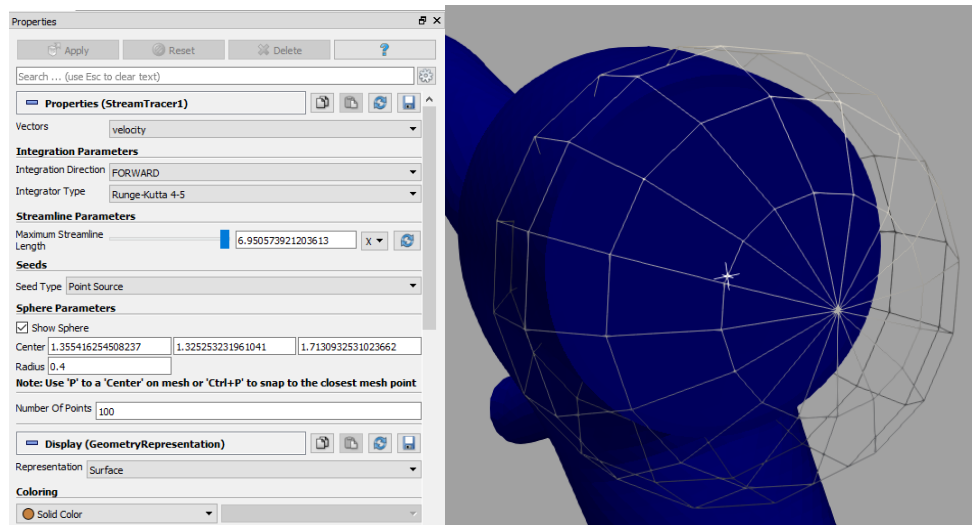


Figure 46: a) Streamline tracer icons interface with the Vector, Integration Direction, Radius and Number of Points selected as Velocity, FORWARD, 0.4, and 100. b) Wireframe circle roughly matches the radius of the inlet and the asterisk in the center part of the CFD simulations inflow.

4. Add Volume to Streamline

- a. To add volume to the streamlines, select **Filters** → **Alphabetical** → **Tube** on the top toolbar.
- b. Adjust the **Radius** of the desired tubed-streamlines, for the **coloring**, select Velocity (see Figure 47). Then press **Apply** (4).
 - i. This step creates the 3D geometry needed for VR and creates the vertex color information indicative of velocity magnitude.

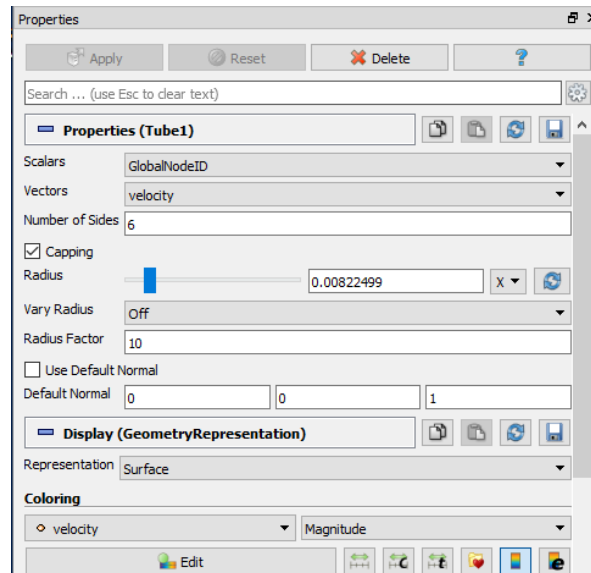




Figure 47: Tube interface with Radius set as $8.6297\text{e-}05$ and the Coloring selected as Velocity.

- c. Select **Choose preset icon** (). A new window should pop up, in Preset select jet and press **Apply** then press **Close** (Figure 43 b).
- d. Select the **Rescale to data range over all timesteps** () in the Color Map Editor (Figure 44 a)
 - i. A new Window should pop up where then the user should then select **Rescale and disable automatic rescaling** (Figure 44 b) or in an older version of ParaView click **Yes**
- e. Record the min and max Velocity information for later use in Unity

5. Run Python Script

- a. To run the python script, create a new folder on the desktop and name it "ParaView_vrml".
- b. Locate the ParaView_export.py python script
- c. Open up the ParaView_export.py file
 - i. On line 8, change the number to the end time step of the simulation.
 - ii. On line 20, change the location of the ParaView_export.py file
 1. For example, the line of code is originally this:
 - a. `fname = "C:/users/9376vennj/Desktop/ParaView_vrml/foo_res" + format(x) + "_velocity.vrml"`
 - b. Only change the highlighted part
- d. Save the ParaView_export.py file.

- e. In ParaView, select **Tools → Python Shell** on the top toolbar.
- f. In the Python Shell window, select **Run Script** (5).
 - i. A new window should pop up, locate the ParaView_export.py file and select run **OK**. (Figure 48)
 - ii. In the main ParaView window, the CFD model should be going through each desired time step.

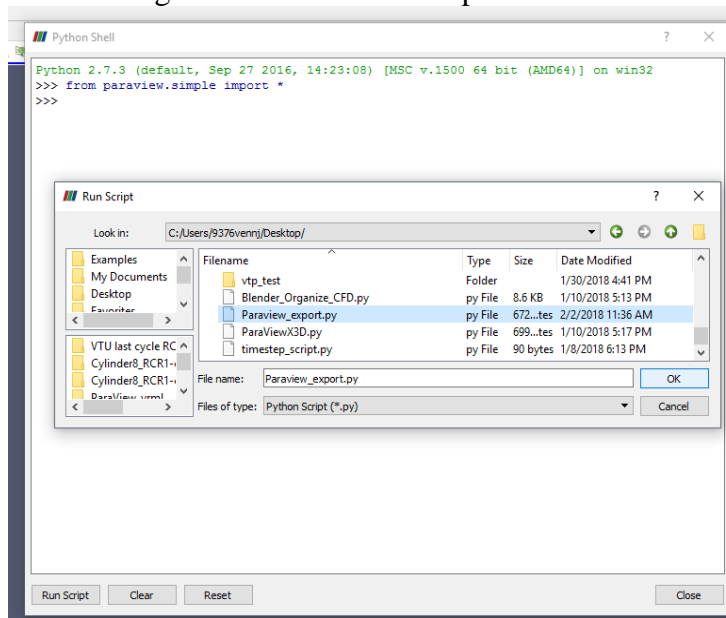


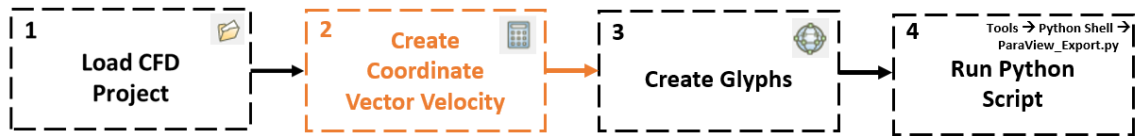
Figure 48: Python Shell window

- g. After the Python script is complete, go to the ParaView_vrml file, in the file there should be the desired number of time steps in .vrml form.



This completes the steps to convert CFD results into 3D format if the user would like to depict his or her CFD results using streamlines.

Appendix B: Convert CFD Results into Glyphs Using ParaView

Below, are the specific commands to create velocity glyphs. Icons are identified with **bolded** font, keyboard commands are underlined., and keyboard entries are *italicized step number* (i.e. (1)).



1. Load CFD Project

- a. Open ParaView
- b. If your CFD simulation has multiple cardiac cycles create a new folder that only includes the last cardiac cycle
 - i. For example: If you have 3500 Time Steps going over 7 cardiac cycles (500 time steps per cycle) with the number of time steps between restarts as 20 you would select all of the files that were 3020 and greater and put them in a separate folder
- c. Load the CFD results created in step 1.b (.vtu, or .vtp) or a .case files into ParaView by selecting **File → Open**.
 - i. A new window should pop up, select the CFD simulation and Press **OK**
- d. In the properties window press **Apply** (1) 
 - i. The model should now appear in the Layout Window
- e. Select the Coloring as whatever texture gradient the wall mesh should be (Example: Pressure, wss, or Solid Color) (Figure 49 a)
 - i. Select **Choose preset icon** () A new window should pop up, in Preset select jet and press **Apply** then press **Close** (Figure 49 b).

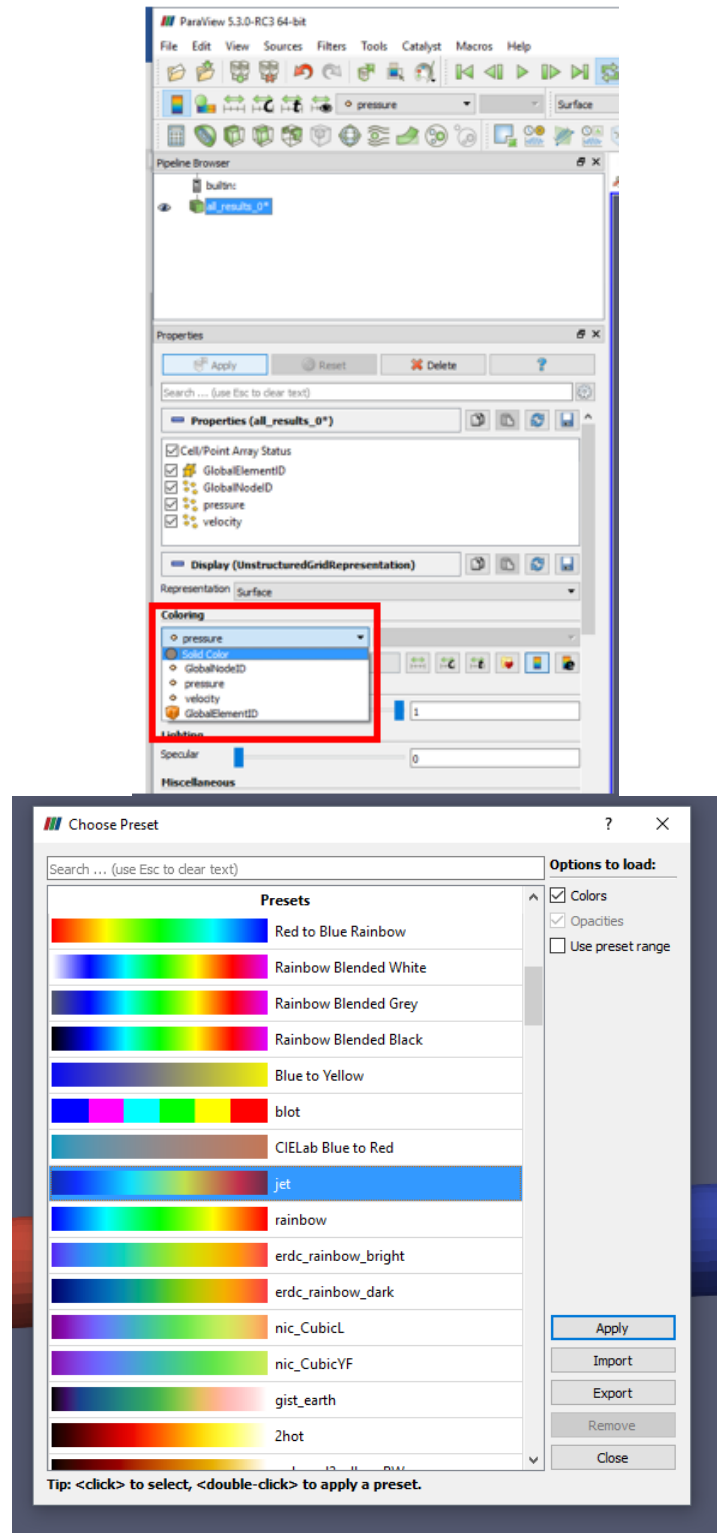



Figure 49 : a) Select the wall value you would like to show for each time step. b) Choose Present jet scale that should be used for all of your scaled

- ii. If you are going to select **pressure** for each time step the scale needs to be set/adjusted

1. Select the **Rescale to data range over all timesteps** () in the Color Map Editor (Figure 50 a)
 - a. A new Window should pop up where then the user should then select **Rescale and disable automatic rescaling** (Figure 50 b) or in an older version of ParaView click **Yes**

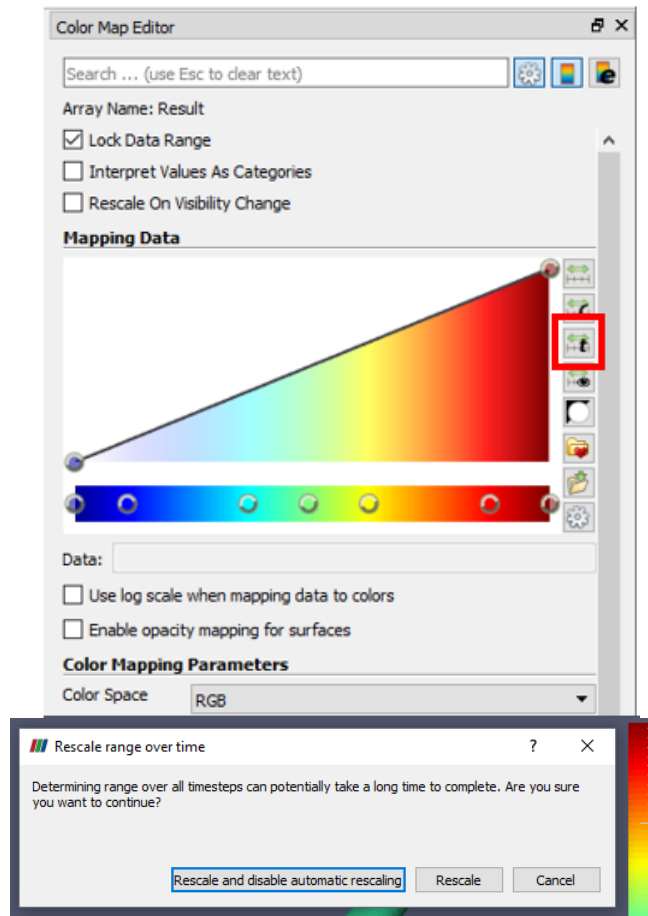



Figure 50: a) Color Map Editor b) setting range of scale

2. Create Coordinate Vector Velocity- Not always necessary
 - a. If Velocity is not given but, x_velocity, y_velocity, and z_velocity where (Figure 51 a), scalar velocities must be converted to a coordinate vector velocity.
 - b. Click the **calculator icon** () (Figure 51 b); type in iHat*velocity X+jHat*velocity Y+kHat*velocity Z.

- i. Note, if user's scalar velocity information was $x_Velocity$, $y_Velocity$, $z_Velocity$ the user would type in:
 $i\hat{H}at*x_Velocity + j\hat{H}at*y_Velocity + k\hat{H}at*z_Velocity$.
- c. In the **Result Array Name**, type in Velocity.2 and press **Apply** (2).
 - ii. An accurate representation of coordinate vector velocity is now depicted in the CFD model.
- d. If Velocity is already provided, the previous step should be skipped.

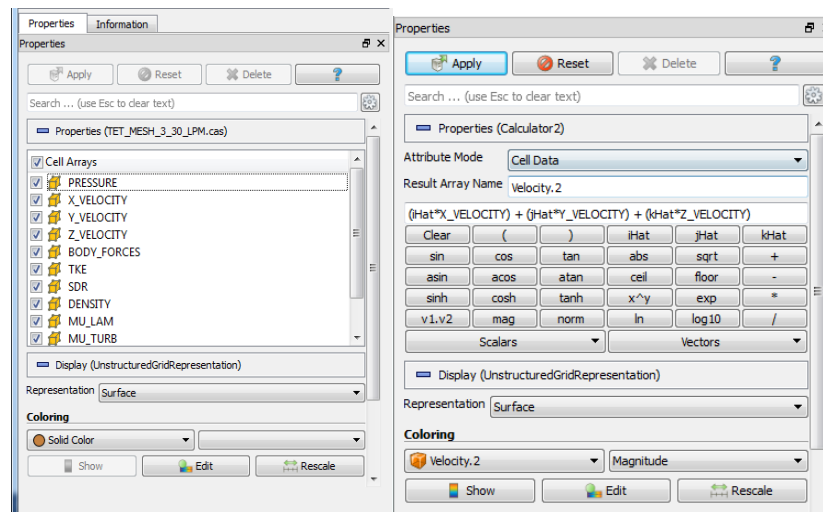



Figure 51: a) Example (without velocity) of CFD results after loading the CFD file into. b) Calculator icons interface with the velocity equation and the Results Array Name.

3. Create Glyphs

- a. To create glyphs representing velocity profile, click on the **glyph icon** () and in the glyph interface (*Figure 52 a*).
 - i. Select **Scalars** as None and **Vectors** as Velocity. Select **Scale Mode** as Vector. Select the desired **Number of Points** (streamlines).
 - ii. Select the time step that is peak systole for your simulation (*Figure 52 b*)
 1. For example, the simulation used in *Figure 52 b* peak systole was at the 9th time step
 - iii. In *Figure 52 a*, the Scale Factor (size of glyph) and Maximum Number of Sampling points is selected as 0.005 and 5000, this ratio needs to be adjusted depending on the model **Apply** (3).
 - iv. Select **Coloring** as GlyphVector

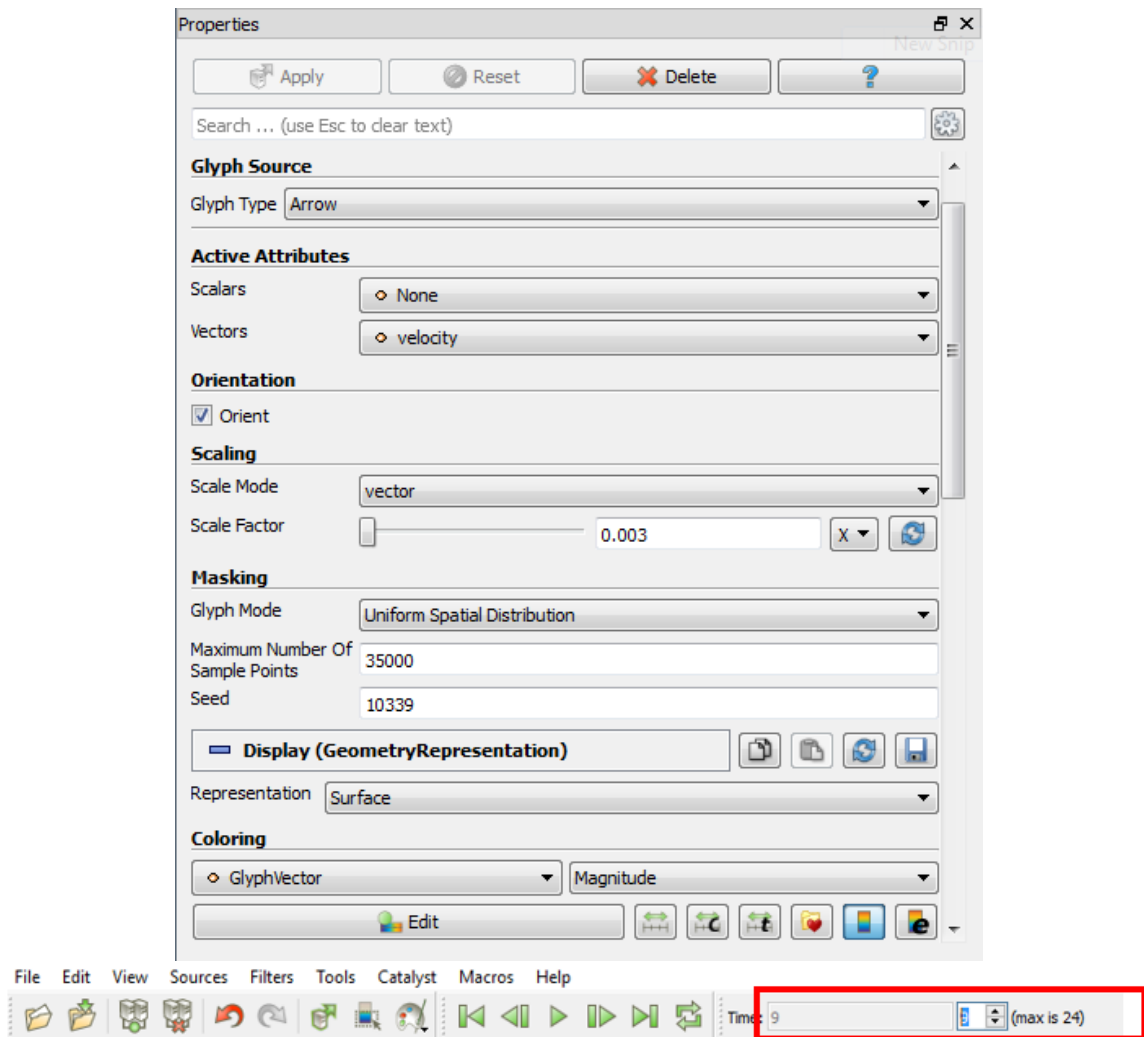




Figure 52: a) Making Glyphs Note that the scalars are set to None, Vectors is set to Velocity, Scale mode is set to Vector, the Scale Factor is very small and Coloring is set to velocity. B) selecting the time step that is peak systole

- v. Select **Choose preset icon** (). A new window should pop up, in Preset select **jet** and press **Apply** then press **Close** (Figure 49 b).
- vi. Select the **Rescale to data range over all timesteps** () in the Color Map Editor (Figure 50 a)
- vii. A new Window should pop up where then the user should then select **Rescale and disable automatic rescaling** (Figure 50 b) or in an older version of ParaView click **Yes**
- viii. Record the min and max Velocity information for later use in Unity

4. Run Python Script

- a. To run the python script, create a new folder on the desktop and name it “ParaView_vrml”.
- b. Locate the ParaView_export.py python script
- c. Open up the ParaView_export.py file
 - i. On line 8, change the number to the end time step of the simulation.
 - ii. On line 20, change the location of the ParaView_export.py file
 1. For example, the line of code is originally this:
 - a. `fname =`
`"C:/users/9376vennj/Desktop/ParaView_vrml/foo_r`
`es" + format(x) + "_velocity.vrml"`
 - b. Only change the highlighted part
- d. Save the ParaView_export.py file.
- e. In ParaView, select **Tools → Python Shell** on the top toolbar.
- f. In the Python Shell window, select **Run Script (5)**.
 - i. A new window should pop up, locate the ParaView_export.py file and select run **OK**. (Figure 53)
 - ii. In the main ParaView window, the CFD model should be going through each desired time step.

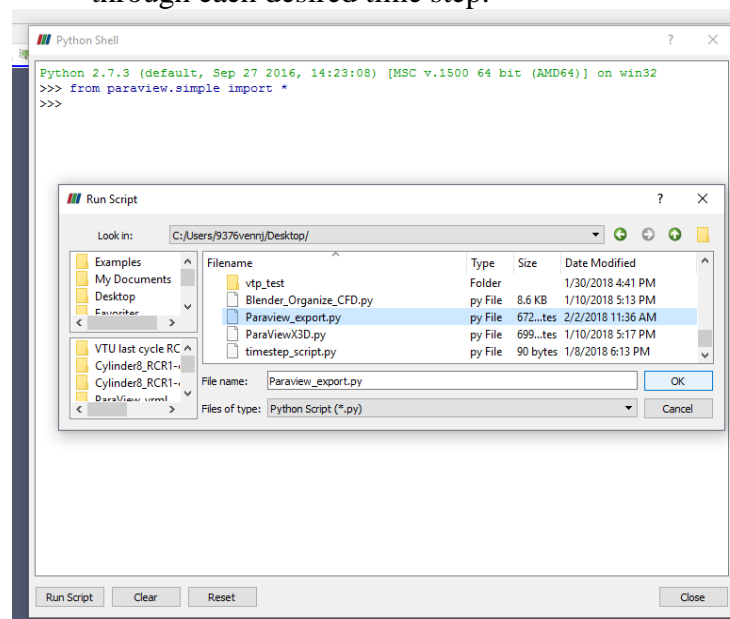


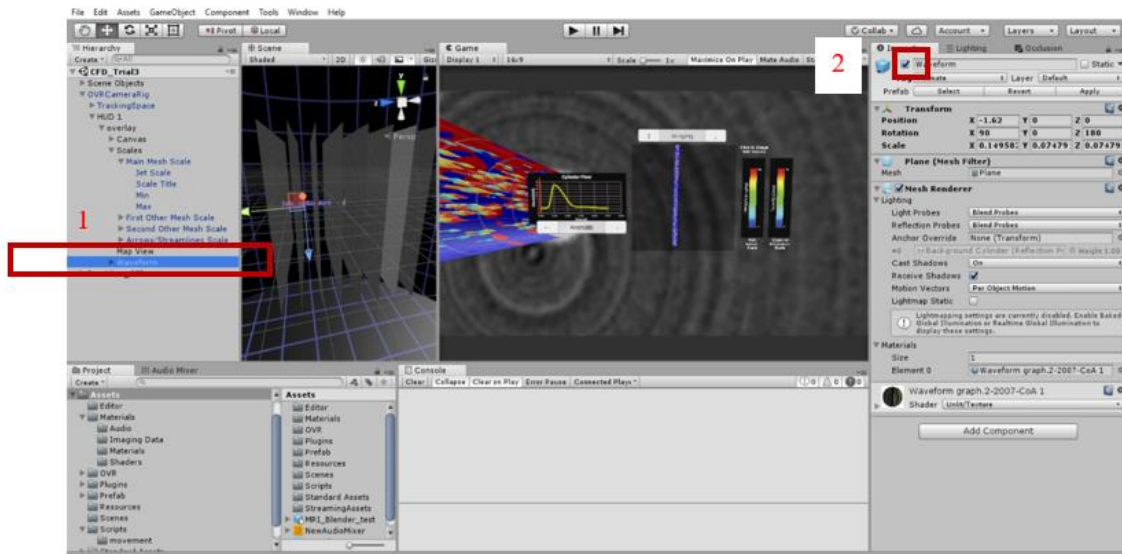
Figure 53: Python Shell window

- g. After the Python script is complete, go to the ParaView_vrml file, in the file there should be the desired number of time steps in .vrml form.

Appendix C: Create/Turn Off Flow Waveform in Unity

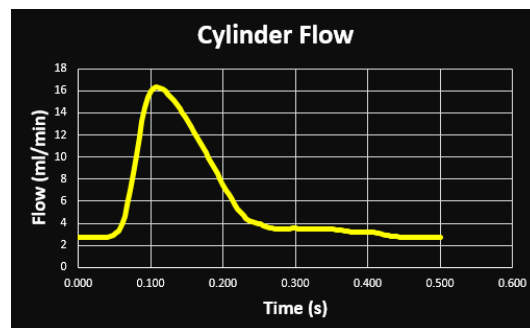
Turn off flow waveform

1. Select the Waveform game object under OVRCameraRig/HUD 1/Overlay/Waveform in the Hierarchy
2. In the Waveform Inspector, uncheck the box next to the right of the blue cube and above the word "Tag"

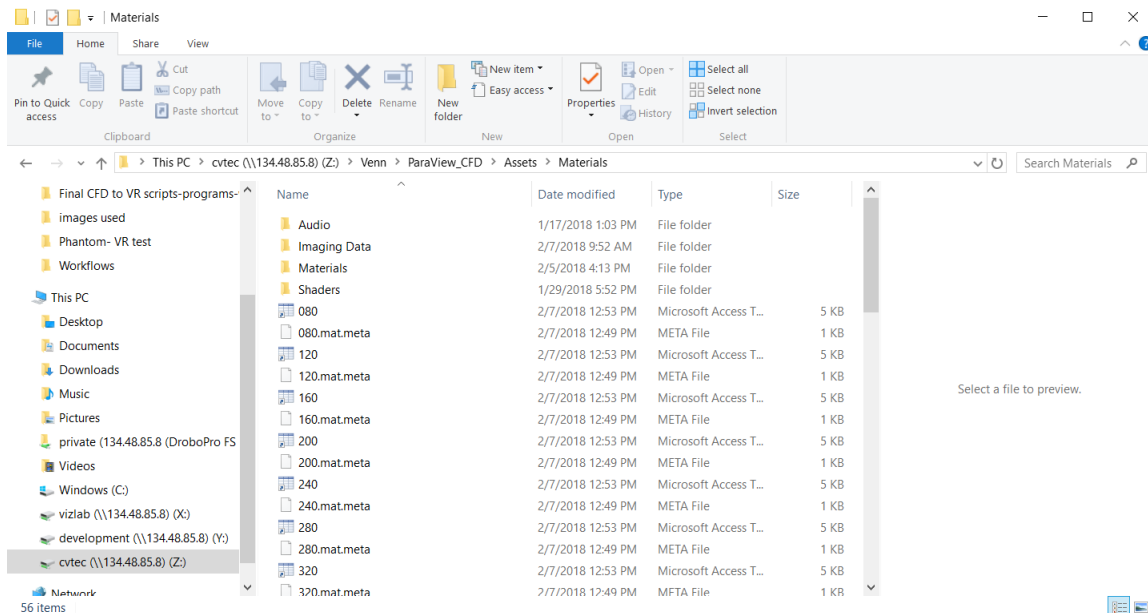


Animate flow waveform

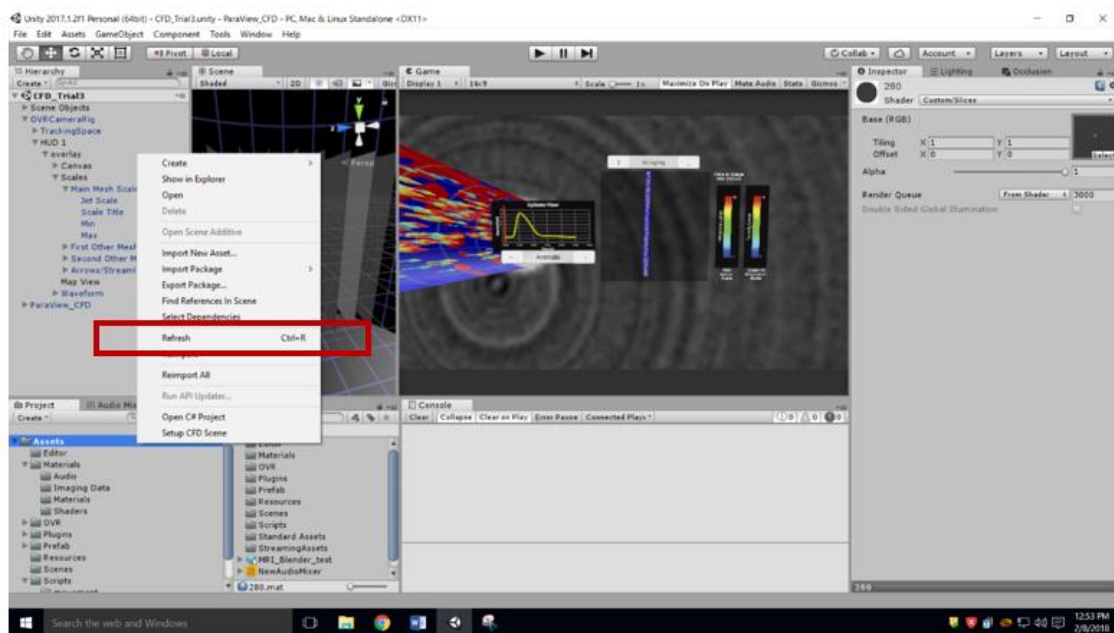
1. Create a flow waveform for your CFD simulation
2. Save the waveform as a PNG image
 - a. Example



3. Using File Explorer () Paste the PNG image in the Unity Template Materials folder

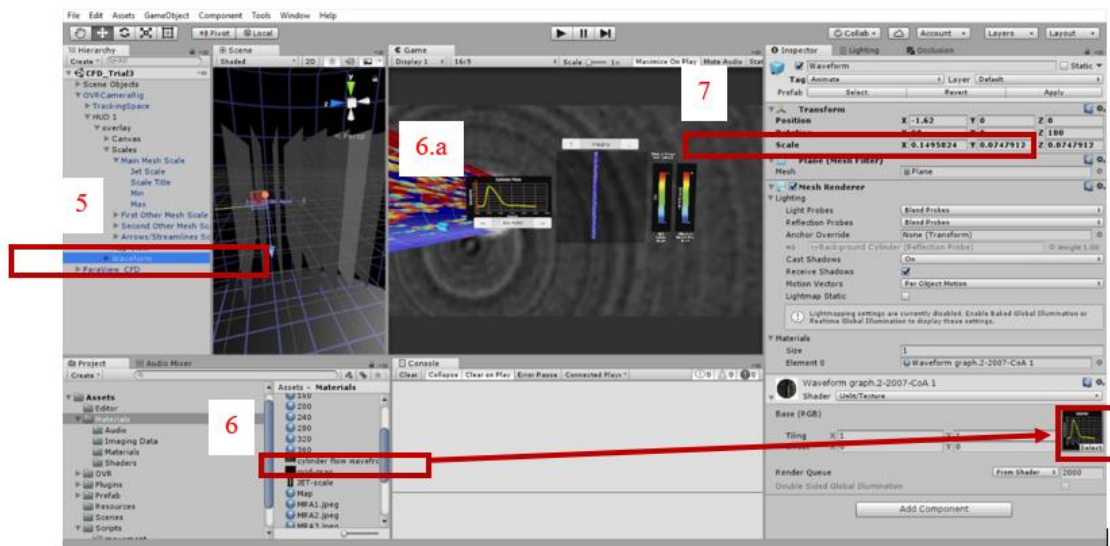


4. Refresh the Unity Scene by right clicking on the Assets folder OR click “Ctrl + R”

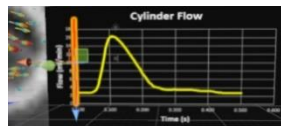


5. Select on the Waveform game object in the Hierarchy
6. In the Materials folder, drag and drop the waveform PNG image into the shader texture box
 - a. Your flow waveform should now appear in the Unity scene

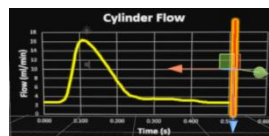
7. The scale of the waveform might need to be adjusted- to do this adjust Transform Scale X and Z

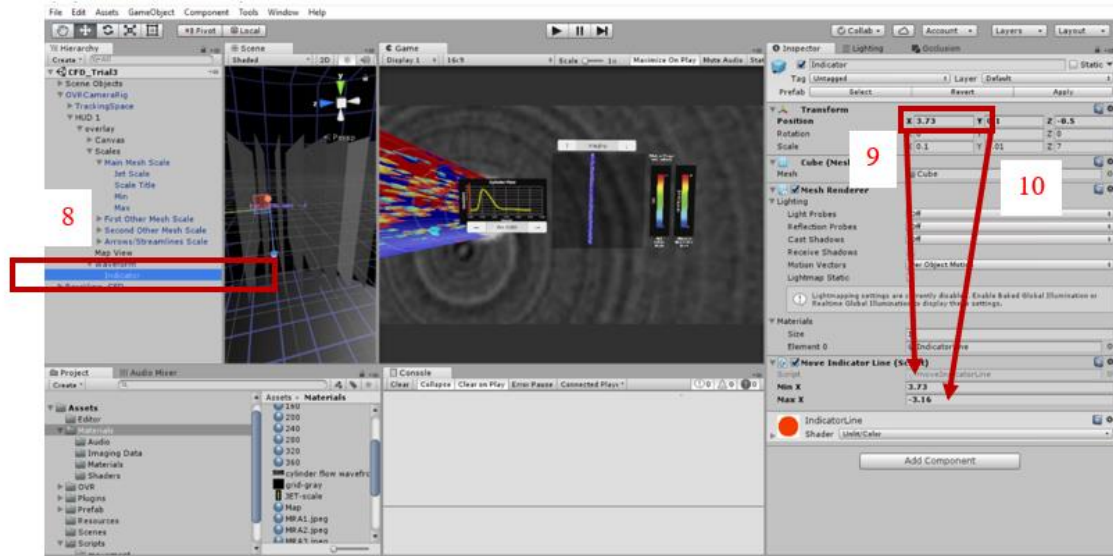


8. Select the Indicator object underneath the Waveform in the Hierarchy
 - a. The line on your waveform should appear orange in the Unity scene
9. Adjust the Transform, Position X of the Indicator to align with the start of the flow waveform
 - a. Copy the Position X
 - b. Past the Position X value to the Move Indicator Script Min X slot



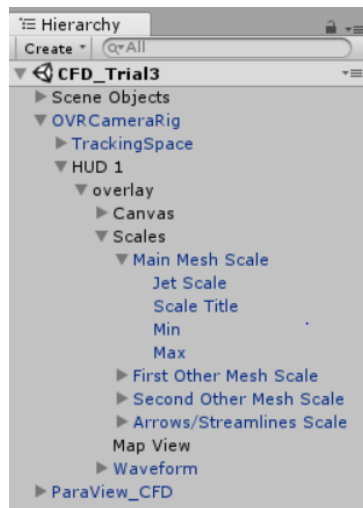
10. Adjust the Transform, Position X of the Indicator to align with the end of the flow waveform
 - a. Copy the Position X
 - b. Past the Position X value to the Move Indicator Script Max X slot





Appendix D: Label Scales in Unity

1. These steps are done for each of your Scales (Main Mesh Scale, First Other Mesh Scale, Second Other Mesh Scale, and or Arrows/Streamlines Scale)
 - a. If you do not have “Other Mesh” disregard the “First Other Mesh Scale” and “Second Other Mesh Scale”



2. Click on your Scale under OVRCameraRig/HUD 1/Scales/WHATEVER SCALE
3. Click on “Scale Title”
 - a. In the Transform, in Text (Script) type in the Title of your scale with appropriate units
4. Click on “Min”
 - a. In the Transform, in Text (Script) type in the minimum number of your scale
5. Click on “Max”
 - a. In the Transform, in Text (Script) type in the maximum number of your scale

