Journal: **PRESENCE**

PERSPECTIVE

Center for Cyber-Physical Systems: Immersive Visualization and Simulation Environment

Thomas Wischgoll

Thomas Wischgoll[1]

[1]Wright State University, 3640. Col. Glenn Hwy., Dayton, OH 45435, thomas.wischgoll@wright.edu

ABSTRACT

Immersive display systems in the form of head-mounted displays or full-size, walkable display systems can provide a very intuitive environment for a multitude of applications. Applications include exploration, simulation for training, or experimental studies to learn about people's behavior. Some display systems utilize large-scale and high-resolution configurations which can be very effective in data exploration and visualization. This paper describes the infrastructure available at Wright State University with its advantages and disadvantages and discusses some of its use cases as well as its setup and administration. Different software frameworks are discussed that are built upon as is or their capabilities extended to provide additional features, such as touch support. Insight is provided into managing and administering a virtual reality laboratory effectively and efficiently by minimizing the effort to keep the infrastructure operational.

AUTHOR SUMMARY

Thomas Wischgoll is currently a full professor and NCR Endowed Chair at Wright State University. His research interests include flow and scientific visualization, virtual environments and display technologies, as well as biomedical imaging and visualization. Dr. Wischgoll devised different algorithms for analyzing and visualizing different flow data sets, medical data, including CT and MRI, and other types of data sets.

He utilized various display systems for virtual reality applications, ranging from head-mounted displays to full-scale walkable immersive systems, and applied these display systems to different virtual and augmented reality applications, including highly immersive experiments involving human subjects for a better understanding of human behavior. His research work in the fields of scientific visualization and data analysis resulted in more than ninety peer-reviewed publications, including IEEE and ACM.

Introduction

The visualization and simulation infrastructure at Wright State University comprises the Appenzeller Visualization Laboratory and the Immersive Visualization and Animation Theater. These two laboratories serve the common goal of making a variety of display systems available to the university and its constituents expanding on the approach suggested by OLeary, Sherman, Shetty, Clark, and Hulme (2013). The Appenzeller Visualization Laboratory is more focused on the research side with some teaching components whereas the Immersive Visualization and Animation Theater provides students with 24/7 access to fully immersive display capabilities.

Right from the inception of this infrastructure, it was important to provide access to a variety of diverse display systems as different applications require different features. This diverse configuration also allows for direct comparison of different display environments to identify the most suitable one for a specific application or to interconnect display systems for a collaborative environment (Koehler, Berger, Rajashekar, Wischgoll, & Su, 2019).

Virtual and augmented reality has a long history. One of the earliest systems was Headsight which combined a screen with a tracking system. This system was used to train military personnel by, for example, simulating flying in complete darkness. Shortly after, Sutherland demonstrated the Ultimate Display (Sutherland, 1965) to mimic the physical world using a tethered head-mounted display configuration. Technical advances enabled tighter integration by combining cell phone displays and accelerometer-based tracking into a single device, such as the first-generation Oculus Rift. Higher resolution and frame rates in current-generation devices, such as the HTC Vive, improve the user experience even further. Anthes, García-Hernández, Wiedemann, and Kranzlmüller (2016) provide an overview of some of the head-mounted VR technology and options for tracking and input devices.

Instead of using head-mounted devices, screen-based systems can also be used to achieve immersive environments. The original CAVE system developed by Cruz-Neira, Sandin, DeFanti, Kenyon, and Hart (1992) is one of the early examples in which projectors displayed content on multiple walls surrounding the user. Active shutter glasses and a tracking system to determine the user's position then enable the immersive experience. More current versions of this design have used LCD displays instead of projectors to cut down on the additional footprint needed and increase the overall resolution the system is capable of displaying, such as the CAVE2 by Febretti et al. (2013). Dincelli et al. provide an overview of some of the VR technologies and their use cases (Dincelli & Yayla, 2022).

Industrial use of VR is growing with more and more companies using VR for testing and exploration (Berg & Vance, 2017). Some stores allow customers to explore designs before purchasing, such as kitchens. Others use VR for training their employees, such as Walmart, or to provide a safe training environment for using various tools or other industrial equipment (Carruth, 2017). Virtual reality techniques can also be used to increase empathy in medical applications as illustrated by Roswell et al. (2020) and Patel, Pei, Vasoya, and Hershberger (2023). Some studies report an increased retention rate of the learned knowledge when using virtual reality (Smith et al., 2016) or augmented reality technology (Menon, Holland, Farra, Wischgoll, & Stuber, 2022).

With respect to visualization software, there tends to be a gap between the visualization software developed in a research domain compared to the needs from an application perspective. As an example, Gillmann et al. (2021) discuss ten open challenges in the medical domain. That paper was the result of one of the Application Spotlights held at the IEEE VIS conference to discuss issues that visualization laboratories often face when it comes to getting their visualization techniques used more broadly within the application domain. The workshop series *VisGap* (*VisGap*, 2023) strives to close this gap within the visualization domain as well. Running a virtual reality laboratory within a research environment poses very similar problems as outlined in these Application Spotlights and the VisGap workshops in that typically there is limited or no funding for dedicated personnel to maintain the infrastructure or develop the necessary software.

This paper outlines the various display systems and environments available at Wright State University and discusses both the hardware and the software aspects of administering and supporting these

Figure 1. Students using an HTC Vive Eye head-mounted display with their custom-developed software also showing on the screen in the back.

environments in the following sections. It provides insight into managing and administering a virtual reality laboratory by minimizing the effort required to keep all the systems operational.

Hardware Environments

To support the various activities in our laboratories, a diverse variety of display systems are installed. This ranges from desktop systems and head-mounted displays to large wall displays and walkable CAVE-type systems. The different displays are housed in the Appenzeller Visualization Laboratory which is geared more toward research and the Immersive Visualization and Animation Theater which is used more for teaching purposes. The rooms have square footages of around twelve hundred and eight hundred square feet, respectively.

Head-mounted displays (HMDs) provide a cost-effective way to provide immersive display technologies to students and researchers. In our laboratories, the HTC Vive Eye HMDs and HP mixed reality devices are available. Figure 1 shows students exploring their custom-developed software using one of the head-mounted displays. We also utilize the Magic Leap One augmented reality devices. These stand-alone devices, similar to the Microsoft Hololens, provide an overlay image on top of the real world suitable for augmented reality applications. We have successfully used these devices for nursing education in which

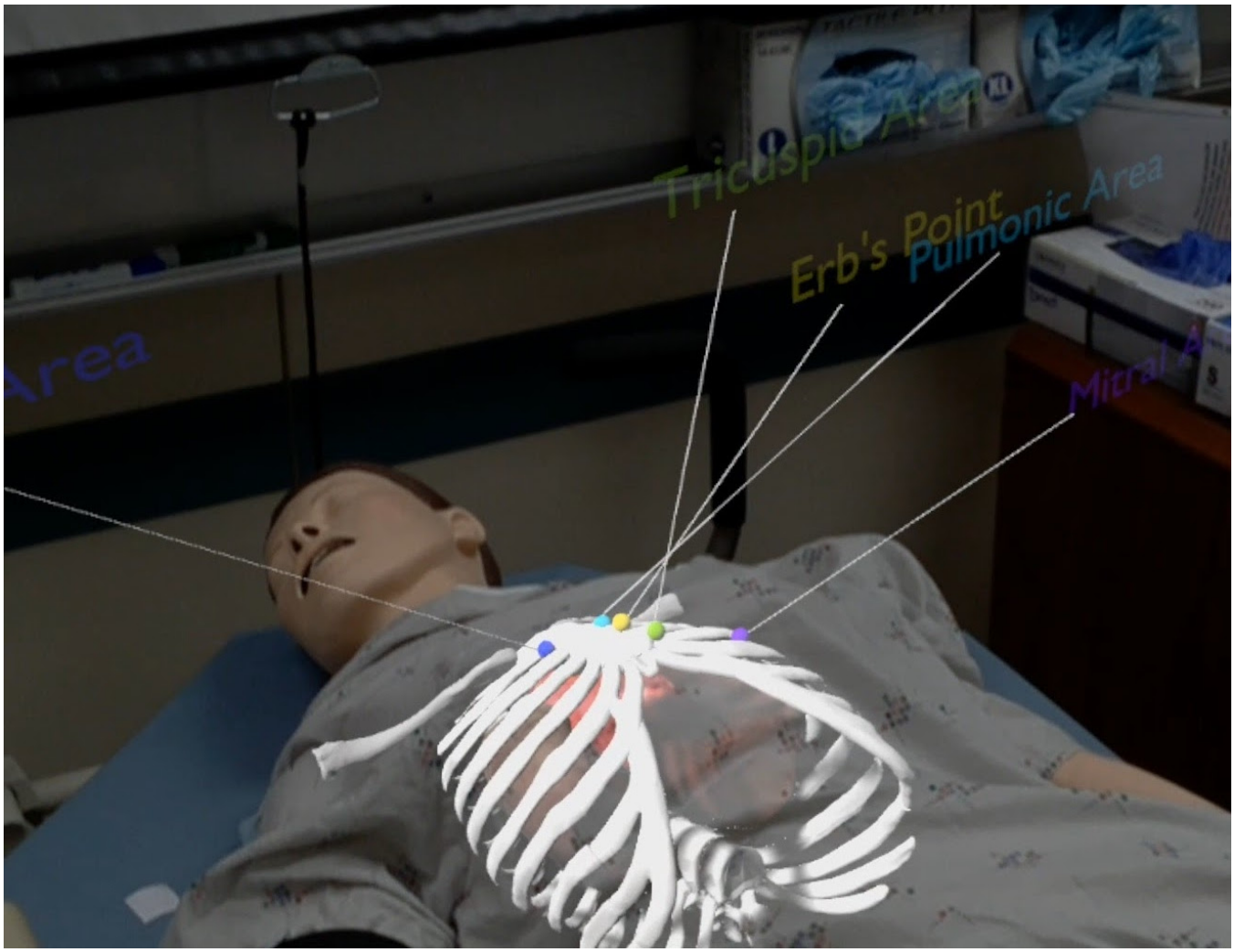

Figure 2. Image captured through the Magic Leap One AR device depicting the traditional training environment for nursing students overlaid with additional information, such as rib cage, fully animated organs, and markers for auscultation sites.

4 **Figure 3.** Teaching lab with access to AR/VR technology including head-mounted displays and graphics workstations.

overlay images of fully animated organs and other internal structures are displayed on the traditional manekins (Menon et al., 2022; Menon, Wischgoll, Farra, & Holland, 2021). Figure 2 shows an example of the training application for nursing students supplementing the traditional training environment by displaying the rib cage and fully animated organs, such as the heart and lungs, through the AR device. Another application was for assisting surgeons to visualize fractured ribs through the skin based on a CT scan during corrective surgery (Menon, Wischgoll, Farra, & Holland, 2020; Sensing, Parikh, Hardman, Wischgoll, & Menon, 2021). This allowed the surgeons to reduce the size of the incision to lessen the impact to the patient.

The teaching lab pictured in 3 provides access to 10 head-mounted VR displays and another 15 AR devices to provide the students with a sufficient number of devices. The HTC Vive devices share one pair of lighthouse tracking devices whereas the HP mixed reality devices use inside-out tracking and thus do not need any external sensors. Since the VR head-mounted displays by design remove the user from any input from the real world as much as possible, the walkable space is marked within the virtual environment to keep users from accidentally walking into furniture and other obstacles.

Different projector-based display systems are available, such as a Barco CADWall and a mobile projection screen with support for passive stereo. These are large-screen displays but do not provide very high resolutions. The projection screen is full HD whereas the Barco CADWall uses two projectors with 300 pixels of overlap to seamlessly blend between the images from these two projectors and thus has an overall resolution of 2500 by 1050 pixels. For applications that require higher-resolution display

Figure 4. High-resolution tiled display system satellite imagery. The high resolution of the display can reproduce the fine details of the imagery when exploring or discussing this type of data.

environments, tiled configurations are a common way of supporting those applications. One of our configurations uses a 2-by-3 setup comprising Sony's 50-inch 4K TVs (Wischgoll, 2017) depicted in figure 4. The entire system is driven by a single computer with two graphics cards to allow easy deployment of standard software packages. To provide intuitive input modalities, this system uses a Logitech touchpad T650 to enable smartphone-style interaction metaphors and a wireless gyro-based mouse and keyboard combination built into one device.

An in-house built system, the Display Infrastructure for Virtual Environments (DIVE) (Wischgoll et al., 2018), utilizes 27 55-inch full-HD LED-backlit displays with small bezels. Specifically, we used Samsung's UA55E large-format displays as those devices are commercial-grade displays. Arranged in a 3-by-3 configuration per wall using three walls, the system provides a 12-by-12 foot walkable footprint with a height of 87 inches. Each wall is driven by a single computer with three graphics cards running each of the Samsung displays in the side-by-side HDMI stereo mode. Since active stereo glasses are used for this system, all graphics cards are frame-looked using AMD's FirePro S400 sync cards. This provides a high-resolution display system bright enough to be used in an environment with the lights fully turned on. To interact with the system, a NaturalPoint OptiTrack optical tracking system is used as well as a Logitech wireless gamepad. In addition, our custom pinch glove uses the electronic components of a wireless mouse with contacts on the fingers and thumb wired to where the mouse buttons used to connect. These pinch gloves are also fully tracked in 3D space using the optical tracking system. This type of walkable display

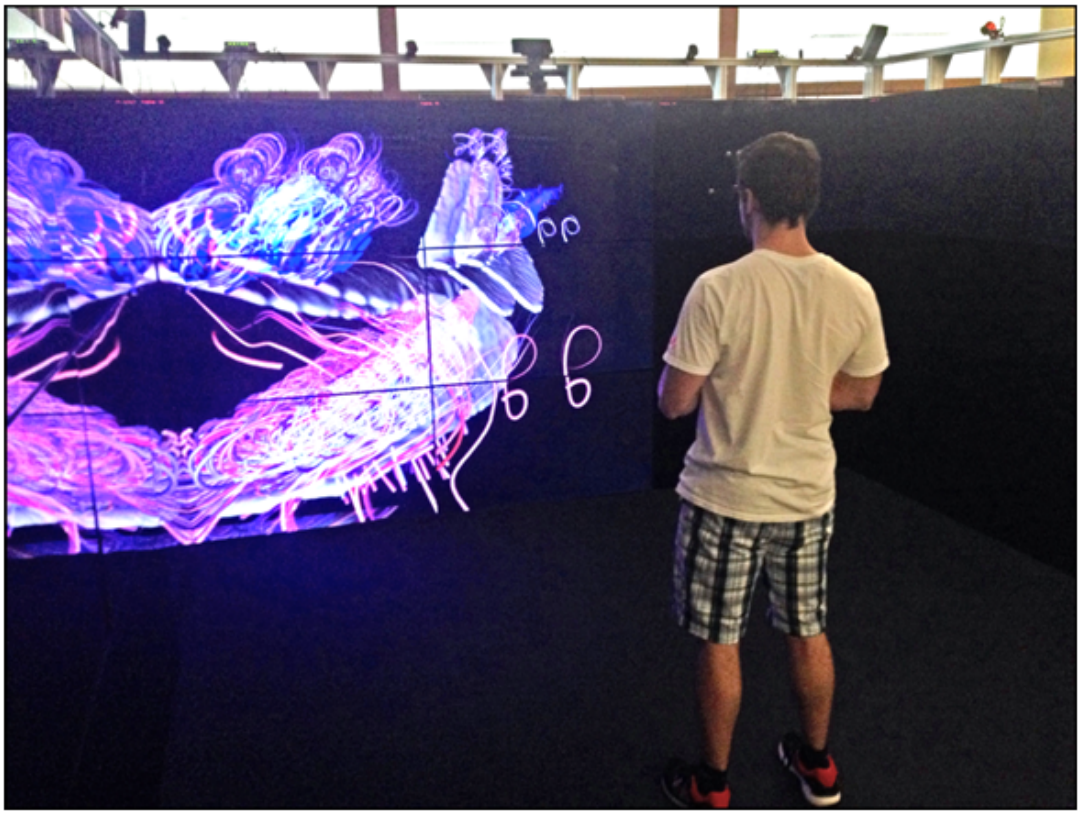

Figure 5. A user exploring a CFD data set describing the flow around a dragonfly's wings. The immersive and high-resolution capabilities of the DIVE system make this setup more suitable for allowing the user to analyze the intricacies of the flow and its vortices.

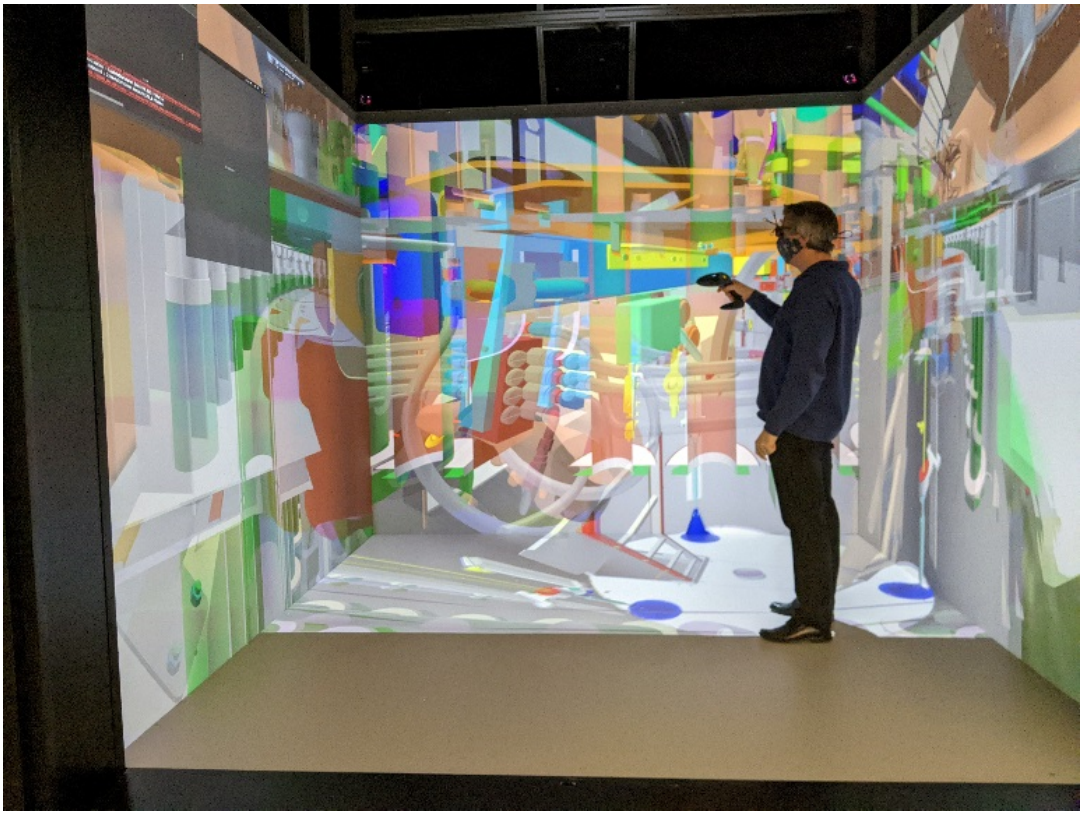

Figure 6. A user interactively exploring the design of a wind turbine in the Virtalis ActiveCube. The system enables to user look inside hidden objects or move elements blocking the view interactively.

system provides the maximal field-of-view humans are capable of perceiving. This enabled us to use the DIVE system successfully for various human perceptional studies, such as the one performed by Guthrie et al. (2018). In this study, a participant automatically moved through a virtual environment with shelves on either side. The task was to find objects that differed from the majority of objects. Based on the head tracking data, the effort needed to find these objects in different configurations and at varying speeds was identified. This study revealed that some geometric shelf layouts were clearly superior to others.

Figure 5 shows a user analyzing the flow around a dragonfly's wings. The DIVE system supports this task by immersively visualizing the data and the high-resolution capabilities portray the intricate details of the flow data to better observe the transition of the vortices between the wings. These are just two examples as the DIVE system can be used with a wide variety of data sets to support many visualization and virtual reality tasks.

To provide access to a fully immersive walkable projection-based display system, we chose the Virtalis ActiveCube to upgrade from the previous Barco I-Space. Our configuration includes a typical 10-by-10 feet footprint with projections on three walls and the floor. Figure 6 shows this system depicting a highly detailed model of a wind turbine that can be interactively explored and individual elements of the model be moved using the Flystick2. The system uses Barco's F80 series projectors which are laser-based projectors to avoid continuous bulb replacements. Each side wall uses two projectors resulting in about 2716 by 2716 pixels per wall. The edge blending to achieve a smooth transition between the images generated by the two projectors is handled within the projectors directly. The system uses Da-Lite's fixed acrylic screens for a rigid projection surface with minimal movement to ensure a smooth transition between projections at the edges. To drive the projectors, each pair of projectors is connected to an Nvidia Quadro RTX 5000 graphics card. The graphics cards are frame-locked to support the active stereo glasses used by this system. The display system is combined with an A.R.T optical tracking system with a Flystick2 for input. The nodes driving the projectors and the node running the software for the tracking system are interconnected with a 10 Gbit network connection to maximize the communication speed available for all the nodes and minimize any delays. A 5.1 surround sound system provides the audio needs for this system. The Barco I-Space this system replaced had a similar footprint but lower resolution at 1400 by 1050 pixels per wall. Thus, the ActiveCube's significantly higher resolution and brighter image make this configuration much better suited for many visualization tasks and other applications.

Software Environments

Many of the display systems are powered by Linux or support a dual-boot environment in which Linux and Windows are installed side-by-side. Additional software is used to realize different virtual environments as outlined in the following sections. We have a variety of software packages installed, including ParaView (Ahrens, Geveci, & Law, 2005) and FreeVR (Sherman, Coming, & Su, 2013). However, the ones listed in the sections below are the ones most frequently used.

VRUI

One of the libraries we have successfully used to develop more sophisticated software is VRUI created by Kreylos (2008). This library supports a variety of display configurations and is fully customizable with excellent support for different tiled display configurations run by one or more computers. Support for

several input devices, such as gamepads, is also available. Keyboards and mice can be used as input devices as well. Hence, our pinch gloves using the electronic components of a wireless mouse are directly supported already. Multiple 3D tracking systems are supported either natively, such as the Intersense systems, or through network-based protocols, such as VRPN or A.R.T's dtrack. Support for additional input devices can also be added based on the already available drivers. For example, we implemented support for touch-based input devices. This allows us to utilize the Logitech T650 touchpad or any other touchscreen to enable smartphone-style input metaphors, such as pinch-zoom, rotation, or panning. This can provide a very intuitive input mechanism for many applications.

The implementation for our touch interface for VRUI is available on github as part of the VTK framework for VRUI (Wischgoll, 2023). The *TouchNavigationTool* class is loosely based on the *MouseNavigationTool* provided by VRUI. It assumes that a link in */dev/touch* points to the proper event input device for the touchpad or touchscreen. We created a dedicated rule for the *udev daemon* to create this symbolic link with the appropriate permissions for the device file. In order to support touch devices in other VRUI-based applications, one only needs to copy the *TouchNavigationTool* class into the project and initialize the touch tool. A factory for initializing the tool is already in place so all one has to do is to call the *createTouchTool* method in the *TouchNavigationTool*.

The *TouchNavigationTool* will then create a background thread in which it continuously processes the input device file for any potential input. The implementation supports multitouch devices. One of our touch screens supports up to ten finger touch. The *TouchNavigationTool* tracks the positions of as many fingers as are supported by the device up to a maximum of twenty. Whenever the driver detects a change in the position where a finger is touching the screen, an event is generated by the driver with the updated coordinates. The *TouchNavigationTool* then keeps track of all fingers and their current coordinates.

Once the finger positions are tracked, the *TouchNavigationTool* implements different modes for panning, zooming, and rotation based on the number of fingers touching the screen. This could obviously be adjusted based on the needs of the specific applications. With one-finger touch, the *TouchNavigationTool* allows the user to rotate the view. This follows the same mechanism as the mouse rotation in VRUI so that it automatically rotates around the visible center. It should be noted that the visible center may vary greatly depending on the display system. In a desktop environment, it would be the position that relates to the

11 **Figure 7.** User interacting with a volumetric model using two-finger touch on a touch screen.

center of the screen. In a CAVE-type environment, this will usually be the center of the entire display system.

With two fingers touching the screen or device, zoom and rotation is implemented similar to how map applications on cell phones work, i.e., a pinch-type zoom mechanism and a rotation around the z-axis when the fingers rotate on the screen uniformly. Whenever three fingers touch, the view is panned in the direction in which the fingers move. This typically provides a very intuitive input paradigm, especially since most users are very familiar with interacting with cell phones which provide similar interaction styles. Figure 7 shows an example of a user exploring a volumetric model using the multi-touch support. The image on the touch screen is mirrored on the CADWall screen to make this a very intuitive interface for interacting and explaining a data set to a larger audience.

In order to render content, VRUI creates an OpenGL context for all the displays involved in the specific configuration. For simple setups, this may be just a single one. For more complex ones, multiple OpenGL contexts may be created. For example, the CAVE-type display using the Samsung TVs describes earlier renders onto a column of 3 TVs using a dedicated graphics card. Each wall is formed by a 3×3 configuration of TVs so there will be three OpenGL contexts created for each wall. This allows VRUI to fully take advantage of the accelerated 3D support provided by the graphics cards. VRUI then renders into

those graphics contexts as needed which typically results in rendering the graphics content nine times: three times per wall for three walls.

The advantage of providing an OpenGL context is that any OpenGL-based application can be supported. One can use a traditional approach to render content using OpenGL commands to draw the required geometry. However, any other OpenGL-based library can be used as well. For example, we also use OpenSceneGraph (*OpenSceneGraph*, 2023) and have it render any geometric content into the OpenGL context. Assuming the scene graph is already configured as desired, the *osgViewer* class can then be used to allow OpenSceneGraph to render into the existing OpenGL context. First, the scene graph is updated using the *updateTraversal* method. Then, the current camera is obtained from the *osgViewer* instance and its settings are updated based on the current view settings in VRUI. Specifically, the viewport, projection matrix, and view matrix are set to identical values within the camera of the *osgViewer* as used in VRUI. Finally, the *renderingTraversal* method is called to initiate the rendering process which will issue the OpenGL calls based on the scene graph.

Any library building onto OpenSceneGraph can then be supported in this way as well. Additionally, the bullet physics engine (*Bullet Real-Time Physics Simulation*, 2023) is used to provide a physics model to create realistic animations. This framework makes creating virtual environments easier due to the support the individual elements of the framework provide. OpenSceneGraph supports a large variety of 3D formats that can be directly ingested. Additional formats can be supported through the plugin mechanism deployed by OpenSceneGraph. The bullet physics engine then provides a mechanism for adding realistic physical properties to different objects. When creating a virtual environment with this framework, the developer can decide which objects are handled by the physics engine. For example, static objects obviously do not require any physical properties to be managed. This then reduces the computational burden by not assigning any physical properties to these objects. Lastly, SmartBody (Thiebaux, Marsella, Marshall, & Kallmann, 2008) is used for character animations. SmartBody is a powerful library that supports a behavior markup language (BML) to control individual characters. For example, this allows one to tell a character to walk to a specific location. Basic animation models are included in SmartBody so that the character can be realistically animated. One can even include a map so that the character can avoid objects without running into them.

The other big advantage of VRUI is that it is centrally configured with a few configuration files. In our case, these configuration files are located on the server that all of our Linux installations mount. The parameters for all of our display systems are collectively specified in these configuration files. This then allows VRUI to identify the computer it is running on based on its hostname and grab the configuration for that system. As a result, the same software can then be run directly on all of our display systems without changing the software or even the need for recompiling it. This provides a very elegant software development platform in an educational environment by allowing students to develop on a simpler hardware configuration, for example, a 4K stereo-capable TV with Natural Point's optical tracking system as provided in the Immersive Visualization and Animation Theater. The students can then simply take their software and run it on the DIVE system, the Virtalis ActiveCube, or any other display system immediately without requiring changes to the software.

Unity

The game engine Unity (*Unity Game Engine*, 2023) has gained a lot of popularity within the virtual and augmented reality community. This is mainly due to the fact that many manufacturers support Unity directly for their devices. This is the case for pretty much all head-mounted virtual reality devices, such as the HTC Vive series or HP's mixed reality devices, but also for augmented reality devices, such as the Magic Leap devices or Microsoft's Hololens. In addition, Unity provides a relatively easy entrance to developing virtual and augmented reality software since there is a lot of support for a large variety of devices and controllers as well as additional support for developing 3D environments. Unity can also be used in traditional CAVE-type configurations. MiddleVR provides a commercially available integration for using Unity with CAVE-type displays (Kuntz, 2015). An open-source integration was developed at the University of Wisconsin, Madison by Tredinnick, Boettcher, Smith, Solovy, and Ponto (2017) that is freely available called Uni-CAVE. Since Uni-CAVE may require more configuration compared to MiddleVR, Davis et al. provide additional tutorials for setting up Uni-CAVE (Davis et al., 2022). Once configured, Unity-based software developed for other devices, such as head-mounted displays, can be ported over by importing the content into the new Unity environment. Any interaction mechanism and input devices are typically handled through C# scripting. Different display systems often have different input devices. Hence, switching between display systems requires these C# scripts to be adjusted to the different input devices. The Virtalis ActiveCube in the Appenzeller Visualization Laboratory uses the A.R.T optical tracking

system with the Flystick2 and thus relies on the dtrack protocol. Uni-CAVE's examples all use the VRPN protocol. A.R.T provides Unity support for dtrack which then has to be integrated with the Uni-CAVE configuration. Uni-CAVE relies on the remote procedure call (RPC) mechanism to transfer tracking and other synchronization data over from the master node to the client nodes. So once the master node is able to obtain the tracking data through the dtrack protocol, this same mechanism can be used. So while a good solution, Uni-CAVE definitely involves more work when switching between display systems compared to VRUI.

There are currently a few projects using equipment in our lab that started out based on Unity. Some of these use augmented reality devices and others are tablet- or phone-based for portability reasons. All of these projects revolve around different training scenarios for medical personnel. While porting over a Unity application to a different display device takes quite a bit of effort and additional programming, the ability to reuse the majority of the application is a great way for exploring the potential use of a CAVE-type system for training purposes with these specific applications. All the modeling and animation effort to make the application work in Unity carry over directly. However, any interaction mechanism needs to be adjusted to work with the input devices provided by the CAVE-type system. There are usually large segments of the interaction scripts in Unity that can be reused but some adjustments are usually required to make it work.

Overall, Unity provides a good framework for a CAVE-type system, especially if large portions of a project are already available in Unity. While being a game engine, it is not perfectly suited for all immersive display systems. However, it provides all the basic components to make it work in CAVE-type systems. About three years ago, Unity made some changes to the stereoscopic rendering modes and moved to an XR plugin mechanism. This is based on the assumption that manufacturers can create official plugins for stereoscopic rendering. While this provides a lot of flexibility for a variety of devices, stereoscopic projectors are typically not considered a device in that sense. Thus, there is no manufacturer providing a direct plugin for Unity for traditional stereoscopic supported by the graphics hardware directly. This is why we still use Unity 2019 on our systems.

Several of our projects utilized Unity as the basic framework. This includes the nursing training pictured in figure 2 and the simulation environments.

12 **Figure 8.** Simulated training environment for medical personnel used to better prepare physicians and improve different metrics, such as empathy.

More recently, we received funding from the Ohio Department of Medicaid for developing simulation environments for virtual training of medicaid professionals. These training environments are intended to reduce bias among the trainees toward patients with addictions (Hershberger et al., 2022) or autism spectrum disorder (Patel et al., 2023) among others. Figure 8 shows an example of such a training environment. It depicts a refugee consulting a physician. The training environment includes the entire interaction with the physician where the physician has different options as to how to respond to the patient. The patient then reacts in different ways based on the physician's choices. The simulation also provides the patient's background to allow the participant in the simulation to better understand the patient. Survey questionnaires are built into the simulation to evaluate before and after the physician's interaction with the patient to track the change. With 230 participants in this study, improvements in all metrics were observed, albeit not all of them were statistically significant. Most notably, there was an increase in compassion and reduction in anxiety observed. This series of studies prepared several simulations with different types of patients to cover a wide range.

Visionary Renderer

Visionary Renderer (*Visionary Renderer*, 2023) is a commercial software package provided by Virtalis. Visionary Renderer can be run on a standard desktop computer or within a CAVE-type environment. It also supports head-mounted VR displays. Visionary Renderer is capable of ingesting a variety of different CAD formats natively. These models can then be explored or edited within the virtual reality environment.

The models can be fully animated through lua (Ierusalimschy, De Figueiredo, & Celes, 2006) scripting. This can be used to create simulated environments for training purposes of using some type of equipment or machinery, as just one example. Visionary Renderer provides a very high visual quality and flexibility in configuring the virtual environment. Figure 6 shows a use-case of this software.

VTK

VTK (Schroeder, Avila, & Hoffman, 2000) can also be used for creating virtual reality applications, particularly for the purpose of visualizing or exploring different types of data sets. VTK supports the HTC Vive directly in Linux and Windows. It requires SteamVR to be installed. Other devices, such as the Magic Leap, could be supported through other means, including WebXR. There are other efforts using OpenXR and OpenVR to support virtual reality devices with VTK (*VTK::RenderingVR*, 2023).

Another way of utilizing VTK is by combining it with VRUI. Since VRUI provides an OpenGL context that anything could technically be rendered into, one only needs to get VTK to render into the context provided by VRUI instead of using its own render windows. In earlier versions of VTK, this required the use of render passes. This approach lets one pick which render passes to run and then execute this set of render passes. This would then issue the necessary OpenGL commands to render into the existing context. While a working solution, this tended to be sensitive to what algorithms in VTK one wanted to use. For example, there is a dedicated render pass for volumetric algorithms which has to be selected if volume rendering is going to be used.

Since version 6.2, VTK provides the *vtkExternalOpenGLRenderer* and *vtkExternalOpenGLRenderWindow* classes specifically for the purpose of rendering into existing OpenGL contexts. This made this significantly more stable and reliable. In addition, the code needed to make this work became shorter and much simpler. It essentially follows the traditional pipeline setup with VTK but instead of using the traditional render windows and renderers *vtkExternalOpenGLRenderer* and *vtkExternalOpenGLRenderWindow* are used. O'Leary et al. also incorporated VTK with VRUI using this

approach (O'Leary et al., 2017). However, our initial approach predates this publication as the original implementation had to utilize the rendering pass feature in VTK to get VTK to render into an existing OpenGL context as the external rendering capabilities were not yet available. Obviously, VTK's external rendering capabilities simplify this significantly.

As always with VRUI, one has to pay attention to how things are rendered to make sure any stored content, such as textures and display lists, is available on the specific graphics card. This is especially important for CAVE-type systems or any system that relies on multipipe rendering, i.e., any system that uses more than one computer to drive the displays or systems that rely on multiple graphics cards used per computer. The DIVE system utilizes both of these mechanisms and as such is especially sensitive to this issue. This then requires the proper use of the *dataItem* in Vrui so that the textures are transferred and graphics content is rendered into each graphics card as otherwise some areas of the display system would not be able to show the rendered content at all. Hence, the VTK rendering needs to be handled entirely within the *dataItem* and as a result, one has to make sure that any necessary files are available at each rendering node. However, this is typically the requirement when using other rendering libraries combined with VRUI as well.

We have implemented a variety of visualization approaches based on VTK, such as a molecular visualization and a volumetric renderer. The base code for our implementation is available on github (Wischgoll, 2023).

Summary

Providing a variety of frameworks can be beneficial when running a virtual reality laboratory like this. Each framework has its pros and cons and not every framework supports every device. For example, a VRUI-based application makes it very easy to switch between different display systems without the need to change or even recompile the software. This allows one to easily experiment with different display modalities to identify the most suitable. Since it relies solely on OpenGL, highly customized rendering methods can be used to accommodate large data sets, for example. This was used to render a large vascular model that included arteries down to the capillary level resulting in a geometric representation too large to be rendered using conventional methods (Wischgoll, 2013).

On the other hand, Unity can be a great tool if one is looking for proliferation as it can support a wide range of devices. It does, however, require creating dedicated applications for each platform and some platforms may require some modification of the user interface. This was the reason we chose Unity as the platform for the simulation environment for training physicians described previously as it allowed us to provide applications for different platforms people could then use at their own time.

Adminstration

Running a virtual reality laboratory can be challenging from an administrative perspective. On the one hand, augmented and virtual reality devices and display systems tend to be more expensive. The advent of modern head-mounted displays has brought down the cost of entry significantly with some headsets starting at just a couple hundred dollars. But many are still at $1000 or higher in price. Augmented reality devices tend to be even costlier and can go for even more than $3000. Walkable display systems are typically even more expensive with many CAVE-type display systems costing hundreds of thousands of dollars or more. However, in our experience, these display systems can be advantageous, especially for novice users as they are not completely detached from the real world as would be the case with a head-mounted display. It is also a little easier for the user to take off the 3D glasses compared to a head-mounted display. In our experience, this makes these systems easier to use for novices when showcasing the technology. Another advantage can be field-of-view. Head-mounted displays typically have a limited field of view of 140 degrees or less. The peripheral vision a typical human is capable of perceiving extends beyond that. We have performed studies in our laboratory where covering the full field of view including the peripheral vision was important which is why we chose the DIVE system for those studies (Guthrie et al., 2018).

Another issue with directing a virtual reality laboratory is maintenance. In our case, we do not have dedicated technicians to support the systems. This makes it important for the systems and the software to require as little maintenance as possible. Based on the design choices we made for both the hardware and software configurations, we were fairly successful in achieving this goal. The following subsections provide more detail on administrative aspects, such as funding the laboratories, maintaining the equipment, and developing software for these systems.

Funding

The initial investment for the laboratories came from the Ohio Third Frontier program to set up the Appenzeller Visualization Laboratory augmented by private donations. These funds allowed us to purchase our first CAVE-type system, a Barco I-Space, and a large display wall, a Barco CADWall. Additional investments from the College of Engineering at Wright State University were used to add the TV-based CAVE-type display system (DIVE) and additional large tiled display walls.

Recently, a grant from the Ohio Department of Higher Education funded a replacement of the original CAVE-type system with the Virtalis ActiveCube, and additional head-mounted virtual and augmented reality displays. This effort deploys virtual and augmented reality techniques for workforce development.

On the computer science side, this includes developing software for these types of systems and managing all the components to make these systems work, including the displays themselves as well as the tracking systems. We offer courses specifically geared toward the use and development of software for these devices.

In addition, virtual and augmented reality techniques can provide a safer learning environment both for the participant and the elements the participant is working with. In a virtual environment, the participants are less likely to hurt themselves or break anything. For example, Menon et al. (2021) demonstrated a training environment for nursing students to increase the student's learning and retention. Additional grant funding from the U.S. Air Force, U.S. Army, the Ohio Department of Medicaid, and the National Science Foundation supported the software development of a variety of visualization and extended reality software. The software developed for these projects includes simulation software for different environments used for testing subjects under a variety of scenarios. Different examples were described in the previous sections of this manuscript.

Supporting a set of laboratories at this scale can be challenging. We have been very fortunate to have been able to fund our efforts through a variety of sources as outlined above. While several efforts in the laboratory were supported by grant funding from sources such as NSF and NIH, we have also sought funding through non-traditional sources. This has been a very effective strategy in our experience. Wright State traditionally has maintained very close ties to the Air Force due to its proximity to one of the largest Air Force bases in the country. This has allowed us to obtain significant amounts of funding from the Air Force but also the Army. Other local funding sources include the Ohio Department of Higher Education

and the Ohio Department of Medicaid who funded our training and workforce development efforts using different virtual reality-based strategies. We also collaborated with different companies over the years funding some of our efforts.

The majority of our display systems currently in use with the exception of the Barco CADWall and the Virtualis ActiveCube were designed and built in-house. This allowed us to significantly cut down on the cost of these display systems. At the same time, it ensured that the necessary knowledge for administering and maintaining these systems is available in-house directly as well. This then cuts down on maintenance costs and downtime as a direct result.

Maintenance

In our experience, Linux has served us well in supporting the diverse set of display environments available in our laboratories. The Windows operating system enforces updates which tend to break things at times. In Linux, we have updates disabled by default, and any updates are applied manually, typically on all systems at the same time. The installation is identical for all systems. In fact, we do not use the installers that come with the Linux distribution but instead just create a copy by unpacking a tar archive and then adjusting a few settings. This typically includes changing the IP address and in some cases updating a network or graphics driver. All of our Linux systems tie in with a common server infrastructure where home directories and additional software packages are installed and accessed by all clients driving the displays. This creates a very homogenous environment in which installed software packages become available to all clients immediately through the server. Similarly, the same software can be run on various display systems without requiring any changes using software libraries, such as VRUI. From a maintenance perspective, this makes this type of setup require significantly less support than Windows or other configurations. The Linux installation very rarely requires us to make any kind of adjustments.

Virtual reality laboratories tend to have a large amount of visibility given the display systems' ability to provide easy access for anyone to the content they display, such as different types of data sets that are visualized in a CAVE-type system. As a result, there are fairly frequent requests for open houses or last-minute visits. The laboratories typically see well over a hundred visitors in a year and the utilization from students and researchers is similar in numbers. In order to make this as easy as possible, we use a dedicated demo account with many of the software packages that we developed over the years and their

configurations installed directly within this account. This isolates all the settings and the software from any software development activities allowing us to have working versions of these software packages available at all times in case we need to do a quick demonstration for an impromptu visitor.

We also try to find hardware configurations that require only minimal maintenance. This is why our updated CAVE-type system, the Virtalis ActiveCube, utilizes laser-based projectors to avoid bulb replacements. This cuts down on maintenance, downtime, and costs all at the same time. Another maintenance issue with projection-based systems can be alignment. All of our systems that use more than a single projector and thus require precise alignment of the images use fixed projection screens made out of glass or plexiglass. This provides the most rigid setup. The Appenzeller Visualization Laboratory is temperature and humidity controlled with its dedicated air conditioning unit to avoid structural movement as much as possible.

The in-house built systems using different types of LCD displays or TVs have been very reliable as well. Over the years, we had to replace just a single TV for the DIVE system. This occurred fairly early on and was still covered under warranty. All the computing equipment driving these displays was configured and assembled entirely in-house and use high-quality name-brand components. As a result, we had very few hardware failures over the years.

Software Development

Developing software for augmented and virtual reality applications can be challenging in an educational environment. For research projects, graduate students are usually developing a lot of code for a variety of software projects. Developing software for teaching purposes can be more challenging, albeit educational research funding is sometimes available for this purpose. However, graduate students typically develop prototype software for their research projects. Once the student graduates, there is a bit of a knowledge drain with respect to that software. As a result, managing larger software projects can be difficult in such an environment.

Providing students with basic frameworks to develop their software can make integrating various components into a larger framework easier. This then still poses the need for integrating the different components to form a larger software package with the additional requirement of testing the final version of the software. Alternatively, the students can contribute directly to a common software repository.

Establishing coding standards, testing procedures, and best practices can help to ensure a quality standard of the resulting software.

Given the fact that these laboratories are part of a university environment, the majority of software was developed as part of a research project. This includes all the examples previously listed in this manuscript.

Conclusion

In conclusion, the visualization and simulation environment at Wright State University has been very successful. The students appreciate and enjoy having access to this type of equipment. The laboratories have contributed to a variety of research projects, some of which are highlighted in this manuscript. Especially the larger display systems are met with a lot of excitement from the general population during open-house events as well.

These display systems were successfully used for experimental studies (Parikh et al. (2018)) and comparative visuslizations (Marangoni and Wischgoll (2015)), for example. Virtual reality and simulations have also been shown to be effective tools for education and training with clear benefits to the learning process. We have used our virtual reality and simulation laboratories for training nurses (Menon et al. (2022, 2021)), medical personell (Hershberger et al. (2022)), and our students (Wischgoll (2019)).

In our experience, providing a set of software environments is beneficial as one solution does not fit everyone's needs. Relying on stable software configurations to minimize maintenance requirements has served us well in the past. Similarly, building the majority of systems in-house has made hardware maintenance significantly easier.

The systems available in our laboratories are very well-suited for a variety of applications, including visualization, training, and experimentation. The systems are met with a lot of excitement from students and visitors. We are planning to integrate the use of this technology more into the curriculum within the engineering disciplines and beyond to better prepare the students for the workforce. Given the nature of these displays and their versatility to be used in a wide variety of applications we envision their use in additional virtual experiments and visualization tasks in the future. We are regularly approached with visualization needs from within the university and outside. Due to their immersiveness and interactivity,

these display systems can play an important role in meeting those needs as illustrated by the examples listed in this manuscript.

Acknowledgments

The activities outlined in this paper were supported by the National Science Foundation, the U.S Air Force, the U.S. Army, the Ohio Third Frontier program, the Ohio Department of Higher Education, Ohio Department of Medicaid, and private donations from Robert C. Appenzeller.

REFERENCES

Ahrens, J., Geveci, B., & Law, C. (2005). Paraview: An end-user tool for large data visualization. *The visualization handbook*, *717*(8).

Anthes, C., García-Hernández, R. J., Wiedemann, M., & Kranzlmüller, D. (2016). State of the art of virtual reality technology. In *2016 ieee aerospace conference* (pp. 1–19).

Berg, L. P., & Vance, J. M. (2017). Industry use of virtual reality in product design and manufacturing: a survey. *Virtual reality*, *21*, 1–17.

Bullet real-time physics simulation. (2023). https://pybullet.org. (Accessed: 2023-11-06)

Carruth, D. W. (2017). Virtual reality for education and workforce training. In *2017 15th international conference on emerging elearning technologies and applications (iceta)* (pp. 1–6).

Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., & Hart, J. C. (1992). The cave: audio visual experience automatic virtual environment. *Communications of the ACM*, *35*(6), 64–73.

Davis, C., Collins, J., Fraser, J., Zhang, H., Yao, S., Lattanzio, E., … Palaniappan, K. (2022). CAVE-VR and unity game engine for visualizing city scale 3d meshes. In *2022 ieee 19th annual consumer communications & networking conference (ccnc)* (pp. 733–734).

Dincelli, E., & Yayla, A. (2022). Immersive virtual reality in the age of the metaverse: A hybrid-narrative review based on the technology affordance perspective. *The journal of strategic information systems*, *31*(2), 101717.

Febretti, A., Nishimoto, A., Thigpen, T., Talandis, J., Long, L., Pirtle, J., ... others (2013). Cave2: a hybrid reality environment for immersive simulation and information analysis. In *The engineering reality of virtual reality 2013* (Vol. 8649, pp. 9–20).

Gillmann, C., Smit, N. N., Gröller, E., Preim, B., Vilanova, A., & Wischgoll, T. (2021). Ten open challenges in medical visualization. *IEEE Computer Graphics and Applications*, *41*(5), 7–15.

Guthrie, B. R., Parikh, P., Whitlock, T., Glines, M., Wischgoll, T., Flach, J., & Watamaniuk, S. (2018). Comparing and enhancing the analytical model for exposure of a retail facility layout with human performance. In *Proceedings of the 2018 IISE annual conference.*

Hershberger, P. J., Pei, Y., Crawford, T. N., Neeley, S. M., Wischgoll, T., Patel, D. B., ... others (2022). An interactive game with virtual reality immersion to improve cultural sensitivity in health care. *Health Equity*, *6*(1), 189–197.

Ierusalimschy, R., De Figueiredo, L. H., & Celes, W. (2006). *Lua 5.1 reference manual.* Lua. org.

Koehler, C., Berger, A., Rajashekar, R., Wischgoll, T., & Su, S. (2019). Dynamic collaborative visualization ecosystem to support the analysis of large-scale disparate data. In *2019 ieee international conference on big data (big data)* (pp. 3968–3977).

Kreylos, O. (2008). Environment-independent vr development. In *International symposium on visual computing* (pp. 901–912).

Kuntz, S. (2015). MiddleVR a generic VR toolbox. In *2015 ieee virtual reality (vr)* (pp. 391–392).

Marangoni, M., & Wischgoll, T. (2015). Comparative visualization of protein conformations using large high resolution displays with gestures and body tracking. In *Visualization and data analysis 2015* (Vol. 9397, pp. 135–147).

Menon, S. S., Holland, C., Farra, S., Wischgoll, T., & Stuber, M. (2022). Augmented reality in nursing education–a pilot study. *Clinical Simulation in Nursing*, *65*, 57–61.

Menon, S. S., Wischgoll, T., Farra, S., & Holland, C. (2020). Medical education and assisted surgery by ar. In *The campus alliance for advanced visualization (thecaav20).*

Menon, S. S., Wischgoll, T., Farra, S., & Holland, C. (2021). Using augmented reality to enhance nursing education. *Electronic Imaging*, *2021*(1), 304–1.

O'Leary, P., Jhaveri, S., Chaudhary, A., Sherman, W., Martin, K., Lonie, D., ... McKenzie, S. (2017). Enhancements to vtk enabling scientific visualization in immersive environments. In *2017 ieee virtual reality (vr)* (pp. 186–194).

Openscenegraph. (2023). http://www.openscenegraph.org/. (Accessed: 2023-11-06)

OLeary, P., Sherman, W. R., Shetty, N., Clark, J., & Hulme, D. (2013). Providing a progression of immersive visualization technologies. In *Vistech'13 workshop: Visualization infrastructure and systems technology at the international conference for high performance computing, networking, storage and analysis (sc'13).*

Parikh, P., Guthrie, B. R., Whitlock, T., Glines, M. A., Wischgoll, T., & Flach, J. (2018). Validation of models to estimate exposure in a retail layout using a 3d virtual environment. *Industrial and Systems Engineering Research Conference (ISERC).*

Patel, D. B., Pei, Y., Vasoya, M., & Hershberger, P. J. (2023). Computer-supported experiential learning-based tool for healthcare skills. *IEEE Computer Graphics and Applications*, *43*(2), 57–68.

Roswell, R. O., Cogburn, C. D., Tocco, J., Martinez, J., Bangeranye, C., Bailenson, J. N., ... Smith, L. (2020). Cultivating empathy through virtual reality: Advancing conversations about racism, inequity, and climate in medicine. *Academic Medicine*, *95*(12), 1882–1886.

Schroeder, W. J., Avila, L. S., & Hoffman, W. (2000). Visualizing with vtk: a tutorial. *IEEE Computer graphics and applications*, *20*(5), 20–27.

Sensing, T., Parikh, P., Hardman, C., Wischgoll, T., & Menon, S. S. (2021). Augmented reality headset facilitates exposure for surgical stabilization of rib fractures. In *16th annual acadmic surgical congress.*

Sherman, W. R., Coming, D., & Su, S. (2013). Freevr: honoring the past, looking to the future. In *The engineering reality of virtual reality 2013* (Vol. 8649, pp. 47–61).

Smith, S. J., Farra, S., Ulrich, D. L., Hodgson, E., Nicely, S., & Matcham, W. (2016). Learning and retention using virtual reality in a decontamination simulation. *Nursing education perspectives*, *37*(4), 210–214.

Sutherland, I. E. (1965). The ultimate display. In *Proceedings of the ifip congress* (Vol. 2, pp. 506–508).

Thiebaux, M., Marsella, S., Marshall, A. N., & Kallmann, M. (2008). Smartbody: Behavior realization for embodied conversational agents. In *Proceedings of the 7th international joint conference on autonomous agents and multiagent*

systems-volume 1 (pp. 151–158).

Tredinnick, R., Boettcher, B., Smith, S., Solovy, S., & Ponto, K. (2017). Uni-cave: A unity3d plugin for non-head mounted vr display systems. In *2017 ieee virtual reality (vr)* (pp. 393–394).

Unity game engine. (2023). https://unity.com/. (Accessed: 2023-11-06)

Visgap. (2023). https://visgap.gitlab.io.

Visionary renderer. (2023). https://www.virtalis.com/software/visionary-render. (Accessed: 2023-11-06)

Vtk::renderingvr. (2023). https://docs.vtk.org/en/latest/modules/vtk-modules/Rendering/VR/README.html. (Accessed: 2023-11-06)

Wischgoll, T. (2013). Visualizing vascular structures in virtual environments. In *Visualization and data analysis* (pp. 86540S-1–86540S-8).

Wischgoll, T. (2017). Display systems for visualization and simulation in virtual environments. *Electronic Imaging*, *2017*(1), 78–88.

Wischgoll, T. (2019). XR-based workforce develop in the southwestern region of ohio. *The Campus Alliance for Advanced Visualization (THECAAV21)*, 27–28.

Wischgoll, T. (2023). *VtkVrui GitHub repository.* https://github.com/wischgol/vtkVrui.

Wischgoll, T., Glines, M., Whitlock, T., Guthrie, B. R., Mowrey, C. M., Parikh, P. J., & Flach, J. (2018). Display infrastructure for virtual environments. *Electronic Imaging*, *2018*(1), 060406–1.