Survey of Research Software for Building Virtual Reality Systems

Dylan Kobayashi

University of Hawai Manoa

dylank@hawaii.edu

Abstract

This literature review will cover publications about software written and distributed to support virtual, augmented, and mixed reality. The trends of software will be covered after a brief hardware history which is necessary to see they influence each other. History will also give context as to why and how augmented reality has had an intertwined past with virtual reality and later show indications of a shared future fate. The trends in software over the paste decade lead up to a convergence towards utilizing commercial solutions. Within the trends some dangers can be identified which need to be addressed or avoided as seen with the coverage of XR's potential future.

1 Introduction

Virtual reality(VR) and augmented reality(AR) software have reached a crossroads. The path taken will have influences that extend beyond research and affect the general public's perception and acceptability. While the grand challenges of VR and AR have not changed throughout the decades, there have been significant changes in definition, format, and implementations to achieve the grand challenges. It is doubtful that either will ever lose its place in research. The current hype around VR and AR is providing an opportunity to solidify its place in mainstream usage, but when viewed from the lens of the past, there are observable trends which may prevent adoption. Unless special effort is made to carefully navigate past the pitfalls, VR and AR may end up disappearing from the market or gain the stigma of just another gimick and limited to a gaming console competitor. To explain why such a pivotal point has been reached, this survey will first need to establish a common ground through defining terminology. Although this survey is focused on software, VR and AR has a deeply intertwined with each other and their software with hardware. To fully appreciate or understand the changes in one, being aware of the other is critical. A brief review of hardware history will be covered for this reason. The following section will illustrate the grand challenges and how they relate to hardware and software. The definitions of VR and AR will then guide the selection of papers which describe research software distributed for developing systems. Based on those papers the sections after will be trends, dangers, and future potential. Finally, a conclusion to wrap up.

2 What is Virtual Reality?

Depending on when and who you ask, virtual reality (VR) can have a variety of different definitions or meanings. This applies to consumers ***and*** researchers. Before further discussion of VR can be held, a common set of vocabulary must be established and will be followed some history for context. The following definition list is the result of reviewing literature of the field and identifying the minimum commonality between the different authors. Included are some definitions that must also be clarified due to their different

meanings within other fields.

Virtual - Computer generated.

Virtual Environment (VE) - An environment generated by a computer.

Virtual Reality (VR) - Method or implementation to fully immerse a user within a virtual environment. To fully immerse a user a VR system must, at minimum:

- Track the user
- Provide stereoscopic viewer centered perspective
- Provide spatialized audio
- Enable user interaction

Augmented Reality (AR) - Method or implementation in which a user's view is enhanced with additive, interactive virtual components. An AR system must, at minimum:

- Track the user
- Track the real world around the user
- Provide stereoscopic viewer centered perspective
- Provide view of real world when not obstructed by virtual objects
- Enable user interaction

Mixed Reality (VR) - Method or implementation to merge a virtual environment with the real world. Typically MR sits between VR and AR, but is often a catchall for cases not fitting into either VR or AR. More on this later.

XR - Refers to any or all of the previously mentioned reality terms. X is a placeholder for V, A, or M.

Viewer centered perspective - View provided to match the position of a user's head. This allows a user to see the virtual environment the same way they see the physical world, by moving their head. For example, when looking at a static object, a user lowering their head will receive view updates of a lower perspective matching the amount of movement.

To briefly explain why these definitions are necessary, during the 1980s VR was commonly defined as a hardware requirement or a medium which a user interacts with, similar to a telephone or television. If a computer had a particular set of hardware components, then it was a VR system. This association is attributed to Jaron Lanier, a pioneer of VR and whom was also the CEO of Virtual Programming Languages (VPL). At the time VPL was a company considered as the name brand for equipment to supplement virtual experiences (Much like modern day's Vive and Oculus). They used the term virtual reality as a catch-all description for their projects. In 1992, Steuer used VR to describe a minimum set of experiences a user would be exposed to within a computer generated environment [Steuer 1992]. Regardless of the hardware, by providing a minimum set of experiences, one could claim having a VR system.

The definitions and requirements used for this survey are greatly influenced by the reality-virtuality continuum by Milgram and Kishino [Milgram and Kishino 1994] (Figure 1). They identified that authors were often inconsistently using the term virtual reality to denote a variety of other environments containing various amounts of immersion and synthesis. Inconsistency was further exacerbated by the, at the time, new term: augmented reality. The reality continuum is comprised of two extremes: the **real environment** and **virtual environment**. The continuum represented a scale of how much a user was immersed within the

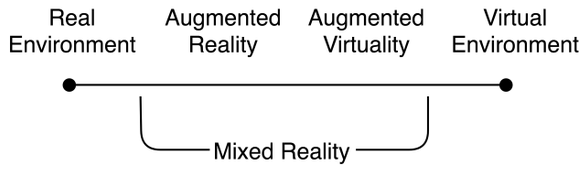

Figure 1: Milgram and Kishino's Reality-Virtuality continuuum

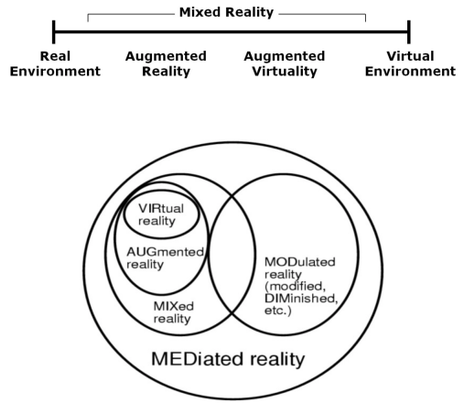

Figure 2: Mann's modification of the Reality-Virtuality continuuum

virtual environment. The far right of the spectrum represented a user fully immersed within the **virtual environment**. Their view is entirely that of the virtual environment and the real world has no relation to what is seen. Moving towards the real, the next classification is **augmented virtuality**, the entire view is produced by computer graphics, but correlates to the real world. One step away from entirely real is **augmented reality**, where a user's view is enhanced (augmented) with virtual cues about the real world. **Mixed reality** was considered a subclass of VR related technologies encompassing anything between fully real and fully virtual.

An example of an early system providing augmented virtuality is described by Edwards et al. [Edwards, J. P. Rolland, and Keller 1993] where they mounted cameras on an optically opaque head mounted display(HMD) to give the illusion of seeing through the HMD. Their work was inpsired by their work was inspired by Bajura et al.'s work on superimposing ultrasound scans over view of patients [Bajura, Fuchs, and Ohbuchi 1992]. Although the HMD was opaque, the user could still see the real world through the view provided by the captured camera feed. This is commonly referred to as video see through. The viewer sees the real world, but the view is entirely computer generated allowing superimposition of additional visuals as well as distortion of the real. Augmented reality usually ensures a user's view is unobstructed by using a see-through HMD, which at the same time, enables addition of visuals to the view. Ronald Azuma, the pioneer of AR, described three requirements: 1) combines real and virtual, 2) is interactive in real time, and 3) is registered in three dimensions [Azuma 1997].

Milgram's Reality-Virtuality continuum was further built upon by Mann who added a Y axis indicating the level of mediation or filter performed on the user's view of the real or virtual world [S. Mann 1994] (Figure 2). A VR system that severely mediated the view not only showed the virtual world, but added to or modified the view and make visible components that did not exist within the world. For example, personal control panels only visible to the owner; or object highlighting / outlining. One common aspect of VR, MR, and AR is the viewer's presence of a reality is influenced and changed. Each has a place on the virtual continuum along the X axis. Mann et al. recently identified the common usage of X Reality, or XR, referring to any of the focuses along the X axis [Steve Mann et al. 2018]. This survey will use the term XR to refer to VR, MR, AR, or any of the related technologies.

The overlap of XR is fairly significant and have some of the same core requirements: tracking the user,

providing an immersive viewer centered perspective, and enabling user interaction. While the differences can be significant, the same core allowed AR to rise from and diverge from VR while still maintaining close enough proximity that advancements in one provides opportunities which can translate to the other. Some of these correlations can be seen within the next section.

3 The Evolution of Virtual Reality Hardware Systems

Hardware and software for virtual reality has an intertwined history with feedback between each other. These changes through history played a role in terminology, as well as the direction of research that took place. To fully appreciate some of the directions taken in software, it is necessary to see how hardware has changed throughout history. This is in part due to the diverse requirements of XR (e.g. tracking and visual presentation) requiring hardware typically not included with computers (Figure 3). Tracking is one component to an XR system and has gone through a number of changes over the years which include different software to interface with, as well as physical changes which influence the user experience. Among the earliest used with computers were magnetic trackers [Fisher et al. 1987]. In modern times a variety of options exist, from infrared (for tracking reflectors ball configurations) to purely camera implementations (e.g. depth cameras like Kinect). XR needs to be visually immersive which is commonly achieved through stereoscopic approaches. While 3D TVs had become popular for viewing 3D movies with the release of Avatar, they did not reach mainstream popularity as part of standard computer configurations. This section covers some of the significant hardware milestones of XR history. For in-depth details it may be best to refer to another survey. Some starting points are those focusing on high resolution displays [Ni et al. 2006], gloves [Ni et al. 2006], tracking [Baillot, Davis, and J. Rolland 2001], MR survey [Costanza, Kunz, and Fjeld 2009], collaborative virtual environments technology survey [Frécon 2004], and a very comprehensive AR survey by Billinghurst et al. [Billinghurst, Clark, G. Lee, et al. 2015].

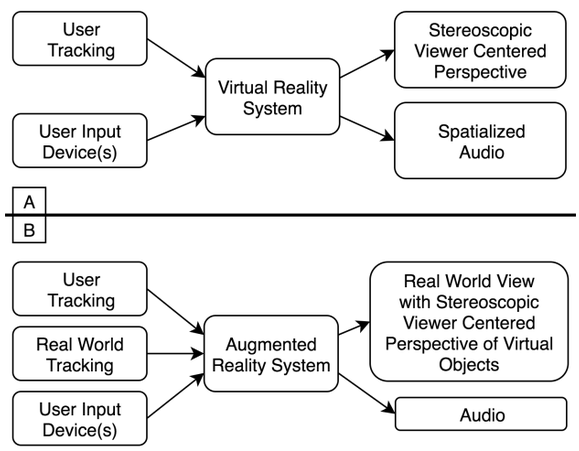

Figure 3: Components of A) Virtual Reality and B) Augmented Reality.

While XR will typically invoke thoughts of modern hardware and equipment, its roots extend before electronics. The origin of stereoscopic techniques can be tracked back to 1838, when Wheatstone built a stereoscope (Figure 4) made with mirrors at 45 degree angles to view images to the left and right demonstrating the human brain would fuse the two images and perceive them as one 3D object [Wheatstone 1852]. Sir Brewster improved upon the design in 1849 by adding lens to make the lenticular stereoscope (Figure 5) which became the first portable 3D viewing device [Brewster 1856]. William Gruber designed the successor to these devices in 1938, the View-Master, which was light, portable, largely marketed as an entertainment piece, and could easily swap between content. Even today most people recognize the View-master [Gruber

1940] (Figure 6).

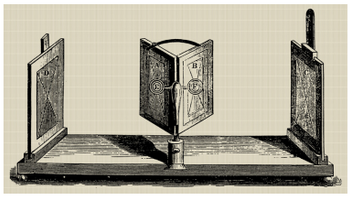

Figure 4: Charles Wheatstone's stereoscope

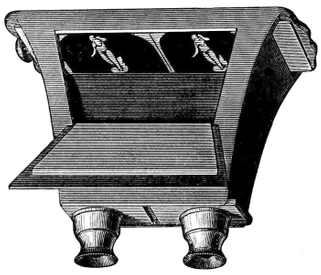

Figure 5: Sir Brewster's lenticular stereoscope

Figure 6: William Gruber's View Master

Morton Heilig, a cinematographer by trade, published "The Cinema of the Future" in 1955 [M. L. Heilig 1992], shortly after he after he built the Sensorama (Figure 7) as a prototype to show that future. It had the appearance of a booth with a seat and was the first video device which stimulated multiple senses of the viewer through vibration, sound, smell, and generation of wind [M. Heilig 2002].

Figure 7: Morton Heilig's Sensorama

The start of VR in computer research is arguably instigated by Ivan Sutherland's description of the Ultimate Display in 1965. Achieving the Ultimate Display is one of the grand challenges of VR. As Sutherland described it, "the ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed

in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked" [Ivan E. Sutherland 1965]. The fictional StarTrek Holodeck is considered to be the closest mainstream media representation of the Ultimate Display (Figure 8). Sutherland built the Sword of Damocles (Figure 9) in 1968 which is regarded as the first HMD [Ivan E Sutherland 1968]. For the next couple decades, development of XR was focused towards hardware. There simply was a lack of available means to create a display which Sutherland envisioned. But simply viewing the virtual environment wasn't enough, interaction was also necessary. The hardware to provide a view, track the user, and capture interaction lead to diverse hardware research which included building new computer peripherals like data gloves, omni-treadmills, 6dof (degree of freedom) tracking devices, and see through displays.

Figure 8: A picture of the Holodeck from StarTrek. The Holodeck is often considered the closest mainstream media representation of Ivan Sutherland's Ultimate Display. Within the Holodeck, the room produces hologram so realistic that one cannot determine the difference between holograms and reality.

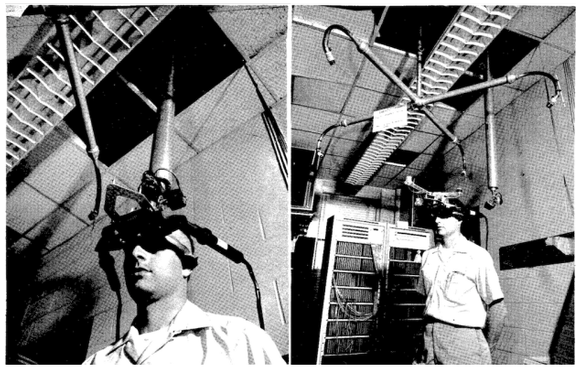

Figure 9: Ivan Sutherland's Sword of Damocles, the first HMD.

Microsoft Kinect and Playstation Eye have roots back to 1983 when Myron Krueger (Figure 10) described the VideoDesk [Myron William Krueger 1983] and later used it to build VideoPlace [Myron W Krueger, Gionfriddo, and Hinrichsen 1985]. VideoPlace took live video of the user, displayed their silhouette, and added virtual elements to the displayed silhouette view. Based on users' movement, they could interact with the added virtual elements.

Figure 10: Myron Krueger's VideoPlace.

The first wired glove, or data glove, is credited to be built by Thomas Defanti and Daniel Sandin [Defanti and Daniel Sandin 1977] in 1977. They named it the "Sayre Glove" (Figure 11) to give credit to Richard

Sayre, a member of their research group who first proposed the idea. Years later in 1982, Thomas Zimmerman filed a patent for an optical sensors which is sensitive to bending that became the basis for the data glove [Zimmerman 1982]. Their goal was to enable natural and intuitive interaction within VR.

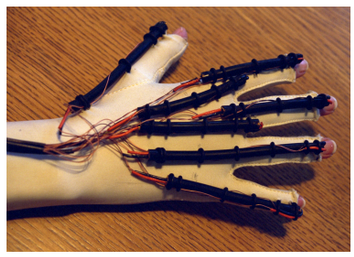

Figure 11: The Sayre Glove, the first wired glove designed for interaction with a computer.

Modern design of HMDs with lenses can be attributed to Eric Howlett (a VR pioneer) who proposed the Large Expanse, Extra Perspective (LEEP) design (Figure 12). It isn't clear exactly when his proposal was first made, some researchers place his original proposed somewhere between 1979 and 1983 [Jalkanen 2000; Frécon 2004]. Howlett himself filed a patent in 1983 [E. M. Howlett 1983] and a research document in 1990 [Eric M Howlett 1990], but did not mention when he first started work. 3D Shutter glasses were proposed by Lenny Lipton, a filmmaker by trade and pioneer of 3D films. As part of his work to improve 3D film he published a paper describing the criteria for a successful stereoscopic video system in 1984 [Lipton 1984]. Within a couple years, his company Stereographics released the CrystalEyes 3D shutter glasses (Figure 13) system which became an alternative to HMDs. Stereographics was bought out by RealD Cinema in 2005 and is the most widely used 3D system in theaters today. At the time, in order to provide stereoscopic view, HMDs required two input sources, one for each eye. Shutter glasses were important by performing this with a single video feed. This was accomplished by swapping between left and right eye views on each frame. The shutter glasses used a timing system to control when the lenses would become opaque. This ensured that the correct eye was seeing the appropriate view, while blocking the sight of the other.

Figure 12: Eric Howlett's LEEP.

Figure 13: Lenny Lipton's CrystalEyes 3D shutter glasses.

Related to HMDs are Virtual Retinal Displays (VRD). In HMDs, the viewer looks at a display with content, but a VRD projects light directly onto the retina. This lead to the technology used by Magic Leap. A review by Lin et al. [Lin et al. 2017] identified Kazuo Yoshinaka as the inventor of VRDs. The only public submission was a Japanese patent file as part of his work for Nippon Electric Co. [Yoshinaka 1986] in 1986. A different patent filed in United States by Sverdrup and Belenkii also attest to Yoshinaka being the inventor of VRD [Sverdrup and Belenkii 2010].

Jaron Lanier is credited with coining the term virtual reality in 1987. Both he and Thomas Zimmerman started the Virtual Programming Languages (VPL) Research company. The company was formed in 1989 and developed different forms of hardware to support virtual experiences. They used terms like virtual worlds, virtual cockpits, and virtual workstations to describe the different projects they worked on and Virtual Reality as the umbrella under which each of the projects related to [Myron William Krueger 1983]. This originally created an association that VR was a medium with which a user could interact with the virtual, like a telephone is to remote voice communication [Steuer 1992]. VPL created hardware pieces to experience VR like the EyePhone HDM, DataSuit [Zimmerman 1982](Figure 14A), and the DataGlove [Zimmerman and Lanier 1987].

The Binocular Omni-Oriented Monitor (BOOM) (Figure 14B), system was presented in 1989 in an attempt to solve a couple problems with HDMs. HMDs at the time simply provided a view and tracking needed a 3rd party solution. All in one solutions were not available. The BOOM system was basically two small CRT monitors in a box shaped to conform to a face and mounted on a mechanical arm. The mechanical arm made movement easy by reducing work to lift and keeping cords organized. The position of the arm provided tracking for the view [McDowall et al. 1990]. However, the system itself was bulky, had limited movement, and was reported to be awkward to adjust.

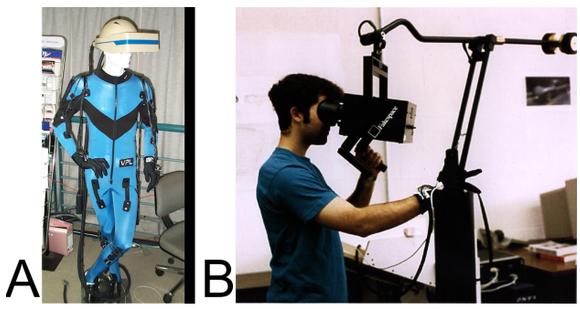

Figure 14: A) VPL's EyePhone HMD and DataSuit and B) The Binocular Omni-Oriented Monitor (BOOM).

Prior to 1992, the term augmented reality did not exist. Thomas Caudell and David Mizell, Beoing software engineers, coined the term to describe their work where they used a see-through HMD (Figure 15) to guide assembly of aircraft wiring. Visuals helped guide and identify where to lay wire, which to use, and how it was to be secured. The implementation was successful and he predicted that it would be successful in other manufacturing situations to help reduce cost and error [Caudell and Mizell 1992].

Figure 15: Thomas Caudell wearing his AR headset used to gude aircraft wiring.

The CAVE Automatic Virtual Environment (CAVE) was the most well known system prior to the introduction of Vive and Oculus. The CAVE represents the golden standard of immersive VR systems where it kick started the lineage of large fullview immersive systems users could walk into rather than look into (e.g. headsets). The first CAVE had a three wall with floor projection system (Figure 16) first built in 1991 by Cruz-Neira et al. [Cruz-Neira, D. J. Sandin, T. A. DeFanti, et al. 1992] [Cruz-Neira, D. J. Sandin, and T. A. DeFanti 1993]. Its origin was slightly different from other systems in that it was designed for the purpose of generically viewing scientific visualizations. The CAVE was developed as a Virtual Reality Theater to meet

the criteria of SIGGRAPH Showcase. They advocated for an environment for computational scientists to interactively present their research at major professional conferences in a one-to-many format.

Figure 16: View from within the CAVE.

The CAVE revealed a couple of aspects lacking in prior VR systems: the ability for others to share the experience and a mismatch between work environment. The Electronic Visualization Laboratory (EVL) utilized the CAVE technology and produced a smaller, mobile system they called ImmersaDesk [Czernuszenko et al. 1997] (Figure 17). As a fully contained, mobile VR system its goal was to mimic the typical work situation of scientists, architects, pilots, physicians, and office workers: a tabletop. The interaction was fairly unique: a user's hands with a tracked wand controller went under a 45 degree semi-transparent display. This enabled viewing of the actual hand and any superimposed virtual elements. It went through a number of revisions [Pape et al. 1999] and was later restructured as the PARIS system [Andrew Johnson et al. 2000]. Perhaps in direct competition and developed in the same year (1994) the Responsive Workbench (Figure 18) was presented by Kruger et al [W. Krueger and Froehlich 1994; Kruger et al. 1995]. They also used back projection, but from underneath to project onto the surface of the table. The Workbench provided in some cases a more intuitive usage for table based applications, but was more limited in the the variety of virtual objects that could be displayed due to viewing angle. Although the workbench style systems were designed to mimic normal working conditions these types of systems had a rather short lived popularity. Larger systems providing a user with a full view into the virtual environment found more success.

Figure 17: The ImmersaDesk.

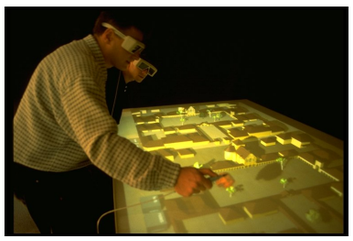

Figure 18: The Responsive Workbench.

By mid 1990s, AR was recognized as a separate and distinct research aspect to VR. But Azuma noted that a major inhibitor to AR's growth was the equipment requirements and costs which created barriers to leaving research laboratories [Azuma 1997]. Almost in response, Feiner et al. created the Touring Machine [Feiner,

MacIntyre, Höllerer, et al. 1997] (Figure 19). Prior to this, AR implementations were mostly restricted to rooms or enclosures due to lack of tracking accuracy. Using off-the-shelf hardware they created a system roughly 40 pounds in weight supported by a backpack to label buildings around Columbia University. They did acknowledge the system's tracker inaccuracy, but found that for the purpose of labeling buildings, was not a large issue. Their success inspired research into AR as a mobile platform.

Figure 19: Feiner et al.'s Touring Machine.

Magic lens refer to AR systems (usually hand-held) where a display shows camera feed edited to include virtual information. The first magic lens AR system used a palmtop configuration with video see-through, called the NaviCam (Figure 20) by Rekimoto and Nagao [Rekimoto and Nagao 1995] in 1995. But the system still required a direct connection to a graphics workstation. Wagner and Schmalstieg [D. Wagner and Schmalstieg 2003] showed in 2003 that off-the-shelf personal digital assistant (PDA) had enough power to be a self contained AR system (Figure 21). Although PDAs eventually died out, their purpose and functionality were merely transferred to the increasingly popular mobile phone scene. The release of the first iPhone in 2007 and Android phone in 2008 marked the start of AR browsers. This generation of mobile phone had widespread popularity, strong enough for AR, and easy to develop for [Billinghurst, Clark, G. Lee, et al. 2015]. AR browsers refer to mobile applications which superimpose information over the view of the mobile device's camera, then display the result on the mobile display. Unfortunately this also marked the decline of immersive AR systems. The AR browsers do not fully qualify as AR for this survey for two reasons: a stereoscopic view is not provided and device centric. Device centric meaning, both the tracking and view was centered around the device, not the user.

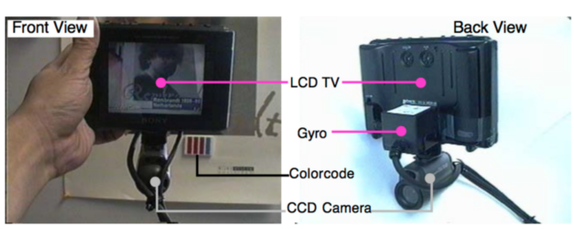

Figure 20: Rekimoto and Nagao's NaviCam.

Figure 21: Wagner and Schmalstieg's using an off the shelf PDA for augmented reality.

The arrival of autostereoscopic displays sparked hope that the burden imposed on a user from wearing equipment could be fully eliminated. Autostereoscopic displays are those that are capable of delivering a stereoscopic view without the viewer needing to wear lenses. The Sharp Corporation first offered autostereoscopic displays since the mid 1990s [Woodgate et al. 2000]. However, usage of such displays within XR systems was largely avoided. Halle identified render requirements [Halle 2005] was a major limiting factor towards adoption. In order for autostereoscopic devices to provide a viewer centered stereoscopic perspective, the rendered view needed to include each possible perspective supported by the display. These additional perspectives were critical for maintaining immersion when a viewer moved their head, but consequently resulted in a loss of resolution in a single perspective. Furthermore, the different displays (even those from the same companY) had vastly different and unique render requirements. In 2005, Sandin et al. created Varrier, the first VR system providing an immersive stereoscopic viewer centered perspective without requiring the user to wear tracking equipment or glasses (e.g. polarized lenses or shutter glasses). Varrier (Figure 22) overcame the resolution limit by tiling displays together and supporting a reduced amount of perspectives to ensure virtual environment rendering could be maintained in real time. Tracking was done through a camera array view processed using artificial neural networks trained to identify faces. Despite the potential shown by Varrier, no other systems like it have been seen since.

Figure 22: The Varrier system, comprised of 35 autostereoscopic displays to provide a stereoscopic viewer centered perspective without headgear.

Interaction with the virtual environment is necessary for immersion. The pursuit of natural hand usage has been continuously sought throughout XR history. Measuring hand and finger position has traditionally needed gloves. However, the Kinect's release in 2011 was significant for being a viable alternative to hand and body tracking [Frati and Prattichizzo 2011]. Being an RGB-depth camera, available commercially, and providing a mature software development kit made the Kinect ideal for augmented mirror systems (Figure 23). Augmented mirror systems are rooted back to Kruger's VideoPlace [Myron W Krueger, Gionfriddo, and Hinrichsen 1985]. Its usage for augmented mirror systems was initially fueled by Microsoft's own advertisements of Holoflector[1] and game usage. Shortly after release others also considered the Kinect to have great potential [F. Anderson et al. 2013; Blum et al. 2012]. Usage ultimately had to stop when the Kinect was officially discontinued[2][3]. Around the same time of Kinect was Leap Motion(Figure 24). Although founded in 2010, they didn't produce a product until 2013. Its initial release was not accurate or consistent enough as an alternative to professional tracking [Weichert et al. 2013]. It wasn't until 2017, when an updated model was considered accurate enough to be usable for creating XR applications. It potential lay in the low cost and small form factor which presented a variety of integration opportunities such as being mounted on the front of HMDs [Cordeil et al. 2017].

The CAVE's official successor, CAVE2 (Figure 25), was presented in 2013 [Febretti, Nishimoto, Thigpen, et al. 2013]. Boasting a size of roughly 3,100 cu. ft. with a resolution of 36 Megapixels, the truly walk-in

[1] https://www.microsoft.com/en-us/research/video/holoflector/
[2] https://www.polygon.com/2017/10/25/16543192/kinect-discontinued-microsoft-announcement
[3] https://gizmodo.com/kinect-one-of-microsofts-most-revolutionary-ideas-is-1819845998

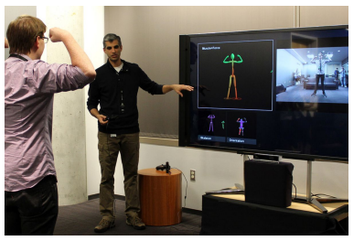

Figure 23: Microsoft Kinect.

Figure 24: Leap Motion hand tracker.

nature of the system could comfortably fit a small class inside. One of the driving goals was the ability to support full group immersion within the data they were exploring. A custom micropolarizer was developed for the system to improve off-axis performance. Micropolarizers are thin film sheets applied to displays which enable passive stereo by correctly filtering views intended for the left and right eye when the viewer wears polarized glasses.

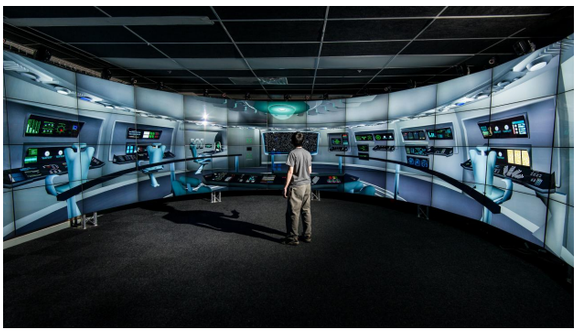

Figure 25: The CAVE2, direct descendent of the CAVE.

In an effort to popularize AR again, a developer version of Google Glass (Figure 26) began in 2013. Although there was much promise an expectation for making impacts in medicine through topics like education [Kamphuis et al. 2014] and surgery [Muensterer et al. 2014], public opinion very quickly waned by the end of its public release in 2014 with the fear or perhaps disgust of its capability for recording.

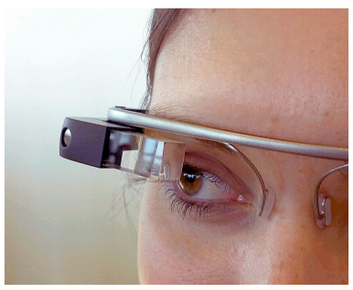

Figure 26: Google Glass.

The battle of the modern HMDs arguably started around 2013 when Oculus (Figure 27) first shipped its development kit. HTC Vive (Figure 28) and Meta development kits were shipped in 2014. Both Oculus and Vive had official releases in 2016, but for the Hololens (Figure 29) 2016 marked its first shipping of

development kits. A competing HMD company, Meta, fell behind and in 2019 declared itself insolvent. Both the Oculus and Vive now play major roles in XR research. For example, ACM's Symposium on Virtual Reality Software and Technology (VRST) submissions that declared using an HMD, 78% of them explicitly stated using an Oculus or Vive.

Figure 27: Oculus Rift.

Figure 28: HTC Vive.

Figure 29: Hololens.

Outside of companies, the Destiny-class CyberCANOE (Figure 30) was also released in 2016 by University of Hawaii's LAVA. A descendent of the CAVE2 and boasting an even higher resolution of 128 Megapixel stereo resolution (3.5x more than CAVE2). The resolution was a significant land mark. Under normal viewing conditions by someone with 20/20 vision, the pixels could not be differentiated. In addition, Destiny was the first system to use software to correct for off axis viewing instead of what has been a traditionally hardware correction [Kawano et al. 2017].

Figure 30: The Destiny-class CyberCANOE.

4 Attributes For Success

This section will cover the the two classes of attributes which XR researchers strive to meet: grand challenge and software success attributes. They have been identified based off the set of papers selected for this review. The grand challenge attributes described are those which can be applied across the XR spectrum, not necessarily unique to one format. These attributes are a way to measure the progress towards creating

an ideal XR system. Complementary to the grand challenge attributes are software success attributes. If the software success attributes are low, regardless of how well it may fulfill grand challenge attributes, the software will have difficulty attracting users and maintaining support.

4.1 Grand Challenges Of Immersion

Figure 31: The Holodeck from StarTrek, a room in which holograms are given mass and cannot be visually differentiated from reality.

The grand challenge of VR started as the goal of reaching Sutherland's ultimate display. This section will cover the hardware and software attributes researchers strive for, in order to obtain an idealized XR system. Within modern media, Sutherland's ultimate display would be likened closest to StarTrek's Holodeck (Figure 31). The Holodeck in StarTrek is depicted as a closed off room where objects and people can be simulated with the fictional "holomatter". Using holomatter, holograms appear solid to the touch and are able to emulate nearly any material. To recall Sutherland's description of the ultimate display, it would "...be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal..." [Ivan E. Sutherland 1965]. While AR and VR share the same roots, they have different direction of involvement. AR has the intention of bringing the virtual environment into our world to be sensed. While VR has the intention of transporting the user's senses into a virtual environment. It would now be more accurate to say, AR's ultimate goal is the Holodeck; while VR's ultimate goal is closer to the experience of the Matrix. Characters in the Matrix experience a virtual environment by connecting their consciousness with a computer. Rather than actually moving or touching with one's body, the computer simulates the view and feeling of the environment by directly interfacing with the user's brain. Reaching either of these ultimate goals will take significant effort. Brain interfaces to recreate the Matrix and matter fabricators for a Holodeck still remain as fictional concepts. Based on the set of papers selected for this review, there are overlapping attribute goals between AR and VR which can be pursued for the sake of achieving end goal implementations:

- **Persistence** - XR environments tend to reset by nature of applications. In some cases this is done intentionally, but as a "reality", one major aspect necessary for mimicking our own is the user's ability to make lasting changes and effects.
- **Sensory immersion** - Of our five senses, XR systems have made the most effort for two: sight and sound. For a truly immersive experience all five senses will need to be simulated.
- **Resolution** - As a sense, sight is so important that there are two related grand challenges: resolution and visual realism. The grand challenge of resolution is being able to create a system where individual pixels are imperceptible to the human eye. With current software and hardware it is possible to achieve such an immersive system under conditions. But, said system (e.g. tiled displays/projection) will still

have its pixels identifiable upon closer inspection. Other systems like HMDs still have more progress to go.

- **Visual Realism** - The ultimate goal is life-like graphics indistinguishable between reality. From another perspective, the requirement may not be the single goal of life-like realism, instead this could be understood as a moving target: graphics that are sufficient enough for immersion. The problem is subjectivity and expectation. As a direct example, imagine the first movie or television show you saw. How real did it look? How was the resolution quality? Special effects? Imagine watching it today on a 4k display. Would it be pixelated? How would it compare to a AAA movie? Probably you would watch it with fond memories, but ultimately no longer look as immersive as before.
- **Feedback** - In many ways directly related to sensory immersion. However, there are some key differences in terms of practicality. The ultimate display described a couple of unusual aspects: "A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal" [Ivan E. Sutherland 1965]. The ultimate form wouldn't just convey the feeling of sitting in a chair. It would allow the user to take a sitting position or truly convince the user they are sitting in a chair. Related to current research, this involves addressing simulator sickness, often attributed to the body rejecting the inconsistencies between senses [Kolasinski 1995].
- **User tracking** - Accurate tracking down to the millimeter and with millisecond delay is possible with current technology. However this requires a controlled environment looking for known objects within that space. Furthermore, tracking is typically limited to the head and hands, beyond that is uncommon. While possible to track down to the fingers without impediments (e.g. Kinect or Leap Motion) such systems are also uncommon as well as tracking the rest of the body (e.g. elbows, shoulders, chest, waist, legs feet).
- **World tracking** - Tracking the real world for incorporation into the virtual environment has traditionally been a desire of AR. Only when the system knows the real world, can it accurately add the virtual to it. Markerless tracking, enabling the virtual to interact with the real world, and usage outdoors at unknown locations are just some of the goals. Furthermore, knowing about the real world benefits VR by allowing it incorporate the real into the virtual (e.g. aligning a virtual wall with a real wall). Most VR systems currently use headsets which blocks the users view of the real for the sake of immersion. Knowing the environment also benefits VR through safety (e.g. proximity warnings) and maintaining immersion through techniques like redirected walking. Redirected walking is a method used by VR to maintain an illusion that users are walking through a much larger environment than their physical space allows for. This is typically done by making small gradual changes to the view, causing the user to change direction without them being aware.
- **Simulation** - This is perhaps the most difficult and a continuously moving target. If ultimate immersion is perfectly emulating real life, the more detailed and accurate the environment, the more intensive both computation and storage requirements become. A fully manipulatable environment and realistic physics calculations are goals shared by other focuses of research. Supercomputers would become necessary for XR, but as other fields have shown, there would still be a limited to what can be done in real time. This may end up a subjective achievement based on compromise.
- **Curtail Encumbrance** - There are a couple different interpretations that can be applied to this. First is the physical burden on the user. When requiring a user to wear tracking or viewing equipment, the

devices have weight, comfort, and correct usage that influence the user experience. If a user is unable to support the weight, they logically cannot use the system. Even extended interaction with just one's hand can cause fatigue, referred to as gorilla arm syndrome [Boring, Jurmu, and Butz 2009]. Mental fatigue comes from the user needing to put conscious effort into using the system. This commonly is caused by needing to remember how to interact with the system. Physical encumbrance can be reduced by removing equipment needed (e.g. HMD) or using equipment with such a small footprint that the user doesn't notice. Mental encumbrance can be reduced through implementation of natural interaction methods which do not require conscious management by the user.

4.2 Successful Software

The previously identified grand challenge attributes identify goals necessary to build fantastical systems. However, such a fantastical system does not automatically make it a popular or well received one. As researchers strive towards grand challenge goals, they have also identified an additional set of attributes which they use to evaluate their handiwork. Software success attributes are a means to evaluate adoption and longevity. In many cases it can be observed that the software attributes had a larger influence over success than the contribution to grand challenge attributes. It is unfortunate that the software success attributes are also highly subjective. The following are success attributes commonly discussed and brought to attention both when authors present their own contributions and discuss others' software:

- **Performance** - General usability for nearly any system (not just XR) depends on running in realtime, with minimal lag, at high frame rates. For XR, this greatly influences the ability to maintain immersion. The human eye can perceive differences in a time as small as 2-5ms to detect motion depending on luminance [Scharnowski, Hermens, and Herzog 2007].
- **Maintaining Truth** - XR systems often involve multiple computers. Be it a cluster of computers to drive one system, client-server architectures, or supporting multiple users at once. The means to ensure all components have the correct and necessary set of data at a given time is crucial for operation. Rephrased, maintaining truth is necessary for a system comprised of more than one computer to ensure that across each, the view and environment is synchronized.
- **Flexibility and Extensibility** - Flexibility refers to being able to use the software for multiple purposes or in different ways. Extensibility in software engineering refers to being able to add new capabilities. These two attributes are listed together because of how related they are in supporting the creation of a designer's idealized environment.
- **Ease of use** - All authors of frameworks claim their system is in some way easier to use than others. In some cases, there are roundabout descriptions implying that due to a particular feature, aspect, or optimization a developer will be able to create a higher performance system or application, but it may take a bit of getting used to.

5 Research Trends of Software

This section will cover research trends in XR software and how they relate to the successful software attributes and grand challenges. In general, software has taken a slow shift in priority from from performance; to truth maintenance; then supporting flexibility and extensibility; and finally ease of use.

The papers covered are primarily focused on those which described software frameworks, libraries, and packages for building XR systems. Qualification for consideration meant supporting requirements described

earlier for VR or AR and make available their software for others to use. Papers that reported upon the existence of the software without making them available are not considered. To be clear, there has been a significantly larger amount of software publications than is covered in this review. However, most were not freely distributed. Another key aspect of qualification is the ability to support building an environment with a customizable logic update. This requirement will excludes all "as-is" applications and those which act as data loaders, viewers, and editors. There are a couple exceptions which will be pointed out later due to the way in which the features became a commonly implemented or integral aspect. With these restrictions it may seem that there is not a lot of software VR research, but specifically, contributions which are also code-wise distributed are small by comparison to those not distributed or software focused (e.g. algorithm reporting, optimization, user studies, case studies, best practices, and user interaction methods). A timeline can be see for all software considered for trends in Figure 32.

5.1 Performance

XR systems that could not maintain updates in realtime were doomed to fail. The early software contributions were all focused towards getting systems to run in realtime while providing the requirements necessary to for XR. The performance challenges faced by early software can be seen in their: difficulties of proving initial support, maintaining frame rates, influence of Silicon Graphics workstations, the CAVE's lasting influence, and the emergence of augmented reality systems.

5.1.1 Difficulties of providing initial support

In the beginning, making software to generically support XR systems was non-existent. As was seen in hardware history, prior to the 1990s was largely a development period to create hardware capable of supporting VR. Most applications were bespoke solutions that were research oriented and not easily transferable outside of a laboratory due to the hardware restrictions. RB2 [Scharnowski, Hermens, and Herzog 2007] is commonly recognized in literature as the first distributed software framework. In actuality Blanchard et al. (authors of RB2) were part of VPL a company that sold VR equipment, but found many of their users were having difficulty building environments with it. At the time of publication they weren't ready for distribution and remarked that it would be available shortly. RB2 required VPL equipment and Polhemus trackers, but significantly lowered the amount of work to have a working VR system.

Shaw et al., authors of the Minimal Reality Toolkit (MRToolkit), commented that it wasn't until the 1990's when a single workstation was strong enough to provide a stereoscopic view [Shaw, Liang, et al. 1992; Shaw, Green, et al. 1993]. For comparison, RB2 require two SGI workstations in order to produce the stereoscopic view, but automated and ensured the view was correctly synchronized. Prior to RB2, developers had to manually synchronize the view between two systems themselves or only provide a monoscopic view.

In the early 1990's there were mainly three software options available and each had fairly large restrictions. RB2 required Silicon Graphics workstations and VPL equipment, but significantly lowered the barrier of entry as result. MRToolkit requires users to handle hardware drivers and integration themselves, but provided an easier means to draw graphics and provided some automated thread optimization. The final was NPSNET [Mackey 1991; M. J. Zyda, D. R. Pratt, Monahan, et al. 1992; M. J. Zyda, Falby, et al. 1993; M. J. Zyda, D. R. Pratt, Osborne, et al. 1993; M. Zyda, D. Pratt, et al. 1993] which was the most restrictive. Originally it was designed to interface with SIMNET [Thorpe et al. 1987], a research project towards creating large scale interactive simulators for combat. This made NPSNET a fairly monolithic piece of software and came with a lot of extras not used by the average user. While it may appear that each of these were focused

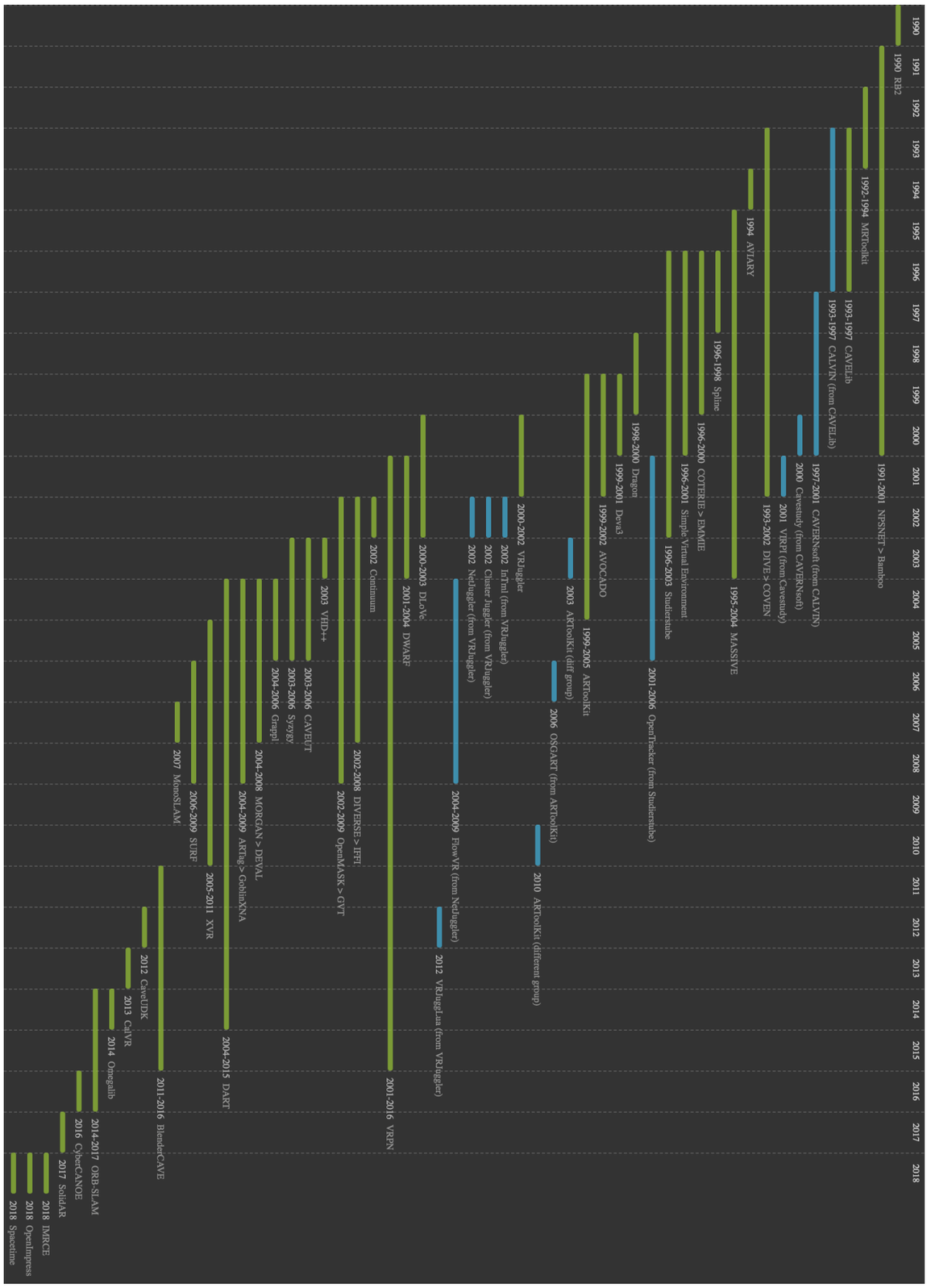

Figure 32: Timelines of all software considered for trends in this review. The timelines are based on publications describing the implementation or improvement to the software. For example, actual usage of the software is not portrayed. Even if the software was used in a study later on, unless it explicitly stated code improvements or changes and made available, those publications were not considered for the timelines.

towards ease of use, they were actually designed to ensure a minimum level of frame rates which will be discussed next.

5.1.2 Maintaining frame rates

In modern times 60 frames per second (fps) is a fairly common expectation, but for the 1990s "acceptable" and "good" values were much lower. It is interesting to note that back in 1990 RB2 reported "interactive rates of 10Hz or greater". NPSNET was satisfied with sustaining 10fps [M. J. Zyda, D. R. Pratt, Monahan,

et al. 1992]. MRToolkit aimed for 20fps, while accepting drops to 10fps [Shaw, Green, et al. 1993]. 1993's CAVELib prided itself with 30fps [Cruz-Neira, D. J. Sandin, and T. A. DeFanti 1993]. Nearly all of the early frameworks had some emphasis on their assistance with maintaining FPS through environment or data management. They placed great value on the performance of a system to support resolution, visual realism, simulation, and world persistence.

From a software perspective there are two major cycles for consideration when attempting to maintain higher frame rates: render updates and application updates. Render updates involve computations for displaying the view of the virtual world. The time necessary for rendering is directly related to how complex the environment is. For a 3D environment, one of the contributing factors is geometric detail. The more polygons in a model, the more computations necessary. There are many other factors like texturing, shadows, and perspective effects to name a few. Render updates operate on the virtual environment in a specific state. Environment updates are typically not allowed while rendering the view. Application updates involves all aspects needed to get to the state being rendered: update logic, data retrieval, data sending, and setting values of the environment. Commonly, after a system update cycle completes, a visual update is initiated.

While a reasonable assumption is that a geometrically simple virtual environment would naturally achieve higher frame rates, this is not always the case. A complex application update can reduce fps even with primitive visuals. For example, between two cases that have the same virtual environment, the system which runs a complex physics engine will require more time to update the application state than if a physics engine wasn't used. Many of the early frameworks approached the fps problem by attempting to reduce both render and application time by minimizing virtual world resources. NPSNET [Mackey 1991; M. J. Zyda, D. R. Pratt, Monahan, et al. 1992; M. J. Zyda, Falby, et al. 1993; M. J. Zyda, D. R. Pratt, Osborne, et al. 1993; M. Zyda, D. Pratt, et al. 1993; Macedonia, M. J. Zyda, et al. 1994] was designed for military simulations and scenarios to evaluate battlefield conditions and tactics. This resulted in virtual environments that needed to span tens or even hundreds of kilometers. Their revisions into the mid 1990's were each oriented towards optimization. They started with requiring a specialized terrain format and storage system. This meant the environment had to be pre-processed by their system before an application could run. Additional restrictions also included limiting virtual entities and entity details. The preprocessing generated terrain tiles in a specially formatted database for more efficient retrieval. Then, they shifted towards focusing on better management algorithms for the virtual environment. Rather than load everything into the environment, they swapped out visuals based on where a user could see, this allowed higher detail models and terrain without losing performance (Figure 33).

MRToolkit [Shaw, Liang, et al. 1992; Shaw, Liang, et al. 1992] took a fairly different approach. Its API exposed a development requirement such that one thread was dedicated to view rendering while another thread handled system updates. While rendering a view, the environment could not be changed, any updates from the system thread were queued for after the render was complete. Through this manner, render time was directly related to the computations necessary to generate the view. Logically, this placed the burden on the developer to ensure the render requirements of the environment could sustain their desired fps.

Barrus's SPLINE [Barrus, Waters, and D. B. Anderson 1996; Waters et al. 1997] was perhaps short lived; it came out after both DIVE and MASSIVE, but was discontinued first among the three. SPLINE was designed to support worlds representing thousands of kilometers in size. However such worlds could not be easily stored in memory. The concept of locales was implemented to ensure this was possible, but ended up limiting how large the visible environment could be. The idea was that within a world, users would not be able to see everything from a given spot and travel would take time. The world was split into locales which

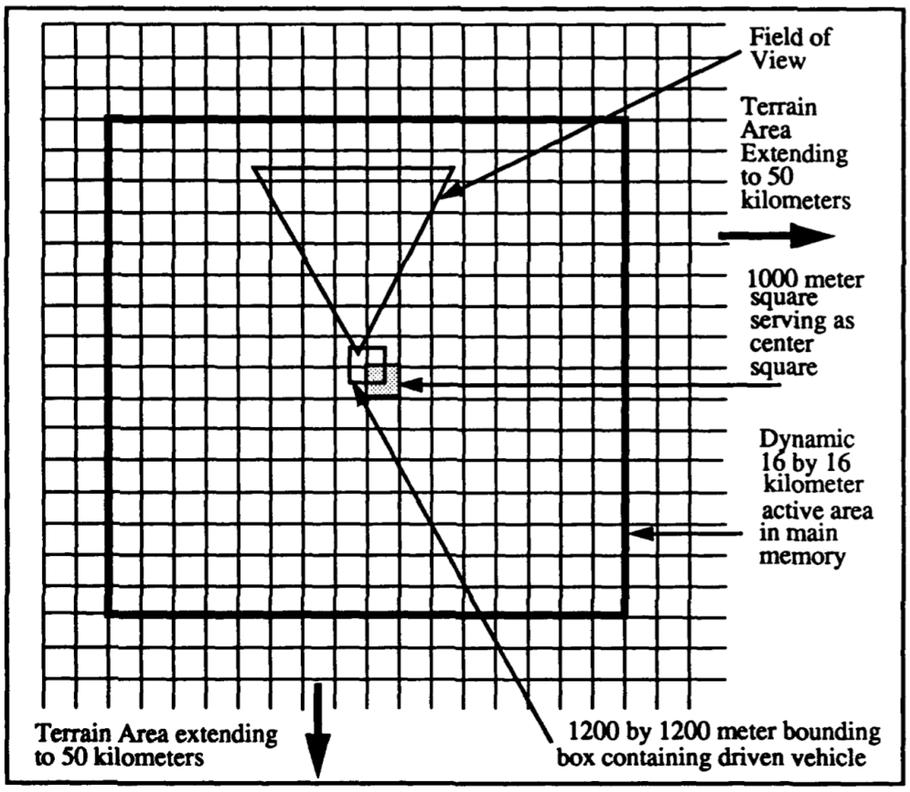

Figure 33: NPSNET required preprocessing the terrain which would be stored in a particular format as tiles. The viewer's located and orientation would determine which tiles were loaded into memory.

represented complete and segregated areas; including hardware and memory separation. Users within would fully load a locale, but as consequence became unaware of anything beyond. This was particularly effective for buildings but lost efficiency for open areas.

Carlsson and Hagsand's DIVE [Carlsson and Hagsand 1993; Hagsand 1996] was a framework oriented towards traversing persisting worlds. It was designed to support large environments of roughly town and eventually city scales. Because of this they identified the need to support more efficient rendering, which was provided in the form of auras. User clients moving through the virtual environment would only load as much of the environment as was within their aura. Their initial version of the aura was basic and sometimes created obvious "drop zones" in the world where beyond that point nothing was visible. Their aura implementation slowly shifted towards being more dynamic in nature. In later implementations auras became affected by buildings of interest, user's orientation within the environment, and area zoning (a hybrid of locales that was implemented in SPLINE).

As the aura system became more effective DIVE could support more complex and detailed environments. However, the increase in world details had to be addressed in order to maintain frame rates. Towards the late 1990s, development was handed off and some fairly extensive rewrites to the network system and graphics handling were implemented. In 2001, DIVE was rebranded as COVEN [Frécon, G. Smith, et al. 2001; Steed, Mortensen, and Frécon 2001] A number of changes to the aura system were added which included incremental rendering, level of detail, and bill boarding. Rather than one set aura size, it was further divided into distances from the user. The closer zones would trigger incremental rendering first, ensuring that objects in close proximity to the user would render before objects further away. Those closer objects also were allowed to render in higher levels of detail. This was only possible if the original model supported multiple levels of detail. Billboarding was reserved for objects at the edges of the aura. Billboarding is when one render view is generated on virtual objects, but that view then replaces the virtual objects. This is like taking a picture of a scene, printing the picture, placing the picture to cover such that the actual

objects could be removed. At a distance viewers will have a difficult time detecting they are looking at a 2D replacement.

Greenhalgh and Benford described their MASSIVE framework as being designed to support virtual reality teleconferencing [Greenhalgh and Steven Benford 1995; Greenhalgh and Steve Benford 1995; Steve Benford, Greenhalgh, and Lloyd 1997; Steve Benford and Greenhalgh 1997]. With the expectation of multiple users meeting, voice support was build into the framework. DIVE was used as inspiration for their own modified version of auras (Figure 34). Although MASSIVE's environment was smaller on average (usually building sized), they needed a means to efficiently reduce the render and data transfer load. As a teleconferencing system, they allowed for fairly detailed avatars (for the time) but also had a significant amount of data transfer for audio. The aura went through a number of changes: dynamic sizing based off of groups, based off of virtual objects auras instead of individuals, and directed cone based off of user facing. In the end they used a modified adaptation of locales (SPLINE) with auras (DIVE). Rather than users having auras, the environment rooms or objects determined locales. Rooms ended up being the standard unit of a locale with large rooms being further subdivided by objects(e.g. tables users can gather around). Locales were used used for data prioritization. Certain forms of locales (e.g. rooms or buildings) would case data restrictions (i.e. users could not see outside of their room). Whereas, artifact based locales created areas of data priority (i.e. users around a table would have their positions updated between each other at a higher priority compared to users at a different table).

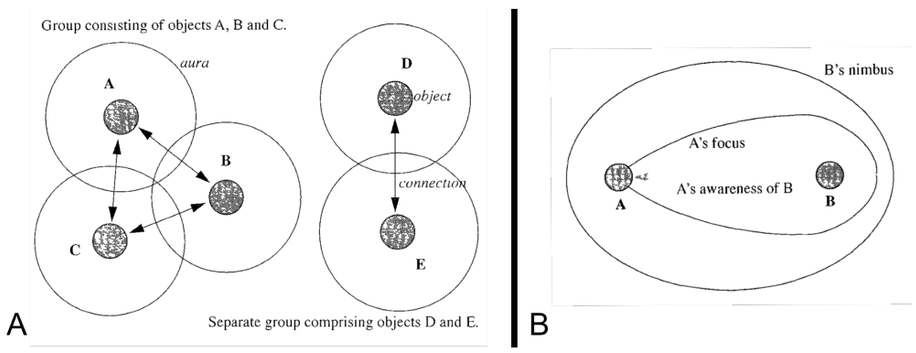

Figure 34: MASSIVE used auras based around users to determine how to reduce data transfer. A) Shows the standard aura which is affected by groups. B) Shows an alternative which was influeced by user gaze.

The importance of these optimizations slowly became less important for a developer to worry about. Graphics libraries and render engines started including such features over the years. Some of the basic formats of aura, level of detail, and view based rendering are commonly supported. Park [S. Park et al. 2001] observed that SPLINE, DIVE, and MASSIVE were each designed to support large environments similar to massive multiplayer online (MMO) servers. The popularity of MMOs drove the need for improvements and with the many similarities, commercial solutions for MMOs became beneficial for VR systems supporting large worlds as well. This is directly related to the game engine trend described later.

Another major reason virtual environment management was critical during the 1990's was availability of RAM and hard disk space. In 1990, it cost about $100 dollars per MB (megabyte) of RAM and about $1,000 per GB (gigabyte) of hard disk space. By 2000, the cost had dropped about two orders of magnitude. What was once $100 per MB of RAM became dollars whereas hard disk space dropped to tens of dollars per GB. But it wasn't just the cost. RAM simply was not available in GB quantities as it is nowadays; similarities applied to hard drives. SPLINE, DIVE, MASSIVE, and many other systems like NPSNET, CALVIN [Leigh, A. E. Johnson, Vasilakis, et al. 1996], AVIARY [Snowdon and West 1994], Dragon [Durbin et al. 1998], and Deva3 [S. R. Pettifer 1999; S. Pettifer et al. 2000] to name a few all expected a data storage separate from the render computer to store the environment because personal computers were not capable of storing gigabytes

of data. Much of the focus on performance has to do with the delay involved with retrieving the data to support high resolution graphics and realism (for the time).

5.1.3 Influence of Silicon Graphics workstations

Silicon Graphics, Inc. (SGI) was considered the de facto graphics workstation producer during the 1990s. Simply put there was practically no other competitor to their graphics workstation in terms of capabilities. But it was the SGI Performer libraries' influence which had a lasting effect on the industry as a whole. Scene graph systems first appeared in the late 1980s[4][5], but was arguably popularized by SGI due to having a large, devoted user base and in particular distributed a free version of Performer with their workstations. Performer introduced a number of aspects which improved performance through automated optimization of threads.

SGI is a prime example of how hardware influenced software. Many VR frameworks required SGI software (e.g. Performer or OpenInventor) for handling graphics rendering [M. J. Zyda, Falby, et al. 1993; M. J. Zyda, D. R. Pratt, Osborne, et al. 1993; M. Zyda, D. Pratt, et al. 1993; Macedonia, M. J. Zyda, et al. 1994; Macedonia, D. P. Brutzman, et al. 1995; Leigh, A. E. Johnson, T. DeFanti, et al. 1995; Leigh and A. E. Johnson 1996a; Greenhalgh and Steven Benford 1995; Greenhalgh and Steve Benford 1995; MacIntyre and Feiner 1996; Schmalstieg, A. Fuhrmann, Szalavari, et al. 1996; Szalavári et al. 1998; Schmalstieg, Encarnação, and Szalavári 1999; Schmalstieg, A. Fuhrmann, and Hesina 2000; A. L. Fuhrmann et al. 2001; Schmalstieg, A. Fuhrmann, Hesina, et al. 2002; Tramberend 1999; Tramberend 2001; Kelso, Arsenault, et al. 2002; Kelso, Satterfield, et al. 2003; Ponder et al. 2003]. Even when it wasn't required, many reported using SGI machines due to their powerful graphics capabilities [Blanchard et al. 1990; Mackey 1991; M. J. Zyda, D. R. Pratt, Monahan, et al. 1992; Shaw, Liang, et al. 1992; Shaw, Green, et al. 1993; M. Zyda, D. Brutzman, et al. 1997; Cruz-Neira, D. J. Sandin, T. A. DeFanti, et al. 1992; Cruz-Neira, D. J. Sandin, and T. A. DeFanti 1993; Pape 1996; Leigh, A. E. Johnson, Vasilakis, et al. 1996; Leigh and A. E. Johnson 1996b; Barrus, Waters, and D. B. Anderson 1996; Waters et al. 1997; Deligiannidis 2000]. Frameworks that didn't require SGI software quickly identified the importance of the scene graph system and implemented some equivalents of it through OpenGL or used other graphics libraries to gain the same capabilities. [Shaw, Liang, et al. 1992; Shaw, Green, et al. 1993; Snowdon and West 1994; Carlsson and Hagsand 1993; Hagsand 1996; S. R. Pettifer 1999; S. Pettifer et al. 2000; Hubbold et al. 1999; Bierbaum, Just, Hartling, and Cruz-Neira 2000; Bierbaum, Just, Hartling, Meinert, et al. 2001; Olson 2002; Deligiannidis and Jacob 2002; Bauer, Bruegge, et al. 2001; Reicher, MacWilliams, and Bruegge 2003; Margery et al. 2002]. By the mid 2000 the importance of scene graph was clear. Nearly all frameworks provided scene graph interaction often by a commercial library.

5.1.4 The Cave Automatic Virtual Environment's lasting influence

The importance of the CAVE (cave automatic virtual environment) to the VR community cannot be stressed enough. Cruz-Neira's introduction of the CAVE in 1992 [Cruz-Neira, D. J. Sandin, T. A. DeFanti, et al. 1992] was a landmark event that left a lasting inspiration still present to this day. In particular, this was an instance of software influencing hardware. The CAVE was component-wise, comprised of equipment that already existed within the market. Cruz-Neira's software combined the pieces to provide a seamless view which made the CAVE possible and provided a leap in multiple grand challenge attributes: resolution, visual realism, feedback, user tracking, and reducing user burden. The key aspect to making this successful was ensuring the system was able to keep up performance wise, making clusters necessary to drive the

[4] http://deelip.com/a-long-conversation-with-ron-fritz
[5] https://blog.novedge.com/2007/03/ron_fritz_is_a_.html

system. From the 1990s a majority of frameworks attempted to shift away from clustering to provide a single viewer centered perspective. However, the CAVE lineage kept this as a key architecture and performance requirement to support increasingly larger resolutions and visual realism. Figure 35 shows some of the CAVE descendents.

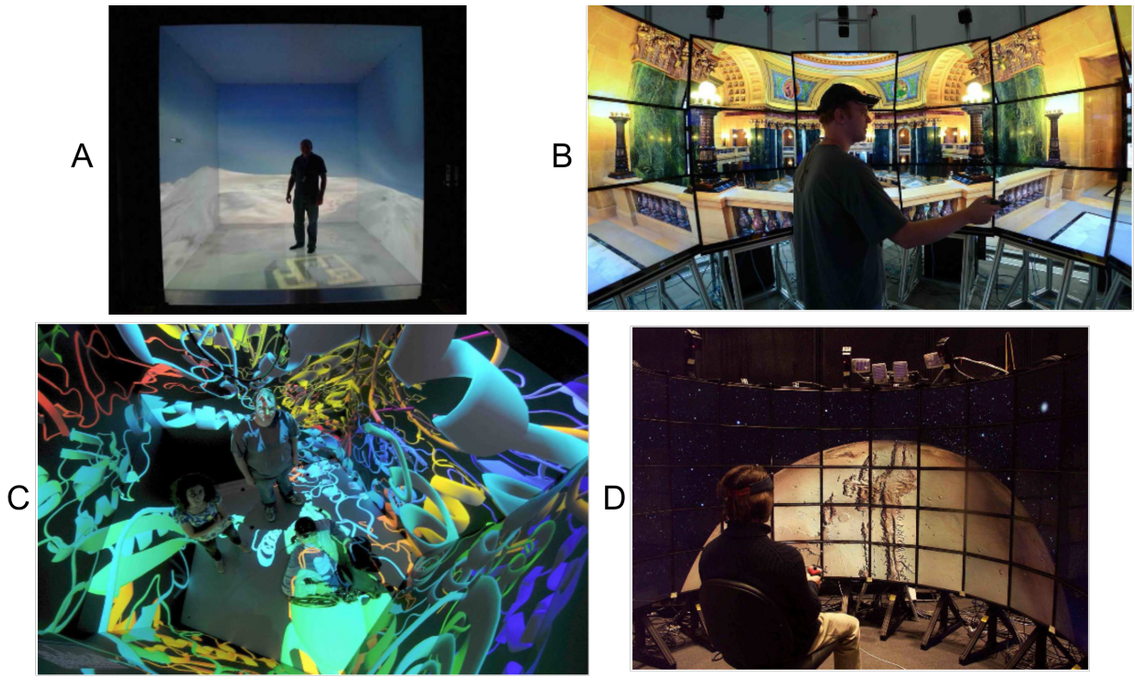

Figure 35: The CAVE spawned an entire lineage of systems. Some mimicked the original as in in the Cornea(A). While others investigated alternative shapes as seen with the NexCAVE(B), StarCAVE(C), and Varrier(D). Nearly all modern systems have moved away from projectors and now use

Prior to the CAVE, VR systems were comprised of a single viewing devices (e.g. HMD, BOOM, etc) which supported only one viewer per device. Being comprised of four viewing area: three walls and floor projection it was a milestone in size and scale. Due to the size and design, it was also the first viewing device to comfortably support multiple people at once. A major contributing factor was reducing the encumbrance on the user; tracked shutter glasses and a controller were the only carrying requirements. Although the system was designed for a single user, the size and usage of shutter glasses allowed multiple people to share the same perspective in VR space [Cruz-Neira, D. J. Sandin, and T. A. DeFanti 1993]. This was significant for illuminating VR's future as a shared collaborative experience.

The CAVE left a lasting impression on the VR community. The software which drove the CAVE, called CAVELib, was initially freely distributed and as result had a number of improvements [Cruz-Neira, D. J. Sandin, and T. A. DeFanti 1993; Pape 1996] and direct descendents including CALVIN [Leigh, A. E. Johnson, T. DeFanti, et al. 1995; Leigh and A. E. Johnson 1996a; Leigh, A. E. Johnson, Vasilakis, et al. 1996; Leigh and A. E. Johnson 1996b], CAVERNsoft [Leigh, Andrew Johnson, and T. A. DeFanti 1997; K. S. Park et al. 2000], CAVEStudy [Renambot et al. 2000], and VIRPI [Germans et al. 2001]. The presence of the CAVE lead to the genesis of an entire lineage of VR systems. The descendents are sometimes referred to as CAVE-likes instead of CAVE, to denote that they are not the original.

Initially, CAVE-likes were able to use CAVELib in order to drive their hardware. But creating software alternatives for driving CAVEs started to become fairly common around the late 1990s and include AVOCADO [Tramberend 1999; Tramberend 2001], CalVR [Schulze et al. 2013], CAVEUT [Jacobson 2003], DIVERSE [Kelso, Arsenault, et al. 2002; Kelso, Satterfield, et al. 2003], DIVE [Steed, Mortensen, and Frécon 2001], FlowVR [Allard, Gouranton, Lecointre, Limet, et al. 2004], Net Juggler [Allard, Gouranton, Lecointre, Melin, et al. 2002], VRJuggler [Bierbaum, Just, Hartling, and Cruz-Neira 2000; Bierbaum, Just,

Hartling, Meinert, et al. 2001], and XVR [Carrozzino et al. 2005]. There are various reasons other groups started development on their own software to drive CAVEs, however, a major cause points to Mechdyne Corporation which by 1998 had licensed the CAVE which included the rights to CAVELib [6]. Since then, access to CAVELib was only available after purchasing a CAVE from Mechdyne. A major aspect supporting this reason is the authors of VRJuggler [Bierbaum, Just, Hartling, and Cruz-Neira 2000; Bierbaum, Just, Hartling, Meinert, et al. 2001] include Cruz-Neira whom did work on the original CAVE and CAVELib.

Throughout the years, CAVEs have been a major part of VR history. Some effects include inciting a continual system improvement competition and variations oriented towards improving immersion (attributes: performance, resolution, and visual realism). As a starting point, further information can be found in a CAVE review by DeFanti et al. [T. A. DeFanti et al. 2011]. The creation of software supporting CAVE systems has continued and the most recent publication on a CAVE framework is by Kawano et al. in 2017 [Kawano et al. 2017] for their creation, DESTINY. However the most recently built CAVE is Calit2's SunCAVE [7]. SunCAVE's construction completed in late 2017 and is currently under consideration for integration into the National Science Foundation's CHASE-CI project [8] [Altintas et al. 2019].

5.1.5 Emergence of augmented reality systems

Augmented reality (AR) is a derivative of virtual reality and until the early 1990s was considered just a different way to implement VR. A brief description of AR's history will be important towards showing how AR split away from VR, and yet between the two there is still a form of symbiosis where advances in one can translate to the other. Through the view of publications, AR came about as a subfocus of VR with origins pointing back to Thomas Caudell and David Mizell's publication [Caudell and Mizell 1992] in 1992 where they are credited as being the first to use the term augmented reality to describe such work. Prior to this, the term augmented reality had not been used. They were Boeing software engineers who created an application that used a see-through HMD to guide assembly of aircraft wiring. Within the same year Bajura et al. [Bajura, Fuchs, and Ohbuchi 1992] and Feiner et al. [Feiner, MacIntyre, and Seligmann 1992] each described what would now be classified as augmented reality applications, but they considered it to be a branch of VR. In Bajura et al.'s system they used a video pass through headset to view ultrasound information over a patient's body. Feiner et al.'s system describes usage of a see through HMD to assist with printer repairs. Both systems fall under what is now a classical grand challenge of AR: practical x-ray vision.

Although concepts of AR existed from the late 1980's, it wasn't until the mid 1990's when AR was established as distinct from VR. Azuma identified equipment requirements as inhibitors to AR's growth which created barrier from leaving research laboratories [Azuma 1997]. His reasoning was that tracking performance was too dependent upon a prepared environment on top of needing known markers. Feiner's Touring Machine [Feiner, MacIntyre, Höllerer, et al. 1997] became the first mobile AR system [Billinghurst, Clark, G. Lee, et al. 2015], used to label buildings around Columbia University. The Touring Machine was implemented using the COTERIE framework [MacIntyre and Feiner 1996] which was originally designed for static installments like fishtank VR systems. As a quick aside, fishtank VR systems treated computer monitors as a viewing lens. They supported viewer centered perspective by tracking the user's head in relation to the monitor, giving the impression of someone looking into a fishtank. Back to COTERIE, its distribution version did not come with the support for the AR capabilities showcased with the Touring Machine. Feiner et al. made a custom server which interfaced with COTERIE that played a pivotal role [Feiner, MacIntyre,

[6] https://www.mechdyne.com/about-history.aspx
[7] http://www.calit2.net/newsroom/article.php?id=3006
[8] https://nsf.gov/awardsearch/showAward?AWD_ID=1730158

Höllerer, et al. 1997].

It wasn't until 1999 when Kato and Billinghurst distributed the first AR software library: ARToolKit [Kato and Billinghurst 1999], which is still in active usage today. Not only was it publicly available, but it worked with commodity cameras to track a set of fiducial markers (Figure 36). It is also important to note that ARToolKit explicitly mentioned having to use a modified approach of a published tracking algorithm to optimize for their markers for the sake of performance [Kato and Billinghurst 1999]. If a marker was within the view, their algorithm was able to detect the orientation and allowing a virtual object to be superimposed over the marker's location. ARToolKit was originally designed for HMDs and worked with video pass through or see-through displays. Shortly after, other groups made their software available. Studierstube in 2000 [Schmalstieg, A. Fuhrmann, and Hesina 2000], DWARF in 2001 [Bauer, Bruegge, et al. 2001], VHD++ in 2003 [Ponder et al. 2003], ARTag [Fiala 2004], and DART in 2004 [MacIntyre, Gandy, et al. 2004]. Each of their publications placed heavy emphasis on the perform their software provided, in particular tracking accuracy and time requirements.

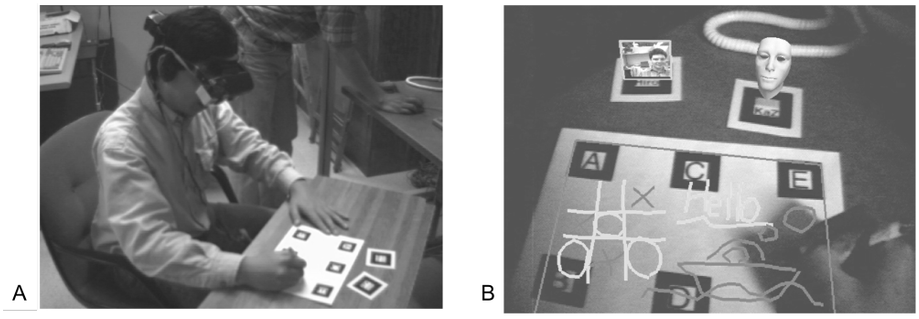

Figure 36: ARToolKit used fiducial markers. A viewer with an HMD could use a stock camera to detect the markers as seen in (A). The augmented view is shown in (B).

Of the software options mentioned, only Studierstube maintained a traditional tracking device and server approach, the rest shifted towards a purely camera solution for tracking. This was arguably influenced by Azuma's statement about equipment requirements were inhibiting the growth of AR. In each case, some part of the system was actively searching for an identifier within the real world. Tracking the identifier produced a location and rotation. The tracked objects needed to be described ahead of time for the software to work correctly.

Schmalstieg et al. first described the visual components of Studierstube in 1996 [Schmalstieg, A. Fuhrmann, Szalavari, et al. 1996]. Two other publications were made in 1998 [Szalavári et al. 1998] and 1999 [Schmalstieg, Encarnação, and Szalavári 1999] describing how Studierstube enables productive investigation and author's work to adapt Studierstube for usage with a VR Workbench [W. Krueger and Froehlich 1994], respectively. But, it wasn't until 2000, where they indicate making Studierstube available. Studierstube was comprised of multiple parts, one a dedicated tracking server. The tracking server required Polhemus Fastrak devices to identify real objects. Studierstube could then poll for specific Fastrak devices, but needed the author to accurately describe the object it was attached to. This was an approach similar to those who created AR applications before [Feiner, MacIntyre, and Seligmann 1992; Bajura, Fuchs, and Ohbuchi 1992]. Although Studierstube hard larger hardware requirements there were a couple of benefits which influenced later versions of Studierstube. The first major benefit was even if the virtual object was not in view of the user, as long as the tracking server was aware of the object, the system could continue to update and track the virtual effects. Hardware tracking was also by comparison quicker and more accurate. The interaction method they provided, personal interaction panel (PIP), enabled fairly intuitive control panel interaction with high accuracy. The other major benefit was the efforts to generalize Studierstube and

combine it with other formats of XR like the workbench [Schmalstieg, Encarnação, and Szalavári 1999] and a tiled display wall [A. L. Fuhrmann et al. 2001].

Bauer et al. claimed that while ARToolKit significantly reduced the barrier for entry to AR, they could further reduce that barrier by providing a modular framework, DWARF [Bauer, Bruegge, et al. 2001]. Although Bauer et al. did praise ARToolKit, they ultimately decided not use use ARToolKit as part of the framework and instead developed their own version. In terms of tracking support, it is interesting to note that a followup paper described how DWARF's modularity allowed the creation of a DWARF interpreter to interface with a Studierstube's tracking server for an increased variety of tracked object support [Bauer, Hilliges, et al. 2003].

Sannier et al. first described the Virtual Human Director (VHD) as a system to support high levels of control over groups of virtual avatars [Sannier et al. 1999]. In 2000, VHD was updated with AR capabilities to track a specific set of patterns. Their usage of AR was to use their avatar system to assist with maintenance and evaluate the realism [Torre et al. 2000; Balcisoy et al. 2000]. But it seems they were dissatisfied with the results, because in 2003, Ponder et al. (including two of the original developers of VHD) completed a significant rewrite to VHD, rebranded as VHD++, they mostly removed the innate AR support and turned their system into a framework supporting detailed avatar orchestration for simulations. Within the rewrite they abstracted view components and a tracking service to support both VR and AR equipment [Ponder et al. 2003].

ARToolKit maintained a steady stream of improvements throughout the years. Users identified a variety of difficulties and desired features which they used to help guide the development process of improved performance. Among those included were: using multiple markers to compensate for visual obstruction issues, ability for real objects to occlude virtual objects, virtual object leaving markers and interacting with other markers, live marker editing interface, and texture tracking, to name a few [Kato, Billinghurst, et al. 2000; Poupyrev et al. 2001; Kato, Tachibana, et al. 2003]. In competition to ARToolKit, Fiala presented ARTag in 2004, which he described as a better solution than ARToolKit [Fiala 2004; Fiala 2005a; Fiala 2005b]. He showed that ARTag's marker tracking worked more accurately and could still operate with partial occlusion. This was made possible by implementing a modified cyclical redundancy check, a concept used with network packets to detect errors.

MacIntyre et al. developed the Designer's Augmented Reality Toolkit (DART) as an extension to Macromedia Director. As a toolkit for designers, they made a clear statement that previous options like ARToolKit, although powerful, were for computer scientists and not easily accessible to designers. DART was comprised of a number of components, one of which interfaced with ARToolKit [MacIntyre, Gandy, et al. 2004] to provide the AR support. However, regardless of the faults identified by other groups, ARToolKit is still used today, perhaps in part due to its open source nature. The most recent publication of an ARToolKit software improvement was by Wang et al. [Wang, Tseng, and Shen 2010] to support live authoring. ARToolKit is still cited in other publications, but the difficulty with tracking its citations is that in modern papers, few refer to the original paper by Kato and Billinghurst; instead citing the code hosting websites [9] [10] [11].

As has been described in this section despite AR and VR being distinct concepts, they still share many similarities. There is also the trend for frameworks to shift from being designed for one but adapted to supporting the other (e.g. COTERIE, Studierstube, and VHD). This concept will be revisited again in a later sections.

[9] `http://www.hitl.washington.edu/artoolkit/`
[10] `https://github.com/artoolkit`
[11] `http://www.artoolkitx.org/`

5.2 Maintaining Truth

In order to provide an immersive experience compliant with XR definition requirements, systems of the 1990s depended upon multiple computers. Sufficient and correct data must be present in a timely manner to ensure the view is correct between machines tasked with generating the view. This still holds true with modern systems and environments supporting multiple users. This section will describe some of the research efforts put into the various system to ensure synchronization: component based systems, automated threading, and network structure implementations.

5.2.1 Component based Systems

Nearly all software for XR systems started to orient themselves towards a component based approach. Part of the reason was the identification that many frameworks were became monolithic in nature. A term some authors used to describe their own work like CALVIN [Leigh, Andrew Johnson, and T. A. DeFanti 1997] and NPSNET [Watsen and M. Zyda 1998]. Although designers commonly stated extensibility as a key aspect, it is closer to say components of a framework that could be run on different computers was implemented for the sake of performance. Spreading across different computers ensured availability of processing power. During the 1990s to the late 2000s the support for these software changes can be seen in the form of data objects, stand alone applications, processes separation, cluster systems, and standalone services.

Self described data objects (SDDO) and standalone applications (SAA) were often similar and closely related to each other. SDDO included three major pieces: resources needed for rendering (e.g. models and textures), dynamic attributes, and code describing how it operated. Having all three pieces allowed objects to migrate between servers and continue to operate correctly or be stored in an offline medium for later recall. SDDO code typically only affected itself. SAA had variations that mimiced SDDO, but often had influence over the virtual environment beyond itself. SAA were often implemented as a client node rather than memory object during the 1990s and slowly shifted towards SDDO format. A contributing factor to the importance of data objects was the hardware expectations during the 1990s. Complex systems often required multiple computers for load and computation balancing. SDDO and SAA were also used beyond for purposes load balancing, like maintaining persistent worlds.

System like CALVIN [Leigh and A. E. Johnson 1996a; Leigh, A. E. Johnson, Vasilakis, et al. 1996], CAVERNsoft [Leigh, Andrew Johnson, and T. A. DeFanti 1997; K. S. Park et al. 2000], AVIARY [Snowdon and West 1994], MASSIVE [Greenhalgh and Steven Benford 1995; Greenhalgh and Steve Benford 1995; Steve Benford, Greenhalgh, and Lloyd 1997; Steve Benford and Greenhalgh 1997; Greenhalgh, Jim Purbrick, and Dave Snowdon 2000], and DIVE [Carlsson and Hagsand 1993; Hagsand 1996; Frécon, G. Smith, et al. 2001] each used SDDO with persistent worlds. Applications and users for these systems were considered temporary entities that influenced the virtual environment in some manner while active. To support lasting effects SDDO could be left behind in the virtual environment by users or applications. For AVIARY and DIVE, users could bring SDDO with them if they traveled to a different world. AVIARY, DIVE, AVOCADO [Tramberend 1999], COTERIE [MacIntyre and Feiner 1996] and Deva3 [S. R. Pettifer 1999; S. Pettifer et al. 2000; Hubbold et al. 1999] used SDDO as a means to load balance. Part of their design made the assumption that multiple servers would be necessary to sustain world management. This was to avoid memory swapping from hard disk or ensure the processor could keep up. SDDO could be transferred between worlds and continue operation. There were two unusual use cases for SDDO. The NPSNET [M. J. Zyda, D. R. Pratt, Monahan, et al. 1992; M. Zyda, D. Pratt, et al. 1993] accessed an offline database containing SDDO and replicated them into a virtual environment. EMMIE [Butz et al. 1999] supported multiple users, but each maintained their own version of

the virtual environment. SDDOs were used to transfer or duplicate an objects, allowing a user to add it into their own maintained version of the environment. After 2000, SAA and SDDO slowly started to disappeared and was just about non-existent by 2010. A major contributing factor was increased performance of CPUs and system storage devices and memory capable of containing the entire virtual environment; leading to fully replicated, distributed scene graphs. This can be seen with MASSIVE [James Purbrick and Greenhalgh 2000; Jim Purbrick and Greenhalgh 2003]; and NPSNET which became BAMBOO [Watsen and M. Zyda 1998; Capps et al. 2000].

Most persistent worlds needed a means to change functionality while active. SAA provided this by taking the form of an instantiatable object or by connecting in as a peer device with virtual environment management and editing capabilities. In addition to the previously mentioned software supporting persistent worlds, Studierstube [Schmalstieg, A. Fuhrmann, Hesina, et al. 2002], VRJuggler [Bierbaum, Just, Hartling, and Cruz-Neira 2000; Olson 2002] and Net Juggler [Allard, Gouranton, Lecointre, Melin, et al. 2002] all supported SAA. All cases except VRJuggler and Net Juggler supported multiple active SA at once. By 2010 SAA support had mostly disappeared due to a shift towards systems supporting launch into a single application.

5.2.2 Automated Threading

Automated threading benefits both performance and maintaining truth. There were mainly two widespread reasons to automate threading: improve render performance and abstract the implementations of distributed shared memory. The benefits of automated threading was recognized from the early 1990s, but wasn't always always user friendly or practical. Among software that have implemented automated threading, there are typically three thread types which assist with increasing performance: render, application, and communication. Applications without threading typically go through a cycle of logic update where rending is performed either at the beginning or end. This ensures the rendered view of the virtual environment is at a specific state and doesn't have any unusual discrepancies. Ensuring the render at a specific state is a critical aspect when generating stereoscopic views, this is because one view is needed for each eye. While rendering between logic updates maintains consistency, this can create large delays between renders. Thread automation for rendering has two major implementations: rendering of the virtual environment is distributed across threads, and/or the render thread is always active and triggered at intervals separate from application updates. For the later case, application updates to the virtual environment will be queued during a render. Some examples of this include MRToolKit [Shaw, Liang, et al. 1992; Shaw, Green, et al. 1993], OpenMask [Margery et al. 2002], and VRJuggler [Bierbaum, Just, Hartling, and Cruz-Neira 2000; Bierbaum, Just, Hartling, Meinert, et al. 2001] which support automation for rendering and application threads.

The third most commonly automated thread is for communication. Software solutions for the distributed and cluster systems need to transfer data between between computers to ensure the virtual environment maintains consistency. Automating and hiding the system upkeep messages reduces the amount of information a developer has to manage but also plays a critical role with the implementation of distributed shared memory. Examples include just about all software supporting clustered and distributed implementations and have carried into modern software solutions: CAVELib, CALVIN, CAVERNsoft, DIVE, COVEN, MASSIVE, SPLINE, COTERIE, Deva3, AVOCADO, ClusterJuggler [Olson 2002], Net Juggler [Allard, Gouranton, Lecointre, Melin, et al. 2002], FlowVR [Allard, Gouranton, Lecointre, Limet, et al. 2004; Lesage and Raffin 2007; Allard and Raffin 2005], DLoVe [Deligiannidis 2000], Continuum [Tran et al. 2002], DIVERSE [Kelso, Arsenault, et al. 2002; Kelso, Satterfield, et al. 2003], Syzygy [Schaeffer and Goudeseune 2003; Schaeffer,

Flider, et al. 2003; Schaeffer, Brinkmann, et al. 2005], CaveUDK [Lugrin et al. 2012], Omegalib [Febretti, Nishimoto, Mateevitsi, et al. 2014], and CyberCANOE [Kawano et al. 2017].

5.2.3 Network structure and implementations

A fairly drastic shift in network utilization has occurred over the years in XR systems. Servers have shifted from a composition of multiple peer-to-peer (p2p) servers into fairly rigid client-server formats. At one point clients slowly gained the capability to support data transfer in a p2p style between each other. Early systems like NPSNET, DIVE, AVIARY, MASSIVE, and SPLINE each started with natively supporting p2p capabilities between server nodes. Their authors expected that hardware capabilities of the time meant that more than one computer would be necessary to sustain the virtual environment. Systems since then have taken the client-server approach most likely due to hardware capabilities making it possible for a client computer to store the environment(usually on hard disk). Even for cluster solutions like ClusterJuggler, FlowVR, Continuum, DIVERSE, CaveUDK, Omegalib and CyberCANOE, they chose to keep the client-server approach. The only exception was Syzygy, which implemented what Schaeffer et al. called a reality peer [Schaeffer, Brinkmann, et al. 2005] and very closely resembled load balancing server nodes similar to that of DIVE, AVIARY, and SPLINE.

Clients became requesters of change rather than an equal entity capable of influencing the world. Cases where stand alone applications, which had the ability to force virtual environment changes independent from server approval, mostly disappeared by the mid 2000s. Systems began supporting applications as data objects, but typically were only allowed to exist in memory on the server (e.g. MASSIVE, VR Juggler). This was especially prominent among cluster implementations from the late 1990s typically developed to support CAVEs. Clients in such systems were mostly implemented for the sake of expanding the physical viewing area. It was typical for the servers to directly access tracking and interaction devices and poll their values, not the clients.

One oddly consistent exception to this is implementations supporting multiple user clients. In a typical client-server approach, clients ask for something to happen. The server will receive the request, determine the effects (sometimes discards the request), and finally conveying results of the request to the client. For the sake of preventing user discomfort, a client view is usually updated based on how a user moves their head in the physical space. In systems that attempt to convey the position of other users, this is one of the few cases where the client still has the right to tell the server where the user's head is instead of requesting permission.

Another aspect that has mostly disappeared from XR software research is network support for transcontinental users. This doesn't mean transcontinental usage is not possible. Simply that transcontinental support is no longer a primary focus among XR software libraries, packages, and frameworks. NPSNET [Macedonia, M. J. Zyda, et al. 1994; Macedonia, D. P. Brutzman, et al. 1995], CALVIN [Leigh and A. E. Johnson 1996b], CAVERNsoft [Leigh, Andrew Johnson, and T. A. DeFanti 1997; K. S. Park et al. 2000], DIVE [Frécon and Stenius 1998; Frécon, Greenhalgh, and Stenius 1999], and MASSIVE [Steve Benford, Greenhalgh, and Lloyd 1997; Steve Benford and Greenhalgh 1997] each had publications describing their approach to data transfer over large distances. Initially, the solution looked like the IP multicast backbone (Mbone) but DIVE, CALVIN, and CAVERNsoft each strayed away from Mbone. A contributing reason is the hardware requirements of the Mbone which researchers typically have no influence over. DIVE kept support for Mbone, but ended up developing their own software variant for packet travel efficiency: the DIVEBONE. Initially the DIVEBONE was a mode of operation a DIVE node could join as. When implemented, DIVEBONE sat be-

tween clients and servers to optimize network utilization. Their reports show significant reduction of network utilization inn comparison to standard Mbone when large numbers of users were participating [Frécon and Stenius 1998; Frécon, Greenhalgh, and Stenius 1999]. CALVIN and CAVERNsoft implemented optimizations usable with all computers through standard UDP and TCP packets. Their methods of port blasting revealed the implementation was especially suited for long distance transmissions(testing was done between United States and Amsterdam) [Leigh and A. E. Johnson 1996b; K. S. Park et al. 2000].

5.3 Flexibility and extensibility

The major barrier to XR software generically supporting developers in creating their envisioned virtual environment was often due to hardware. Such systems were not commonly available until about mid 2000s. These shifts can be in the adoption of the Virtual-Reality Peripheral Network, increased support of hardware implementations, and transition to consumer solutions.

5.3.1 Adoption of the Virtual-Reality Peripheral Network (VRPN)

The Virtual-Reality Peripheral Network sits at a happy medium between maintaining truth, flexibility, and extensibility because all XR systems need some form of tracking. The tracked objects can relate to almost any aspect: users, environment, input devices, and output devices. Software solutions in the 1990s either were built for specific types of hardware (e.g. tracking devices), only supported abstracting that worked within a specific framework, or needed manual solutions. Software like FlowVR [Allard, Gouranton, Lecointre, Limet, et al. 2004], Net Juggler [Allard, Gouranton, Lecointre, Melin, et al. 2002], Simple Virtual Environment [Kessler, Doug A Bowman, and Hodges 2000], and VRJuggler [Bierbaum, Just, Hartling, and Cruz-Neira 2000; Bierbaum, Just, Hartling, Meinert, et al. 2001] supported a separate local process to handle tracking data. But the more advanced solutions provided something closer to a tracking server which could run on a separate device like DLoVe [Deligiannidis 2000], DIVERSE [Kelso, Arsenault, et al. 2002; Kelso, Satterfield, et al. 2003], DWARF [Bauer, Bruegge, et al. 2001; Reicher, MacWilliams, and Bruegge 2003], Morgan [Ohlenburg, Herbst, et al. 2004], Studierstube [Schmalstieg, A. Fuhrmann, Hesina, et al. 2002], and Syzygy [Schaeffer and Goudeseune 2003; Schaeffer, Flider, et al. 2003; Schaeffer, Brinkmann, et al. 2005]. The need for a tracking manager extends across the entire XR spectrum; the afore mentioned software includes a mix of VR and AR. Taylor et al. [Taylor et al. 2001] identified that XR frameworks has started commonly implementing a component to manage abstracted devices. They proposed that rather than each system needing to implement its own version, a general tracking manager that could work as an interchangeable component to any system would be more practical and have better longevity. Taylor et al. presented the Virtual-Reality Peripheral Network (VRPN) as a solution in 2001. They were not alone in this thinking, Reitmayr and Schmalstieg proposed OpenTracker [Reitmayr and Schmalstieg 2001] in 2001 and while Ohlenburg et al. proposed DEVAL [Ohlenburg, Broll, and Lindt 2007] in 2007.

VRPN started as a very simple data server. Clients could set or retrieve data. Entries represented tracked objects or input. Tracked objects had six properties: x, y, z location and roll, pitch, yaw rotation values. Inputs were either a boolean representing button status (pressed or not pressed) or decimal value representing sliders or axis tilt (e.g. a joystick). OpenTracker was also a server, but directly managed and polled for tracked objects and devices. Information about tracked objects included attributes like: position, rotation, button values, confidence rating and time of sample. Furthermore, OpenTracker supported dataflow customization for value filtering and event generation. VRPN was closer to a data intermediary where anything could set and retrieve values. Whereas OpenTracker required devices or interfaces to be directly

connected, but provided event generation options. DEVAL was introduced fairly late by comparison, six years after both VRPN and OpenTracker, in 2007. Like OpenTracker it supported an XML based query and data processing capabilities. DEVAL differentiated itself from the other two by describing itself as an device abstractor for both input and output. For object tracking it worked fairly similar to VRPN. Client connects could declare and set values for a device. But as a device abstractor, it supported a wide variety of values which could then be sent to outputs using a publish-subscribe method. One example they used was speech. An input device could take speech and submit the text to DEVAL. DEVAL could then then forward this data to a connected receiver.

It is arguably VRPN's simple nature which allowed it to survive throughout the years and has continued used today. Despite criticisms from other groups since release, it was integrated as a component or had support written in a variety of other software both AR and VR: DART [MacIntyre, Gandy, et al. 2004], FlowVR [Limet and Robert 2008], BlenderCAVE [Gascón et al. 2011; Poirier-Quinot, Touraine, and Katz 2013], CaveUDK [Lugrin et al. 2012] BlenderVR [Katz et al. 2015], and CyberCANOE [Kawano et al. 2017]. The exact number is not known, VRPN has since become an open source project and it has become common to cite the code hosting website rather than the original paper [12] [13] [14]. As an open source project, this presents additional opportunities in research. For example, Cuevas-Rodriguez et al. [Cuevas-Rodriguez et al. 2015], a completely separate group form the original VRPN authors, presented in 2015 (14 years after VRPN's original release) an addition to VRPN enabling it to support haptics.

5.3.2 Increased support for hardware implementations

VRPN as a tracking solution came at a critical time when XR software was becoming generalized. Various groups started to realized the restrictive nature of their own software and made efforts to support a variety of equipment. Interestingly nearly all of them had a similar approach: support abstraction through scene graph head and eye representations. With the scene graph supporting head and eye representations, their positioning can be used to adjust virtual cameras to ensure appropriate locations when generating stereoscopic views. The following frameworks were supplemented for generalized support of XR equipment: Dragon [Julier et al. 1999], Bamboo [Capps et al. 2000], Simple Virtual Environment [Kessler, Doug A Bowman, and Hodges 2000], COVEN [Frécon, G. Smith, et al. 2001; Steed, Mortensen, and Frécon 2001], AVOCADO [Tramberend 2001], Studierstube [Schmalstieg, A. Fuhrmann, Hesina, et al. 2002], Massive [Jim Purbrick and Greenhalgh 2003], and VHD [Ponder et al. 2003]. This style of scene graph generalization has continued and can be seen into modern software frameworks. Furthermore, implementations of this format are capable of of supporting VR and AR usage.

5.3.3 Transition to consumer solutions

Historically there has been a varying degree of dependence upon commercial solutions. But, between the 1990s and 2000s, a significant shift from directly using OpenGL to working on top of a scene graph software occurred. To explain a bit of what happened requires revising SGI. SGI's role in history was significant as was noted before. However, its influence started to wane by 2000 and the company eventually went defunct. This was greatly influenced by the gaming industry which motivated the continuous cycle of making available increasingly more powerful graphics hardware. One major lineage is the Voodoo card line produced by 3dfx Interactive. The company gained a following from the arcade market when they produced the Voodoo graphics card to drive arcade booths. In the mid 1990s Voodoo was sold as a supplement 3D card which

[12] `http://vrpn.org/`
[13] `http://vrpn.github.io/`
[14] `https://github.com/vrpn`

worked in combination with a traditional 2D video controller (a common practice at the time). By late 1990 companies had shifted towards making single graphics card solutions (both 2D and 3D). The Voodoo line bought by NVIDIA roughly early 2000, eventually becoming the world renown GeForce line.

The gaming industry had a steady increase in popularity since it first started and was considered a major driving factor towards improving graphics hardware. Various researchers also took notice of the potential for usage of game engines within research [Lewis and Jacobson 2002; Meyer 2003; Trenholme and S. P. Smith 2008]. The mid 2000s mark a transition towards commercial software playing a major role in XR software research. Some major examples include DART [MacIntyre, Gandy, et al. 2004; Gandy and MacIntyre 2014], CAVEUT [Jacobson 2003] (Figure 37), CaveUDK [Lugrin et al. 2012](Figure 38), and BlenderCAVE [Gascón et al. 2011; Poirier-Quinot, Touraine, and Katz 2013; Katz et al. 2015](Figure 39). DART was a software extension to Macromedia Director; CAVEUT and CaveUDK used the different versions of the Unreal Game Engine; and BlenderCAVE required the Blender Game Engine. XVR [Carrozzino et al. 2005; Tecchia et al. 2010] was developed by PERCRO with the intention to compete with commercial systems. However it seems development was eventually discontinued; their webpages state usage of Unity from about 2014 [15]. ARTag was eventually rebranded as GoblinXNA to specifically work with the Microsoft XNA Platform [16]. The graphics support provided by game engines automated a large amount of the performance optimizations early systems had to manually implement (e.g. calculating only what was in view, swapping a model's level of detail). But importantly, as a commercial product for game creation, it had to be flexible and extensible to support its user base. If the game engine was not flexible or extensible, game developers would either use a different engine or be forced to create their own solution. For businesses that needed their paying customers to survive, maintaining customer satisfaction played a large role. Game engines will be revisited again throughout the ease of use section.

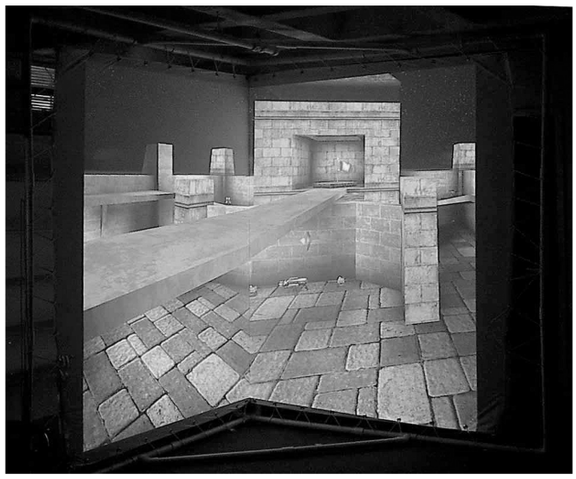

Figure 37: The CAVEUT was build using the Unreal Engine that came bundled with Unreal Tournament. This figure shows a two panel configuration using CAVEUT. To provide a stereoscopic view, the projector needed to swap in the input source between two computers. One for each eye.

5.4 Ease of use

Although all research software will at some point imply (or outright state) their software is easy to use, for whom is it easy to use? Perhaps easy to use for a computer scientists or someone with extensive programming knowledge. How about those without a computer science background? Should software strive to be easy enough to use that someone without a programming background can use it? I think yes, it would be ideal to remove or lower barriers of entry as much as possible. I have observed trends that indicate XR software

[15] `http://www.percro.org/content/motion-and-force-analysis`
[16] `ARTaghttp://monet.cs.columbia.edu/projects/goblin/goblinXNA.htm`

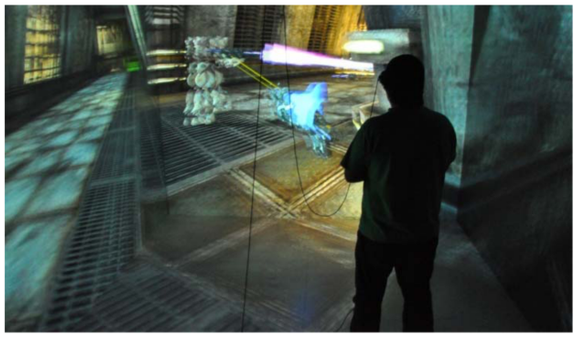

Figure 38: CaveUDK was designed as the successor to CAVEUT, updated to use the Unreal Engine 3 [17].

Figure 39: The BlenderCAVE was built using the Blender Game Engine. Although there seemed to be consistent set of updates through the years ([Gascón et al. 2011; Poirier-Quinot, Touraine, and Katz 2013; Katz et al. 2015]), out of their control was the Blender Foundation ultimately decided to discontinue the game engine[19].

researchers agree, ease of use is a significant factor and software should strive to be usable beyond computer scientists. This can be seen in their identification of: programming language deficiencies, AR's change in direction, the convergence towards Unity, and live editors.

5.4.1 Programming language deficiencies

All computer scientists would agree that the C / C++ languages offers a significant amount of programming power. However, the user friendliness is often brought into question. A number of authors took the time to point out in later publications about their software that a major improvement for the sake of usability was to move away from or add support of languages beyond C/C++ [Capps et al. 2000; Waters et al. 1997; Frécon, G. Smith, et al. 2001]. The first publication of a couple of systems (e.g. VR Juggler [Bierbaum, Just, Hartling, Meinert, et al. 2001], Continuum [Tran et al. 2002] made emphasis that by implementing their software in Java, not only would it assist with cross platform capabilities, but also be easier for usage and extensibility. Recently, Unity was the most used framework at ACM's VRST 2018 and supports C#, JavaScript, and Boo.

The movement away from C/C++ wasn't just difficulties of the language itself; but perhaps moreso to do with organization of thoughts. As a whole, developers over the decades have attempted to create interfaces in better support of creating XR environments. This is most consistently seen in the shift to scene graph usage. Creating immersive virtual environment to that captivate viewers using code alone is incredibly difficult. There are other aspects that include: adding support for data flow graphs (AVOCADO, DLoVe, DWARF, FlowVR, OpenTracker, Studierstube), scripting languages (DART, Syzygy, XVR, VRJuggLua [Pavlik and Vance 2012]), and implementing an event system for reactive processing (Cluster Juggler, Continuum, DIVE, MASSIVE, NPSNET, Studierstube). Along with scene graph usage, event support has become a fairly universal implementation.

Event support is another aspects where C fall deficient. The linear nature almost requires developers

to take an active check approach instead of what has become fairly common with recent programming languages: reactive handling of events. Desinging user interaction for XR systems greatly benefits from event handling. Jacob et al. noted that with virtual environments attempting to mimic reality, a shift to "Non-WIMP" (Windows, Icons, Menus, and Pointers) user interfaces would be necessary to support natural interaction in such environments [Jacob, Deligiannidis, and Morrison 1999]. In literature, user studies to evaluation interaction techniques are common. They often describe in detail how their interaction works, but uncommon is releasing code supporting the interaction tested. Figueroa et al. commented that with their own system, VRJuggler, it was particularly difficult to generate events for user interactions [Figueroa, Green, and Hoover 2002]. They developed InTml for VRJuggler, but determined that it would be of benefit to the community if it was made as a generic component that could be used with other frameworks. Other groups also shared the opinion that a 3D interaction manager would be beneficial to XR systems and released their own versions. IFFI [Ray and Doug A. Bowman 2007] was very similar to InTml, but also supported a means to record events and play them back later (e.g. recording navigation). Grappl differentiated itself by providing the ability automate the layout of a 2D control panel within 3D space. DEVAL [Ohlenburg, Broll, and Lindt 2007], and OpenTracker [Reitmayr and Schmalstieg 2001; Reitmayr and Schmalstieg 2005] were primarily designed as a tracking server, but could perform event generation if tracked objects behaved in predefined ways (e.g. generate a rotation or push event). User interaction studies are still common in XR, but creation of supporting software frameworks have since declined (most now use Unity).

5.4.2 AR's change in direction

The general public's most commonly associated implementation of AR (AR phone browsers) does not align with the definition of AR used in this review. This is because AR phone browsers do not provide a stereoscopic viewer centered perspective. Instead, the view is based on the device and due to using the device's single camera, cannot produce a stereoscopic image. The early 2000s instigated a large interest and popularity with marker based AR (ARToolKit and ARTag) and shortly after significant advances were made with cameras. SURF [Bay, Tuytelaars, and Van Gool 2006; Bay, Ess, et al. 2008] and MonoSlam [Davison et al. 2007] allowed standard RGB cameras to map the environment. There were significant milestone, because prior to this, depth cameras were considered necessary for markerless tracking in order to maintain frame rates. However, this also marked the start of AR browsers which, as mentioned before, by this review's definition of AR do not qualify. As such, AR software contributions dropped to almost zero activity between 2008 and 2014. Regardless it is important to note that this shift is direction is primarily contributed to ease of use, both from a development and user standpoint. For more information, Billinghurst's survey of augmented reality contains an in-depth history of AR with a large section dedicated to AR browsers [Billinghurst, Clark, G. Lee, et al. 2015]. They state a major influence of this shift in direction came from smartphones and tablet devices. It became normal for such devices to come with cameras and mobile processors strong enough to run the computations necessary for AR browsers in real time. With the popularity of AR browsers it wasn't until roughly the announcement of Microsoft's Hololens when interest was renewed in developing AR for headsets. To reiterate: AR research did not stop during this time, it simply entered a direction which doesn't qualify for the definition of AR in this review.

5.4.3 The convergence towards Unity

Each of the previous trends show support behind why Unity has become the most popular framework for supporting XR research. The potential and usage of game engines for XR research is arguably marked

by CAVEUT in 2003 [Jacobson 2003]. Recalling the attributes authors strive for: ease of use, flexibility and extensibility, maintaining truth, and performance. Unity supports each of these attributes with minimal effort from the user. Time has given Unity an edge in performance. Hardware has significantly improved and is expected to continue improving. Unity provides a number of automated rendering optimizations which had to be done manually in previous decades. As a game engine, Unity was designed to be flexible and extensible in order to support developers in building games across all genres. This can be seen in the app store and the fact that all major XR equipment companies (e.g. Microsoft, Oculus, Valve) write extensions for Unity to ease the use of their equipment. Unity has benefited from the gaming industry which has become more popular and socially acceptable over the years and involves tens of billions of dollars annually in the United States [20] [21]. The increased interest in gaming has created a large community with increasing expectations and strong financial interest. As a product, Unity has obtained a sufficiently large community that is invested in Unity's success while providing a steady cash flow to fund continuous improvements. Shortly after 2014 a majority of software in research began using Unity as the core framework. Some examples are CyberCANOE [Kawano et al. 2017], SolidAR [Avveduto, Tecchia, and Fuchs 2017], IMRCE [Salimian, Brooks, and Reilly 2018], OpenImpress [Kolkmeier et al. 2018] and SpaceTime [Xia et al. 2018].

The ACM Symposium on Virtual Reality Software and Technology (VRST) in 2018 has almost 60% of the submissions state they use Unity. For reference, about 20% did not state what software was used to implement their research. Although Unreal had a head start (e.g. CAVEUT and CAVEUDK [Lugrin et al. 2012]), it lost popularity throughout the years. Unity's popularity can be argued in two parts: cost and community. The influence of free is hard to argue. Since the first release in 2005, the basic version was free, and feature-wise has not been too different from the professional version. By comparison, nearly all other professional game engines had only one version and required purchasing a license. This was also true for a majority of text editors and compilers at the time. Unity maintained a steady set of improvements and the power of free provided a large community that fed into the experience. Having a community provided some critical and major benefits for sustaining users: tech support and the app store. Through the forums users could ask questions and often someone else from the community would answer (reducing the need for a developer to respond). Work could be easily distributed and included the ability to modify the editor. Sharing work was both made easy and incentivised through the app store. That is not to say commodification is necessary for success. A good product will need to have sustained support and dedication. Some examples of software survivors are VRPN and ARToolKit which are still used today. Although they may not have publications with regards to implementation in recent years, a steady stream of updates were maintained since their conception. Currently both VRPN and ARToolKit are able to maintain update through their open source communities [22] [23].

5.4.4 Live editors

Much of the early software in research (NPSNET, CALVIN, DIVE, AVIARY, MASSIVE, SPLINE, AVOCADO) involved support of persistent worlds. Unity doesn't naturally support servers or persistence and perhaps indirectly caused a reduction in such research topics. However, something similar but with different purpose are live editors. Rather than needing to code-wise create, place, and adjust the virtual environment and objects within; live editors allow the user to directly manipulate the visuals through a realtime interface. This is different from a scene graph allows graphics position. XR live editors run a modified version of the

[20] As reported by the Entertainment Software Association, a trade association of the video game industry in the United States.
[21] `http://www.theesa.com/about-esa/industry-facts/`
[22] `https://github.com/vrpn/vrpn`
[23] `http://www.artoolkitx.org/`

environment allowing direct modification. This allows seeing exactly what the end user would to enable better control. No coding required, from within the virtual environment designers may literally drag and drop objects until they are satisfied with the arrangement. Live editors got their start in AR (ARToolKit, SURF, MonoSLAM) due to needing to preserve scans of the environment in order to maintain references for virtual objects. Recently there has been interest in brining live editors to VR, like Spacetime [Xia et al. 2018] (Figure 40) and Unity [24]). It is interesting to note that there were two earlier attempts of VR live editors for object animations [G. A. Lee, Kim, and C.-M. Park 2002; Chattopadhyay, Bhandarkar, and Li 2005]. Live editors are a prime example for the importance of ease of use, relating back to the deficiencies of programming. If programming languages were sufficient, there would not be a need for improved means of editing. But also showing that it is possible to create VR environments without needing to code; potentially expanding the audience base for which developers may come from.

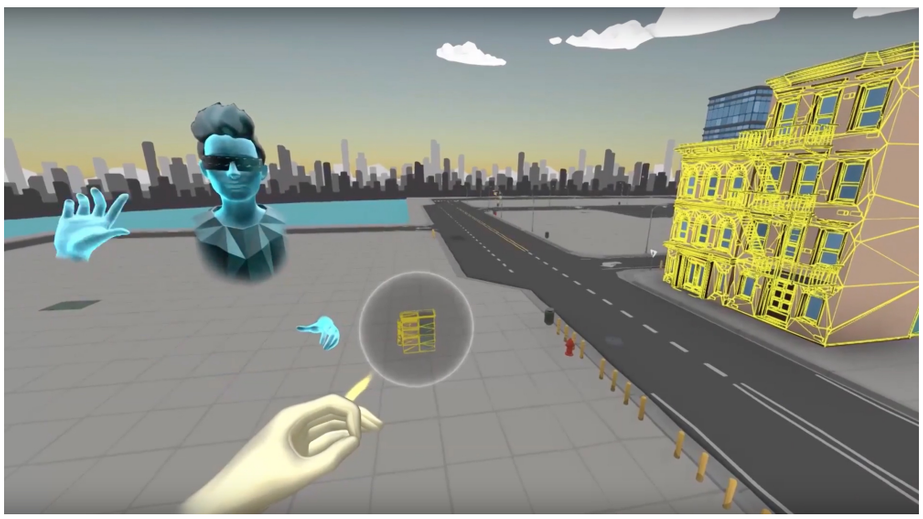

Figure 40: The Space Time editor allows editing the virtual environment from within. In addition, they supported multiple editors at once.

6 Dangers Facing VR

VR is at a software crossroads and in danger of becoming just a gimick. The gaming industry has generated large amounts of public interest while fueling hardware advancements. But, unless VR can establish itself and provide a practical killer app, the general public's perception of VR systems may end up being just another gaming console. Considering how closely VR and AR are related, these dangers have the potential to affects AR as well. The signs of danger come in three forms: user difficulties, trends of the past, and current limitations.

6.1 User difficulties

The hype of VR is losing momentum and disappointment can be seen within the general public. The Consumer Electronics Show (CES) is the largest trade show in North America with attendees from across the world. CES 2019 held an AR-VR-MR Think Tank [25] [26] with a panel of representatives from major companies like Google and Unity. Panelists commented on the overall tone of the general public which was disappointment in VR for lack of progress. Gartner is a company known globally as a research and advisory firm for providing insights and consulting regarding over a wide array of topics including emerging

[24] https://blogs.unity3d.com/2016/12/15/editorvr-experimental-build-available-today/
[25] https://www.ces.tech/Topics/Immersive-Entertainment/Augmented-Virtual-Reality.aspx
[26] https://www.ces.tech/conference/Digital-Hollywood/ARVRMR-Think-Tank-Top-Technology-and-Entertainment-Companies.aspx

technologies. They produced a chart they call the Hype Cycle (Figure 41) which is intended to identify the viability of emerging technology. Since they first made the Hype Cycle chart in 1995, VR was a standard presence. When VR appear on the hype cycle, it was on the downward slope within the trough of disillusion and by 2013, was expected to only get better. However, in 2018 they completely removed VR [27]. They described that there are three challenges that both AR and VR would need to address in order to survive: interface, convenience, and smartphone competition [28]. Elaborating further, the interface doesn't allow someone to just "don their HMD and go to work." Headaches were common among users and there were problems with gesture and movement control. The smartphone provided an AR experience users became accustomed to, in particular the mobility and point-and-see nature which is easier to use. Much of these sentiments were shared across businesses and the gaming industry.

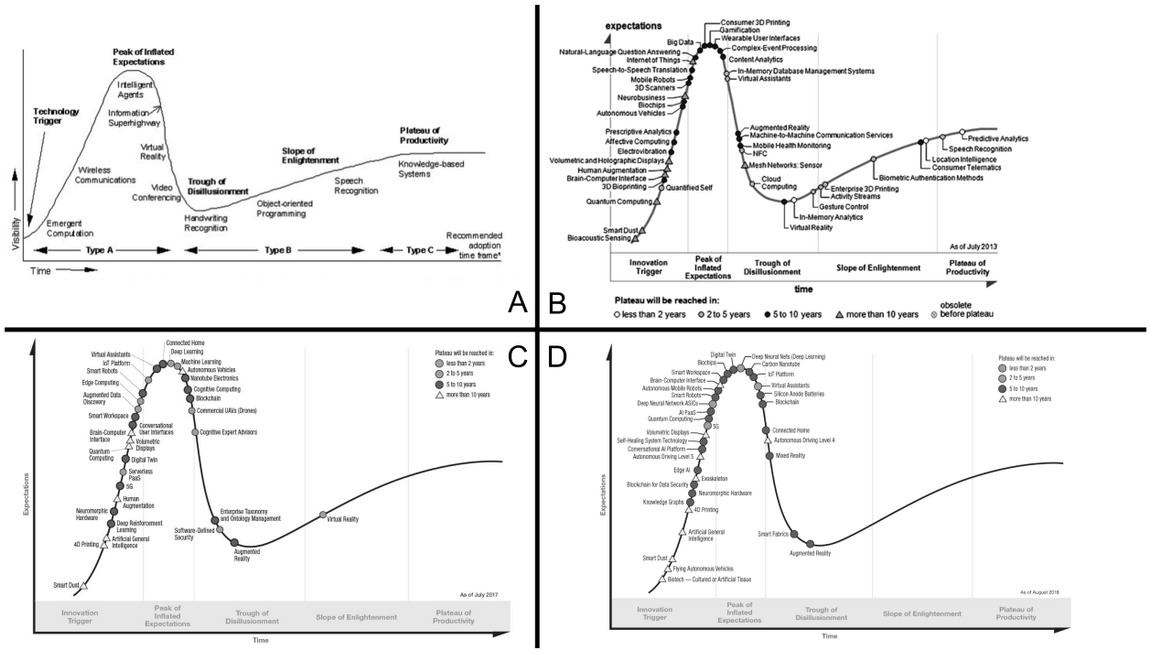

Figure 41: The Gartner Hype cycle is in 1995(A) 2013 (B), 2017(C), and 2018(D). In 2018 Virtual Reality was removed. There are typically five parts of the hype cycle: Innovation Trigger, Peak of Inflation Expectations, Trough of Disillusionment, Slope of Enlightenment, and Pleateau of Productivity. The innovation trigger represents the a new technology gaining hype and popularity. Most technology will then enter a phase where the capabilities are no where close and unable to meet public opinions, the peak of inflated expectations. This mismatch between capabilities and expectations starts will start the descent into the trough of disillusion where most will lose interest and become disappointed. But usually it is after this period which the technology has been given enough time to mature and is able to finally become practical. The plateau of productivity represents a successful technology which has survived the hype cycle and receives mainstream adoption.

6.2 Trends of the past

At the 2019 CES AR-VR-MR Think Tank, panelists made indications that there needs to be more time to develop, but this sentiment has been around since the 1990s. SEGA and Nintendo attempted to take VR mainstream in the 1990s, but was met with failure. Electronic Games, a magazine, reported on SEGA's announcement of developing a VR headset during CES 1993 [29]; however, SEGA ended up cancelling the headset. Nintendo did release the Virtual Boy(Figure 42) in 1995 but that product was discontinued about a year after its release. Since the initial introduction of Oculus and Vive, there have been various complains: the headsets are uncomfortable, VR causes nausea and headaches (simualtor sickness), there is a lack of content, and is mostly a novelty. As someone who has used and worked with these systems, I mostly agree. Since the introduction of the Vive and Occulus in 2014, there hasn't been much improvement regarding

[27] https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/
[28] https://www.gartner.com/smarterwithgartner/3-reasons-why-vr-and-ar-are-slow-to-take-off/
[29] https://archive.org/stream/Electronic-Games-1994-01/Electronic%20Games%201994-01#page/n33/mode/2up

these complains. For consideration, simulator sickness has been known since before 1995 [Kolasinski 1995] yet there are still publications made each year about it.

Figure 42: The Nintendo Virtual Boy was first released in 1995, but wasn't successful and production discontinued within a year.

From a research standpoint, there are additional trends to be aware of. First is that XR research seems to fixate on a topic and then move on. In some cases, the work done is strong enough to last until present day (CAVE, VRPN, ARToolKit). But in other cases, the motivation for investigation isn't revisited. For example: persistent worlds, using XR for remote collaboration meetings, assisting user interaction, support for alternative hardware implementations, and building frameworks. To briefly touch upon each, persistent worlds was a motivator behind much of the effort to maintain truth. CALVIN, CAVERNsoft, DIVE, COVEN, AVIARY, MASSIVE, and SPLINE were all frameworks with a design centered around supporting persistent worlds. However, research software since then has been oriented towards single application launching. The usage of maintaining truth shifted to primarily ensuring cluster implementations or multiple users had correctly synchronized views (e.g. implementing distributed shared memory). Software to support remote collaborative investigation has also mostly disappeared (NPSNET, CALVIN, CAVERNsoft, DIVE, and MASSIVE). This I want to differentiate from multiuser support for applications. For example, education applications are oriented towards distributing information and look to have a promising future [Freina and Ott 2015]. But as an application, these are typically bespoke education solutions and not generalizable(which may hold potential as a research topic). Another significant different how the listed software support collaborative investigation is by allowing users to bring into the environment self defined data objects. These data objects can influence the virtual environment, persist after the owner disconnects, and do not need to be known by the software ahead of time. Assistance with creating user interaction has mostly disappeared (InTml, IFFI, Grappl, DEVAL, and OpenTracker). Each of these systems attempted to make easier the detection of actions and generation of events (e.g. pushing, pointing, rotating, grabbing) to better support 3D interaction. This by default is not a particular bad thing. Perhaps, this indicates frameworks have gotten to the point of being easier enough that assistance is not needed, or perhaps physics engines are the future to enabling natural interaction. This unfortunately doesn't seem likely as each year there continue to be more user studies on interaction and as was mentioned by Gartner, a barrier to VR is one cannot just put on an HMD and get to work. VR systems have become concentrated around two formats: CAVEs and HMDs. Is there no better format? Or simply because this is what hardware companies offer? Which feeds into the last trend. Research does not seem to be producing standalone frameworks to support VR anymore. Just about all publications in the last year were using a commercial solution, and if it was custom, they didn't make the software available. Just like the general public has been hyped up about VR, were the software trends of VR also just hype cycles experienced by researchers? Have researchers lost interest in making frameworks

for the community, or is Unity good enough?

6.3 Current limitations

VR has a number of limitations that I believe are preventing mainstream adoption. Some of these limitations are directly related to user difficulties, but the major ones are as followed: equipment encumbrance, comfort, natural interaction, and practical usage. Of these topics equipment encumbrance and comfort are almost directly related while natural interaction will greatly influence practical usage. Many of these directly relate to AR as well considering the equipment (e.g. HMDs and controller) are almost identical.

Having worked with VR systems, I have observed new VR users tend to exhibit three similar phases. First, they are wowed by what they see. Then, they investigate work and recreation uses. Finally, the verdict: they either become frustrated or willing to wait for improvements. Starting with the hardware, there are a number of problems people tend to dislike about headsets: its not comfortable to wear, it leaves marks, its heavy, it messes up their hair, the glasses don't fit, they get tangled in the cord, can't see the cord, HMD might not be clean(germophobe), the view looks pixelated, the headset gets hot, etc. The list goes on and explains why the CAVE remains so popular, CAVE users are only required to wear light glasses, an encumbrance no greater than sunglasses. But there are also complaints about controllers: the size doesn't always match hand sizes, prefer mouse and keyboard, have to constantly carry it, weight gets tiring over time, doesn't feel accurate, sometimes it doesn't track, hands get sweaty, etc. These complaints are so far related to physical attributes. Interaction is frequently unintuitive, users need to learn how to interact, it doesn't match up what they would have done if using hand instead of controller, navigation is strange, the batteries ran out, although VR is meant to be immersive the real world has very real restrictions, people often damage their surroundings, and have been hurt by their surroundings, etc. Finally the practical usage from a work perspective, why would VR be used over the current alternatives? Not only are there the previously identified complaints, but it also requires additional equipment and costs a lot. Many of the frameworks covered in this review has companion user studies evaluating their use. In particular, MASSIVE had a major focus as providing a framework to build VR alternatives to teleconferencing [Steve Benford, Greenhalgh, and Lloyd 1997]. Yet, nowadays video conferencing is the standard.

6.3.1 Dependence upon Unity

One final topic I would like to cover regarding current limitations is how developers are limited by Unity. As described earlier, a recent research trend was convergent towards using Unity. There are no standalone frameworks developed and provided by researchers in recent years. Unity does exhibit having successful software attributes: ease of use, flexibility and performance. But this also means being at the mercy of Unity release changes. For example, Unity previously supported cluster rendering, but discontinued support[30]. The same is true for UNet, Unity's networking implementation[31]. UNet has been depreciated and will soon be removed from distributions. Developers will have to rely on the app store or write their own solution. Unity as a game engine is really designed to work as a client or in a p2p fashion and lacks support for designing servers for traditional client-server architecture. While multiuser environment are still common, the allowed users is limited. As a commercial game engine, one purpose is to support the creation of standalone games. This causes a couple of restrictions that are very contrary to early XR research. Standalone has the effect of runtime being locked into one application. Using game functionality, it is possible to save and load states, but is unlikely to support the connection and disconnection of applications as objects or client like research

[30] `https://unity3d.com/learn/tutorials/topics/graphics/understanding-clusters`
[31] `https://docs.unity3d.com/Manual/UNet.html`

done in the past. Being standalone also creates a resource problem. The typical usage and expectation means applications must be fully aware of all necessary resources ahead of time. This can be avoided, but doing so must be manually handled by the developer. Finally, is that due to the purpose of Unity, software built using Unity will have difficulty being more than an application.

7 The future of XR

I believe the future of XR depends on mainstream adoption. While the previous section was mostly dangers facing VR, with consideration to research roots and similarity between equipment, the same dangers will affect AR. With the expectation that hardware improvements will continue, even if not XR specific (e.g. batteries, increased CPU performance, display technology, etc) the future of XR research will involve how to better make use of the hardware, what additional capabilities can be included, or how can the software improve along with the hardware. In relation to grand challenges, more progress towards VR's ultimate display can be made. But the grand challenges aren't just about reaching the ultimate display. It can be interpreted as how to make XR appealing and beneficial to the general public which makes the achievement of the ultimate display worthwhile. I predict there will be four major directions which lead XR to mainstream adoption: merging the disciplines, addressing interaction, streaming, and becoming more than an application.

7.1 Merging VR and AR

As was seen previously in this review, originally there was only VR, AR started as a subfocus, but quickly became independent. They share many similar hardware aspects (e.g. HMD and interaction devices) and software also translates well between the two. I believe the future of XR in general will involve the merging of these two back as one for the following primary reason: an AR HMD that turns fully opaque is providing the equivalent of modern day VR. Until technology reaches the point of directly interfacing with the human brain I believe that AR and VR implementations will maintain similar requirements. But this merging can already be seen in safety mechanisms and interaction evaluation.

While using a VR headset, the view of the real world is occluded. For this reason, nearly all applications have recently implemented some form of safety zone. The most well known is perhaps SteamVR's Chaperone system (Figure 43). The concept of a safety zone is not new and has been used in research for redirected walking [Cirio et al. 2009; Williams et al. 2007; Wozniak et al. 2018]. Redirected walking is a method used by VR to maintain an illusion that users are walking through a much larger environment than their physical space allows for. This is typically done by making small gradual changes to the view, causing the user to change direction without them being aware. However, the standard safety box is usually predefined by a user marking off a safety zone. That safety zone doesn't change unless later updated by the user. While that is a low requirement approach, it doesn't account for objects that may move through the space (e.g. other people or chairs), or being able to incorporate the real as part of the virtual experience [Yamaguchi et al. 2018]. Incoporating such aspects requires technology developed for AR, environment tracking. Incoporating this will help VR systems keep users safe and enhance immersion by aligning virtual elements with the real.

A number of users studies have investigated immersion and ownership with relation to avatar properties the user is represented by. In short, the more realistic and aligned with the action taken by the user, the higher sense of ownership the user has. That is, the more likely the user is convinced that the virtual is their own. For more details, the survey by Kilteni and Slater [Kilteni, Groten, and Slater 2012] is a good starting point). This form of research is still continued, last year has publications showing improvement in task completion when with realistic virtual hands and virtual hands that closely aligned with how the

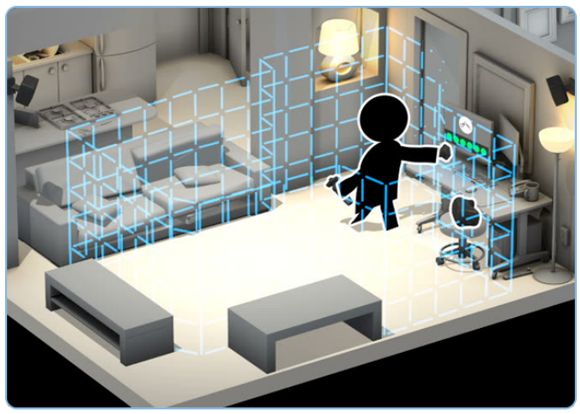

Figure 43: The SteamVR Chaperone system. Users can make an outline of area they set aside for VR applications. If they physically approach while wearing a headset, their view will change to indicate they are at the edges of the area they outlined.

user actually moved [Höll et al. 2018; Jung et al. 2018]. However, this ultimately begs the question, why not allow the user to see their own hand? Are VR developers getting too hung up on showing only the virtual? There are many benefits of merging back VR with AR. First is that, assuming hardware continues to improve, headsets will get lighter, more powerful, and eventually get to a point of being as strong as modern desktops. For this reason, AR with a fully opaque view, is no different from VR. Second, utilizing some of the implementation technologies like environment mapping will help keep users safe. Third, integrating the real can enhance immersion by incorporating the real. This involves concepts like aligning virtual wall with real walls and letting the user see their own hands for better immersion. Conversely, methods like redirected walking need to know the environment in order to help the user avoid the real and maintain immersion.

7.2 Addressing interaction

Despite XR being researched for decades, interaction still doesn't feel natural and there still isn't a generic reason for the general public to want to work in XR, yet. One of the biggest problems is the lack of natural interaction. That is, one must learn how to interact before they can. For many, the interaction methods are strange and unintuitive. A recent study evaluated usage of the Vive and LeapMotion for interaction [Caggianese, Gallo, and Neroni 2018]. They found that despite LeapMotion allowed hands free interaction Vive users performed better; but ultimately, users still had significant difficulties and more work is needed to allow completion of complex tasks. The future of interaction looks to include haptics which has become a common recurring topics especially in user studies. One interesting aspect to point out is typing. Personally, I feel that the ability to type is the biggest barrier towards enabling VR for work purposes. Perhaps researchers agree, 2018 saw a large variety of different text input evaluation. Combining haptics with virtual keyboards [Wu et al. 2017], to joystick input [Yu et al. 2018], to gaze [Rajanna and Hansen 2018], and flat out using real keyboards [Grubert et al. 2018]. Many seem to be of a similar mindset, key input needs to improve. Which relates to the previous section, what is so wrong about letting someone use a real keyboard? At least currently, that is how people naturally type. Aside from typing, XR has other interaction aspects that need to be addressed like movement, especially in physically restrictive areas. A recent survey [Al Zayer, MacNeilage, and Folmer 2018] evaluated locomotion techinques and identified weaknesses in all techniques; indicating more work to be since since an ideal technique has not yet been found.

7.3 Streaming

I believe streaming will play a significant role for XR systems both in the short and long term future. Of the XR grand challenge attributes resolution, visual realism, simulation, and curtailing encumbrance can each directly benefit from streaming, while persistence has an indirect benefit. For the software attributes, streaming directly benefits maintaining truth and performance. Streaming a view into a virtual environment directly benefits the aforementioned attributes by creating a centralized location which the virtual environment is maintained. The major negative is that this will require a fast network with a large bandwidth. However, the benefits will be enormous.

By having a centralized processing location, this removes the need to process the virtual environment on the device(s) the user is directly connected to. This will allow increased performance and lower user encumbrance be reducing the amount of equipment necessary to wear. From an encumbrance aspect this benefits AR moreso than VR. This can also improve VR by no longer needing a desktop for processing. Instead the processing can be done on a server or supercomputer. The average user will not have access to such machines, but can instead be seen as a business product. The gaming industry is already moving towards a streaming environment, and considering how much impact XR has had from the gaming industry, I believe it is best to start now, rather than play catch up later. The benefits of using a server/supercomputer to stream a view directly relate to the grand challenge attributes: simulation, resolution, and visual realism. In wanting the virtual world to mimic the real, this puts an emphasis on aspects like physics simulation. The more interactive the world, the more processing is required for general management, on top of calculating the effects of interaction (e.g. physics). When visual realism and resolution are thrown into the mix, the processing requirements are further increased. Aiming for grand challenge attributes almost certainly indicates XR systems will not be possible to run on consumer grade equipment with decent performance. The final aspect is maintaining truth. A single user, single view XR application doesn't have to worry too much about maintaining truth. But system involving multiple user and cluster setup, maintaining the truth is necessary to ensure users experience the same thing. With a centralized system that views stream out from, at least for the multiuser scenario, difficulties in maintaining the truth will be significantly reduced. From a scene graph perspective, it just becomes a matter of additional camera objects representing locations from which the additional users need to see. This does of course performance concerns. Each additional user is that much more render work, which may already be quite intensive from the resolution and visual realism aspect. But hey, its running on a supercomputer, right?

7.4 XR beyond applications

Currently XR systems are thought of as means to view XR applications. The future of XR involves changing this mindset which is limiting XR to applications. Extending XR beyond an application is necessary to allow integration into daily life. There are a couple of phases I predict this will occur in: browsers, spaces, elimination of equipment, and operating system support.

7.4.1 XR Browsers

AR was on the right track in with browsers. The idea was to have a single AR environment which supported potentially any AR content, a concept that has been around for decades but ultimately still has not been fulfilled [MacIntyre, Hill, et al. 2011; Billinghurst, Clark, G. Lee, et al. 2015; Kooper and MacIntyre 2003; Piekarski and Thomas 2001; Schmalstieg, A. Fuhrmann, Hesina, et al. 2002; Spohrer 1999]. I agree with this belief and that it can be extended to XR in general. The World Wide Web has shown this concept is possible,

the difficulties lies in adaptation. This is a software hurdle has actually been partially addressed before in XR systems supporting persistent worlds. Among the largest barriers was the shift to the application mindset. In the application mindset, it is necessary to know ahead of time (and have locally) resources to view the virtual environment. This assumption is significantly different from browsers and persistent worlds of old; which don't assume such information and support dynamic retrieval of resources necessary to generate the view.

7.4.2 XR as part of the natural space

Part of the mindset restricting XR is the concept of space. Many developers operate with the assumption that users will have limited space to operate XR equipment. Or, one must go to a designated space to experience XR (e.g. CAVE). As displays become larger, cheaper, smaller profile, what if any room could become a CAVE? If it became possible and economical to cover a room in displays as if they were wallpaper, why wouldn't people do so? Simply because there is no application? If wallpaper displays can become a reality, then so too can the concept of rooms, or even buildings becoming CAVEs. I feel this is a concept similar to multiple displays. Computers started off only capable of supporting a single display. The concept of tiled displays was impressive, but concerns about practicality arose. Research dedicated to multiple displays arose [**bradel2013large** ; Czerwinski et al. 2003; Ball and North 2005; Andrews, Endert, and North 2010]. I believe that productivity effects of tiled display walls combined with the desire for immersion are clear indicators that XR will eventually transition beyond HMD's and CAVE, and into everyday spaces.

7.4.3 Elimination of equipment

Tying directly into XR within the natural space, the most obvious problem is how one would see the virtual environment in 3D or interact with it. The answer to this has already appeared, but perhaps in primitive formats: autostereoscopic displays. The Varrier system D. J. Sandin, Margolis, Ge, et al. 2005; D. J. Sandin, Margolis, Greg Dawe, et al. 2001, utilized autostereoscopic with head tracking to support stereoscopic viewer center perspective without the need for glasses or head tracking. Kinect and Leap Motion I believe are only the precursors of technology for enabling interaction without equipment. However that doesn't mean no equipment is the best route. At minimum, the goal is to eliminate equipment that causes a burden on users. Autostereoscopic displays will play a pivotal role breaking XR away from a designated space and allow it transition everyday locations. Another possibility is the advancements of holographic projects to provide the view. Cameras and sensors employing technology similar to the Kinect and Leap Motion will enable interaction with such arrangements.

7.4.4 Supporting XR on the operating system level

As mentioned before, part of the restrictions imposed on XR is the application mindset. But, also contributing to the restriction is device perception. The current mindset is that XR systems involve additional equipment separate from the standard computer. I believe that XR equipment will become integrated as part of a standard computer deployment and be natively supported by operating systems(OS). But when that happens, the hardware formats will probably be in a much different form than available today. The movement away from WIMP (windows, icons, menus, and pointers; i.e. 2D interfaces) and towards 3D interfaces is a view shared by others [Jacob, Girouard, et al. 2008]; and although the view is more towards tangibles, AR could potentially further enhance the experience. Integrating XR equipment as part of the OS is a necessary aspect in eliminating the concept of XR applications. In fact, some commonly used components

were not originally part of the standard computer. For example, mouse, touch screens, multiple displays, and browsers. Each of these are features that have been added over time and became natively supported. To exist as something beyond a novelty, the same must occur for XR devices. Assured longevity of XR means integration as part of operating systems and equipment distributed as part of standard computer configurations. In terms of accomplishing this, VR has shown possibilities and inspirational. The next step involves producing the software in order to make it happen.

8 Conclusion

This review of XR software in research was intended to identify trends in history, dangers facing mainstream adoption, and potential future possibilities. As mentioned before, XR is facing a crossroads where depending on the path taken may end up causing XR to become seen as a novelty rather and preventing it from reaching practical usage. History has shown an symbiotic relationship between software and hardware, AR and VR, research and industry. The identified trends show a reduction in software released by researchers. Considering the vast amount of interaction and usage challenges identified by XR users, researchers who choose to simply report on their findings are not helping XR avoid the perception as a gimick. Researchers must make their work available, especially if they have found something that can benefit the community as a whole. Simply describing it is not enough; as this causes an age old VR disconnection: describing a VR experience to someone who doesn't have any contextual references will not understand the importance unless they also experience it for themself. The last section discussed possible futures where XR can be utilized by the general public. But that path will be difficult and require focus in addressing problems have existed with XR for decades.

References

Al Zayer, Majed, Paul MacNeilage, and Eelke Folmer (2018). "Virtual Locomotion: a Survey". In: *IEEE transactions on visualization and computer graphics*.

Allard, Jérémie, Valérie Gouranton, Loıck Lecointre, Sébastien Limet, et al. (2004). "FlowVR: a middleware for large scale virtual reality applications". In: *European Conference on Parallel Processing*. Springer, pp. 497–505.

Allard, Jérémie, Valérie Gouranton, Loıck Lecointre, Emmanuel Melin, et al. (2002). "Net Juggler: running VR Juggler with multiple displays on a commodity component cluster". In: *Proceedings IEEE Virtual Reality 2002*. IEEE, pp. 273–274.

Allard, Jérémie and Bruno Raffin (2005). "A shader-based parallel rendering framework". In: *Visualization, 2005. VIS 05. IEEE*. IEEE, pp. 127–134.

Altintas, Ilkay et al. (2019). "Workflow-Driven Distributed Machine Learning in CHASE-CI: A Cognitive Hardware and Software Ecosystem Community Infrastructure". In: *arXiv preprint arXiv:1903.06802*.

Anderson, Fraser et al. (2013). "YouMove: enhancing movement training with an augmented reality mirror". In: *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, pp. 311–320.

Andrews, Christopher, Alex Endert, and Chris North (2010). "Space to think: large high-resolution displays for sensemaking". In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, pp. 55–64.

Avveduto, Giovanni, Franco Tecchia, and Henry Fuchs (2017). "Real-world Occlusion in Optical See-through AR Displays". In: *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*. VRST '17. Gothenburg, Sweden: ACM, 29:1–29:10. ISBN: 978-1-4503-5548-3. DOI: 10.1145/3139131.3139150. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/3139131.3139150.

Azuma, Ronald T (1997). "A survey of augmented reality". In: *Presence: Teleoperators & Virtual Environments* 6.4, pp. 355–385.

Baillot, Yohan, L Davis, and J Rolland (2001). "A survey of tracking technology for virtual environments". In: *Fundamentals of wearable computers and augumented reality*, p. 67.

Bajura, Michael, Henry Fuchs, and Ryutarou Ohbuchi (1992). "Merging virtual objects with the real world: Seeing ultrasound imagery within the patient". In: *ACM SIGGRAPH Computer Graphics* 26.2, pp. 203–210.

Balcisoy, Selim et al. (2000). "A Framework for Rapid Evaluation of Prototypes with Augmented Reality". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '00. Seoul, Korea: ACM, pp. 61–66. ISBN: 1-58113-316-2. DOI: 10.1145/502390.502403. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/502390.502403.

Ball, Robert and Chris North (2005). "Analysis of user behavior on high-resolution tiled displays". In: *IFIP Conference on Human-Computer Interaction*. Springer, pp. 350–363.

Barrus, John W, Richard C Waters, and David B Anderson (1996). "Locales: Supporting large multiuser virtual environments". In: *IEEE Computer Graphics and Applications* 16.6, pp. 50–57.

Bauer, Martin, Bernd Bruegge, et al. (2001). "Design of a component-based augmented reality framework". In: *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*. IEEE, pp. 45–54.

Bauer, Martin, Otmar Hilliges, et al. (2003). "Integrating studierstube and dwarf". PhD thesis. CiteSeerX.

Bay, Herbert, Andreas Ess, et al. (2008). "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3. Similarity Matching in Computer Vision and Multimedia, pp. 346–359. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. URL: http://www.sciencedirect.com/science/article/pii/S1077314207001555.

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer, pp. 404–417.

Benford, Steve and Chris Greenhalgh (1997). "Introducing third party objects into the spatial model of interaction". In: *Proceedings of the Fifth European Conference on Computer Supported Cooperative Work*. Springer, pp. 189–204.

Benford, Steve, Chris Greenhalgh, and David Lloyd (1997). "Crowded collaborative virtual environments". In: *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. ACM, pp. 59–66.

Bierbaum, Allen, Christopher Just, Patrick Hartling, and Carolina Cruz-Neira (2000). "Flexible application design using VR Juggler". In: *Technical sketch presented at SIGGRAPH 2000*.

Bierbaum, Allen, Christopher Just, Patrick Hartling, Kevin Meinert, et al. (2001). "VR Juggler: A virtual platform for virtual reality application development". In: *Proceedings IEEE Virtual Reality 2001*. IEEE, pp. 89–96.

Billinghurst, Mark, Adrian Clark, Gun Lee, et al. (2015). "A survey of augmented reality". In: *Foundations and Trends® in Human–Computer Interaction* 8.2-3, pp. 73–272.

Blanchard, Chuck et al. (1990). "Reality Built for Two: A Virtual Reality Tool". In: *SIGGRAPH Comput. Graph.* 24.2, pp. 35–36. ISSN: 0097-8930. DOI: 10.1145/91394.91409. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/91394.91409.

Blum, Tobias et al. (2012). "mirracle: An augmented reality magic mirror system for anatomy education". In: *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*. IEEE, pp. 115–116.

Boring, Sebastian, Marko Jurmu, and Andreas Butz (2009). "Scroll, tilt or move it: using mobile phones to continuously control pointers on large public displays". In: *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7*. ACM, pp. 161–168.

Bradel, Lauren et al. (2013). "Large high resolution displays for co-located collaborative sensemaking: Display usage and territoriality". In: *International Journal of Human-Computer Studies* 71.11, pp. 1078–1088.

Brewster, David (1856). *The Stereoscope; Its History, Theory and Construction, with Its Application to the Fine and Useful Arts and to Education, Etc.* John Murray.

Butz, Andreas et al. (1999). "Enveloping users and computers in a collaborative 3D augmented reality". In: *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. IEEE, pp. 35–44.

Caggianese, Giuseppe, Luigi Gallo, and Pietro Neroni (2018). "The Vive controllers vs. leap motion for interactions in virtual environments: a comparative evaluation". In: *International Conference on Intelligent Interactive Multimedia Systems and Services*. Springer, pp. 24–33.

Capps, Michael et al. (2000). "NPSNET-V. A new beginning for dynamically extensible virtual environments". In: *IEEE Computer Graphics and Applications* 20.5, pp. 12–15.

Carlsson, Christer and Olof Hagsand (1993). "DIVE A multi-user virtual reality system". In: *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, pp. 394–400.

Carrozzino, Marcello et al. (2005). "Lowering the development time of multimodal interactive application: the real-life experience of the XVR project". In: *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACM, pp. 270–273.

Caudell, Thomas P and David W Mizell (1992). "Augmented reality: An application of heads-up display technology to manual manufacturing processes". In: *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*. Vol. 2. IEEE, pp. 659–669.

Chattopadhyay, Siddhartha, Suchendra M. Bhandarkar, and Kang Li (2005). "Efficient Compression and Delivery of Stored Motion Data for Avatar Animation in Resource Constrained Devices". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '05. Monterey, CA, USA: ACM, pp. 235–243. ISBN: 1-59593-098-1. DOI: 10.1145/1101616.1101665. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/1101616.1101665.

Cirio, Gabriel et al. (2009). "The magic barrier tape: a novel metaphor for infinite navigation in virtual worlds with a restricted walking workspace". In: *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*. ACM, pp. 155–162.

Cordeil, Maxime et al. (2017). "Immersive collaborative analysis of network connectivity: Cave-style or head-mounted display?" In: *IEEE transactions on visualization and computer graphics* 23.1, pp. 441–450.

Costanza, Enrico, Andreas Kunz, and Morten Fjeld (2009). "Mixed reality: A survey". In: *Human machine interaction*. Springer, pp. 47–68.

Cruz-Neira, Carolina, Daniel J Sandin, and Thomas A DeFanti (1993). "Surround-screen projection-based virtual reality: the design and implementation of the CAVE". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. ACM, pp. 135–142.

Cruz-Neira, Carolina, Daniel J Sandin, Thomas A DeFanti, et al. (1992). "The CAVE: audio visual experience automatic virtual environment". In: *Communications of the ACM* 35.6, pp. 64–73.

Cuevas-Rodriguez, Maria et al. (2015). "Contributing to VRPN with a New Server for Haptic Devices". In: *Proceedings of the 21st ACM Symposium on Virtual Reality Software and Technology*. VRST '15. Beijing, China: ACM, pp. 193–193. ISBN: 978-1-4503-3990-2. DOI: 10.1145/2821592.2821639. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/2821592.2821639.

Czernuszenko, Marek et al. (1997). "The ImmersaDesk and Infinity Wall projection-based virtual reality displays". In: *ACM SIGGRAPH Computer Graphics* 31.2, pp. 46–49.

Czerwinski, Mary et al. (2003). "Toward characterizing the productivity benefits of very large displays." In: *Interact*. Vol. 3, pp. 9–16.

Davison, Andrew J et al. (2007). "MonoSLAM: Real-time single camera SLAM". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6, pp. 1052–1067.

DeFanti, Thomas A et al. (2011). "The future of the CAVE". In: *Central European Journal of Engineering* 1.1, pp. 16–37.

Defanti, Thomas and Daniel Sandin (1977). "Final report to the National Endowment of the arts". In: *US NEA R60-34-163, University of Illinois at Chicago Circle, Chicago, Illinois*.

Deligiannidis, Leonidas (2000). "A specification paradigm for designing distributed VR applications for single or multiple users". PhD thesis. Tufts University.

Deligiannidis, Leonidas and Robert JK Jacob (2002). "Dlove: Using constraints to allow parallel processing in multi-user virtual reality". In: *Virtual Reality, 2002. Proceedings. IEEE*. IEEE, pp. 49–56.

Durbin, Jim et al. (1998). "Making information overload work: The Dragon software system on a virtual reality responsive workbench". In: *Digitization of the Battlespace III*. Vol. 3393. International Society for Optics and Photonics, pp. 96–108.

Edwards, Emily K, Jannick P Rolland, and Kurtis P Keller (1993). "Video see-through design for merging of real and virtual environments". In: *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, pp. 223–233.

Febretti, Alessandro, Arthur Nishimoto, Victor Mateevitsi, et al. (2014). "Omegalib: A multi-view application framework for hybrid reality display environments". In: *2014 IEEE Virtual Reality (VR)*. IEEE, pp. 9–14.

Febretti, Alessandro, Arthur Nishimoto, Terrance Thigpen, et al. (2013). "CAVE2: a hybrid reality environment for immersive simulation and information analysis". In: *The Engineering Reality of Virtual Reality 2013*. Vol. 8649. International Society for Optics and Photonics, p. 864903.

Feiner, Steven, Blair MacIntyre, Tobias Höllerer, et al. (1997). "A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment". In: *Personal Technologies* 1.4, pp. 208–217.

Feiner, Steven, Blair MacIntyre, and Dorée Seligmann (1992). "Annotating the real world with knowledge-based graphics on a see-through head-mounted display". In: *proceedings of Graphics Interface*. Vol. 92, pp. 78–85.

Fiala, Mark (2004). "Artag revision 1, a fiducial marker system using digital techniques". In: *National Research Council Publication* 47419, pp. 1–47.

Fiala, Mark (2005a). "ARTag, a fiducial marker system using digital techniques". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 2. IEEE, pp. 590–596.

— (2005b). "Comparing ARTag and ARToolkit Plus fiducial marker systems". In: *Haptic Audio Visual Environments and their Applications, 2005. IEEE International Workshop on*. IEEE, 6–pp.

Figueroa, Pablo, Mark Green, and H James Hoover (2002). "InTml: a description language for VR applications". In: *Proceedings of the seventh international conference on 3D Web technology*. ACM, pp. 53–58.

Fisher, Scott S et al. (1987). "Virtual environment display system". In: *Proceedings of the 1986 workshop on Interactive 3D graphics*. ACM, pp. 77–87.

Frati, Valentino and Domenico Prattichizzo (2011). "Using Kinect for hand tracking and rendering in wearable haptics". In: *World Haptics Conference (WHC), 2011 IEEE*. IEEE, pp. 317–321.

Frécon, Emmanuel (2004). *A Survey of CVE Technologies and Systems*. Swedish Institute of Computer Science.

Frécon, Emmanuel, Chris Greenhalgh, and Mårten Stenius (1999). "The DiveBone—an application-level network architecture for Internet-based CVEs". In: *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, pp. 58–65.

Frécon, Emmanuel, Gareth Smith, et al. (2001). "An overview of the COVEN platform". In: *Presence: Teleoperators & Virtual Environments* 10.1, pp. 109–127.

Frécon, Emmanuel and Mårten Stenius (1998). "DIVE: A scaleable network architecture for distributed virtual environments". In: *Distributed Systems Engineering* 5.3, p. 91.

Freina, Laura and Michela Ott (2015). "A literature review on immersive virtual reality in education: state of the art and perspectives". In: *The International Scientific Conference eLearning and Software for Education*. Vol. 1. " Carol I" National Defence University, p. 133.

Fuhrmann, A. L. et al. (2001). "Interactive Content for Presentations in Virtual Reality". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '01. Baniff, Alberta, Canada: ACM, pp. 183–189. ISBN: 1-58113-427-4. DOI: 10.1145/505008.505043. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/505008.505043.

Gandy, Maribeth and Blair MacIntyre (2014). "Designer's Augmented Reality Toolkit, Ten Years Later: Implications for New Media Authoring Tools". In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST '14. Honolulu, Hawaii, USA: ACM, pp. 627–636. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647369. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/2642918.2647369.

Gascón, Jorge et al. (2011). "BlenderCAVE: Easy VR Authoring for Multi-Screen Displays". In:

Germans, Desmond et al. (2001). "VIRPI: a high-level toolkit for interactive scientific visualization in virtual reality". In: *Immersive Projection Technology and Virtual Environments 2001*. Springer, pp. 109–120.

Greenhalgh, Chris and Steve Benford (1995). "MASSIVE: a distributed virtual reality system incorporating spatial trading". In: *icdcs*. Vol. 95, p. 27.

Greenhalgh, Chris and Steven Benford (1995). "MASSIVE: a collaborative virtual environment for teleconferencing". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 2.3, pp. 239–261.

Greenhalgh, Chris, Jim Purbrick, and Dave Snowdon (2000). "Inside MASSIVE-3: Flexible support for data consistency and world structuring". In: *Proceedings of the third international conference on Collaborative virtual environments*. ACM, pp. 119–127.

Gruber, Wilhelm B (1940). *Stereoscopic viewing device*. US Patent 2,189,285.

Grubert, Jens et al. (2018). "Text entry in immersive head-mounted display-based virtual reality using standard keyboards". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, pp. 159–166.

Hagsand, Olof (1996). "Interactive multiuser VEs in the DIVE system". In: *IEEE multimedia* 3.1, pp. 30–39.

Halle, Michael (2005). "Autostereoscopic displays and computer graphics". In: *ACM SIGGRAPH 2005 Courses*. ACM, p. 104.

Heilig, Morton (2002). "The cinema of the future". In: *Translated by Uri Feldman. In Multimedia: From Wagner to Virtual Reality. Edited by Randall Packer and Ken Jordan. Expanded ed. New York: WW Norton*, pp. 239–251.

Heilig, Morton Leonard (1992). "El cine del futuro: The cinema of the future". In: *Presence: Teleoperators & Virtual Environments* 1.3, pp. 279–294.

Höll, Markus et al. (2018). "Efficient physics-based implementation for realistic hand-object interaction in virtual reality". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, pp. 175–182.

Howlett, E. M. (1983). *Wide angle color photography method and system*. US Patent 4406532.

Howlett, Eric M (1990). "Wide-angle orthostereo". In: *Stereoscopic Displays and Applications*. Vol. 1256. International Society for Optics and Photonics, pp. 210–224.

Hubbold, Roger et al. (1999). "GNU/MAVERIK: A Micro-kernel for Large-scale Virtual Environments". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '99. London, United Kingdom: ACM, pp. 66–73. ISBN: 1-58113-141-0. DOI: 10.1145/323663.323674. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/323663.323674.

Jacob, Robert JK, Leonidas Deligiannidis, and Stephen Morrison (1999). "A software model and specification language for non-WIMP user interfaces". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 6.1, pp. 1–46.

Jacob, Robert JK, Audrey Girouard, et al. (2008). "Reality-based interaction: a framework for post-WIMP interfaces". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 201–210.

Jacobson, Jeffrey (2003). "Using CaveUT to build immersive displays with the unreal tournament engine and a PC cluster". In: *Proceedings of the 2003 symposium on Interactive 3D graphics*. ACM, pp. 221–222.

Jalkanen, Janne (2000). "Building a spatially immersive display-HUTCAVE". In:

Johnson, Andrew et al. (2000). "Developing the paris: Using the cave to prototype a new vr display". In: *Proceedings of IPT*, pp. 19–20.

Julier, Simon et al. (1999). "The software architecture of a real-time battlefield visualization virtual environment". In: *Virtual Reality, 1999. Proceedings., IEEE*. IEEE, pp. 29–36.

Jung, Sungchul et al. (2018). "Over My Hand: Using a Personalized Hand in VR to Improve Object Size Estimation, Body Ownership, and Presence". In: *Proceedings of the Symposium on Spatial User Interaction*. ACM, pp. 60–68.

Kamphuis, Carolien et al. (2014). "Augmented reality in medical education?" In: *Perspectives on medical education* 3.4, pp. 300–311.

Kato, Hirokazu and Mark Billinghurst (1999). "Marker tracking and hmd calibration for a video-based augmented reality conferencing system". In: *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*. IEEE, pp. 85–94.

Kato, Hirokazu, Mark Billinghurst, et al. (2000). "Virtual object manipulation on a table-top AR environment". In: *Augmented Reality, 2000.(ISAR 2000). Proceedings. IEEE and ACM International Symposium on*. Ieee, pp. 111–119.

Kato, Hirokazu, Keihachiro Tachibana, et al. (2003). "A registration method based on texture tracking using artoolkit". In: *Augmented Reality Toolkit Workshop, 2003. IEEE International*. IEEE, pp. 77–85.

Katz, Brian FG et al. (2015). "BlenderVR: Open-source framework for interactive and immersive VR". In: *Virtual Reality (VR), 2015 IEEE*. IEEE, pp. 203–204.

Kawano, Noel et al. (2017). "The Destiny-class CyberCANOE–a surround screen, stereoscopic, cyber-enabled collaboration analysis navigation and observation environment". In: *Electronic Imaging* 2017.3, pp. 25–30.

Kelso, John, Lance E Arsenault, et al. (2002). "Diverse: A framework for building extensible and reconfigurable device independent virtual environments". In: *Virtual Reality, 2002. Proceedings. IEEE*. IEEE, pp. 183–190.

Kelso, John, Steven G Satterfield, et al. (2003). "DIVERSE: a framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations". In: *Presence: Teleoperators & Virtual Environments* 12.1, pp. 19–36.

Kessler, G Drew, Doug A Bowman, and Larry F Hodges (2000). "The simple virtual environment library: an extensible framework for building VE applications". In: *Presence: Teleoperators & Virtual Environments* 9.2, pp. 187–208.

Kilteni, Konstantina, Raphaela Groten, and Mel Slater (2012). "The sense of embodiment in virtual reality". In: *Presence: Teleoperators and Virtual Environments* 21.4, pp. 373–387.

Kolasinski, Eugenia M (1995). *Simulator Sickness in Virtual Environments*. Tech. rep. Army research Inst for the behavioral and social sciences Alexandria VA.

Kolkmeier, Jan et al. (2018). "With a little help from a holographic friend: the OpenIMPRESS mixed reality telepresence toolkit for remote collaboration systems". In: *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. ACM, p. 26.

Kooper, Rob and Blair MacIntyre (2003). "Browsing the real-world wide web: Maintaining awareness of virtual information in an AR information space". In: *International Journal of Human-Computer Interaction* 16.3, pp. 425–446.

Krueger, Myron W, Thomas Gionfriddo, and Katrin Hinrichsen (1985). "VIDEOPLACE—an artificial reality". In: *ACM SIGCHI Bulletin*. Vol. 16. 4. ACM, pp. 35–40.

Krueger, Myron William (1983). *Artificial reality*. Vol. 126. Addison-Wesley Reading, MA.

Krueger, Wolfgang and Bernd Froehlich (1994). "Responsive Workbench". In: *Virtual Reality'94*. Springer, pp. 73–80.

Kruger, Wolfgang et al. (1995). "The responsive workbench: A virtual work environment". In: *Computer* 28.7, pp. 42–48.

Lee, Gun A., Gerard Jounghyun Kim, and Chan-Mo Park (2002). "Modeling Virtual Object Behavior Within Virtual Environment". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '02. Hong Kong, China: ACM, pp. 41–48. ISBN: 1-58113-530-0. DOI: 10.1145/585740.585748. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/585740.585748.

Leigh, Jason and Andrew E Johnson (1996a). "CALVIN: an immersimedia design environment utilizing heterogeneous perspectives". In: *Multimedia Computing and Systems, 1996., Proceedings of the Third IEEE International Conference on*. IEEE, pp. 20–23.

Leigh, Jason and Andrew E Johnson (1996b). “Supporting transcontinental collaborative work in persistent virtual environments”. In: *IEEE Computer Graphics and Applications* 16.4, pp. 47–51.

Leigh, Jason, Andrew E Johnson, Tom DeFanti, et al. (1995). “Transcontinental collaborative design in a shared virtual environment”. In: *Demonstration in the CAVE at the Supercomputing ’95 Conference*. Supercomputing.

Leigh, Jason, Andrew E Johnson, Christina A Vasilakis, et al. (1996). “Multi-perspective collaborative design in persistent networked virtual environments”. In: *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*. IEEE, pp. 253–260.

Leigh, Jason, Andrew Johnson, and Thomas A DeFanti (1997). “CAVERN: A distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments”. In: *Virtual Reality: Research, Development and Applications* 2.2, pp. 217–237.

Lesage, Jean-Denis and Bruno Raffin (2007). “A hierarchical programming model for large parallel interactive applications”. In: *IFIP International Conference on Network and Parallel Computing*. Springer, pp. 516–525.

Lewis, Michael and Jeffrey Jacobson (2002). “Game Engines In Scientific Research”. In: *Communications of the ACM* 45.1, p. 27.

Limet, Sébastien and Sophie Robert (2008). “FlowVR-VRPN: first experiments of a VRPN/FlowVR coupling”. In: *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*. ACM, pp. 251–252.

Lin, Junguo et al. (2017). “Retinal projection head-mounted display”. In: *Frontiers of Optoelectronics* 10.1, pp. 1–8.

Lipton, Lenny (1984). “Binocular symmetries as criteria for the successful transmission of images in the stereodimensional (TM) brand stereoscopic video system”. In: *Processing and Display of Three-Dimensional Data II*. Vol. 507. International Society for Optics and Photonics, pp. 108–114.

Lugrin, Jean-Luc et al. (2012). “CaveUDK: a VR game engine middleware”. In: *Proceedings of the 18th ACM symposium on Virtual reality software and technology*. ACM, pp. 137–144.

Macedonia, Michael R, Donald P Brutzman, et al. (1995). “NPSNET: A multi-player 3D virtual environment over the internet”. In: *Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM, 93–ff.

Macedonia, Michael R, Michael J Zyda, et al. (1994). “NPSNET: a network software architecture for largescale virtual environments”. In: *Presence: Teleoperators & Virtual Environments* 3.4, pp. 265–287.

MacIntyre, Blair and Steven Feiner (1996). “Language-level support for exploratory programming of distributed virtual environments”. In: *Proceedings of the 9th annual ACM symposium on User interface software and technology*. ACM, pp. 83–94.

MacIntyre, Blair, Maribeth Gandy, et al. (2004). “DART: a toolkit for rapid design exploration of augmented reality experiences”. In: *Proceedings of the 17th annual ACM symposium on User interface software and technology*. ACM, pp. 197–206.

MacIntyre, Blair, Alex Hill, et al. (2011). “The Argon AR Web Browser and standards-based AR application environment”. In: *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, pp. 65–74.

Mackey, Randall L (1991). *NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation*. Tech. rep. NAVAL POSTGRADUATE SCHOOL MONTEREY CA.

Mann, S. (1994). *‘Mediated Reality’*. TR 260. Cambridge, Massachusetts, http://wearcam.org/mr.htm: M.I.T. Media Lab Perceptual Computing Section.

Mann, Steve et al. (2018). *All Reality: Values, taxonomy, and continuum, for Virtual, Augmented, eXtended/MiXed (X), Mediated (X, Y), and Multimediated Reality/Intelligence*.

Margery, David et al. (2002). "Openmask:{Multi-Threaded— Modular} animation and simulation {Kernel— Kit}: a general introduction". In: *VRIC*, pp. 101–110.

McDowall, Ian E et al. (1990). "Implementation and integration of a counterbalanced CRT-based stereoscopic display for interactive viewpoint control in virtual-environment applications". In: *Stereoscopic Displays and Applications*. Vol. 1256. International Society for Optics and Photonics, pp. 136–147.

Meyer, Tony (2003). "Building cost-effective research platforms: utilising free— open-source software in research projects". In:

Milgram, Paul and Fumio Kishino (1994). "A taxonomy of mixed reality visual displays". In: *IEICE TRANSACTIONS on Information and Systems* 77.12, pp. 1321–1329.

Muensterer, Oliver J et al. (2014). "Google Glass in pediatric surgery: an exploratory study". In: *International journal of surgery* 12.4, pp. 281–289.

Ni, Tao et al. (2006). "A survey of large high-resolution display technologies, techniques, and applications". In: *null*. IEEE, pp. 223–236.

Ohlenburg, Jan, Wolfgang Broll, and Irma Lindt (2007). "DEVAL–a device abstraction layer for VR/AR". In: *International Conference on Universal Access in Human-Computer Interaction*. Springer, pp. 497–506.

Ohlenburg, Jan, Iris Herbst, et al. (2004). "The MORGAN Framework: Enabling Dynamic Multi-user AR and VR Projects". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '04. Hong Kong: ACM, pp. 166–169. ISBN: 1-58113-907-1. DOI: 10.1145/1077534.1077568. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/1077534.1077568.

Olson, Eric Charles (2002). "Cluster Juggler-PC cluster virtual reality". PhD thesis. Iowa State University.

Pape, Dave (1996). "A hardware-independent virtual reality development system". In: *IEEE Computer Graphics and Applications* 16.4, pp. 44–47.

Pape, Dave et al. (1999). "The immersadesk3-experiences with a flat panel display for virtual reality". In: *the proceedings of the Third International Immersive Projection Technology Workshop, Stuttgart, Germany*, pp. 107–112.

Park, Kyoung S et al. (2000). "CAVERNsoft G2: a toolkit for high performance tele-immersive collaboration". In: *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, pp. 8–15.

Park, Sungju et al. (2001). "Scalable Data Management Using User-based Caching and Prefetching in Distributed Virtual Environments". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '01. Baniff, Alberta, Canada: ACM, pp. 121–126. ISBN: 1-58113-427-4. DOI: 10.1145/505008.505033. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/505008.505033.

Pavlik, Ryan A and Judy M Vance (2012). "VR JuggLua: A framework for VR applications combining Lua, OpenSceneGraph, and VR Juggler". In: *2012 5th workshop on software engineering and architectures for realtime interactive systems (SEARIS)*. IEEE, pp. 29–35.

Pettifer, Stephen Robert (1999). "An operating environment for large scale virtual reality". PhD thesis. University of Manchester, UK.

Pettifer, Steve et al. (2000). "DEVA3: Architecture for a Large-scale Distributed Virtual Reality System". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '00. Seoul,

Korea: ACM, pp. 33–40. ISBN: 1-58113-316-2. DOI: 10.1145/502390.502397. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/502390.502397.

Piekarski, Wayne and Bruce H Thomas (2001). "Tinmith-metro: New outdoor techniques for creating city models with an augmented reality wearable computer". In: *Proceedings Fifth International Symposium on Wearable Computers*. IEEE, pp. 31–38.

Poirier-Quinot, David, Damien Touraine, and Brian FG Katz (2013). "BlenderCAVE: A multimodal scene graph editor for virtual reality". In: International Conference on Auditory Display.

Ponder, Michal et al. (2003). "VHD++ development framework: Towards extendible, component based VR/AR simulation engine featuring advanced virtual character technologies". In: *Proceedings Computer Graphics International 2003*. IEEE, pp. 96–104.

Poupyrev, Ivan et al. (2001). "Tiles: A Mixed Reality Authoring Interface." In: *Interact*. Vol. 1, pp. 334–341.

Purbrick, James and Chris Greenhalgh (2000). "Extending locales: Awareness management in MASSIVE-3". In: *vr*. IEEE, p. 287.

Purbrick, Jim and Chris Greenhalgh (2003). "An extensible event-based infrastructure for networked virtual worlds". In: *Presence: Teleoperators & Virtual Environments* 12.1, pp. 68–84.

Rajanna, Vijay and John Paulin Hansen (2018). "Gaze typing in virtual reality: impact of keyboard design, selection method, and motion". In: *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, p. 15.

Ray, Andrew and Doug A. Bowman (2007). "Towards a System for Reusable 3D Interaction Techniques". In: *Proceedings of the 2007 ACM Symposium on Virtual Reality Software and Technology*. VRST '07. Newport Beach, California: ACM, pp. 187–190. ISBN: 978-1-59593-863-3. DOI: 10.1145/1315184.1315219. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/1315184.1315219.

Reicher, Thomas, Asa MacWilliams, and Bernd Bruegge (2003). "Towards a system of patterns for augmented reality systems". In: *International Workshop on Software Technology for Augmented Reality Systems (STARS 2003)*.

Reitmayr, Gerhard and Dieter Schmalstieg (2001). "An open software architecture for virtual reality interaction". In: *Proceedings of the ACM symposium on Virtual reality software and technology*. ACM, pp. 47–54.

— (2005). "OpenTracker: A flexible software design for three-dimensional interaction". In: *Virtual reality* 9.1, pp. 79–92.

Rekimoto, Jun and Katashi Nagao (1995). "The world through the computer: Computer augmented interaction with real world environments". In: *Proceedings of the 8th annual ACM symposium on User interface and software technology*. ACM, pp. 29–36.

Renambot, Luc et al. (2000). "Cavestudy: an infrastructure for computational steering in virtual reality environments". In: *Proceedings the Ninth International Symposium on High-Performance Distributed Computing*. IEEE, pp. 239–246.

Salimian, Mohamad, Stephen Brooks, and Derek Reilly (2018). "IMRCE: A Unity Toolkit for Virtual Co-Presence". In: *Proceedings of the Symposium on Spatial User Interaction*. ACM, pp. 48–59.

Sandin, Daniel J, Todd Margolis, Greg Dawe, et al. (2001). "Varrier autostereographic display". In: *Stereoscopic Displays and Virtual Reality Systems VIII*. Vol. 4297. International Society for Optics and Photonics, pp. 204–212.

Sandin, Daniel J, Todd Margolis, Jinghua Ge, et al. (2005). "The Varrier TM autostereoscopic virtual reality display". In: *ACM Transactions on Graphics (TOG)*. Vol. 24. 3. ACM, pp. 894–903.

Sannier, Gaël et al. (1999). "VHD: a system for directing real-time virtual actors". In: *The Visual Computer* 15.7-8, pp. 320–329.

Schaeffer, Benjamin, Peter Brinkmann, et al. (2005). "Myriad: Scalable VR via Peer-to-peer Connectivity, PC Clustering, and Transient Inconsistency". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '05. Monterey, CA, USA: ACM, pp. 68–77. ISBN: 1-59593-098-1. DOI: 10.1145/1101616.1101631. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/1101616.1101631.

Schaeffer, Benjamin, Mark Flider, et al. (2003). "Tele-sports and Tele-dance: Full-body Network Interaction". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '03. Osaka, Japan: ACM, pp. 108–116. ISBN: 1-58113-569-6. DOI: 10.1145/1008653.1008673. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/1008653.1008673.

Schaeffer, Benjamin and Camille Goudeseune (2003). "Syzygy: native PC cluster VR". In: *IEEE Virtual Reality, 2003. Proceedings.* IEEE, pp. 15–22.

Scharnowski, Frank, Frouke Hermens, and Michael H Herzog (2007). "Bloch's law and the dynamics of feature fusion". In: *Vision research* 47.18, pp. 2444–2452.

Schmalstieg, Dieter, L Miguel Encarnação, and Zsolt Szalavári (1999). "Using transparent props for interaction with the virtual table". In: *SI3D* 99, pp. 147–153.

Schmalstieg, Dieter, Anton Fuhrmann, and Gerd Hesina (2000). "Bridging multiple user interface dimensions with augmented reality". In: *Proceedings IEEE and ACM International Symposium on Augmented Reality (ISAR 2000)*. IEEE, pp. 20–29.

Schmalstieg, Dieter, Anton Fuhrmann, Gerd Hesina, et al. (2002). "The studierstube augmented reality project". In: *Presence: Teleoperators & Virtual Environments* 11.1, pp. 33–54.

Schmalstieg, Dieter, Anton Fuhrmann, Zsolt Szalavari, et al. (1996). "Studierstube-an environment for collaboration in augmented reality". In: *CVE'96 Workshop Proceedings*. Vol. 19.

Schulze, Jürgen P et al. (2013). "CalVR: an advanced open source virtual reality software framework". In: *The Engineering Reality of Virtual Reality 2013*. Vol. 8649. International Society for Optics and Photonics, p. 864902.

Shaw, Chris, Mark Green, et al. (1993). "Decoupled simulation in virtual reality with the MR toolkit". In: *ACM Transactions on Information Systems (TOIS)* 11.3, pp. 287–317.

Shaw, Chris, Jiandong Liang, et al. (1992). "The decoupled simulation model for virtual reality systems". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 321–328.

Snowdon, D and Adrian J West (1994). "The AVIARY VR-system. a prototype implementation". In: *6th ERCIM Workshop*.

Spohrer, James C. (1999). "Information in places". In: *IBM Systems Journal* 38.4, pp. 602–628.

Steed, Anthony, Jesper Mortensen, and Emmanuel Frécon (2001). "Spelunking: Experiences using the D ive System on CAVE-like Platforms". In: *Immersive Projection Technology and Virtual Environments 2001*. Springer, pp. 153–164.

Steuer, Jonathan (1992). "Defining virtual reality: Dimensions determining telepresence". In: *Journal of communication* 42.4, pp. 73–93.

Sutherland, Ivan E. (1965). "The Ultimate Display". In: *Proceedings of the IFIP Congress*, pp. 506–508.

Sutherland, Ivan E (1968). "A head-mounted three dimensional display". In: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, pp. 757–764.

Sverdrup, Lawrence H. and Mikhail Belenkii (2010). *Dynamic foveal vision display*. US Patent US20120105310A1.

Szalavári, Zsolt et al. (1998). ""Studierstube": An environment for collaboration in augmented reality". In: *Virtual Reality* 3.1, pp. 37–48.

Taylor II, Russell M. et al. (2001). "VRPN: A Device-independent, Network-transparent VR Peripheral System". In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '01. Baniff, Alberta, Canada: ACM, pp. 55–61. ISBN: 1-58113-427-4. DOI: 10.1145/505008.505019. URL: http://doi.acm.org.eres.library.manoa.hawaii.edu/10.1145/505008.505019.

Tecchia, Franco et al. (2010). "A flexible framework for wide-spectrum VR development". In: *Presence: Teleoperators and Virtual Environments* 19.4, pp. 302–312.

Thorpe, Jack A et al. (1987). *The SIMNET network and protocol*. Tech. rep. BBN LABS INC CAMBRIDGE MA.

Torre, Rémy et al. (2000). "Interaction between real and virtual humans: Playing checkers". In: *Virtual Environments 2000*. Springer, pp. 23–32.

Tramberend, Henrik (1999). "Avocado: A distributed virtual reality framework". In: *Virtual Reality, 1999. Proceedings., IEEE*. IEEE, pp. 14–21.

— (2001). "A display device abstraction for virtual reality applications". In: *Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation*. ACM, pp. 75–80.

Tran, Frédéric Dang et al. (2002). "An open middleware for large-scale networked virtual environments". In: *Virtual Reality, 2002. Proceedings. IEEE*. IEEE, pp. 22–29.

Trenholme, David and Shamus P Smith (2008). "Computer game engines for developing first-person virtual environments". In: *Virtual reality* 12.3, pp. 181–187.

Wagner, Daniel and Dieter Schmalstieg (2003). *First steps towards handheld augmented reality*. IEEE.

Wang, Ming-Jen, Chien-Hao Tseng, and Cherng-Yeu Shen (2010). "An easy to use augmented reality authoring tool for use in examination purpose". In: *Human-Computer Interaction*. Springer, pp. 285–288.

Waters, Richard C et al. (1997). "Diamond park and spline: Social virtual reality with 3d animation, spoken interaction, and runtime extendability". In: *Presence: Teleoperators & Virtual Environments* 6.4, pp. 461–481.

Watsen, Kent and Michael Zyda (1998). "Bamboo-a portable system for dynamically extensible, real-time, networked, virtual environments". In: *Virtual Reality Annual International Symposium, 1998. Proceedings., IEEE 1998*. IEEE, pp. 252–259.

Weichert, Frank et al. (2013). "Analysis of the accuracy and robustness of the leap motion controller". In: *Sensors* 13.5, pp. 6380–6393.

Wheatstone, Charles (1852). "I. The Bakerian Lecture.—Contributions to the physiology of vision.—Part the second. On some remarkable, and hitherto unobserved, phenomena of binocular vision (continued)". In: *Philosophical transactions of the Royal Society of London* 142, pp. 1–17.

Williams, Betsy et al. (2007). "Exploring large virtual environments with an HMD when physical space is limited". In: *Proceedings of the 4th symposium on Applied perception in graphics and visualization*. ACM, pp. 41–48.

Woodgate, Graham J et al. (2000). "Flat-panel autostereoscopic displays: characterization and enhancement". In: *Stereoscopic displays and virtual reality systems VII*. Vol. 3957. International Society for Optics and Photonics, pp. 153–165.

Wozniak, Peter et al. (2018). "Towards Unobtrusive Obstacle Detection and Notification for Virtual Reality Using Metaphors". In: *Proceedings of the Symposium on Spatial User Interaction*. ACM, pp. 188–188.

Wu, Chien-Min et al. (2017). “A virtual reality keyboard with realistic haptic feedback in a fully immersive virtual environment”. In: *Virtual Reality* 21.1, pp. 19–29.

Xia, Haijun et al. (2018). “Spacetime: Enabling Fluid Individual and Collaborative Editing in Virtual Reality”. In: *The 31st Annual ACM Symposium on User Interface Software and Technology*. ACM, pp. 853–866.

Yamaguchi, Shun et al. (2018). “An Encounter Type VR System Aimed at Exhibiting Wall Material Samples for Show House”. In: *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. ACM, pp. 321–326.

Yoshinaka, Kazuo (1986). *Display Device*. Japanese Patent 61198892.

Yu, Difeng et al. (2018). “PizzaText: Text Entry for Virtual Reality Systems Using Dual Thumbsticks”. In: *IEEE transactions on visualization and computer graphics* 24.11, pp. 2927–2935.

Zimmerman, Thomas G. (1982). *Optical flex sensor*. US Patent US4542291A.

Zimmerman, Thomas G. and Jaron Z. Lanier (1987). *Computer data entry and manipulation apparatus and method*. US Patent US4988981A.

Zyda, Michael J, John S Falby, et al. (1993). “NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation”. In: *Computers & graphics* 17.1, pp. 65–69.

Zyda, Michael J, David R Pratt, James G Monahan, et al. (1992). “NPSNET: Constructing a 3D virtual world”. In: *Proceedings of the 1992 symposium on Interactive 3D graphics*. ACM, pp. 147–156.

Zyda, Michael J, David R Pratt, William D Osborne, et al. (1993). “NPSNET: Real-time collision detection and response”. In: *The Journal of Visualization and Computer Animation* 4.1, pp. 13–24.

Zyda, Michael, Don Brutzman, et al. (1997). “NPSNET-Large-scale virtual environment technology testbed”. In:

Zyda, Michael, David Pratt, et al. (1993). “NPSNET and the naval postgraduate school graphics and video laboratory”. In: *Presence: Teleoperators & Virtual Environments* 2.3, pp. 244–258.