ARCAD:
Augmented Reality
Computer Aided Design

Martin Joseph Costello

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Design

University of Washington
2018

Committee:
Axel Roesler
Jason Germany
Paul Hoover

Program authorized to offer degree:
Division of Design

University of Washington

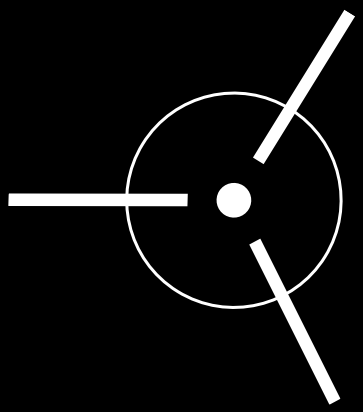
Abstract

ARCAD:
Augmented Reality
Computer Aided Design

Martin Joseph Costello

Chair of the Supervisory Committee
Axel Roesler
Division of Design

This thesis explores the potential for creation of three-dimensional, parametric computer aided design (CAD) models using augmented reality technology. The current standard – a flat screen – can only display two dimensions of information, which constrains the display of three-dimensional geometry as either perspective, parallel, or orthographic projection. Each projection type has trade-offs, resulting in distortion, inaccurate scale, or the need of multiple planar projections to capture three-dimensional space. Augmented reality technology overcomes the limitations of the flat screen – for the first time, holographic, truly three-dimensional display of geometry is possible and can be integrated into a real scene by using a head-mounted AR/Mixed Reality headset, affording a CAD display that imparts better understanding and accuracy during the entirety of the product development process. Using current devices and adapted practices, I developed a prototype mixed reality solid modeling environment to simulate the experience of using such a tool, and test how we might interact with it. This thesis documentation reflects the process of creating the interactive demonstration and reflects on the lessons and feedback gained from the students, enthusiasts, and industry professionals who engaged with it.



ARCAD

Augmented Reality Computer Aided Design

Joe Costello

MDes Thesis Documentation
University of Washington


RONIN
Ronin_Main_ASM

ABSTRACT

This thesis explores the potential for creation of three-dimensional, parametric computer aided design (CAD) models using augmented reality technology. The current standard – a flat screen – can only display two dimensions of information, which constrains the display of three-dimensional geometry as either perspective, parallel, or orthographic projection. Each projection type has trade-offs, resulting in distortion, inaccurate scale, or the need of multiple planar projections to capture three-dimensional space. Augmented reality technology overcomes the limitations of the flat screen – for the first time, holographic, truly three-dimensional display of geometry is possible and can be integrated into a real scene by using a head-mounted AR/Mixed Reality headset, affording a CAD display that imparts better understanding and accuracy during the entirety of the product development process. Using current devices and adapted practices, I developed a prototype mixed reality solid modeling environment to simulate the experience of using such a tool, and test how we might interact with it. This thesis documentation reflects the process of creating the interactive demonstration and reflects on the lessons and feedback gained from the students, enthusiasts, and industry professionals who engaged with it.

TABLE OF CONTENTS

Part 1:

INTRODUCTION

New mixed reality technology has the capacity to fundamentally change the way in which people interact with computer systems. These products are the culmination of hard-won technological leaps made through major investments, both financial and intellectual, the results of which we have barely begun to actualize. Though there is ambiguity surrounding this new technology, with closer inspection there are precedents, relationships, and applications for its integration into the design process.

One potential application is in providing designers a new platform from which to create. Designers who develop three-dimensional products in simulated environments could instead use mixed reality to better understand and interact with their creations—experiencing virtual designs blended into real environments at full scale and with realistic perspective change relative to gaze and point of view. For the past 18 months I have been developing an interactive simulation of what that platform might look like for professional computer-aided modeling and testing it with a variety of potential users. This thesis documentation reflects research into the history and compositional elements of 3-D computer modeling, as well as the process of creating and responding to arguments for an augmented reality computer aided design system.

DESIGNING DIGITALLY

Computer aided design (or CAD) is a method of three-dimensional drawing to create a digital model file within a computerized system. This model is a representation of a tangible design intent, or plan, for how to proceed with actualization. There are numerous applications, all with their own specialized strengths and weaknesses, that facilitate the creation of CAD models. Contemporary CAD applications can be considered within two major categories: surface modeling and solid modeling. When combined, these two categories approach ubiquity in many creative fields (discussed further in section 2). However, in their current form, each distinct application shares known limitations.

Except for artisan goods created by hand, nearly every gadget purchased, building constructed, and car manufactured were initially planned for production on a three-dimensional design program. While such creations were destined for the three-dimensional, physical world, they were conceived of and brought to life by a team of people using a two-dimensional medium: the computer screen. In the process of physical product development, designers must engage with a flattened design environment to create a digital model that simulates physicality. That digital model can then be proven in three dimensions by creating a physical prototype. Until recently, creating a CAD model, and coupling it with physical models of varying types and fidelities, represented the best option for bringing designs into the physical world.

The processes of three-dimensional modeling performed in the entertainment industries follow a similar workflow and are subject to similar constraints. Boundary-pushing film production companies like Lucasfilm Ltd, Pixar Animation, and James Cameron's Digital Domain employ armies of designers working on hundreds of individual

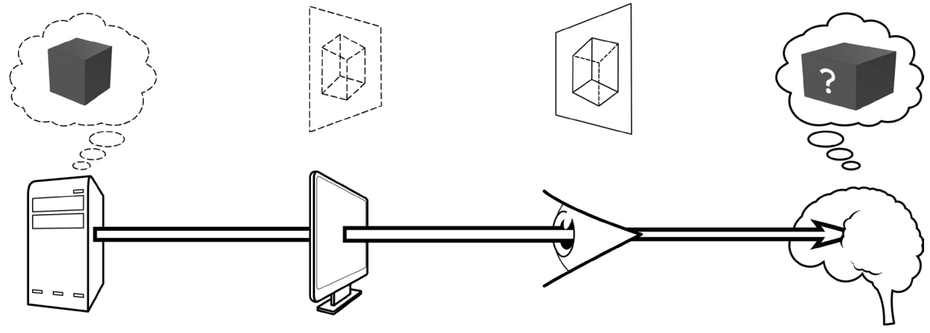


Fig 1: 2-D Perception

workstations laboring to perfect their big-screen visuals. The same is true of large video game development companies like Bethesda, Rockstar, and Blizzard. It is worth noting that these groups have a distinct advantage in that their end-products are intended to be viewed on flat screens amongst contextual geometric data, as opposed to the reality-governed geometry of the physical artifacts of industrial or architectural design. Perhaps their relative lack of physical constraints in the game development domain explains why they have been the first to embrace a new technology that eschews the stationary screen.

MIXED REALITY

Until recently, designers of all stripes have had to make use of interactive, but flat virtual images of digital files to experience the designs on which they are working. This construct can only be viewed through a window sitting atop a desk and could only be interacted with through crude prosthetics plugged into that window. However, in the same way that controlling the movements of the Mars rover from a terminal at NASA does not accurately depict the experience of being on mars, working on a CAD model at a desk provides creators with a fraction of the experience of having that intended product with them in the room. This is why no matter how wasteful, labor-intensive, expensive, and

sometimes dangerous building physical models can be—prototyping is still an immensely important part of the design process.

By the time a CAD workstation is employed in the design process, the designer often has a three-dimensional idea in their mind from which to work. This idea is the result of hundreds of hours spent researching the project, sketching the forms, understanding the ecosystem in which it will fit, deliberating the basic geometry, etc. Upon entering the CAD universe, the designer is forced to pan a camera around a virtual environment as viewed through a 2-D window, interacting with its contents, and modifying them through this restrictive portal. By clicking on different facets, reference markers, and icons to elicit different tools and reactions, one can try to build a digital representation by transforming a mental image into digital data; but the screen represents an enduring disconnect.

Mixed reality – that is the entire span of virtual and augmented reality– has the potential to bridge the cognitive gap for three-dimensional modeling. Computers are now powerful enough to append, or even supplant, reality as we see it with curated simulacra (Baudrillard, 1994). Instead of peering through a window into a disparate digital realm, we can bring the digital data into our world as a hologram that is anchored in, and aligned with, the physical world that surrounds it. We can experience this hologram, at least visually, as we would the real thing. Considering that many designers create three-dimensional virtual files, and mixed reality equipment exists specifically to allow us to experience environments that are essentially complex CAD files, blending the virtual world into the real world presents an immense leap in the design world. It may sound like an obvious marriage, but high-fidelity CAD modeling is not yet possible in mixed reality, thus the interaction paradigm remains firmly in untested ambiguity.

As technology perpetually advances, so too must the interactions upon which the tools of technology rely. Progressing beyond ambiguity is no longer an option, it is a necessity. As Christopher Alexander predicted:

> ***We must face the fact that we are on the brink of times when man may be able to magnify his intellectual and inventive capability, just as in the nineteenth century he used machines to magnify his physical capacity (Alexander, 1964).***

The rules for three-dimensional interface are as yet unwritten, but the tools upon which these rules would rely are here, now. Remarkable new tools create drastically new opportunities. Many current three-dimensional interactions have been adapted from the classic rules of two-dimensional visual design. While having undeniable merit, they often need to be appended, and sometimes overhauled entirely, when transposed into three-dimensions.

TERMINOLOGY

As with any field, especially those on the cutting edge, terminology can vary from within the industry, and even from within a company. In addition, the range of interchangeable terms found throughout existing computerized, synthetic reality research can lead to confusion. To be clear and consistent, this thesis documentation will leverage the following mixed reality terms as defined by “A Taxonomy of Mixed Reality Visual Displays”(Milgram & Kishino, 1994):

Head-mounted Display (HMD): the physical headset, usually containing sensors, display screens, speakers and the necessary apparatus, that one wears to engage in a digitally altered experience.

Virtual Reality (VR): an environment in which the participant-observer is totally immersed in, and able to interact with, an entirely synthetic world with no visibility outside of the HMD. Augmented Reality (AR): any case in which an otherwise real environment is augmented by means of virtual, computer-generated objects usually through transparent, mirror-equipped lenses.

Mixed Reality (MR): an environment in which real world and virtual world objects are presented together within a single display, anywhere on the spectrum between entirely virtual and subtly augmented realities. Microsoft has adopted this as the umbrella term for both their software platform, and suite of “Windows Mixed Reality” branded advanced display hardware.

Additionally, I will refer broadly to this technological sector as *Mixed Reality* (MR), while *Augmented Reality* (AR) and *Virtual Reality* (VR) will also be used intentionally as some differentiation is required.

Part 2:

RESEARCH

PRECEDENTS

Virtual reality, that is an artificial reality created by anything other than natural processes, has existed in a low-tech form for a long time. Storytelling creates meaning through a series of contexts, actions, and characters transmitted to a reader or listener. These principles combine to synthesize an imaginary world in the mind of the reader or listener with the intent of driving a story forward. This tradition has been reflected in music, theater, photography, animation, and film; all of which existed as a passive experience to be had by an audience whose options were limited to paying it attention, or not.

As technological artifacts made their way into our homes in the early 20th century, their design reflected this same disengagement. Radio towers sprung up around the planet beaming news and stories through the air to the specific devices engineered to catch them. The audience could adjust the frequency and the volume of these in-home radios, but no matter how loudly they spoke, the announcer on the other end could never hear them. Thankfully, along the timeline of technological evolution, this one-sidedness was short-lived.

One of the first mechanically-powered simulations of reality appeared in the late 1920's when a young Ed Link took an interest in flying. Finding the cost of flying lessons to be prohibitive, he began building a ground-based simulation

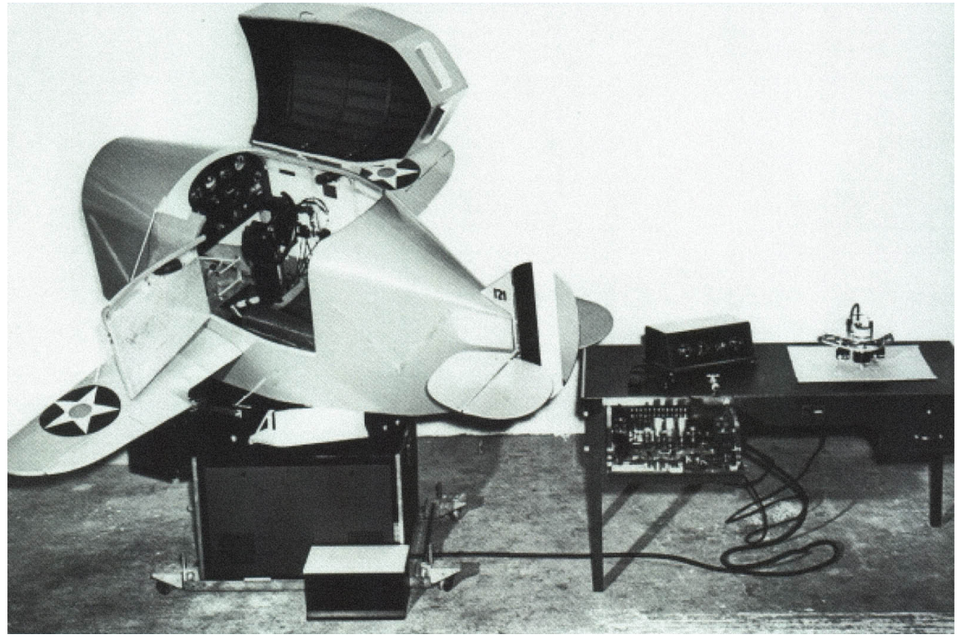
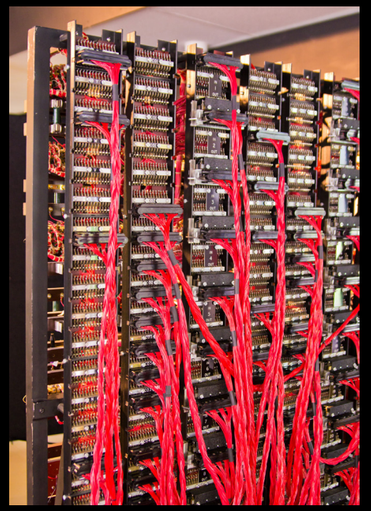
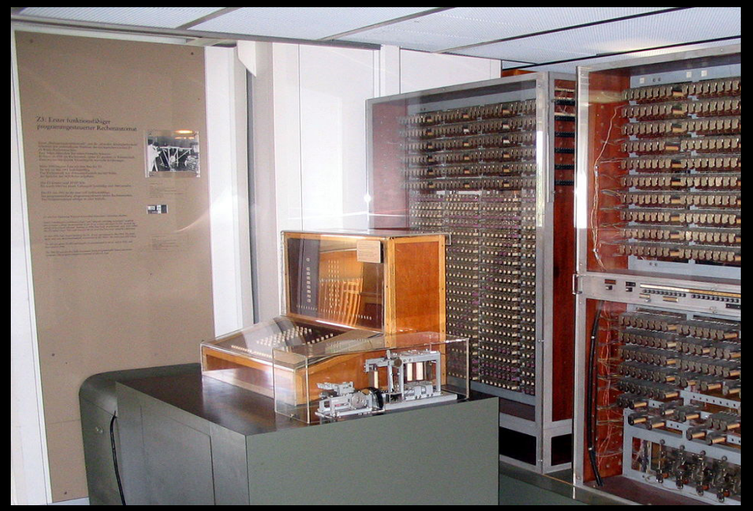


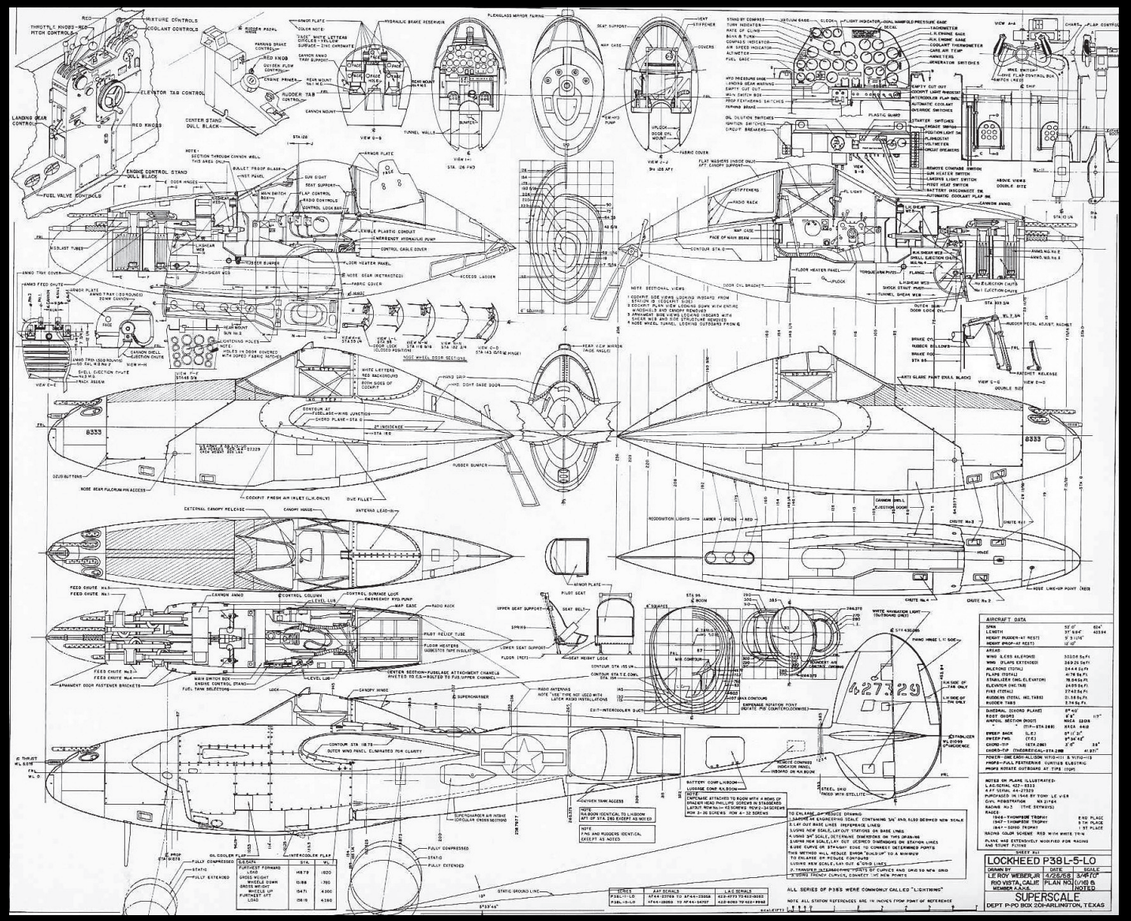
Fig 2: Link Trainer

of the experience of operating an airplane. By 1929 Link had developed the first flight simulator by mounting a smaller-scale airplane to a flexible universal joint. The scale aircraft had a full-scale cockpit, with working controls and gauges that felt like those of true airplanes (figure 2). It could also respond to pilot input, using electric motors and organ bellows, it could pitch and roll to simulate flying (Link, 2018). While not as informative as a true flight lesson, the power of simulation to teach and imbue familiarity was made evident and accessible.

By the mid-1900s, the burgeoning concept of a computing machine had begun to show a variety of applications and uses. In the 1930s, electromechanical computing machines were taking shape in certain arenas, but mostly for crunching numbers, not yet for amusement. At the 1940 New York World's Fair, the nuclear physicist Edward Condon debuted a newly-patented electromechanical







Upper Left:
Fig 3: Turing's Bombe

Upper Right:
Fig 4: Zuse's Z3

Lower Center:
Fig 5: Lockheed P-38 Technical Drawings

computing machine which played the ancient mathematical strategy game "Nim" (Norman J. , 2018). While it only used lights and basic logic circuits, it could certainly be argued to be the first "arcade-style," standalone game. A few years later the electromechanical computing machines would be implemented for much more dire circumstances.

World War II wrought a precipitous expansion of our technology and communications methods. As with any set of inventions this momentous, who invented the "first computer" is debated. Konrad Zuse created the Z3 in Germany (figure 4), Howard H. Aiken and IBM built the Mark-1 at Harvard University, and J. Presper Eckert and John W. Mauchly created ENIAC at the University of Pennsylvania Moore School of Engineering (Kalay, 2004). The allied forces developed computing machines, like "Colossus" (Crypto Museum, 2012a) and Alan Turing's infamous "Bombe" (Crypto Museum, 2012b) (figure 3), to decrypt axis communications to anticipate movements and orders, keeping them one step ahead of the Nazi war machine. One could imagine a very different outcome without these minds and machines.

The computers and humans that cracked codes were vital to winning the war, but so too were the machines used on the front lines. The arms, planes, tanks, and boats developed during this time were all mapped out on flat sheets of paper with pencils, rulers, and engineering know-how. Designers today still marvel at the imagination and tenacity it must have taken to develop three-dimensional, curvilinear, organic forms using scale models and blueprints; but two-dimensional schematics to represent three-dimensional forms were difficult to modify and could only represent a portion of the design planning. These technical drawings (figure 5), and the ones like them spanning back to DaVinci and earlier, would become digitized by the next generation and computer-aided drafting would be born.

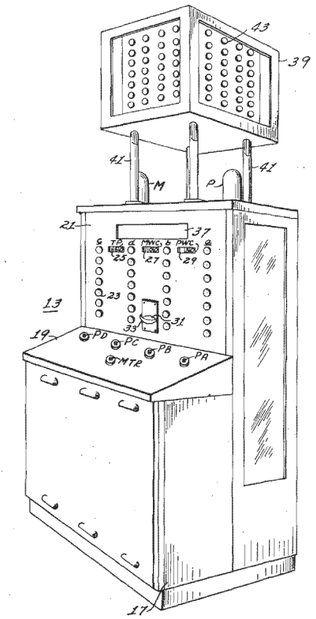


Fig 6: Game of Nim

Most of the technological progress made during the war was in the name of national self-preservation. The end of the war saw much of that technology publicized, while the mental and material resources could be redirected away from the war effort. The late forties and fifties were a time of national expansion. Technological progress became a national obsession and several artifacts, like the television, started making their way into the norm. By 1958 nearly 42 million television sets had been sold (Buffalohistory.Org, 2018), but almost all of them were destined to elicit the same lack of feedback that was endured by radio listeners. By appropriating this visual technology and attaching it to a new machine, researchers began to close the feedback loop.

INSTITUTIONALIZATION

After the war, computers, as we would come to know them, started popping up in academic and governmental institutions. MIT's TX-2 computer's main interface was very similar to that of a television, though what it showed was not based on the reception of local programming but on what was programmed to be shown by the room-sized computer to which it was attached. On that TX-2 computer, Ivan Sutherland of the MIT Lincoln Laboratory debuted a program called "Sketchpad" in 1958 (Dalakov, 2018). By drawing on the screen using a "light pen," or *stylus* as we would call it today, Sketchpad was programmed to produce mathematically-constrained lines and shapes that mimicked that which our compasses and rulers had been doing for centuries (figure 8). When making sketches and blueprints on paper the designer had a single attempt at connecting two points or drawing a circle around a center-point. If that attempt was in any way imperfect, it had to be erased and re-attempted. Sketchpad was different. It was able to

Fig 7: Photo by Evert F. Baumgardner

store various points in its memory and if an end- or center-point needed readdressing, one simply grabbed those points, repositioned them on the on-screen canvas, and the program updated the lines in real time (Sutherland I. E., 1963). Whether he could imagine it being precedent or not, Sutherland's Sketchpad was the first of hundreds of programs capable of working this way.

Even the early computers were proving to be flexible enough to be used for both work and play. Elevation drawings, or blueprints, were the perfect application for this simple two-dimensional geometry. Computer aided drafting was born of a computer's ability to draw exacting lines and shapes and adjusting them when needed; this was to be the business side. But just as our ancestors had horse-drawn plows and racing chariots, a mathematical workhorse could produce both schematics and amusement.

A year later, in 1959, the Digital Equipment Company began producing the Programed Data Processor 1, or PDP-1. While it was initially sent to academic facilities, their goal was bringing computer interaction to all. MIT received their first PDP-1 in late 1961. By spring of 1962 the first truly engaging computer game was created by Steven Russell and a team of MIT programmers in their spare time (Lok, 2005). "Spacewar!" is a contest between two spaceships, which moved based on the switch-inputs of two humans, orbiting around a blackhole battling to shoot each other down for points (figure 9). There was no intended productive output to Spacewar!, it was purely for amusement. Because it was programmed to operate only on the PDP-1 and the team never planned to sell the game, it was given away to other colleges with the same computer systems. Thus, the first program to be dubbed a digital computer game was science-fiction-themed, based on newtonian physics, multiplayer, and open-sourced. These precedents are visible even now as the tools for this thesis demonstration were developed.

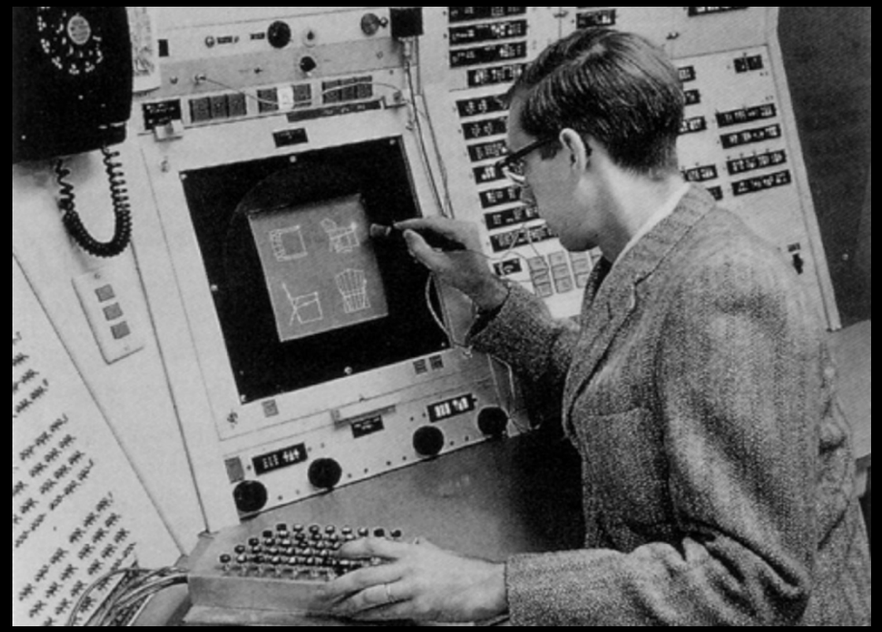


Fig 8: Sketchpad

Fig 9: Spacewar!

Sketchpad and Spacewar! Were the first programs that closely resemble the way in which we use our computers today. Instead of passively observing what appeared on-screen, as on a television, the operator was meant to engage with it and create a dialog. The computer displays something, the human changes something through built-in inputs, and the computer displays a response in real time. Because the results of that dialog manifested as digital visual representations, instead of a paper-based output, the feedback loop tightened significantly, as did the speed and quality of interaction. The inputs, computer rendered graphics, and programming language were underdeveloped by today's standards, but both efforts kickstarted the computer graphics world as we know it today. It also showed that games and design programs were siblings of the same parent, the computer, and would be destined to influence each other for decades to come. From this, the Graphical User Interface (GUI) would be born.

MINIATURIZATION

The use of computerized systems would continue to expand as the systems became smaller and more affordable. There were several important information technology milestones reached in 1964. The IBM System/360 was introduced using eight-bit bytes, integrated solid-state circuits, and removable tape drives for data storage, allowing computer system use in a much wider array of businesses (IBM, 2018). IBM, in partnership with General Motors, completed a computer program that could turn sketches into three-dimensional data (Computerhistory., 2018) which could then be sent to a computerized milling machine to be cut out. Christopher Alexander published *Notes of the Synthesis of Form* whose diagrammatical and algorithmic approach to the design solution discovering process hinted at the need for the use of computerized tools (Alexander, 1964).

The very next year Sketchpad's creator, Ivan Sutherland, presented a short but mind-bending paper (Sutherland I. E., 1965). In it he describes a display as a looking glass into the wonderland of mathematics, a need for curvilinear spline geometry, predicts the enduring ubiquity of a typewriter-like keyboard, explains the interaction benefits of a stylus, describes an early version of highlighting and dragging text, espouses the virtues of adjusting a viewing angle of an on-screen perspective view using a three-rotation joystick, and describes the shortcomings of interacting with a computer which can display only two-dimensions. Those benefits and complications were ongoing in 1965, but Sutherland goes on to describe solutions based upon computers capable of tracking our muscle movement and eye motion and their ability to respond accordingly; something he calls a kinesthetic display:

> ***There is no reason why the objects displayed by a computer have to follow the ordinary rules of physical reality with which we are familiar. The kinesthetic display might be used to simulate the motions of a negative mass. The user of one of today's visual displays can easily make solid objects transparent - he can "see through matter!" Concepts which never before had any visual representation can be shown, for example the "constraints" in Sketchpad. By working with such displays of mathematical phenomena we can learn to know them as well as we know our own natural world. Such knowledge is the major promise of computer displays (Sutherland, 1965).***

This paper was well ahead of its time in 1965. And its final paragraph remains slightly ahead of even our time:

> ***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 (Sutherland, 1965).***

Three years later, in 1968, Sutherland created a head-mounted display that projected simple, Sketchpad-like three-dimensional graphics, which reacted to viewer movement, on to transparent lenses (Sutherland I. E., A Head-Mounted Three-Dimensional Display, 1968). The lenses and their requisite assemblies and cables were suspended from the ceiling and looked a bit medieval, but the precedent set by these are visible to this day (figure 10). These works, and the ones that followed, would garner Sutherland sixty patents and twenty prestigious awards to date (Burton, 2012).

The physicality of the computer continued to advance over the subsequent decades along a mathematically-predictable line. In the mid-sixties, the computer scientist Gordon E. Moore noticed that the number of transistors capable of being put upon a processor chip had roughly doubled each year and he estimated that manufacturers would continue along that trajectory (Moore, 1965). In 1975 this estimation was amended from twelve months to twenty-four and “Moore’s Law” has been met or exceed since, though there are signs that this rate may slow (Simonite, 2016). As the physical transistor count increased, so too did the information

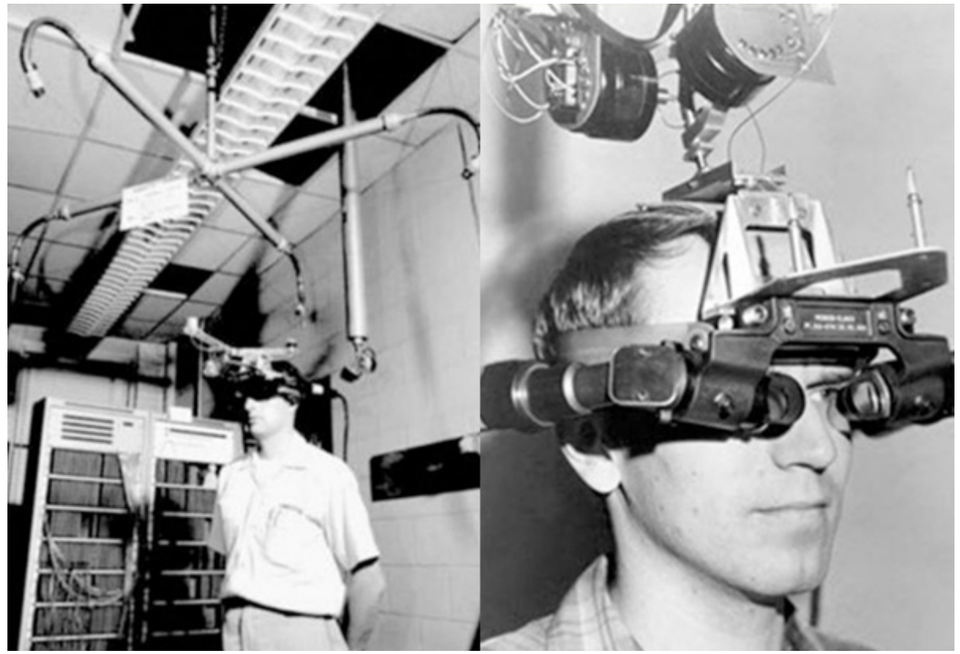


Fig 10: Sutherland’s Ultimate Display

throughput speed, which allowed our devices to grow in sophistication, usefulness, and accessibility while shrinking in physical size and cost. Intel says their current microprocessors are 3,500 times faster, 90,000 times more efficient, and 60,000 times cheaper compared to their first Intel 4004 microprocessor (Intel, 2018). Though, no matter how powerful these systems became, they remained beyond the comprehension of those less adept at the computer sciences.

DEMOCRATIZATION

Starting 1970, the team at Xerox-PARC created the Alto workstation computer and would come to produce, among several others, four important precedents: The first mouse designed for mass-production; layered window graphical user interface (GUI); bitmapping “Superpaint” program; and

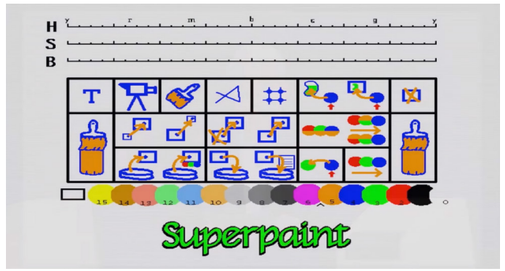


Fig 11: Xerox-PARC's Superpaint

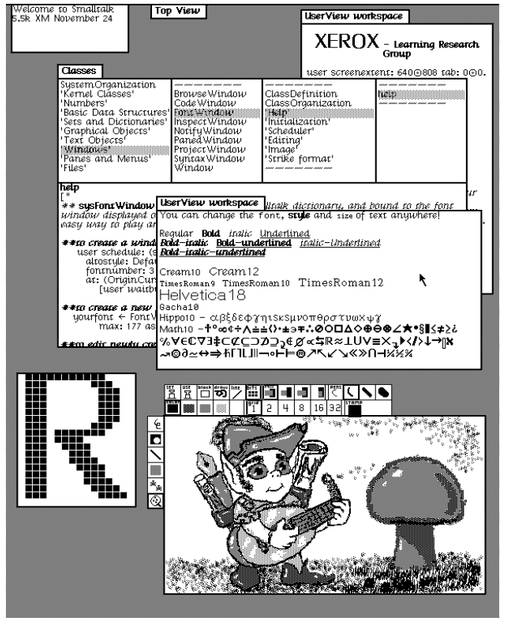


Fig 12: Xerox-PARC's Smalltalk

Sketchpad-like program called "Draw" (PARC, 2018). The organization of this system was oriented heavily on the input of the mouse. While they did not invent the mouse, they arranged an entire two-dimensional interface paradigm of windows, folders, icons, and layers around it. The "desktop" metaphor began here with rectilinear squares referencing different text input fields and tools. In Superpaint (figure 11) and Draw, the operator could create a vast array of shapes and lines with a variety of tools that were differentiated from each other by icon representations at the left of the screen. They were working to create a workstation that was expressly made to be the blueprint for "the office of the future" (PARC, 2018). While that could be considered a lofty charter for any group, they proved successful given that their influence in making the computer more usable for the average person is still evident in the products of Microsoft, Apple, Adobe, CAD developers, and more in the subsequent decades.

While PARC was working on interfaces in the seventies, large companies and universities were beginning to acquire relatively affordable workstations and were exploring how to implement the use of computing power for design and production. Computer aided design began to split during this time. Vehicle manufacturers needed to create complex surfacing geometry that could assist engineers in vehicle production. Architects, and by extension the building industry, needed to design in terms of layouts of simple shapes and volumes and were much less concerned with complex surfaces (Kalay, 2004). Dozens of approaches to these needs surfaced as stand-alone programs but were hamstrung by underdeveloped hardware and a lack of interoperability between systems. As the equipment evolved, CAD developers would persevere.

Just as was seen in the creation of Nim and Spacewar!, computers began making themselves useful and lucrative for amusement as well. In 1972, Magnavox introduced a television peripheral called "Odyssey" that could be plugged into a television and very simple digital games could be played, sometimes in concert with analog implements like dice and cards. This unit broke new ground and became known as the first home video game console (National Museum Of American History, 2018). It was ultimately ill-fated due to several factors, but one of the games was a simplistic table-tennis-like game would make an impression beyond the short-lived Odyssey in that it inspired the progenitor of an entire industry.

After the commercial failure of his first Spacewar!-facsimile arcade game "Computer Space" and having played the Odyssey's table tennis game (IGN, 2008), Nolan Bushnell enlisted Al Alcorn, a young electronics engineer, to recontextualize and improve upon the idea of a digital ping-pong game. "Pong" was supposed to be Alcorn's first attempt at building an arcade game so that he could learn the process. It was released in 1972 and became a runaway success. Proceeds from the game would fund the creation of the first video game development company Atari. While those in the laboratories at universities had been in contact with computers for some years, Pong was the first device upon which much of general public would first interact with a playful computer. Before the end of the 1970s Atari would

produce 34 more distinct arcade gaming computers (Atari Museum, 2018) and release their first home console: the Atari 2600.

The 1980s brought an expansion of computing systems to the masses as computers of varying forms started making their way into the homes, schools, businesses, and popular culture of Americans. As Moore's Law of computing power endured, the rise in computing power and reduction in price brought about innumerable new fields into existence; including research into virtual reality. As with the advent of the first computing machines, attributing the true first experience of synthetic reality is hotly debated. What is well-documented is that the precursors of virtual reality began to cross-pollinate with the smaller, faster, and more affordable computers of the 1980s allowing researchers like Henry Fuchs, Jaron Lanier, and Thomas Zimmerman to leverage it to try to change human-computer interaction.

In the mid 1980's VPL Research was founded by Jaron Lanier and the term "Virtual Reality" was officially coined (Lanier, 2017). Working with the likes of NASA's Ames' Aerospace Human Factors Research Division (NASA, 1990), VPL debuted their first working VR system in 1989. This system included a head-mounted display that is quite like the equipment available today, and dataglove that contained fiber optic cables that detected hand movement and extended the viewer's reach into the digital space (figure 14). These pioneering inventions landed on the front pages of the *Wall Street Journal* and the *New York Times* (Parkin, 2014) but the advanced technology was inaccessible to most with its price tag in the low six-figures. Commercial viability aside, these artifacts represented the first of hundreds of products to be made in their image.

These inventions, and the technology that emerged in the subsequent decades because of them, brought about the technological revolution that shaped the world we know today. As of 2015, the consumer technology sector represented 10.3 percent of the United States' GDP and was responsible for 8.4 percent of total U.S. employment (PricewaterhouseCoopers LLP, 2016). These numbers are projected to keep growing and the ten largest tech firms are projected to exceed a combined sales figure of one trillion dollars in 2018 (Shinal, 2018). From modified oscilloscopes and televisions to a tenth of the world's largest economy in just over half a century, computing machines have grown to be an integral part of daily lives both at work and at play.

Fig 13:
NASA's VIEW

Fig 14:
VPL Eyephone Debut
1989

COMPUTER AIDED DESIGN

Technological systems have found their way into nearly every corner of the creative fields today and there are countless professional creation programs serving a variety of functions. While the breadth of applications is dizzying, there are some foundational principles upon which they are commonly built. The following section enunciates some of the important principles of which one would need a working understanding to create for mixed reality.

Two-dimensional Graphics

Starting with the work at Xerox PARC, our digital imagery to this point has been based around the pixel. The pixel is a square box that is lit in an arrangement created by the computer to project a variety of colors onto a flat, two-dimensional screen. These boxes are arranged by the millions on a grid of, for example, 1,920-wide by 1,080-high on a standard high-definition screen. Each of these pixels are mapped on an X-axis for the width, and Y-axis for the height, and their state of display can be updated at upwards of 120 times per second. How these pixels are mapped depend on how the program being used tells the computer to display them. In general, there are two methods of displaying imagery: bitmap and vector.

Bitmap, or raster, are the file-types most commonly utilized to create life-like digital images; a photograph taken on a digital camera or mobile device, for example, is of this type. In its coding, a bitmap is a map like any other except, instead of cities, it cites the location and colors of pixels within a region as dictated by the file's resolution. At a high enough resolution, these images can appear very true-to life, but zooming in far enough on any given area will reveal the compositional pixels and their relative address. Superpaint on the Xerox Alto, as well as Microsoft Paint, MacPaint, CorelDRAW, and Adobe Photoshop were created specifically for working with these files.

Vector graphics, on the other hand do not seek to replicate photographic color depth. Arising from early oscilloscopes and first appearing on Sutherland's Sketchpad (Kalay, 2004), they are instead based on geometric spatial relationships and applied mathematics. Instead of pixel addresses, vector graphic programs assign addresses to user-selected (or process-created) points and draw mathematically-calculated lines between them to create geometric shapes. These lines can be straight, curved, round, and even complex curvilinear splines resulting in potentially complex polygons which, when creating a closed shape, can be filled in with solid color or raster-like gradients. Because the points are given a relative XY address that are connected by the program, these files are not subject to a loss of fidelity when zoomed into and could not become "pixelated" (figure 15). As a result, the same vector file can be used on a business card or billboard simply by changing the assigned physical distance between points. Because of the inherent precision and scalability, vector art is the standard for web pages, advertising, and, perhaps most commonly, typographic fonts. Although two-dimensional vector art applications have existed for several years in the design industry (e.g. Adobe Illustrator), but this same level of precision can be brought into a third-dimension, too.

Fig 15: Zoomed in Raster (Left) versus Vector (right)

Three-dimensional Modeling

Three-dimensional representation works similar to that of two-dimensional artwork but, in addition to an address about the X width and Y height, a third dimension, Z for depth, is created. In three dimensions, the flat canvas is no longer the referential priority in assigning point location. Points are instead given an address relative to a common “origin point” that is shared file-wide.

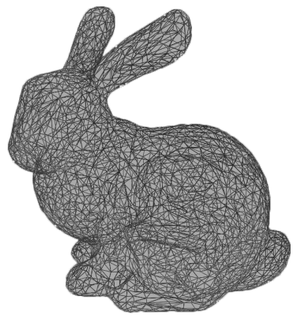
With enough of these points in three-dimensional space one can start to get an impression of the shape of the object in the form of a “point cloud” (Autodesk, 2018a). 3-D scanning and motion capture systems have made interesting use of the technology, but to further understand a surface you need lines connecting the points, which are then called vertices, denoting a true mesh “topology.” This wireframe mesh is the oldest form of computerized graphical representation of shapes (Kalay, 2004). With enough overlapping lines in a representational mesh, though, the overall form can become confounding, especially when viewed on a flat screen. To further describe the form, one would need to see surfaces.

As in vector graphics, a set of lines connecting vertices can create a closed polygonal shape; when created in the third dimension this becomes a face and the lines become edges. Viewing a set of opaque faces allows much more of the topology to be understood, especially when combined with motion. With a high enough polygon count, sets of faces begin to blend together to form smooth surfaces capable of describing complex organic forms (Fuchs, Kedem, & Uselton, 1977).

To further smooth out the polygonal faces as the complexity grows, the points in space can be interpolated into a smooth curve, called a spline. To represent a spline, and its connections to other splines as a 3-D surface, many design programs employ the mathematical model called NURBS, or “Non-Uniform Rational B-Splines.” NURBS can represent everything from a simple line, to extremely complex organic surfaces with accuracy and flexibility. Most exchangeable CAD file-types are derived from this mathematical model and the geometric information can move between programs for modeling, rendering, animation, and analysis (Rhinoceros, 2018).

Modeling programs like *Maya, Z-Brush,* and *Alias* that are surface-geometry-based have become industry-standard for transportation design, movie production, animation, and video games where appearances are the main priority. Because the files amount to an empty shell, with geometry that

Point Cloud



Wireframe

Fig 16:
Point Cloud (left)
Wireframe (Right)

Polygon-surfaced

NURBS-refined

Fig 17:
Polygons (left)
NURBS (Right)

has little-to-no basis in the physical world, a high-quality appearance is easier for the computers to render efficiently. That approach certainly has its utility, but these balloon-like, empty husk files lack one unit of measure that is important to industrial design, mechanical engineering, and architecture: volume.

Solid modeling, as it is called, was created to convey all necessary measurable attributes. These models are the closest to physicality as they describe vertices, edges, surfaces, and volume. They are constructed as solid masses, like clay or metal, and can be added to, subtracted from, and intersected with other masses to yield desired forms with extreme precision. This replication of physicality is important because the mass of these models translates directly to testable strength, weight, manufacturability, and material usage when paired with a known physical material in a simulation. A single file, or part, can also be paired with numerous other parts into subassemblies and assemblies so that mechanisms and spatial relationships can be simulated and verified. The accuracy of these simulations are high-stakes for those that employ the use of solid modeling because the process of these models gaining physicality, in the form of an injection mold, a skyscraper, or a space shuttle, can mean the investment of large amounts of resources. Often a mistake of any magnitude can have drastic effects and must be dutifully avoided.

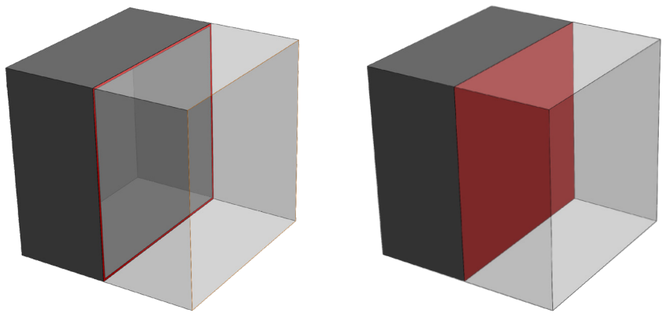


Fig 16:
Surface Model (left)
Solid Model (Right)

Surface Model

Solid Model

Fig 17: Ford GT Clay Model

However, for as simultaneously accurate, complex, and beautiful as a fully developed CAD model can be, they are still unable to impart the implicit understanding that comes with a physical model. This is why physical models of varying fidelity are still created. Ford Motor Company's design department uses 200,000 pounds of clay every year to create full scale models (Ford, 2018) of everything from their $400,00.00 GT supercar to the quintessential work truck F-150. In an interview with Ford's North American design director Moray Callum, Wired Magazine's Elizabeth Stinson explains that "while a computer-generated rendering might be precise, a computer model won't tell you what it's like to actually experience a car's design, standing next to it." Callum described it as such: "People still buy real cars ...They don't buy digital cars" (Stinson, 2018). But this assertion is only partially true.

SIMULATION

The average citizen may not be able to afford the Ford GT or have any particular need for a large F-150, but there are ways to simulate the experience of driving one. As of

August 2018, a cursory search on the PC gaming outlet *Steam* yields 214 results for “driving simulators” (Valve Software, 2018). For a comparatively low cost, one can buy a game that simulates driving everything from a Ford GT to a diesel railcar. A search for flight simulators, like the ones that evolved from the Link Trainer, yield another 192 results. The high-performance computers in use today, coupled with the high-level of craftsmanship with which game developers now create, can yield surprisingly realistic simulations. This is especially true if you take a driving simulator experience into virtual reality, of which 15 of the games listed are capable.

The equipment available today, particularly in VR, has progressed far enough to be quite effective at creating an illusion of the natural. Behind the scenes at most every gaming software development company are people designing three-dimensional surface models of characters, vehicles, buildings, and environments which are, in concert with a whole host of other encoded parameters, plugged into a “gaming engine” to yield a game. Game asset designers are bound by constraints that are inherently different than those of their physical design counterparts in fashion, transportation, architectural, and landscape design, but their results are not entirely dissimilar.

As a designer of 3-D physical products, and someone who has enjoyed the immersion of 3-D displays, an obvious question emerges: what if we were able to use this technology to stand next to our designs? Are our tools now capable of creating something like Sutherlands kinesthetic display? Moreover, what if we could combine the benefits of a kinesthetic display with the benefits solid CAD modeling to actively design a digital model while experiencing it in the physical world; just as we would a clay model, but cleaner, quicker and with much more precision?

My theory was that with enough effort we could, but what form would that take? To quote Christopher Alexander, I was:

> ***…searching for some kind of harmony between two intangibles: a form which we have not yet designed, and a context which we cannot properly describe. The only reason we have for thinking that there must be some kind of fit to be achieved between them is that we can detect incongruities, or negative instances of it (Alexander, 1964).***

As a designer, I had detected and experienced incongruities and frustrations between our current 2-D screens and new 3D viewers and intuited a strong potential solution for both a viable and desirable fit within the designer’s future toolkit. This initial hunch has since developed into a year-long investigation, research, ideation, and reflection. This process has validated my existing hypotheses about the limitations and possibilities of existing tools and systems. As such, I feel confident in my ability to propose a strong and informed answer to the aforementioned what-ifs, a result of delving into even more ambiguous territory (as articulated in section 3).

EXISTING TECHNOLOGY

A robust survey of the existing technologies was imperative before exploring a new path into Augmented Reality. Though I have a professional background in solid modeling, I began my research by experiencing as many screen-based computer aided design programs as was possible, discussing their various strengths and drawbacks with industry colleagues and students, analyzing tools and work-flows to understand the root of their interactions, and identifying the ways in which these programs could be improved. Simultaneously, I felt it important that I gain a familiarity and fluency with the burgeoning mixed reality market by analyzing a cross-section of games and creativity-based programs, talking to industry professionals, and experiencing them for myself. The following is a cross-section of the common 3-D modeling applications, virtual reality hardware, and VR modeling software in use today (Summer 2018).

CAD Programs

Except for the companies that are intentionally eschewing a digital process, computer aided design is how most physical artifacts are now designed. This omnipresent requirement for design creates the demand and necessity for several dozen programs capable of creating three-dimensional digital models, with innumerable niches and strengths. While learning to develop for mixed reality, many of these programs were referenced by designers with different workflows and, having encountered several in the field, I chose a few well-regarded examples to evaluate/study in-depth. Below is my brief synopsis of relevant programs.

Sketchup: A very simplistic 3-D surface modeling program, with a free version, that is especially adept at architecture modeling and champions itself as being “3D modeling for everyone” (SketchUp, 2018). It is noteworthy because it is a great program for educating first-time 3-D modelers as it is extremely simple. It is also connected to a “3D Warehouse” repository of free model files that are royalty free and compatible with numerous other programs.

Rhinoceros: A free-form, surface modeling program that is versatile and surprisingly simple to use. It relies on NURBS mathematical wireframing to produce an intuitive, clay-modeling feel that is especially adept at creating complex organic forms. Its fundamentally simple nature makes it a great introductory program for academia and is often part of an industrial design curriculum. Advanced users can also dig beyond the initial simplicity to access fine-grained surface controls and volumetric data.

Blender: A free, community-derived, open-source, versatile, 3-D modeling program. This program is fundamentally a surface modeling program that has been built with the support of a large community of users. The flexible and transparent program structure has spawned a vast user-base who create plug-ins and tools for use in the main program. The resulting options and uses are as complicated as one would imagine but a large community of users means a broad range of tutorials.

Autodesk Suite: *Autodesk* has become one of the largest companies in the 3-D modeling industry by creating a suite of programs that are devoted to tackling every aspect of professional 3-D modeling. One of their biggest draws is interoperability as the files created in each program are fundamentally compatible with most other programs in their suite. Their range of programs can be broken down into, and purchased as, the collections of architecture, product design, media, and *Alias* creation tools (Autodesk, 2018b):

Architecture: *AutoCAD*, the quintessential architectural and 2-D blueprint software; *Revit*, a structural engineering and building information modeling program; *Civil 3D*, a civil engineering and construction documentation program; *Infraworks*, a platform for geospatial and engineering planning and analysis; and several building and civil engineering-focused components are included.

Product Design: *Inventor Professional* is a parametric, solid modeling program with a mechanical engineering focus; *Fusion 360*, the first cross-platform, cloud-based modeling program which also blends solid and surfacing tools; *Recap Pro*, a 3-D scanning utility; *Nastran*, a finite element analysis tool; and several other production-oriented utilities are grouped with this package.

Media: *Mudbox*, a *Rhinoceros*-like, digital clay sculpting and modeling program; *Maya*, a 3-D animation and modeling program with an emphasis on organic characters and creature animation; *3DS Max*, a surface modeling program with support for animation and photorealistic rendering; plus, additional brainstorming and rendering utilities are included in this package.

Alias: A stand-alone family of programs designed predominantly for vehicle design. They are built to be used in chronological order, increasing in complexity as they progress. Listed respectively, they are *Alias Concept*, *Alias Speedform*, *Alias Surface*, and *Alias Auto Studio*.
The portfolio of programs available from *Autodesk* is broad and they are constantly innovating and introducing new tools. They have recently introduced a VR visualization tool through *3DS Max*, but they are limited to inspection, material application, and minor component arrangement (Amanda Di Pancrazio, 2017).

SolidWorks: Industrial design's primary solid modeling program, *SolidWorks* is a powerful desktop CAD program from Dassault Systems. This program is capable of complex surface and solid mass modeling, as well as intensive physics simulations and photorealistic rendering. It is quite precise, complex, versatile and infamous for its accuracy demands and confusing UI, thus having a steep learning curve compared to, for example, clay-type digital modeling.

The most basic mass in *SolidWorks* is a solid body; one or more bodies can comprise a part file; one or more part files (which can contain multiple bodies therein) can be saved as an assembly file (figure 17); for organization's sake, multiple assembly files can even be saved into one higher-level assembly file. Each of these parts and assemblies can be built to have complex physical relationships to one another for testing a variety of physical conditions. This assembly file structure is obviously complex but effective and was created to be effective at organizing very complex plans, from vehicles to entire factories.

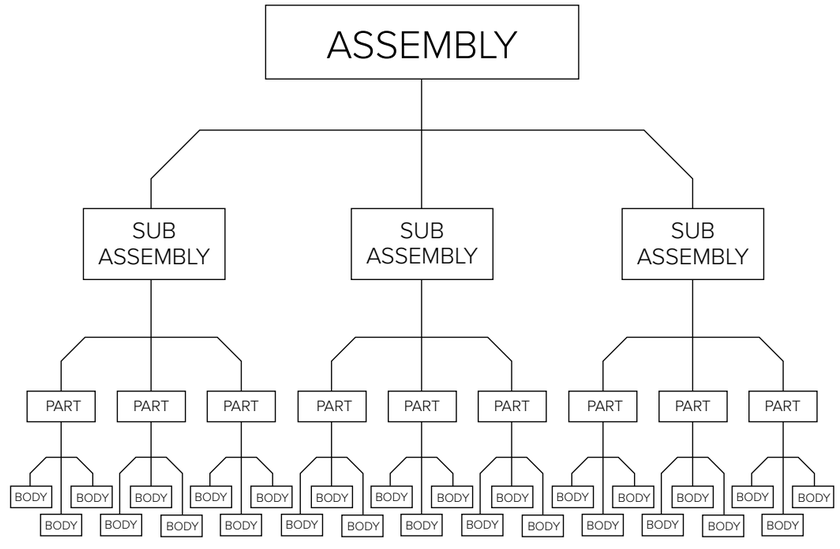


Fig 17: Typical Solidworks Assembly File Structure

Geomagic Freeform: As part of my survey I visited a custom medical implants design team to see their unique process. This set of designers and engineers use a volumetric clay modeling program called Freeform to build 3-D printed prosthetics for human implantation. *Freeform* is a clay modeler, like *Rhinoceros*, that the team couples with a device called the *Touch X Haptic*. This peripheral has articulating arms that have been designed to simulate the physical feedback of pushing and pulling real clay while modeling. The organic modeling capabilities of this arrangement are mandatory as they work with primarily 3-D scanned, biological CAD files to create prosthetics that help to repair bone damage or birth defects.

In my examinations, I found that several professional-grade CAD programs, especially solid modelers, work in surprisingly similar ways. The parameter-based model creation process begins with a vector sketch on a plane, that is dimensioned to a specific size across two axes, as dictated by the position of the plane, and brought into the third by a specific tool. For example, if a 24-inch cube was the goal, you merely draw a two-foot by two-foot square on a desired plane and extrude the square by two feet in the direction perpendicular to the drawing plane, thus achieving a cube. To cut a circular hole through that cube, a designer selects the face or plane onto which you will begin the hole, draw a two-dimensional circle's diameter, then subtractive-extrude the circular hole. The 2-D sketch, the plane on which to draw the sketch, and the tools to bring it into the third dimension was at the core of these programs and needed to be addressed in any transitional attempt.

Fig 18: 3D Systems Touch X Haptic

HEAD-MOUNTED DISPLAYS

The virtual reality market is currently saturated with devices of varying quality which can generally be broken into two categories: mobile and gaming. I knew at the beginning of my research that I wanted an experience with the most technical possibilities. So instead of the less powerful, mobile-handset-based headsets, I acquired the gaming-grade virtual reality equipment that was reasonably affordable, compatible with many existing programs, and could serve as benchmarks for performance:

HTC Vive: A gaming-oriented VR system from electronics company HTC that is closely tied to Valve Software's PC gaming platform *Steam*. It has dual AMOLED screens that are 3.6 inch diagonally, with a resolution of 1080 x 1200 pixels each, running at 90Hz, and combined providing a 110-degree field of vision (HTC, 2018). It uses a minimum of two fixed, tracking stations, placed within the room of use in line of sight, which track location sensors on the HMD and controllers. This wireless room tracking is coupled with front-facing camera, G-sensor, gyroscope, and proximity sensors built-in to the equipment to monitor location and movement while in use. This data is fed to a PC via USB tether, interpreted by the computer, and the rendered visual result is fed back to the dual AMOLED screens in the HMD through the second HDMI tether.

Samsung Odyssey HMD: The *Odyssey* was released, amongst other offerings from Lenovo, Acer, and HP, as part of the first wave of Windows Mixed Reality devices. It has dual AMOLED, 3.5inch screens, with a resolution of 1440 x 1600 pixels per screen, running at 90Hz, also providing a 110-degree field of vision (Samsung, 2018). While it is branded a "mixed reality" headset, and it was built for use on the Mixed Reality platform, the result is technically a VR

experience. The difference between Windows MR devices and the conventional VR devices such as the *Vive* or *Oculus Rift*, is that they don't required fixed, external room sensors to track their whereabouts. The system combines measurements from an array of internal sensors, similar to the *Vive's* accelerometer, gyroscope, compass, and proximity sensors, with two front-facing cameras that scan the space in which they are active to locate flat planes (like floors and walls) to record and store the space's geometry in memory. It actively rescans, while referencing the internal motion sensors, to update the observed location as the operator moves about a space. The cameras also track the controllers, which also contain their own movement sensors, by monitoring the relative location of a ring of small LED lights projecting outward from the handsets. This data is similarly sent via USB and Bluetooth to the computer, which then renders the result and feeds it to the HMD's dual screens via HDMI tether. This tracking method, dubbed "inside-out tracking," is the result of earlier Microsoft work on the Kinect and HoloLens (Microsoft, 2018a).

COMPUTER EQUIPMENT

Earlier versions of virtual reality fell flat, mostly because the supporting technology could not render a digital environment at a high enough frame-rate to feel natural to the human eye. This has, for the most part, been addressed in the current generation of mainstream devices thanks to refined componentry and the availability of graphically-powerful gaming computers which can render a VR environment at an impressive 90Hz. I used three computers in my exploration and their specifications are as follows:

Custom Desktop: Windows 10 Home, Intel Core i7-6700K @ 4.1GHz, 48GB DDR4 RAM @ 2999MHz, EVGA GeForce GTX 1080 FTW2 GAMING with 8GB GDDR5X, 500GB m.2 SSD, 500GB Samsung Evo SSD.

Dell Inspiron 7567 Gaming Laptop: Windows 10 Pro, Intel Core i5-7300HQ @ 2.5 GHz, 24 GB DDR4 RAM @ 2400MHz, Nvidia GTX 1050Ti with 4GB VRAM, 256 GB m.2 SSD, 500GB Samsung Evo SSD.

Microsoft Surface Pro 4: Windows 10 Pro, Intel Core i5-6300U @ 2.4 GHz, 8 GB LPDDR3 RAM, Intel HD Graphics 520, 256 GB SSD.

CURRENT VR MODELING SOFTWARE

Model creation in virtual reality has been researched as far back as 1992 (Butterworth, Davidson, Hench, & Olano, 1992) and is possible in a few well-developed programs now. By professional CAD standards, the resulting model files are decidedly underwhelming. However, a closer look reveals that much more can be gleaned by exploring how such interaction hurdles have been tackled/solved. The following VR modeling applications are used wearing one of the head mounted displays described in the previous section. It is also worth noting that because interaction is immersive, it requires two device hand controllers.

Google Tilt Brush: Google's easy and intuitive three-dimensional painting program with an obvious recreational focus. The controls are divided across both hands, which can be swapped, and each have their own distinct duties. One hand is the "brush," with which the user is given the option of various brushes sizes which appear to be shot out of the controller when a trigger is depressed. Surrounding the other hand is a set of four tool panels which rotate around the hand by swiping the corresponding trackpad. These panels contain the various brush types, file options, tools, and color wheel. Though these panels have a decidedly loose and sketchy UI, the tools represented are obvious and navigating them is intuitive and fun. However

enjoyable, the 3-D files created in Tilt Brush are very unrefined and, while they are fun to create, they have very little utility beyond the confines of the program itself.

Google Blocks: Also from Google, this free program allows the player to arrange, combine, and mildly modify pre-existing primitive shapes that come included with the program. Like *Tilt Brush*, one hand "holds" a tool panel, and one hand holds the selected tool. Each tool on the panel represents an optional effect one could have on the primitives created in the space. Shape to insert a primitive into the space, stroke to draw a line, paint to apply color, modify to stretch, and grab to reposition the primitives, along with the requisite file management options, are available on the tool panel. The panels are flat with subtle three-dimensionality and the tools appear blocky and as primitive as the shapes they create. Blocks is a bit more complex than *Tilt Brush* so somewhat less intuitive, but the results are significantly closer to real CAD models than is possible with *Tilt Brush*. That said, limited primitive shapes yield models at such low-fidelity that they border a retro or whimsical aesthetic. However, it is possible to export the files to share or import into more powerful programs.

Gravity Sketch: This tool employs freeform sketching in three-dimensional space, like *Tilt Brush*, but also allows the designer to create complex surface geometry that is like existing surface modeling programs such as *Alias* and *Blender*. Unlike *Tilt Brush*, these lines and surfaces can be adjusted after they are created to allow for further refinement. A tool panel is affixed to one hand and the tool is affixed to the other, unless a two-handed surfacing tool is in use. The UI is comprised of high-resolution primitive shapes that are refined and unobtrusive. The simple shapes work like physical button would in the real world and the brush hand turns to a drumstick-like object when in menus. The files created are at a high enough resolution to can be exported for use in other programs.

MasterpieceVR: This software is a modeling program that, in the tradition of *Rhinoceros* or *Z-brush*, acts like a clay sculpture studio brought into the virtual realm. The operator of this program can inject lumps of clay into the world around them and sculpt, pushing and pulling, areas of the surface to achieve the desired look. The two-handed approach is employed here, too, with a panel on one hand and the shaping tool on the other. The UI is minimalist and clear, if a little crowded. *MasterpieceVR* was among the first to allow multiple users to join in the same virtual "room" to collaborate in file creation. The files created can be very high resolution and can also be exported to more powerful programs, or even 3-D printed.

There exists another half-dozen programs of this type but I found these to be the most instructive and well-executed. While none of them can meet the precision necessary for professional modeling, but they existed as a proto-proof of concept for what I had theorized. The digital sketch models that I created in *Gravity Sketch* and *MasterpieceVR* felt extremely true-to-life, almost as if I could physically touch them, despite floating in a digital room stored somewhere on my hard drive.

Fig 19: Google's Tilt Brush

Part 3:

PROCESS

Digital creation tools and games have had an enduring presence in my life; but I also grew up building physical objects like Legos, models, erector sets, and playing with tools and mechanical toys. While these sets came with instructions, I found it more fun to deconstruct multiple sets and combine them into what I envisioned. The first time I tried to apply this dexterity under the hood of a car, the seeming chaos of large lumps of metal, plastic, wires, hoses, and shrouds might as well have been an alien spacecraft. Intellectually, though, I knew that human hands had crafted this self-perpetuating assemblage of parts, and that cars, like Lego sets, had their own instruction manuals. With the guidance of friends and repair manuals, I learned how to properly dissect these machines when needed, and how to put them back together better than they were before. I busted knuckles, lost bolts, installed improper parts, and pondered repairs for hours, but eventually, opening a hood was no longer an intimidating and confusing experience. Before I knew it, the automobile had been entirely de-mystified.

That same spirit was invoked for the synthesis of this thesis demonstration. Technology has always been a preoccupation of mine, but never had I programmed or written my own code. My professional experience in industrial design afforded me a tacit, functional knowledge of a variety of two- and three-dimensional design programs,

as well as the real-world implications of designing physical objects. In the Master of Design program at the University of Washington, I formally trained in the increasingly important field of Interaction Design and posited that 3-D mixed reality was the perfect realm by which to explore the combination of Industrial Design and Interaction Design principles.

Prototyping as Research

Research prototypes in design take a variety of forms and methods and, having tried a broad range of MR experiences, I could have created analogous physical prototypes and imagined an interaction workflow that would be beyond actualization for another decade, but would be unable to truly experience it in a genuine way. Instead, after some cursory research into creating content for mixed reality, I took the decidedly pragmatic approach of building a digital prototype. This took an immense amount of time and experimentation, some of which is documented in the following section, but I felt it to be the only way to be immersed in the current ecosystem and create for the technology with maximum authenticity. Through the products of this effort, I hoped to share a portion of the experiences I imagined with others, push the boundaries of current devices, and then envision the future from a deepened understanding of the present. To begin to achieve these ends, I would first need to develop a workflow for bringing CAD models into mixed reality.

My first steps in creating mixed reality files were humble. Since Microsoft released the “Windows 10 Fall Creators Update” in 2017, all Windows 10 machines now had “Windows Mixed Reality (VR),” “Paint 3D” and “Mixed Reality Viewer” programs built into the main operating system (Ruiz-Hopper, 2018). The “Mixed Reality Viewer” is a simple, three-dimensional file viewer that displays surface models like a photo viewer. It can display these models in a basic grey environment or access an attached webcam to place files on a flat surface in the camera’s field of vision. FBX, STL, OBJ, GLB, GLTF, PLY, 3MF are the importable file-types, none of which are fundamentally solid modeling files. To best utilize my existing skills and programs to create 3-D assets, I had to figure out how to translate native *SolidWorks* files, into common surface exchange files. STL, commonly used for machining or 3-D printing was allowable in the Mixed Reality Viewer, but when exported from *SolidWorks*, even at the highest settings, the resulting models were low-resolution and lacked color information.

In addition to the basic Mixed Reality Viewer, the next step was to work with the more advanced game development application Unity as platform for the creation of Mixed Reality applications, so I also investigated which 3-D models are importable to it. Unity supports direct file plug-in with surface modeling programs like *Blender, Maya,* and *3DS Max* (Unity 3D, 2018a), but entirely lacked support for any solid modeling program. Somehow, I needed to transcode to OBJ.

After further research, I discovered a thread on the official *SolidWorks* forum with a free, downloadable solution (Larsen, 2018). In his post, Neil Larsen supplies a macro that can be included as an “add in” to *SolidWorks*. A “macro,” or macroinstruction, is a set of simple coded instructions that translates a specific input to a predetermined output (Computer Hope, 2018). This particular macro strips away the volumetric data of a solid model and exports just the outer faces as a surface mesh, or set of meshes, and a second MTL file detailing the color attributes. It allows for fine-grained control of the output file’s resolution (figure 20) and, while the output is one OBJ file,

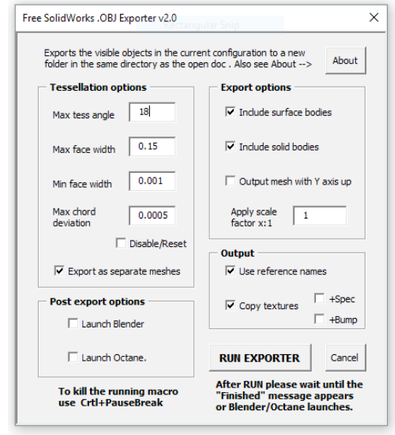


Fig 20: OBJ Export Macro

it can export the file with “submeshes” based on the visual attributes of the original file (figure 21); a color differentiation workflow that is also common for rendering photorealistic images in external programs like *Keyshot*.

Put simply, if one exported a cube which had material attributes, or colors, that were entirely blue except for one red face, the resulting OBJ would be comprised of two submeshes: a five-faced blue submesh, and a single-faced red submesh. It is important to remember that were the cube entirely red, it would be one red, cubic mesh and the individual faces would lack manipulability or the ability to be colored individually after export. It should also be disclosed that this macro struggles with complex arrangements of subassemblies; therefore, it is best to export from one, multi-body part file, or one assembly containing all needed parts, with material attributes that are unique or grouped by type (all aluminum parts the same color, for example).

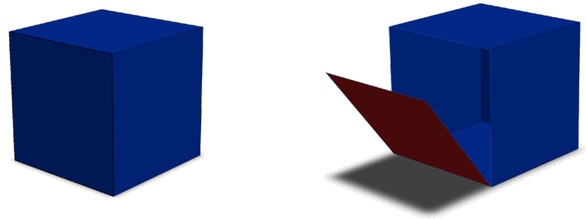


Fig 21:
single mesh cube (left)
two-submesh cube (right)

After developing this workflow, I was able to bring *SolidWorks* files into the Windows Mixed Reality Viewer and, using the Surface tablet’s rear-facing camera with depth sensing, viewed my first “solid” model in mixed reality (figure 22). Gone was the need to awkwardly spin a digital world to get a sense of the model’s shape, and the inability to determine the true scale: it was sitting right there on the table. This was a revelatory step, but it still amounted to viewing a prototype through a window held out in front of me However, I had a much more ambitious goal in mind—moving beyond a conceptual “looks-like” demo and build a “works-like” modeling shop, that could actually build prototypes.

Fig 22: Windows Mixed Reality Viewer

HOLOLENS

It was at this point in my research that I was given access to the Microsoft HoloLens for testing. The HoloLens is the first mass-produced device to define the category of augmented reality, or as Microsoft calls it, “the first fully untethered holographic computer” (Microsoft, 2018b). It is a groundbreaking device in that it can project holograms (otherwise known as 3-D models) onto transparent lenses and render them in real time. It employs six cameras, an inertial measurement unit, four microphones, and an ambient light sensor to track the room and its position in it. A set of processors crunch this input data and push the renderings out through two “light engines” (otherwise known as projectors) which project 2.3 million points of light (otherwise known as pixels) onto transparent lenses, creating one 1268 x 720-pixel transparent display per eye (Microsoft, 2018b). Compared to the main VR products available, the specifications are underwhelming when assessed by the visual specifications alone. More impressive is understanding that Microsoft has packed the equipment necessary to do all that processing, plus a battery, Wi-Fi and Bluetooth radios, speakers, and the head-mounting apparatus into a stand-alone mobile computing device that works as intended.

Fig 21: HoloLens

I downloaded and experimented with an entire spectrum of games and applications available for the HoloLens. There were 3-D model viewers and 360-degree videos, digital sharks and measurement tools, but at the time, a mostly a somewhat underdeveloped ecosystem of applications that was commensurate with the cost-prohibitive hardware on which it ran. Once I felt I had a reasonable grasp on what others had done, I took to the tutorials to start creating my own environments and interactions.

UNITY 3D

It is perhaps fitting that video gaming has led the way in mixed reality development. The computer aided design software companies create enterprise tools that emphasize user-familiarity and stability, thus it tends to be conservative, painfully steeped in incrementalism, and slow to adopt new technologies. The level of precision and reliability required by CAD can put a workstation through its paces when creating renderings or physics simulations, but those tend to be secondary to model-creation processes.

Conversely, in the realm of gaming, especially PC gaming, software is constantly pushing the boundaries of data throughput for graphics processing, memory resources, and disk speed. For example, widely-distributed games must include adjustable graphics quality because a newly-released game may not be playable at maximum quality on a graphics card that is six months old. Flexibility is foundational to gaming and, thanks to options like pre-existing game engines, developers can react quickly to hardware advancement.

During development, most video games require a common foundation of coded utilities and underpinnings upon which the overt content is stacked before functionality is achieved. This core set of principles and parameters can be created from scratch or acquired in the prefabricated form of a “game engine.” Unity is one example of this and explains itself as such:

> ***A game engine is a framework for game development that supports and brings together several core areas. You can import art and assets, 2D and 3D, from other software, such as Maya or 3s Max or Photoshop; assemble those assets into scenes and environments; add lighting, audio, special effects, physics and animation, interactivity, and gameplay logic; and edit, debug and optimize the content for your target platforms (Unity 3D, 2018b).***

As a service to streamline the process of creation, game development companies started licensing their games' fundamental framework to other developers in exchange for licensing fees. There are dozens of engines which are treated with different approaches, ranging from closely guarded intellectual property to free to download; Unity is of the latter. Microsoft has selected Unity as a development partner for creating content for their mixed reality platform, allowing anyone who wishes to tinker to download the engine, follow tutorials created by Microsoft, and learn the basics of creating for mixed reality environments. The home of these free tutorials is called the "Mixed Reality Academy" (Microsoft, 2018c).

To begin familiarization myself with Unity, I followed the specific installation method listed on the "Install the Tools" page of the academy website (Microsoft, 2018d). It is important for the sake of time-savings that the Unity installation instructions be followed exactly, as the tutorials in the academy provide not only video instructions, but prefabricated, downloadable Unity environments (called scenes) complete with all the necessary files, called assets, and the computer code to get them running. These various components were created by the Microsoft teams using whichever long-term-support (LTS) version of Unity is listed, plus additional required Unity elements that are listed in the "optional installation" panel in the Unity install dialog and need to be manually checked for installation - "Universal Windows Platform," ".NET scripting backend," and "Visual Studio". These steps are mandatory, and any subsequent progress will be hampered if these foundational instructions are not followed exactingly.

The Mixed Reality Academy tutorials, especially the "basic" and "input" sets, introduce several important terms and concepts for designing Unity scenes for mixed reality, but do very little to define them. Some of the terms that I found important, as defined by the Unity User Manual (Unity 3D, 2018c), appear in Appendix I at the end of this document. It is by no means an exhaustive list of terms but can serve as a high-level glossary of the basics. The online Unity User Manual is well-managed and should be consulted for further research.

MIXED REALITY TOOLKIT

Once I had completed the tutorials on the Mixed Reality Academy site, I realized that the academy files are merely scratching the surface of what is available for those of us who might wish to dig deeper. When following the instructions at the Mixed Reality Academy, scene files will be downloaded from a file repository on GitHub.com. A visit to the main section of GitHub from where these files came reveals a burgeoning community of developers working on, and sharing freely, open-source assets and coding that form the overarching Mixed Reality Toolkit. (Microsoft, 2018e).

After completing the package-importation steps and the "getting started guide" on the toolkit site, the local installation of Unity is automatically configured for experimentation in mixed reality. The tedious steps in scene, camera, and environment setup can now be automated with one click of the button: "apply mixed reality settings to scene." Included in the installation package are a number of scenes with prefabricated arrangements of assets and interactable elements meant to demonstrate what is possible with the toolkit. By experimenting with these elements' behaviors and reactions to input, and looking closely at how they are assembled, I was able to understand how they were structured and what could be extracted for my own demonstration, and what I would need to learn to build for myself.

VERSION A: THE MOUSE

[DECEMBER 2017 - MARCH 2018]

My original vision was a holographic console that could serve as the movable base of a desktop workstation, so I went about building it. As soon as I saw the console through the HoloLens, I received my first lesson in moving beyond the screen: many existing digital design paradigms are a result of the limitations of hardware. Holographic 3-D models are purely software and need not suffer these same constraints. A console, like a screen, is a physical constraint created by a lack of available display space and the need to concentrate a plurality of related controls to one area of access. An augmented reality workstation could exist in a nearly infinite digital space. Thus, the console disappeared as quickly as it appeared.

For the primary simulation of the "model being created," I wanted something that was simple and easily understandable for anyone experiencing the demonstration. I found a simple computer mouse to act as a placeholder CAD file and imported into the unity scene. I applied a glossy, plastic-like dark gray color which taught me a second important lesson in designing for mixed reality: the HoloLens projects holograms comprised of colored light onto its lenses and is incapable of "emitting darkness," thus visual attribute decisions should be made accordingly. I could see the white light reflecting off the glossy plastic but not the gray plastic itself. A shift to a very light gray increased the model's visibility dramatically.

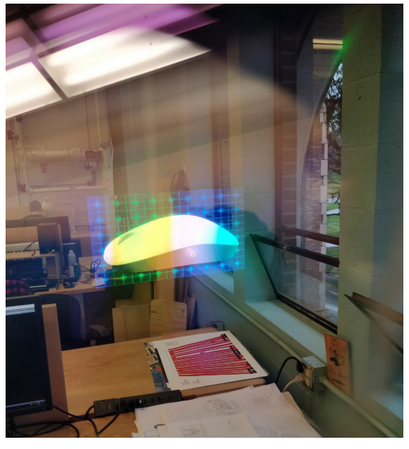


Fig 21: Mouse model in HoloLens

I knew early-on that a boundless digital workspace would require an overt feedback loop that explicitly acknowledges operator focus and reacts to commands. Like a mouse cursor on a screen, the primary input that extends the user's control into digital space was a three-dimensional cursor. As the main point of interaction with the models, it required careful consideration. The MR Toolkit provides a four-state cursor system that reacts to gestural input viewed by the HoloLens sensors. In a normal state, it is an unobtrusive light sphere in the center of vision; when the cursor is cast upon a holographic model, the sphere becomes more opaque and a doughnut-like torus appears around it, like a fat-ringed Saturn. When the HoloLens detects a hand within the gesture frame, the cursor grows in size to acknowledge input readiness; and during an air tap the ring collapses inward leaving just the orb. When cast upon the surface of a 3-D model, the torus rides along the surface helping to describe a model's topology and the cursor's location on it. On the whole the cursor's feedback system works rather well, but the meshes that comprised it are rather bulky and imprecise, unfit for complex modeling. I dug into the MR Toolkit's "InteractiveMeshCursor" prefab to locate the mesh cursor assets – the orb and the torus – to replace the ring with one of my own design. I wanted something with greater precision that could still react to the surface conveying its orientation. After a few iterations, I found my requirements met by reducing the orb's size by 75 percent and replacing the torus with a thin ring with a three-pronged, concave crosshair with extruded fins (figure 22). This cursor design is subtle enough to not impede on visibility, but precise enough to select minor details and convey its orientation on a surface, even at a distance.

Building upon the familiar, I created one of the foundational elements of solid modeling programs next, the plane. These planes are an important grounding point for a three-dimensional file as they give the user a sense of directional orientation and allow the foundational sketches to be created before being drawn into a third dimension. Planes in *SolidWorks* are flat surfaces

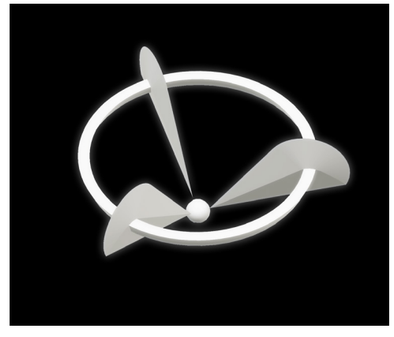


Fig 22: ARCAD Custom Cursor

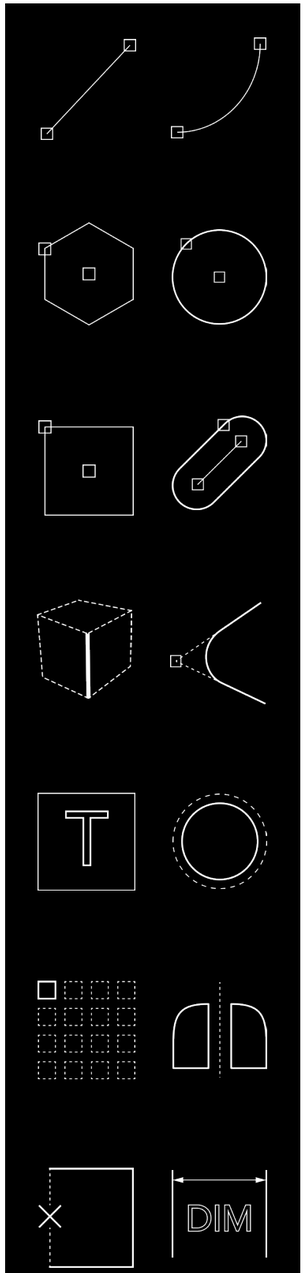


Fig 23:
Sketch Tool Icons

that scale relative to the model file and are prominent without being intrusive. In *Fusion 360*, planes only appear when the designer is about to start a sketch and seem to extend infinitely but have a graph-paper grid on them that assist in orientation. Combining the benefits of both I used Unity cubes with 3:5 height to width ratio and a thickness of 0.001 meter to create a transparent blue material for them, and placed them in the scene attached to the common orientations of side, front and top.

I found in my survey that existing CAD environments are typically rife with visual clutter, but unnecessarily so as certain tools only needed to appear when their use makes sense. Sketch tools, for example, need only be present when creating or modifying a sketch, and sketches typically exist on planes. With this in mind, I created brackets that acted as the indicator of which plane was presently in focus, and a dedicated "sketch-edit" button on each plane. Using the MR Toolkit's "Interactive Button" script assigned to the plane, a single click would reorient the bracket to align with that plane, and a "long-press" of the plane caused the entire system to rotate to align the plane with the operator. When the sketch edit button was clicked, the system entered "sketch-edit mode" causing the 3-D model to become transparent, the other planes to hide away, and the brackets to slide open revealing tools associated with the creation and manipulation of sketches. Within the left bracket, a set of common sketch tool icons appear which were minimalist, monochromatic, and, like the sketches with which they interact, two-dimensional (figure 23). These icons were necessarily simple graphical representations as the amount of coding that would be required to create interactive sketches was simply beyond the scope of this thesis.

Within the right bracket, though, I created an active set of controls for the plane's position, relative to the operator, to test the necessities of movement while interacting with sketches in AR.

- **Exit:** Replacing the "edit sketch" in the upper right of the plane, where it is commonly placed on screens, is an exit button that closes sketch-edit mode, thus sliding closed the brackets, hiding the tools, and causing the rest of the system to appear again.
- **Model Transparency:** The first button within the brackets was a model transparency button. Currently, when in a sketch-editing mode in current programs, the only thing visible is the sketch being created or edited; sometimes this is advantageous, and sometimes a nuisance. This button allows the transparent model with which the sketch is associated to be toggled on and off.
- **Follow Operator:** When starting a sketch on a plane in conventional CAD programs, it is common for the program to automatically orient the camera to a view that is orthographic (without simulated perspective) and exactly perpendicular to that plane. When using AR, the designer is the camera and perspective is not falsely simulated on a 2-D screen, but sketches are still often best viewed perpendicular to the viewer lest they become skewed by perspective. With that in mind, I appropriated a script from the MR Toolkit that causes the sketch plane to "Billboard," that is "face the user," and created a toggle switch to turn this feature on and off.
- **Manual Repositioning:** The third control button allows the plane, and indeed the whole environment, to be clicked and dragged in 3-D space for manual repositioning without respect to the position of the camera. I imagine this ability should be purposefully toggled, so as not to disrupt sketch-editing.
- **Tilt Plane:** Depending on the size and location of the model, simply tilting the plane about the horizontal axis could be useful. Inspired by the drafting tables used before CAD, and with help from another code from the MR toolkit, the forth button cycles through a tilting of the plane backwards from vertical 0, to -30, -45, -60, -75, and -90 degrees (horizontal).
- **Dotted Grid:** Most CAD and vector art programs have a visual grid for relative scale and positioning of sketches, the fifth button toggles on and off the visibility of a plane with a "grid of dots" material applied.
- **Line Grid:** Like the dotted grid, a graphing-paper-like line grid can be toggled on and off by the sixth button.

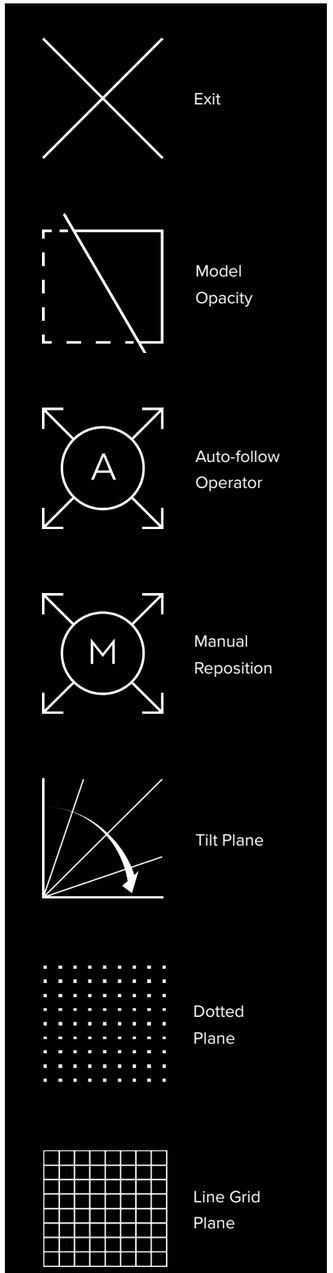


Fig 24:
Plane Control Icons

With no screen to confine the system and flexibility becoming increasingly important, the plane control features would prove necessary in the work and testing to come.

Another nagging drawback of the limited real estate of traditional screens is that if you delve into a foundational sketch to modify basic dimensions, you must exit that sketch to see how the changes will affect the finished model. Furthermore, CAD models being created or modified are typically displayed with oversimplified material attributes, and the designer must enter a rendering mode to get a photorealistic static image or "rendering." *Keyshot*, a dedicated rendering program, can create photorealistic representations a speed approaching real time, even while arranging the shot for rendering. Clay modeling programs that are based on direct surface manipulation are similarly able to show the results of modifications in real time. Combining the strengths of rendering and direct editing, allowing the designer to modify sketches while simultaneously seeing the results of their decisions in rendered form, could prove advantageous in form-giving.

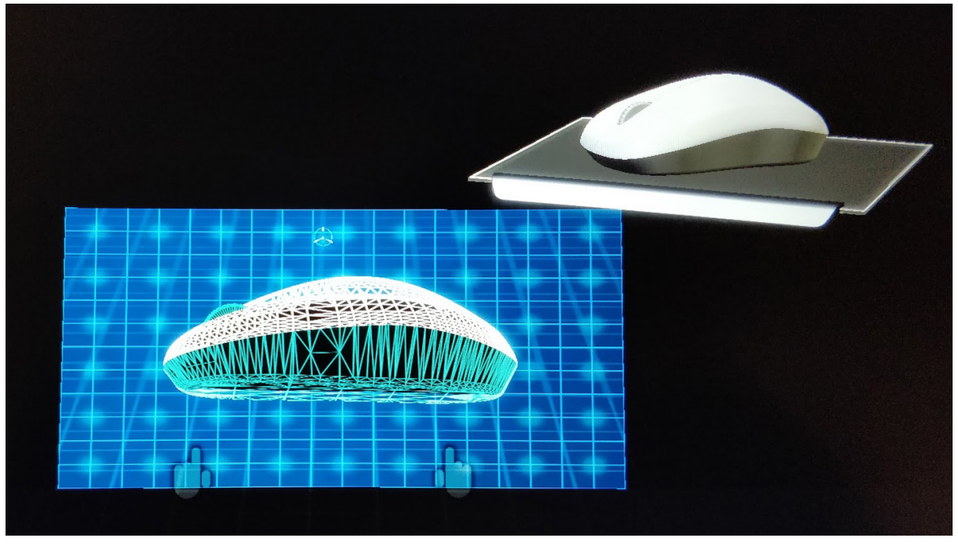


Fig 25: Version A

To test how this would feel, I made a copy of the mouse model and placed it on a tray-like surface separate from the planes and tools to allow for a "finished rendering" zone as separate from the "under construction" zone (figure 25). I imagined that there could be two instances of the model, and that one could be operated on while the other displayed the photorealistic consequences in real time. This is admittedly added complexity, and would strain the computers of today, but as an exploration it was enlightening.

Outside of the sketching and rendering environments, another compulsory component of every CAD program is the palette of tools that modify a model's topological features in three-dimensions; rounding the sharp edges with a fillet, for example. The tools that create and modify 3-D features have always been relegated to 2-D buttons with graphics that, up to now, weren't capable of accurately representing the three-dimensional features they invoke. This representational disconnect contributes to the steep learning curve of programs like *SolidWorks* and was quite obvious in the confused faces of the undergraduate industrial design students I observed. While brainstorming how to show these tools in the system, I realized that this shortcoming presented an opportunity to break from convention and exploit the nature of AR. Another lesson for mixed reality design: In addition to reduced constraints, mixed reality offers an opportunity to represent some items more authentically; that which is inherently three-dimensional can be presented as such.

Fig 25: 3-D Tool Icons

In response to the findings of my survey of CAD programs, I selected a set of primary tools that were common to any CAD system and created simple 3-D models as representative icons. Intuitive functionality should be the goal of any tool, as is taught by the study of product semantics, and with these icons a three-dimensional function could be inferred with greater immediacy compared to their artificially-flattened, screen-based counterparts. Secondarily, existing programs often separate adding mass and subtracting it into distinct tools – for example “extrude” and “extrude cut”. Since they are effectively the same process and could be derived from the same sketch, I omitted the unnecessary differentiation by creating an additive or subtractive button to accompany each feature. Since they were at times used in concert with a sketch and other times not, the “tool module,” as I would come to call it, became a free-floating palette of tools that could be moved about in 3-D space and collapsed to a single 3-D icon when not in use.

A shortcoming of the HoloLens began to make itself apparent as the complexity of my demonstration increased. As was stated by Microsoft, it was designed to be a stand-alone holographic computer that required no other inputs beyond what was observed by its tracking cameras, micro-phones, or included Bluetooth clicker. While wearing the headset, a cursor is placed directly at the center of your vision, which is combined with gestures made in front of the user within the “gesture frame,” or tracking cameras’ visible range. The HoloLens recognizes only two gestures as primary input mechanisms: the bloom, an upward-facing palm opening that acts as a home button, and the two-state air tap, a mid-air pinch with the fingers pointed outward that is akin to a mouse click (Microsoft, 2018f). For designers familiar with a mouse and keyboard, this was not an intuitive way to interact with a digital world.

I gave my first demonstration of an early version of the AR CAD system to a Vice President of Product Development and his design team. Their company is very CAD-oriented and designs precision military equipment at their product development arm in Denver, Colorado. As I booted up the system and placed the holographic assembly at the center of a conference table, the intuitiveness of the visuals was immediately clear. Upon donning the HoloLens, each team member walked around the model, testing its three-dimensionality and perceiving it from all angles. The designers appeared to readily understand the idea behind the demonstration and expressed desires to dig deeper into the model’s construction, as they would with a conventional CAD program, but were unable to do so given the demo’s singular “overall shape” sketch, represented in the sketch edit mode. Understanding the gaze tracking and gesture input was, however, less natural for the design team. The group felt impeded by the gesture frame and wanted an external implement to control the cursor. I felt the same way throughout the process and was determined to experiment with alternatives as the demonstration evolved.

ARCAD REVISION B: THE RONIN

[MARCH 2018 - JUNE 2018]

The Vice President who tested the first version of the demonstration was also involved in a design exercise that resulted in a short-run production of 47 intriguing motorcycles dubbed the “The Ronin.” For his initial exercise, a motorcycle was purchased, stripped down to the engine and frame, then almost entirely redesigned. Following the first proof of concept prototype, 47 more motorcycles were purchased and subjected to the same hand-built treatment (Ronin Motorworks, 2018). As is typical with mass-production, even a short-run like this, every major component of the entire motorcycle was turned into CAD part and assembly files. I was fortunate enough to be furnished with a copy of these CAD files for the next iteration of the system.

Unlike the mouse model, careful forethought was required in preparing this complex model to be exported from *SolidWorks*. The Ronin top assembly file was organized as several dozen subassemblies that had to be simplified to one assembly. A motorcycle is comprised of a myriad of material types, each of which needed to be grouped for accurate material attributes to be applied later, or divorced entirely, for individual manipulability post-export. A few components were missing, as were minor details like tire tread, which had to be researched and recreated for accuracy. Similar to preparing a model for rendering in an external program, if any of the file structure decisions were imperfect, they would need to be re-exported and patched into the Unity scene later. After a few inaccurate exports, I was finally able to bring one of the most complex CAD models I had ever worked with into mixed reality.

Mode States: After importing the Ronin into the system, I realized that the floating motorcycle seemed like a basic, ungrounded hologram with no file or state information

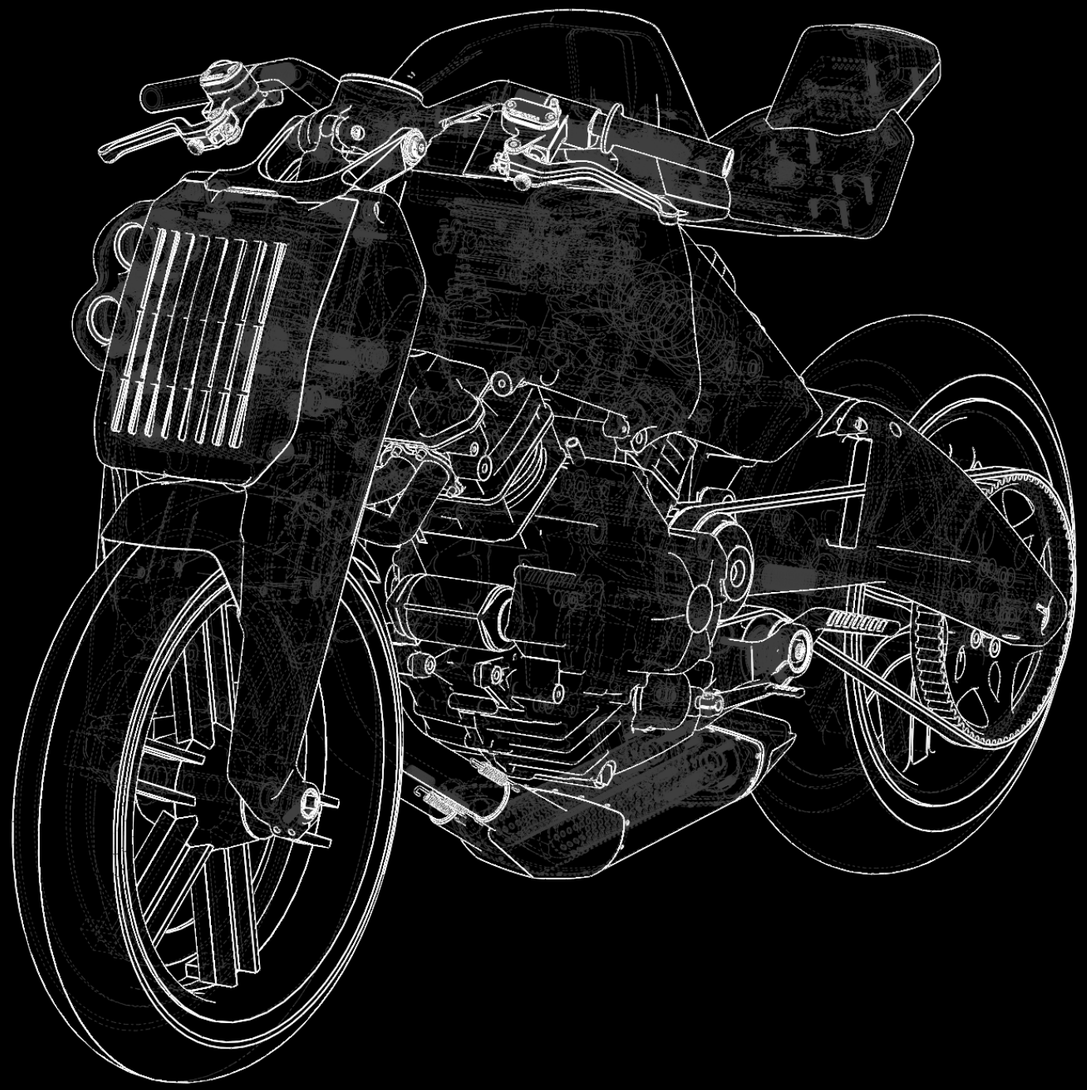


Fig 26: Ronin CAD Model

indicating that the hologram was a true CAD file. The holographic equivalent of a menu bar that stuck with the model and conveyed basic information needed to be created. Instead of something like an intrusive floating billboard, I borrowed an elegant physical solution from the world of artistic sculpture: the plinth. In *SolidWorks* I created a simple 3-D placard and D-shaped bar that encircled the entire footprint of the motorcycle. I placed a filename graphic on the front face of the placard and assigned the MR Toolkit billboarding script to the bar and placard assembly, instructing it to follow the designer for visibility at all angles without regard to the motorcycle's position.

While organizing the various meshes and materials which make up the motorcycle, I began to realize that the consideration needed to be paid to the complexity of large assemblies, subassemblies, and parts. The ability to "dig deeper," as was the urge of those who saw the earlier version, is an expected part of professional CAD systems and is already coded into the procedural memory of CAD operators. Any simulation of a CAD system should replicate this depth of function. To achieve this, I had to foundationally restructure the entire system. I began by adapting the existing workflow of screen-based CAD assembly interaction. I set aside the idea of disparate "render" and "construction" views, instead overlapping them in the same workspace with different modes: "inspection" and "edit."

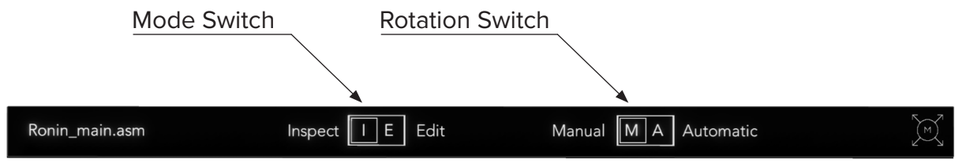


Fig 27: Front Panel Mode Switches

Inspection Mode: The ability to share digital files on screens to communicate design intent with other stakeholders is commonplace in product design and engineering today. It is preferable to physical models, because it saves both resources and time. Extending this interactively-shallow, "inspection using MR," ability was a logical first step in providing more intuitive communication and is used today at places like Nasa's Jet Propulsion Laboratory (NASA, 2018). While inspecting in MR is not especially innovative, it makes sense as the equivalent of an opening screen in my system.

While in this mode, viewers can move about the model freely without editing anything. This feature is a prime candidate for sharing the design work with clients or colleagues without divulging the complexity of the model's base geometry. To allow for the rotation of the bike, I applied the "NavigationRotateResponder" script from the MR Toolkit to an empty parent containing all the bike meshes. With this in place, air tapping and dragging on any part of the model caused it to rotate about the Y-axis. To not disrupt the ability to only rotate the motorcycle, the plinth alone received a "Hand Draggable" script which allowed the entire assembly to be repositioned in X, Y, and Z space while the model's rotational orientation was maintained. The plinth face also received a pair of two-state toggle switches; one to switch from manual model rotation to a rotating turntable mode with adjustable speed, and another to switch from the default inspection mode to the deeper edit mode.

Edit Mode: With the switch in the "edit mode" state, the system became more like a conventional solid modeling program with an assembly file open. The designer in this mode would likely be trying to look at the assembly holistically and select individual parts and subassemblies with which to interact. I needed to demonstrate an AR version of that workflow to simulate the process.

Assembly Mode: In a typical CAD program, assemblies are used to look at the groups of parts as a whole and assign relationships between individual parts and assemblies. To give feedback as to which section the cursor is hovering over, surfaces of parts in most screen-based CAD programs light up to a common color, like an illuminated blue, to confirm the selection. This was an effective method in two dimensions and I wanted similar feedback functionality for the gaze cursor. Recreating this feature for the thousands of parts that make up a motorcycle would have taken months to achieve, so to demonstrate the idea I selected the very prominent, gas-tank-like airbox cover[1] on the top of the motorcycle for in-depth simulation.

Fig 28: Airbox Highlight

To accomplish the highlighting effect, two copies of the airbox mesh were made to inhabit the exact same space in the scene, one standard-color mesh that is visible, and one copy with an illuminated blue material set to invisible. Using the MR Toolkit's "OnFocusEvent" script, I told the system to use the presence of the cursor to dictate which copy would be visible. The script told Unity to enable the visibility of the of the illuminated blue copy's "MeshRenderer" when the cursor collided with the mesh, or a "FocusEnterEvent," and disable the MeshRenderer when the cursor stopped colliding, or a "FocusLostEvent." This resulted in an effective version of the hover-over highlight effect.

Next, I assigned another "InteractiveButton" script to the airbox mesh that when clicked causes a panel of ordered buttons to appear. This bracketed panel of buttons were stored as invisibly dormant but, using Unity's animation system, they subtly rise from the airbox mesh to a prominent position above the model and billboard to face the designer. These buttons describe the hierarchy of the assembly, subassembly, and part file to which this airbox belongs, each of them invoking a corresponding mode that is revealed by clicking on them. The system defaults to the current main assembly until another button is clicked, as indicated by the "Ronin_Main.asm" button being highlighted in blue. This button can also be clicked at any time to return to the main assembly.

Subassembly Mode: By clicking the "upper_body.sasm" button, the system enters subassembly mode. To make clear the parts comprising the subassembly, the collection of upper body panels remains opaque while other subassemblies, like the front suspension, drivetrain, rear suspension, etc., become transparent. This mode is similar to the assembly mode except that it segments certain aspects of the design for the examination of relationships of parts, help with organization, and provide less encumbered access to individual parts. With further development, and by adding additional inputs, subassemblies would be an essential part of an AR CAD system.

1: On a technical note: the body panel on the top of the Ronin motorcycle (which resembles the fuel tank of most other motorcycles) is a cover panel containing the air inlet box and several electronic components. The fuel is stored in the hollow frame of the motorcycle.

Part Mode: When the “airbox_cover.prt” button is clicked, the part file mode is invoked. Like the subassembly, the part file in focus maintains its opacity while the rest of the parts and subassemblies become transparent. Solid-modeled part files are deceptively complex as they must inherently contain every step that was taken by their designer to achieve the finished form. This set of steps is commonly referred to as a “feature tree,” “timeline,” or “build order.” Representing this bulk of information is essential to the part file and presented a new opportunity for design in AR.

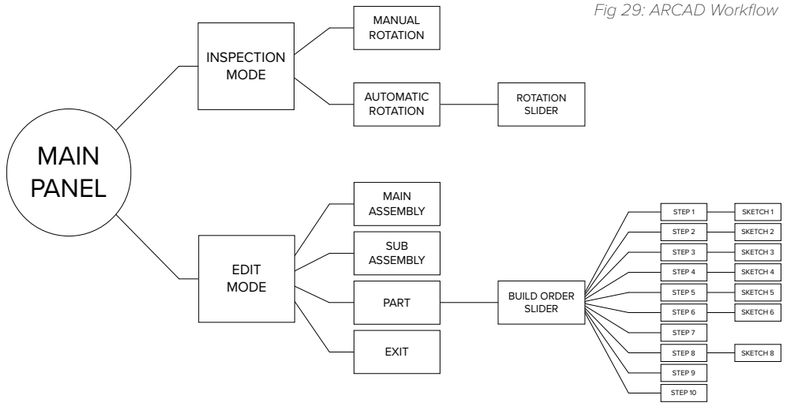
Build Order Timeline: Current build orders amount to a chronological list of features that were applied to a model and are often shown as unintuitive icons, a list of words, or both. To see what effect a feature has on a build, the designer must roll back the timeline to that point and watch the part model devolve accordingly. Considering the additional visual real estate, I decided to represent the features not by words or icons, but by the results of the features on the part file itself. To create this effect, I rebuilt the airbox part file from scratch in *SolidWorks*, exporting each of the 10 steps of the build as I went. Each export yielded OBJ files that had submeshes representing the airbox, the sketch(s) and the feature being applied. After importing each of the meshes into the scene, applying the necessary material attributes, and carefully placing them on a timeline beneath the assembly hierarchy buttons, a new kind of timeline was starting to emerge.

The next component proved to be a bit more complex because I needed an interactive signifier to both highlight an individual airbox build step along the timeline, and cause that step’s corresponding mesh to be reflected in the larger assembly. To create a sliding-signifier I appropriated a gesture-compatible, physical slider from the MR Toolkit, stripped it of extraneous animation, stretched the dimensions, and replaced the existing “slider handle” with mesh brackets of my own design. That way a set of brackets that fit the design language could slide along the timeline using a HoloLens-compatible air tap and drag gesture. Next, I added one cube-shaped, invisible collider to each airbox construction step on the timeline. Using Unity collider trigger tutorials (Omnirift, 2017), I wrote a script of my own called “OnTriggerEvent” and applied it to each of the collider-equipped construction steps. I then imported a mesh of each of the 10 steps of the airbox’s construction process, placed them at full scale within the larger motorcycle assembly in the scene, and set them all to invisible. I then configured each step’s “OnTriggerEvent” instance so that, should the slider brackets enter its collider, the script would toggle on the visibility of its corresponding full-scale airbox mesh and turn off the other nine.

Now when the “airbox_cover.prt” button is clicked, the brackets surrounding the hierarchy buttons stretch downward to reveal a timeline populated by labeled, three-dimensional snapshots of each feature step and their incremental effect on the airbox. The frustration of blindly sliding a timeline back and forth trying to guess which instance of which feature created which effect could be easily averted using this approach. Instead, the designer need only walk up to the timeline and look closely. In a fully-realized AR CAD file workflow, when the desired feature is found, the designer can air-tap and drag to that feature and modify at will.

Sketch Edit Mode: As the deep end of the simulation, most everything led to the sketch mode, as it does in conventional CAD. Initial impressions indicated that it was successful enough to remain largely unchanged from its first incarnation. The planes, tools, and animations needed only to be scaled up, and worked as intended, even when paired with something as complex as a motorcycle. The system,

Fig 29: ARCAD Workflow



while in its limited beta format, was unable to actually create vector-like original sketches, but it served its purpose simulate the experience of a true AR CAD workflow.

Out of Context: Harkening back to the differentiated views of “construction” and “render,” I remained convinced that there was utility in putting to use the “room-scale” abilities of an AR CAD system by allowing a part to be extracted from an assembly while leaving both files visible. By air-tapping the engine, just as with the airbox, the engine subassembly opens with its own set of planes, and the rest of the motorcycle becomes transparent. The engine mesh was entirely separate so that, by clicking and dragging the manual relocation button, it could be extracted from the main assembly. When the engine exited a pre-determined envelope, the full motorcycle returned to opaque. As a result, it was like having an engine model being edited with the main assembly simultaneously in full view.

Holographic Emulation: During its construction I regularly used the MR Toolkit’s Holographic Emulation feature to experience the demonstration scene holographically. With the computer and the HoloLens connected to the same Wi-Fi network, Holographic Emulation wirelessly connects the two to run the Unity scene in AR. This is typically used for iterative testing between builds, as it saves the developer the time it takes to build and install – called deploying – the scene as a stand-alone application on the HoloLens. When using this feature, the HoloLens acts only as a set of sensors and a 3-D display. The sensor data supplies Unity with the room-scale position of the HoloLens which is applied to the scene’s camera object. The Unity-running computer crunches the graphically-intense data of what is seen by the camera and feeds it back to the HoloLens to display. This happens remarkably quickly and with little noticeable latency, provided there is a capable Wi-Fi network.

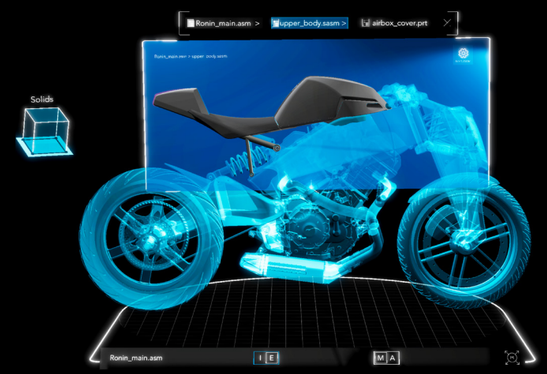
With all my elements in place, the scene outgrew the HoloLens’ on-board computing capabilities. I could successfully deploy the scene to the HoloLens as a stand-alone application and it appeared on the application list as any other. However, when I tried to open it, the Unity splash screen would appear, then disappear, then I was kicked back out to the main operating system shell. By putting the HoloLens in debugging mode and watching in Visual Studio, I was able to see the problem: with all the high-polygon-count models, animations and features stacked in place, the RAM necessary to run the scene well exceeded that which is on-board the HoloLens (figure 31). Thus, all further demonstrations would require the use of the Holographic Emulation, the implications of which are discussed further in the next section.

With a working framework assembled for an augmented reality computer aided design (ARCAD) system, I began usability testing by letting colleagues in both the academic and professional spheres experience it for themselves. The simple description of “like CAD modeling in augmented reality” was all that was required to make sense of the idea to many of those who have worked closely enough with traditional CAD to feel the screen-based drawbacks. Some were surprised that this approach had not yet been actualized in the MR realm already, and it was characterized by most as a step in an obvious direction.

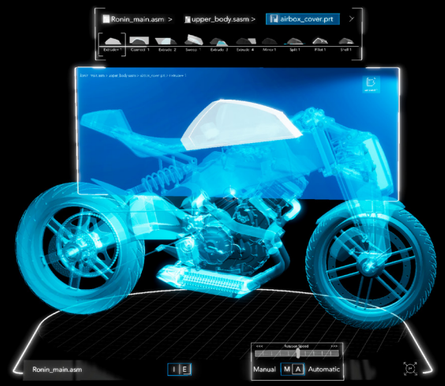
Assembly Mode

Build Order

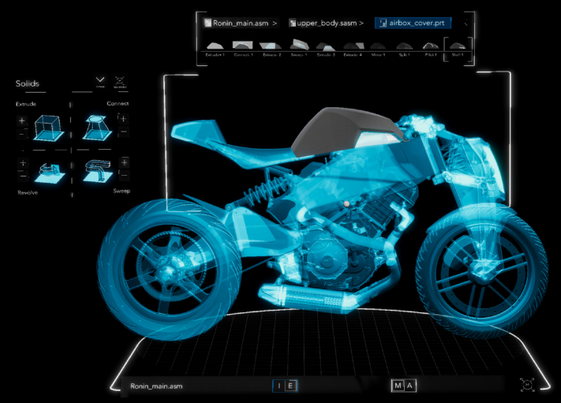
Subassembly Mode



Build Step One



Part Mode



Sketch Edit Mode

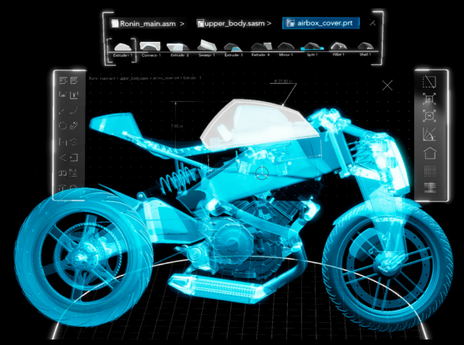


Fig 30: Mode States

Part 4:

DISCUSSION

In the previous section, I outlined how the necessary features and interactions were created, but in addition to the technical knowledge that I sought out and dissected for specific uses, a series of design decisions had to be made. Overarchingly, these decisions were made with one goal in mind: to leverage a new way of computing to engender heightened cognitive comprehension to aid in the physical design process.

The entire demonstration was designed with consideration paid to the psychological foundations set forth by James J. Gibson's "The Ecological Approach to Visual Perception" (Gibson, 2015), and considering the platform of augmented reality through this lens is fascinating: an HMD like the HoloLens can see through the (terrestrial medium of) air, gather information about an environment's layout of surfaces, substances, and objects, then alter the observer's perception of that environment with a second artificial ambient array, the holographic projection of the virtual geometry that is blended into the natural optic array. Gibson mentions experiments with glass panes and flight simulators in his writing, but regarding mixed reality this quote is even more prescient:

> ***If the artificial array is the same as the natural array, it will yield the same perception. There will arise an illusion of reality without a genuine reality (Gibson, 2015)***

For ARCAD I sought to appropriate this illusory perception for design by transposing three-dimensional information into the environment in a way that is much more natural to human perception than the current two-dimensional display on a flat screen. Instead of spinning a flattened scene on a screen, a CAD model can be directly perceived as the 360-degree data set that can be experienced along ambulatory tracks of vision that are infinitely adjustable.

> ***Perspective-taking along view tracks, resulting in perspective change and motion parallax, forms a framework for observations by an individual situated in an environment. When we know where the camera is, we understand what is not visible and how to obtain a new point of view location to reveal what is currently hidden from view (Roesler, 2011)***

While in this demonstration, the system makes its operator the camera, provides self-location by segueing on the proprioception of the operator wearing the head mounted display, and supplies infinite views within the environment. By imbuing a heightened understanding and affording the ability to make changes in-situ, I believe that finished designs could be achieved more quickly and with better results. To achieve this, however, augmented reality systems demand that some of the existing guidelines for interaction be reimagined.

There is no guidebook of steadfast rules for using a game engine to produce an interactive framework of a CAD simulation to be experienced through a groundbreaking device. This is the job of designers and the number of variables they will encounter has the potential to produce ambiguity in the process and its result. However, with the words of Don Norman in mind that:

> ***. . . because the fundamental principles of designing for people are the same across all domains. People are the same, and so the design principles are the same (Norman D. , 2013)***

In my own experimental process, I have tried and tested existing tenets that I have implemented or adapted from established best practices for design in all three major design concentrations: Visual Communication, Industrial, and especially, Interaction Design.

It is worth noting, though, that the three distinct concentrations are all branches of design and have many overlapping precepts. Industrial Design and Visual Communication Design have a longer history as fields of study, resulting in more established knowledge bases, but in many cases the knowledge for interaction design has adapted patterns from both fields and coupled them with the psychology of cognition, machine interaction, and human behavior to create useful, usable and understandable interactions (Roesler, 2009). Following the interaction design guidelines and teachings from my thesis chair, and books like *About Face* (Cooper, et al., 2014) and Bill Buxton's *Sketching User Experiences* (Buxton, 2007) I found that, while many of the tenets of Interaction Design were intended for on-screen interface experiences, several of the intellectual considerations were apropos, while others less so. Thus, attribution of the lessons employed here to specific concentrations is less important than the lessons themselves.

The ARCAD prototype draws its most overt inspiration from the CAD workflow of the Industrial Design product development process. The tools and options that appear in the demonstration and video prototype come from needs identified by my background in product design

and the survey of existing screen-based and mixed reality design programs. Some of the fundamental interaction decisions, like the sovereign posture of an AR platform, were pre-determined by the ARCAD precept, but analyzing and creating the affordances, signifiers, and feedback for its line-of-sight interface came from adapting screen-based, digital interface design principles and translating them into mixed reality.

Where applicable, the demonstration was built with basic rules of 2-D visual design in mind. Typography is used consistently, sparingly, and without embellishment, but plays an important description role when employed. The assembly hierarchy and build order module is, at its core, a large panel with 3-D icons and text that floats above the main model when invoked. Despite it being intended for viewing in three dimensions, the panel's information is arranged about a 2-D grid, in a visually-hierarchical order, with legible labels carefully aligned to their associated 3-D components. The icons that comprise the sketch and panel control tools are intentionally subdued and minimalist. The assembly hierarchy buttons, planes, and 3-D icons use a restricted color palette that serves as a backdrop upon which changes in color can convey importance.

The mapping of the prototype's interactive component-parts was the result of careful consideration. The conceptual models of those who would engage with ARCAD are likely steeped in the precedents set by contemporary CAD systems, but the ARCAD system is not similarly constrained to a screen with a carousel of tools stuck to the top. Don Norman asserts that best mapping for natural understanding comes from controls "mounted directly on the item to be controlled" (Norman D. , 2013). While designing a mixed reality experience, the lack of physicality means a corresponding lack of physical constraints which normally hinder the implementation of Norman's suggestion. Thus, functional grouping of corresponding items is quite naturally achievable, as seen with the sketch and panel adjustment tools directly on the sketch plane. The second-best mapping scheme being controls that are "as close as possible to the object to be controlled" can be seen in the assembly hierarchy and build order module just vertically adjacent to the main model.

Animated motion was assigned to certain interactions to provide feedback and call attention, but it was used carefully and subtly to maintain its effectiveness. The plane brackets quickly swing to the selected planes when tapped, and then slide open to reveal the tool panels when entering sketch mode; the assembly hierarchy module emerges from the mesh selected, then the module's brackets grow to reveal the build timeline when entering part mode; as the slider is manipulated along the timeline, the mesh of the corresponding step grows by 20 percent to confirm its selection before shrinking again; the 3-D tool panel can collapse to a small icon when not in use, and expand again to reveal the full features and options; mode state switches slide deliberately between states; icon graphics elevate slightly when the cursor hovers over them to confirm intent; and various typographical tooltips appear clearly but subtly to explain a hovered-over option. These microinteractions were created to convey meaning to new users but remain unobtrusive to those who might become intermediate or expert users.

The sensibilities of product semantics in Industrial Design (Krippendorff & Butter, 1984) were invoked in the hope that the variety of 3-D interactive assets that were created could immediately convey their function in context. However, because they were never intended to gain physicality, they fall closer to the definition of digital metaphor per the guidelines of Interaction Design. "About Face" (Cooper, et al., 2014) asserts that we are living in

a post-metaphor age, but that assertion is made with reference to the past iconography of skeuomorphism, in which 2-D representations disingenuously emulated the manual affordances of their 3-D counterparts. When in mixed reality, the metaphor can become a more literal three-dimensional representation of a three-dimensional function. This I believe to be a very valuable addition.

A metaphor not used here, though, was that of the desktop, workstation, or clay modeling studio: the primary theme was genuine, purposeful use of the AR platform and its appropriate visuals. Iconography was as literal as possible, manipulable items were bright with pliancy made apparent, and textual descriptions were intentionally minimal. I found in my survey of MR design programs that the developers of some of these programs had succumbed to the temptation to make everything as three-dimensional as possible – I would caution against this. Just because nearly everything could be three-dimensional doesn't mean that it should be. Panels and two-dimensional text have useful precedents in our daily lives now, thus an incrementalist approach which understands that some iconography and text in MR is still best viewed in two dimensions is appropriate. I recommend often stopping to think "is this improved by being three-dimensional?"

Pragmatically speaking, ARCAD was created to simulate the main hub of a set of tools to facilitate digital creation of three-dimensional content – a set of tools that can be as large as a room or as small as a workstation. This is analogous to real-world workshop. As viewed through the lens of an architect, creating an immersive environment In VR can be called cyberplace-making (Kalay, 2004), but in AR, less concern need be paid to social or temporal indicators for context, though that may change as our technology advances. Place-making in spatial computing did require an understanding of the importance of adaptability.

In a workshop, as in a CAD program, usefulness is gained from the ability to facilitate a variety of work. With that in mind, my demonstration of the ARCAD prototype allowed for every major item to be movable; whether through mere rotation, or through repositioning palettes, subassemblies, or the entire system. In a fully actualized version, this reconfigurability would be combined with fine-grained control of scaling to allow maximum flexibility.

The preceding section presents some of the considerations that I found appropriate for the realization of my prototype as a proof of concept, but as the technology matures with more testing and development, I anticipate that the guidelines for best practices for mixed reality interactions will continue to evolve.

Part 5:

FINDINGS

Over the course of its construction and refinement, the ARCAD system was experienced by several dozen people from a wide range of backgrounds. For many, this was their first foray into an augmented reality experience, so wonderment at the AR illusion was typically the first reaction. As the novelty wore off, any confusion and awkwardness made it clear how important the interactions were. To quote one professor “the model is intuitive... engaging with it is the trick.”

Each end of the spectrum of mixed reality represents its own distinct opportunities. For one industrial design professor, trying to use the system in his office was wholly distracting. He could see all of the elements in AR, but also all of the books in the physical world behind it. He thought that he might like it better in VR where he could focus on the task at hand. The owner of an exhibit design company in Seattle Washington felt the same. He sculpts entire exhibits using MasterpieceVR and will 3-D print them directly from the files created in VR. He said he “rather likes putting on a headset and headphones and blocking everything out, so he can work.”

After experiencing ARCAD, a professional industrial designer wished he had had this a few months ago, explaining that he was designing a crane truck that was so large that there was no way to physically model it. He spent months heavily focused on designing a form and

had resigned himself to the fact that building a model at full scale, while possible, was quite impractical. He imagined bringing the system to a parking lot next to a building and being able to position and extend the crane arm, adjusting his parameters to get the assembly's geometry perfect.

An undergraduate student learning CAD as part of her Industrial Design training was excited to try "a different way of doing it, because the screens seem so small." While in the system, she said "wow, this just makes more sense. I don't like the gestures, but this could definitely be a thing."

A motorcycle builder stared at the rear suspension for some time during a demonstration. When I asked what he was looking at he said that he wished it was a fully functioning assembly with mates, as in *SolidWorks*, so that "he could adjust some of the rear suspension geometry and test the swingarm travel. Or maybe raise the handle bars and check the seating position."

A cohort of mine, who had not yet experienced AR, tested ARCAD during the first version. She approached it with trepidation, but her eyes widened when I started the demonstration. She walked from one end of the room to the other trying to manipulate the models. "I get it now; this hype makes sense. It could change everything"

Professional Interaction Designers were very helpful, too. One searched around for the usual spatial indicators from a CAD program and exclaimed "Oh, the orientation signifiers are moot, it is natural!" A design director from a Seattle-based consultancy helped by finding a few behavioral and graphic elements he would recommend modifying, but not before throwing his leg over the holographic motorcycle and looking down at his hands to check the span of the handlebars.

I presented the system to a design team from a Seattle consultancy who had not yet included mixed reality in their process but thought that they might in the future. While demonstrating, their President – also an active designer – excitedly imagined the prospect of being able to create a product in the context of the environment in which it will eventually fit; like designing a medical device in a real hospital room. Building on that, another designer imagined being able to accurately arrange holographic versions of internal componentry within an existing physical enclosure; or conversely, testing multiple holographic enclosures around an existing internal structure. Was there a future for designing with this technology? "No doubt" he said.

DRAWBACKS OF CURRENT DEVICES

Input: In demonstrations and interviews, without fail, the people engaging with the system felt stifled by a lack of input options. The gestural input created for the HoloLens was the only method allowed by the HMD at that time. Air taps were made but not recognized by the sensors, tapping and dragging outside of the frame caused the event it was meant to elicit to continue indefinitely. One designer likened the gaze cursor to "operating a touchscreen phone with a stylus in your mouth" compared to VR controllers, so I sought a better solution.

In the main operating system of the HoloLens, I was able to pair a Bluetooth mouse to the system, causing a standard arrow cursor to replace the gaze cursor. This cursor, like the gaze cursor, conformed to the surfaces found by the HoloLens' spatial tracking. This method of control is admittedly less direct, but also much less intrusive and far more intuitive in computing's current context. Unfortunately, the mouse would stop working upon starting most applications, including anything running from Unity.

Knowing that the Windows MR headsets used an "inside-out" tracking system that is similar to the HoloLens, and that the Universal Windows Platform (UWP) configuration in the Unity scene is programmed to be compatible with

both kinds of HMDs, I tried pairing the Bluetooth Windows MR controllers to the HoloLens as well. These were even less compatible and refused to work even where the mouse had. Similarly disappointing results were had with the Google daydream VR controller. I spent several days researching developer forums trying bridge this gap through coding but was ultimately unsuccessful. Future AR systems, like the newly-released Magic Leap One headset which comes bundled with a controller, might endeavor to alleviate some of these issues.

Vision: The HoloLens has a narrow field of view on which holographic information can be shown; by some measurements only 34.5 degrees (Delwo, 2018). Compared to the HTC Vive's 110-degree screens, there is considerable room for improvement. There is also the issue of focal plane. When scanning a room without a headset, the eyes can shift their focus between near and far, even with one eye. The screens in MR are at a fixed distance from the eyes so the holographic illusions are not subject to the natural adjustment of focus. This will not be an easy hurdle to overcome technologically, but several HMD companies are said to be considering this too.

Ergonomics: At a listed weight of 1.28 lbs. (Microsoft, 2018b), much of which is transferred to the bridge of the nose, the presence of the HoloLens while in use is apparent in more than just vision. Even though it is an untethered computer that weighs only .07 lbs. more than the tethered Vive headset, several observers of my demonstration complained of nose and neck pain after donning the HoloLens.

Holographic Emulator: The on-board memory of the HoloLens was a hurdle for me to overcome throughout the process. The HoloLens was built with 2 GB of RAM on-board, which limits the complexity (figure 31) of what can be run on the HMD alone. Because of these hardware limitations, my demonstration had to be performed through the MR Toolkit's holographic emulation system. This process depends on strong and stable Wi-Fi connections to operate properly. One of the early demonstrations was severely hampered by the system not connecting the HoloLens to the Unity-running laptop. Whenever possible, Unity scenes should be deployed to run as stand-alone applications on the HoloLens. If that is not an option, as was my case, using a dedicated Wi-Fi hotspot or router is advisable for the stability of the demonstration.

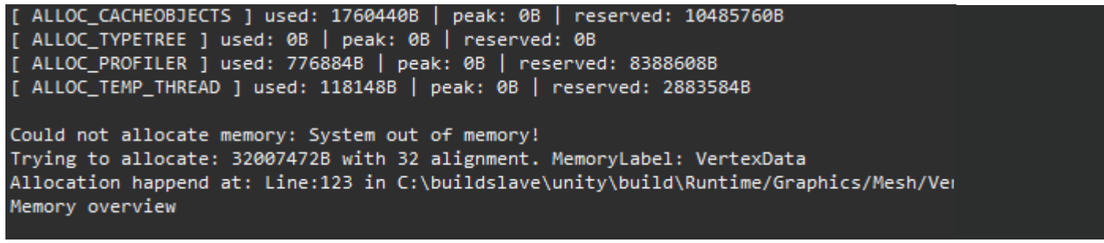


Fig 31: Memory Error

CREATING IN AR

Through creating and testing the ARCAD system I have learned lessons that can be used for further AR development. Overall, the best advice for designing for augmented reality is the most obvious: test your work with actual AR equipment whenever possible. Some of the new variables to consider when working in this realm are:

- Unlike computer screens which can simply dim a pixel to display black, holographic models are comprised entirely of light, with and without color.
- Elements appear as varying degrees of transparent, and when viewed through AR color is not displayed as accurately as it is on highly-evolved flat screens.

- White elements command the most light be displayed thus become the most opaque. This can be harnessed to draw visual attention but too much white can be taxing on the eyes.
- Scaling is important but sometimes difficult to predict because it changes depending on the proximity of the viewer. Google's Daydream VR team studied the subject and design their elements in terms of Distance-independent Millimeter, or DMMs, instead of pixels (Google, 2018). 1 DMM amounts to 1millimeter of height at 1 meter from the viewer. 4DMMs is their baseline size for text.
- Consider the comfort zones of the eyes and neck. The comfortable movement limit of our eyes is approximately 30 degrees from center in any direction, and 60 degrees (biased upward) for our necks (Google, 2018) (figure 32). Primary elements should be in the eye comfort zone, and secondary elements within the neck comfort zone.

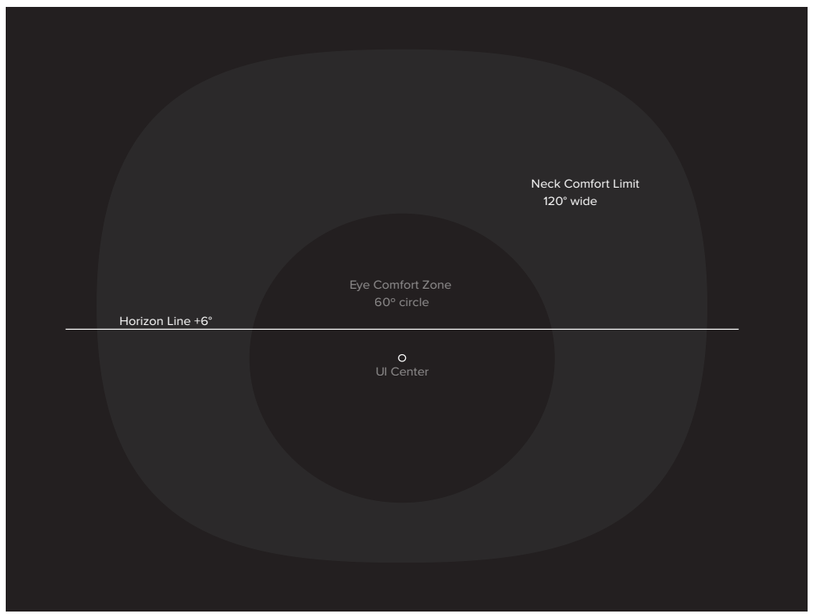


Fig 32: Visual Range (Google, 2018)

- Organized layout of elements into a visual hierarchy remains appropriate but can get lost on large flat cards of information. Sheets should be broken into smaller cards or columns that can encircle the viewer.
- If elements appear outside the field of vision, a directional indicator attached to the cursor can be used to direct attention to that element when needed. The MR Toolkit contains elements to facilitate this approach.
- Movement or animation is a natural way to reflexively draw attention to an event (Johnson, 2014), especially in the periphery, but too much movement can cause habituation and hinder effectiveness.
- We are accustomed to relative spatial freedom in un-augmented reality, and the environments in which an AR system might be used can vary greatly. To quote Jaron Lanier: "User interfaces are about people, and people live out in the big unfenced world, and we can't specify them perfectly" (Lanier, 2017). Therefore, flexibility in element arrangement should be afforded the viewer whenever possible and appropriate.
- Forcing elements to tag-along in the main field of vision (head locking) is uncomfortable; placing elements in an environment (world-locking) and setting them to face the viewer (billboarding) is more natural hence more advisable.
- Most everything that appears in AR can be made to appear 3-D. But, as stated in the findings, the unnecessary addition of three-dimensionality can actually hinder visibility and should be avoided. If it is not three-dimensional in the natural world, should it be in an augmentation of the world?

CREATING IN UNITY

For designers familiar with the typical suite of design-focused programs, the Unity game engine will be a moderately different computing environment to begin with. Some considerations for beginners include:

Scene element hierarchy: While assembling the assets and objects that comprise a scene, the parent-child hierarchical structure should be carefully organized. If, for example, a script is applied to a parent, every child element is subject to the effects of that script.

Movement: Motion can be accomplished using positioning and rotation in coding, or through the Unity animation systems. These can be used interchangeably but will compete, causing a glitch if invoked simultaneously.

GUI-gesture incompatibility: Unity's built-in system for 2-D interfaces does interact with the HoloLens gestural input system, but not reliably, as it was designed for mouse input. Clicks of buttons are recognized, but air-tap and drag does not work for GUI sliders. The MR Toolkit's 3-D versions are designed for MR use and are better equipped to respond to gestural input.

Measurement: Unity's unit of spatial measurement isn't overtly expressed but is important to know for MR development. By default, 1 unit equals 1 meter.

Emissives: To be emissive means to emit light at some variable intensity, and Unity allows the application of an emissive property to any visible component in a scene. Where applied, white or lighter colored emissive components appear to glow which can heighten their call for attention.

FUTURE DEVELOPMENT:

Admittedly, there was one major component missing in my demonstration: the ability to sketch. The process of generating two-dimensional vector graphic data within the Unity game engine exceeded my coding abilities. Beyond a demonstration, it would be an essential part of any CAD system and demand careful cultivation.

As mentioned in the drawbacks section, 3D input still exhibits "major limitations regarding effective, accurate selection of targets in real-world applications" (Argelaguet & Andujar, 2013). The amount of head movement that would be required to create one sketch of average complexity for CAD modeling renders the gaze cursor infeasible. Sketch manipulation via laser-pointer-style raycast from VR style controllers is one existing solution that could be adapted; externalizing the two-dimensional sketch to a two-dimensional medium like a tablet computer is another possibility; being able to drag the sketch panel to a flat surface, like a desk or wall, to be modified using touch gestures is another. Sketching is just one of the problems posed by CAD modeling with a gaze cursor; the complexity and precision of input demanded in all aspects of CAD only serve to reinforce the point that further exploration is required.

The term "Mixed Reality" evokes the entire spectrum of VR and AR, thus a true MR headset system would be a confluence of the two disparate technologies we know today. An HMD that could go completely opaque to create a virtual experience and back to transparent to augment reality would be the optimum hardware. Were it able to go entirely opaque, one could assume that it could be selectively opaque too, allowing for improved holographic representations in AR.

Design almost always requires collaboration with others and designing in AR should be no different. It is common practice in CAD modeling to visit the screens of other designers to see the work being performed. Unfortunately, in my demonstrations, I had to monitor the Unity-running laptop screen to see where my participants were looking. This was acceptable for the purposes of demonstration, but the isolation of a singular environment, in a stand-alone computer, visible by only one viewer, is unsustainable in design practice. Research into shared VR experiences stretches back to the late 1980s (Blanchard, et al., 1990) but the sharing abilities of the HoloLens is somewhat limited. Collaborative sharing of the environment-based tools facilitating creation is a mandatory consideration.

The advancements made in the resolution of our displays and the craftsmanship of our computer-generated graphics are producing quite realistic environments now, but most of the energy has been spent on improving the visible aspects of VR and little else. There is some vibrational haptic feedback in the VR controllers made today but to some “input is more important than display. Your input in VR is you” (Lanier, 2017). Simulating the physicality of a model through a force feedback system, like the one in use at the medical modeling firm I visited, could heighten the understanding of an object and the tactile feeling of pushing and pulling on digital clay, as one would with physical clay, could allow a new direct-modeling analogue for use in AR.

Fig 33: ARCAD in use

Fig 34: Left to right; the Vive, Odyssey, and HoloLens

Part 6:

CONCLUSION

Computer aided design has long been subjected to the inherent hindrances of the technology on which it is viewed. The designers working in these simulated two-dimensional environments are incapable of truly designing in a real-world context, instead, employing a familiar but flawed flat screen to achieve their goals. To quote James Gibson “The very notion of an image as a flattened-out object, a sort of pancake of solid body, is … misleading” (Gibson, 2015). But without an effective alternative, CAD has flourished despite the visual, and thus experiential, incongruities.

The Augmented Reality Computer Aided Design system is a response to these drawbacks and offers an alternative method for designing through a new visual modality in computing. Ivan Sutherland’s Sketchpad was introduced as a way of “talking to the computer using a graphical language,” a language now spoken daily by designers. New mixed reality tools present the potential to elevate the dialogue between designer and computer, and by extension the designs being created with those systems. This thesis work demonstrates a potential form upon which this new dialogue could be conceived of and built.

To achieve this, I thoroughly surveyed precedents upon which our current tools were built, the functional operation that resulted in the various approaches to modeling, the current generation of screen-based CAD systems common to industry, and the new generation of

virtual reality modeling. From there, I created a workflow for importing professional-grade CAD models into the Unity game engine, and open-source processes from the augmented reality development community were used to experience models through a HoloLens. Next, based on a cross-pollination of best practices in virtual reality coupled with teachings from the current fields of design, an interactive workflow was built to simulate a CAD design environment in augmented reality. This demonstration was shown to a variety of interested parties, the results of which I incorporated and documented into the overall design.

I designed and built the ARCAD system to be an incremental improvement to the digital tools that facilitate 3-D creation now, brought to a logical realm of mixed reality. Tools are usually created for one of two reasons; as response to a need in a process, or as a technology matures enough to uncover a use not yet realized. Technological tools, like the computer, can have a plethora of applications and can grow to be an indispensable part of a practice, like that of design. As in the transition to digital drafting, evolutionary tools can come to define a time frame's identity because they assimilate into the process of which they are applied, drastically affecting the results. As a tool that can more accurately communicate form in an intuitive way, 3-D modeling in a truly 3-D environment heightens a designer's cognition by adding visual cues for dynamic spatial perception – a significant addition to the visual representation of spatial geometry as a naturalistic 3-D display that could radically change the design process for those who create content in three dimensions.

ACKNOWLEDGMENTS

This thesis would not have been possible without those who supported me along the way:

First and foremost, my thesis chair Axel whose help and guidance changed not only the trajectory of my academic career, but also fundamentally the way I see and think about the world.

The committee of Jason and Paul: Your feedback and challenges were invaluable.

My cohort team: Aubree, Chris, Coreen, and Emma– I am forever in your debt.

And by my grandfather, mother, and families– both by birth and by choice.

Immense appreciation to the faculty and staff of the University of Washington and those who have taught me along the way. Especially Ahn, Ann L-F., Audrey, Chris O., Clare, Derek, Dom, Fernd, Franklin, Heidi, Jade, James O., Jeff S., Joe (No. 1), Josh K., John T., John W., Kendell, Kurt, Mayberry, Melissa, Mylee, Phil, Piper, Rei, Richelle, Sarah R., Scott H., Scott T., Steph D., Tate, Ted S., Tyler Y., Walter and William T. Also, the teams at Artefact, Magpul, Metro, Ronin, and Tactile.

Dedicated to Helen Nussbaum.

BIBLIOGRAPHY

Alexander, C. (1964). *Notes on the synthesis of form.* Cambridge, Massachusetts: Harvard University Press.

Amanda Di Pancrazio, e. a. (2017, June 30). *3D to VR: The Essentials*. Retrieved from Area by Autodesk: https://area.autodesk.com/tutorials/series/3d-to-vr-the-essentials1/

Argelaguet, F., & Andujar, C. (2013). A survey of 3D object selection techniques for virtual environments. *Computers and Graphics, 37*(3), 121–136.

Atari Museum. (2018, August 1). *Atari Museum Arcade*. Retrieved August 1, 2018, from Atari Museum: http://www.atari-museum.com/videogames/arcade/arcade-menu/arcade-menu.htm

Autodesk. (2018a, August 2). *About Working With Point Clouds*. Retrieved from Autodesk Knowledge Network: https://knowledge.autodesk.com/support/autocad/getting-started/caas/CloudHelp/cloudhelp/2018/ENU/AutoCAD-Core/files/GUID-C0C610D0-9784-4E87-A857-F17F1F7FEEBE-htm.html

Autodesk. (2018b, August 8). *List Of All Products*. Retrieved from Autodesk: https://www.autodesk.com/products

Baudrillard, J. (1994). *Simulacra and simulation.* Ann Arbor: University of Michigan Press.

Blanchard, C., Burgess, S., Harvill, Y., Lanier, J., Lasko, A., Oberman, M., & Teitel, M. (1990). Reality built for two: a virtual reality tool. In . *Proceedings of the 1990 symposium on Interactive 3D graphics (I3D '90)* (pp. 35-36). New York: ACM.

Buffalohistory.Org. (2018). *Number of TV Households in America*. Retrieved July 10, 2018, from Buffalo History Museum: http://www.buffalohistory.org/Explore/Exhibits/virtual_exhibits/wheels_of_power/educ_materials/television_handout.pdf

Burton, R. (2012). *Ivan Sutherland - A.M. Turing Award Laureate*. Retrieved July 30, 2018, from A.M. Turing Award: https://amturing.acm.org/award_winners/sutherland_3467412.cfm

Butterworth, J., Davidson, A., Hench, S., & Olano, M. T. (1992). 3DM: a three dimensional modeler using a head-mounted display. *Proceedings of the 1992 symposium on Interactive 3D graphics (I3D '92)* (pp. 135-138). New York: ACM.

Buxton, B. (2007). *Sketching User Experiences.* San Francisco: Elsevier.

Computer Hope. (2018, August 4). *What is a Macro?* Retrieved from Computerhope Definitions: https://www.computerhope.com/jargon/m/macro.htm

Computerhistory. (2018, July 12). *Computerizing Car Design: The DAC-1 - CHM Revolution*. Retrieved 2018, from Computerhistory.org: http://www.computerhistory.org/revolution/computer-graphics-music-and-art/15/215.

Cooper, A., Reimann, R., Cronin, D., Noessel, C., Csizmadi, J., & LeMoine, D. (2014). *About Face: The Essentials of Interaction Design.* Indianapolis: John Wiley & Sons, Inc.

Crypto Museum. (2012a, February 24). *Colossus*. Retrieved July 27, 2018, from Crypto Museum: http://cryptomuseum.com/crypto/colossus/index.htm

Crypto Museum. (2012b, November 23). *Bombe*. Retrieved July 30, 2018, from Crypto Museum: http://cryptomuseum.com/crypto/bombe/index.htm

Dalakov, G. (2018, July 25). *History Of Computers And Computing, Birth Of The Modern Computer, Software History, Sketchpad Of Ivan Sutherland*. Retrieved from History-Computer.Com: http://history-computer.com/ModernComputer/Software/Sketchpad.html

Delwo, R. (2018, August 22). *What AR Field of View Feels Like* . Retrieved from Medium: https://medium.com/hologram-inc/what-ar-field-of-view-feels-like-ecadd16afc5e

Ford. (2018, August 3). *Molding a More Sustainable Future: How Ford Recycles Clay to Reduce Landfill Waste*. Retrieved from Ford Media Center: https://media.ford.com/content/fordmedia/fna/us/en/news/2016/10/17/molding-a-more-sustainable-future--how-ford-recycles-clay-to-red.html

Fuchs, H., Kedem, Z. M., & Uselton, S. P. (1977). Optimal surface reconstruction from planar. *Communications of the ACM, 20*(10), 693-702.

Gibson, J. J. (2015). *The Ecological Approach to Visual Perception.* New York: Psychology Press.

Google. (2018, August 23). *Google VR*. Retrieved from Google Developers: https://developers.google.com/vr/design/sticker-sheet

HTC. (2018, August 5). *VIVE™ | VIVE Virtual Reality System*. Retrieved from Vive.com: https://www.vive.com/us/product/vive-virtual-reality-system/

IBM. (2018, July 30). *System 360: From Computers to Computer Systems*. Retrieved from IBM: http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/system360/

IGN. (2008, March 10). *Al Alcorn Interview.* Retrieved August 1, 2018, from IGN: http://www.ign.com/articles/2008/03/11/al-alcorn-interview.

Intel. (2018). *50 Years Of Moore's Law*. Retrieved July 30, 2018, from Intel: https://www.intel.com/content/www/us/en/silicon-innovations/moores-law-technology.html

Johnson, J. (2014). *Designing with the mind in mind: Simple guide to understanding user interface guidelines.* Waltham: Morgan Kaufmann.

Kalay, Y. E. (2004). *Architecture's new media: principles, theories, and methods of computer-aided design.* Cambridge, Massachusetts: The MIT Press.

Krippendorff, K., & Butter, R. (1984). Product semantics: Exploring the symbolic qualities of form. *Innovation, 3*(2), 4-9.

Lanier, J. (2017). *Dawn of the New Everything: Encounters with Reality and Virtual Reality.* New York: Henry Holt and Company.

Larsen, N. (2018, August 3). *Free Solidworks OBJ Exporter v2.0* . Retrieved from Forum.solidworks.com: https://forum.solidworks.com/thread/54270

Link. (2018, August 16). *Our Innovation Heritage*. Retrieved from L3 Link Training & Simulation: https://www.link.com/about#our-innovation-heritage

Lok, C. (2005, June 1). *The Start of Computer Games*. Retrieved June 10, 2018, from https://www.technologyreview.com/s/404233/the-start-of-computer-games/

Microsoft. (2018a, August 5). *How inside-out tracking works*. Retrieved from Mixed Reality Enthusiast Guide: https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/tracking-system

Microsoft. (2018b, August 5). *HoloLens hardware details*. Retrieved from Windows Dev Center: https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details

Microsoft. (2018c, July 20). *Academy - code, tutorials, and lessons*. Retrieved from Windows Dev Center: https://docs.microsoft.com/en-us/windows/mixed-reality/academy

Microsoft. (2018d, August 8). *Install the tools - Mixed Reality*. Retrieved from Windows Dev Center: https://docs.microsoft.com/en-us/windows/mixed-reality/install-the-tools

Microsoft. (2018e, August 10). *Microsoft-MixedRealityToolkit-Unity.* Retrieved from GitHub: https://github.com/Microsoft/MixedRealityToolkit-Unity

Microsoft. (2018f, March 20). *Gestures - Mixed Reality*. Retrieved August 11, 2018, from Windows Dev Center: https://docs.microsoft.com/en-us/windows/mixed-reality/gestures

Milgram, P., & Kishino, F. (1994, December). A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems, E77-D, No.12*.

Moore, G. E. (1965, April 19). Cramming more components onto integrated circuits. *Electronics Magazine 38*, pp. 114–117.

NASA. (1990). *The Virtual Interface Environment Workstation (VIEW)*. Retrieved August 22, 2018, from NASA.gov: https://www.nasa.gov/ames/spinoff/new_continent_of_ideas/

NASA. (2018, August 13). *New 'AR' Mobile App Features 3-D NASA Spacecraft*. Retrieved from Nasa.gov: https://www.nasa.gov/feature/jpl/new-ar-mobile-app-features-3-d-nasa-spacecraft

National Museum Of American History. (2018). *Magnavox Odyssey Video Game Unit, 1972*. Retrieved July 23, 2018, from http://americanhistory.si.edu/collections/search/object/nmah_1302004

Norman, D. (2013). *The Design of Everyday Things.* New York: Basic Books.

Norman, J. (2018, July 10). *NIMATRON: An Early Electromechanical Machine to Play the Game of Nim*. Retrieved from History of Information: http://www.historyofinformation.com/expanded.php?id=4472

Omnirift. (2017, July 18). *Unity Collisions Tutorial - How To Use Colliders and Triggers in Unity*. Retrieved August 12, 2018, from Youtube: https://www.youtube.com/watch?v=6C4KfuW2q8Y

PARC. (2018, July 31). *PARC History*. Retrieved from PARC: https://www.parc.com/about-parc/parc-history/

Parkin, S. (2014, March 7). *Enthusiasts Go Back to the Future in a Virtual Reality Boom*. Retrieved August 24, 2018, from MIT Technology Review: https://www.technologyreview.com/s/525301/virtual-reality-start-ups-look-back-to-the-future/

PricewaterhouseCoopers LLP. (2016). *U.S. Economic Contribution of the Consumer Technology Sector.* Arlington: The Consumer Technology Association.

Rhinoceros. (2018, August 23). *Rhinoceros - NURBS*. Retrieved from Rhino3d: https://www.rhino3d.com/nurbs

Roesler, A. (2009). Lessons From Three Mile Island: Visual Information Design In A High-Stakes Environment. *Visual Language, 43.2/3*, 170-195.

Roesler, A. (2011). A New Model for Perspective: Patterns of Perspective Change and the Ecology of Point of View and Resulting Image. *International Journal of the Image, 1*, 228.

Ronin Motorworks. (2018, August 20). *The Ronin Story*. Retrieved from Ronin Motorworks: http://www.the47.com/

Ruiz-Hopper, M. (2018, August 6). *What's new in the Windows 10 Fall Creators Update*. Retrieved from Windows Experience Blog: https://blogs.windows.com/windowsexperience/2017/10/17/whats-new-windows-10-fall-creators-update/

Samsung. (2018, August 3). *Samsung HMD Odyssey the Ultimate Windows Mixed Reality Experience*. Retrieved from Samsung Newsroom: https://news.samsung.com/us/samsung-hmd-odyssey-ultimate-windows-mixed-reality-experience/

Shinal, J. (2018, January 21). *The 10 biggest US tech companies will top $1 trillion in sales this year*. Retrieved August 24, 2018, from CNBC: https://www.cnbc.com/2018/01/21/ten-largest-us-tech-firms-2018-revenue-seen-topping-1-trillion.html

Simon, H. (1996). *The Sciences of the Artificial.* Cambridge: The MIT Press.

Simonite, T. (2016, March 23). *Intel Is Putting The Brakes On Moore'S Law*. Retrieved July 30, 2018, from MIT Technology Review: https://www.technologyreview.com/s/601102/intel-puts-the-brakes-on-moores-law

SketchUp. (2018, August 8). *3D modeling for everyone.* . Retrieved from SketchUp: https://www.sketchup.com/

Stinson, E. (2018, August 3). *The Secret Sauce to a Mustang's Design Is Still Clay and Tape*. Retrieved from WIRED: https://www.wired.com/2014/04/the-secret-sauce-of-a-mustangs-design-is-still-clay-and-tape/

Sutherland, I. E. (1963). *Sketchpad, a man-machine graphical communication system.* Cambridge, Massachusetts: Massachusetts Institute of Technology.

Sutherland, I. E. (1965). The Ultimate Display. *Proceedings of IFIP World Computer Congress* (pp. 506-508). New York City: Verlag wechs.

Sutherland, I. E. (1968). A Head-Mounted Three-Dimensional Display. *Fall Joint Computer Conf., Am. Federation of Information Processing Soc. (AFIPS) Conf. Proc. 33* (pp. pp. 757-764). Washington, D.C: Thompson Books, .

Unity 3D. (2018a, August 3). *Manual: Model file formats*. Retrieved from Unity Documentation: https://docs.unity3d.com/Manual/3D-formats.html

Unity 3D. (2018b, August 5). *Game engines - how do they work?* Retrieved from Unity: https://unity3d.com/what-is-a-game-engine

Unity 3D. (2018c, August 9). *Manual: Unity User Manual*. Retrieved from Unity Documentation: https://docs.unity3d.com/Manual/

Valve Software. (2018, August 3). *Steam Search*. Retrieved from Store.steampowered.com: https://store.steampowered.com/search/?sort_by=Released_DESC&tags=1644

Appendix I

A selection of important terms pertaining to the Unity game engine (Unity 3D, 2018c):

Project: A project is the dedicated, highest level file folder on the hard drive of the computer in use in which all of the information for a game, inclusive of scenes, models, texture maps, audio file, scripts, etc., are stored.
Scene: A Scene contains the environments and menus of your game. Think of each unique Scene file as a unique level. In each Scene, you place your environments, obstacles, and decorations, essentially designing and building your game in pieces.
Camera: A Unity scene is created by arranging and moving objects in a three-dimensional space. Since the viewer's screen is two-dimensional, there needs to be a way to capture a view and "flatten" it for display. This is accomplished using Cameras. A camera is an object that defines a view in scene space. The object's position defines the viewpoint, while the forward (Z) and upward (Y) axes of the object define the view direction and the top of the screen, respectively
Skybox: a wrapper around your entire scene that shows what the world looks like beyond your geometry and are rendered around the whole scene in order to give the impression of complex scenery at the horizon. Internally skyboxes are rendered after all opaque objects; and the mesh used to render them is either a box with six textures, or a tessellated sphere.
Asset: An asset is representation of any item that can be used in your game or project. An asset may come from a file created outside of Unity, such as a 3D model, an audio file, an image, or any of the other types of file that Unity supports. There are also some asset types that can be created within Unity, such as an Animator Controller, an Audio Mixer or a Render Texture.
GameObject: The fundamental object in Unity scenes, which can represent characters, props, scenery, cameras, waypoints, and more. A GameObject's functionality is defined by the components attached to it.
Canvas: The area that contains all UI elements in a scene. The Canvas area is shown as a rectangle in the Scene View.
Prefab: An asset type that allows you to store a GameObject complete with components and properties. The prefab acts as a template from which you can create new object instances in the scene.
Collider: An invisible shape that is used to handle physical collisions for an object. A collider doesn't need to be exactly the same shape as the object's mesh - a rough approximation is often more efficient and indistinguishable in gameplay.

Script: A piece of code that allows you to create your own Components, trigger game events, modify Component properties over time and respond to user input in any way you like.
Parent: An object that contains child objects in a hierarchy. When a GameObject is a Parent of another GameObject, the child GameObject will move, rotate, and scale exactly as its Parent does. You can think of parenting as being like the relationship between your arms and your body; whenever your body moves, your arms also move along with it.
Event System: A way of sending events to objects in the application based on input, be it keyboard, mouse, touch, or custom input. The Event System consists of a few components that work together to send events.
Raycasters: The Event System needs a method for detecting where current input events need to be sent to, and this is provided by the Raycasters. Given a screen space position they will collect all potential targets, figure out if they are under the given position, and then return the object that is closest to the screen.
Clipping Plane: A plane that limits how far or close a camera can see from its current position. A camera's viewable range is between the far and near clipping planes.
Animation: Unity's animation system is based on the concept of Animation Clips, which contain information about how certain objects should change their position, rotation, or other properties over time. Each clip can be thought of as a single linear recording. Animation Clips are then organized into a structured flowchart-like system called an Animator Controller. The Animator Controller acts as a "State Machine" which keeps track of which clip should currently be playing, and when the animations should change or blend together.
Shader: A small script that contains the mathematical calculations and algorithms for calculating the color of each pixel rendered, based on the lighting input and the material configuration.
Bump map: A bitmap image texture used to represent geometric detail across the surface of a mesh, for example bumps and grooves.
Normal Map: A type of Bump Map texture that allows you to add surface detail such as bumps, grooves, and scratches to a model which catch the light as if they are represented by real geometry.