Machinima Filmmaking:

The Integration of Immersive Technology for Collaborative Machinima Filmmaking

A Thesis

Submitted to the Faculty

of

Drexel University

by

Jonnathan Mercado

in partial fulfillment of the

requirements for the degree

of

Master of Science in Digital Media

2016

Dedications

Nine years ago, a group of people said, “You can’t do it, you are going to *fail*.” Upon hearing such burdening words, a blockade of motivation emerged from within. The motivation to do one thing, follow my dreams. I thought if I did what I loved and loved what I did, there was no way I was going to fail. I believed in myself and turned all negativity into positive energy, persevering through hardships. It was not until I was older that I realized, we all *have* to fail. How can one succeed without knowledge of failure? The key to success is then not of accepting failure, but of the courage and willpower to get back up after you fail. And so, I want to thank and dedicate this body of work to the people who indirectly exploited my hidden potential, and to Yasmin (mother), Edwin (father), Orlando (stepfather), Giovannie (brother), and Mayleen (sister), who taught me “*it’s ok to fail, just get back up*.”

Acknowledgements

I would like to first and foremost give thanks to my adviser and thesis committee members, Theo. A. Artz, Glen Muschio, and Santiago Ontanon, for all the support and constructive feedback, which has been instrumental in the culmination of my thesis. Ted, your passion for art and broad knowledge base provided me with new creative outlets to explore. You encouraged me to strive and reach beyond my initial vision, which has greatly facilitated my growth as an artist. Glen, your articulate feedback and constant reminder to remain focused contributed profoundly to my professional and personal growth. With your mentoring, I was able to set a work-life balance, build self-confidence, and get some sleep. Santiago, although briefly communicated, your witty anticipation and feedback regarding eye tracking influenced the design approach of my tool. With your feedback, I was able to identify and compensate for problems arising while using eye tracking as a virtual cinematography tool.

I would also like to thank the Digital Media department for dedicating countless hours of support, in-and-out of class. Also, a big thank you to my fellow graduates. Over the past two years, we have spent wearisome hours struggling in our perspective areas within Digital Media. Each and every one of you offered masterful insights, uniquely attributing to the final delivery of my thesis. Now the time has come to showcase our labor, let’s triumph together while committee’s in favor.

Table of Contents

List of Figures

Abstract

Machinima Filmmaking:

The Integration of Immersive Technology for Collaborative Machinima Filmmaking

Jonnathan Mercado

Advisor: Theo. A. Artz, Glen Muschio, and Santiago Ontanon

This Digital Media MS project proposes to create a flexible, intuitive, and low-entry barrier virtual cinematography tool that will enable participants engaged in human computer interaction (HCI) activities to quickly stage, choreograph, rehearse, and capture performances in real-time. Heretofore, Machinima developers have used limited forms of expressive input devices to puppeteer characters and record in-game live performances, using a gamepad, keyboard, mouse, and joystick to produce content. This has stagnated Machinima development because *machinimators* have not embraced the current evolution of input devices to create, capture, and edit content, thereby omitting game engine programming possibilities that could exploit new 3D compositing techniques and alternatives to strengthen interactivity, collaboration, and efficiency in cinematic pipelines. Our system, leveraging consumer-affordable hardware and software, advances the development of Machinima production by providing a foundation for alternative cinematic practices, to create a seamless form of human computer interaction and to positively convince more people toward Machinima filmmaking. We propose to produce an Unreal Engine 4 system plugin which integrates virtual reality and eye tracking, via the Oculus Rift DK2 and Tobii EyeX, respectively. The plugin will enable two people, in the roles of *Director* and *Performer*, to navigate and interact within a virtual 3D space to productively affect collaborative Machinima filmmaking.

1. Introduction

Machinima is a hybrid medium building on traditions from cinematic productions, video games, and live performance. In 1996, during Machinima's "video capture" period of development, pre and post-production tools used for creating, editing, storing, and displaying narrative performances resided within the game engine. First Person Shooter (FPS) games associated with this period include *Quake*, *Doom*, and *Warcraft III*. Video capture enabled a player to record a performance in real-time by means of automated game engine scripts which stored and replayed captured keyboard and mouse input. Players would then distribute their films within the game archive to present their high-level performances to other players seeking to improve their own skills. The distribution and accessibility of Machinima films during this period were limited to players who owned a copy of the game. This narrowed the target audience significantly; essentially limited to hard core gamers/owners only.

In the next stage of development there was a shift from video capture to "screen capture," Machinima's post-production tools shifted to using out-of-game resources such as FRAPS, After Effects, and Adobe Premier, thus enabling machinimators extended flexibility and ease of use to capture, edit, arrange, and distribute game-recorded footage in various digital formats. This led to a revolutionary shift that expanded participatory communities, production tools, and Machinima's target audience. No longer was the accessibility of Machinima films constrained to gamers who owned a copy of the game, but was now available to far wider audiences across the web; gamers and non-gamers, alike. Red vs. Blue is one of the best known examples of this use of screen capture as a mode of production.

Over the past decade, however, Machinima has stagnated and failed to live up to the expectation of becoming a mature distinct medium capable of revolutionizing new systems of filmic production [17], [24]. Optimistic articles were written about video game engines offering technological advancements that would dramatically change how cinematic films were created, watched, shared, and experienced. Machinima was even regarded as an agent of change and a distinct medium by institutions such as the American Museum of the Moving Image, the Film Society of the Lincoln Center, and Sundance Film Institute [13]. Since Machinima is a hybrid medium containing aspects of film, video games, and live performance traditions, it presents great opportunities to explore novel ways of using human computer interface (HCI) to integrate physical and digital spaces.

However, a problem arises. While the visual quality and technical advances of game engines and technology have significantly advanced, the use of these technological advances have been stymied by HCI devices. A gamepad, keyboard, and mouse are examples of technologies still in use after two decades of Machinima production, limiting methods to puppeteer characters and record performances during runtime. Considering advancements of technology, researchers ponder whether Machinima can benefit greatly from using a game engine as a production environment and post production tool, using emerging input devices to interact, perform, and capture live performances within a virtual space [20].

Our system will reenergize the currently stymied development of Machinima. It introduces a new platform whereby a *Performer* and *Director* interact and collaborate within a shared virtual space. As we leverage consumer-affordable technology, we aim to create an intuitive, flexible, modular, and affordable tool for filmmakers. The expectation is that such democratization of filmmaking tools will positively affect more people involved in using a game engine as a production environment and post production tool. To reinforce Machinima as a hybrid revolutionary medium, we will facilitate new production methods and integrate eye tracking and virtual reality as a means to exploit new in-game techniques and possibilities for collaborative HCI. We use these two technologies to establish dramatic agency, which should influence future designs of real-time collaborative interfaces, while providing new avenues for machinimators to explore in producing Machinima films.

2. Background

2.1 What is Machinima?

Machinima is a vaguely defined medium which offers experimental possibilities to employ digital interactive narratives for computer display. It has typically employed input devices such as game pads, keyboards, mouse, head mounted displays, and eye tracking devices. The meaning behind the craft of Machinima varies from player to player and person to person. However, a widely accepted definition derives from The Academy of Machinima Arts and Sciences and the Academy of Motion Picture Arts and Sciences which relatedly state, "The art of making animated films within a real-time virtual 3D environment [13]." Considering that use of a game engine to create a film is a common distinguishing factor between traditional filmmaking and Machinima, perhaps it is wise to follow this viewpoint in order to provide Machinima a chance to be defined as a distinct medium. After all, a central goal of Machinima is to achieve its independence as an alternative filmmaking technique [23]. However, Paul Marino and Katherine Anna Kang disagree with The Academy of Machinima Arts and Science's definition of Machinima, and they classify the medium as just another form of film production [17]. They are not wrong, but there is a fundamental difference between tools used for film production (screen capture) and tools used in film production using a real-time engine (code).

2.1.1 Video Capture Period of Development

Machinima origins took the form of demo recording, allowing players to record their in-game performance, archive it, and allow players with the same copy of the game to view archived replays (see Fig 2.1). The target audiences were limited to those who owned the game, but varied from those players seeking enjoyable viewing of in-game replays to those learning from highly skilled players on how to improve their own gameplay skills [13]. First person shooter titles such as *Quake*, *Doom*, and *Warcraft III* exemplify the origins of code-based Machinima production in 1996. What is meant by code-based is that game replays were not films, "rather they are sequences of commands or scripts that tell the game engine what to do, by repeating the effects of keyboard and mouse input in the same sequence as executed by the player when playing the game [11]." In other words, Machinima during its early development depended largely on programming to analyze a player's input on the controller, replicate the input in real-time, archive the sequence of inputs using programming scripts, and replaying the *actual* performance as if the player was re-performing it in-game. This unique code-based functionality of recording a player's digital performance in real-time was a key ingredient within the origins of Machinima. Machinimators during this time period used the software, coding, virtual space, and tools inside a real-time engine to perform *and* capture the recording of an event from the perspective of the player.

Figure 2.1 Video Capture Period: Real-time Performance Capture and Archiving

2.1.2 Screen Capture Period of Development

The release of *Tritin's Quad God* in the year 2000 marked a transitional period for Machinima, drastically reorienting its code-based attributes (video capture) to capture attributes (screen capture). In other words, Machinima's video capture was no longer edited through code, but reliant on using non-linear editing programs (screen capture systems) such as FRAPS, Adobe Premier, and Adobe After Effects (see Fig 2.2) [13]. It is important to note the transitional period of Machinima did not occur immediately. Hugh Hancock of Strange Company (1997) and ILL Clan (1999) were the first two companies to produce Machinima using out-of-game software such as FRAPS. Although post-production tools used with their movies *Eschaton: Darkening Twilight* and *Apartment Huntin'* were not immediately accepted as common practice, the two films managed to enter a new space for Machinima production, providing machinimators the opportunity to explore new forms of Machinima practices not confined to in-game software. The use of non-linear editing software such as After Effects and Adobe Premier did not become a common practice until the release of Quad God. After Effects and Adobe Premier offered convenient editing tools for moving images, leading machinimators to incorporate this new out-of-game form of post-production.

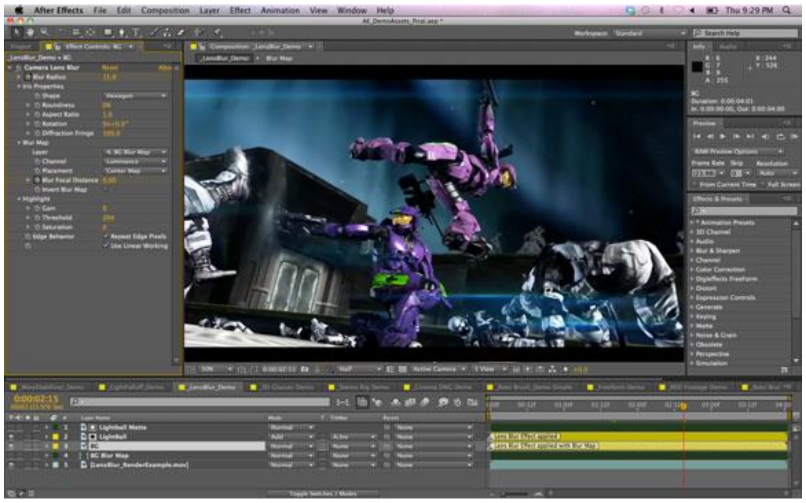


Figure 2.2 Screen Capture Period: Machinima's Post Production Tool Shift to 2D Compositing

2.2 Velvet Revolution: Transformation of Moving Images and its Impact on Machinima

Machinima films owe much to the transformation of moving images during the rise of digital media [24]. If not for the affordability and accessibility of mainstream media content, which permitted many to participate, Machinima would have been limited to only gamers owning the same copy of the game. Perhaps it is not coincidental that the birth of Machinima and rise of Manovich's "Velvet Revolution" which marked a shift away from time-based (temporal) to composition-based (spatial) production occurred at the same time [23].

According to Manovich, the software package, "After Effects" ushered in the Velvet Revolution with its ability to combine distinct media such as animation, typography, live action, and graphic design all in a single, affordable, software environment [16]. Although content from varying media met within a single space, it did not mean all content looked the same, but rather the techniques employed were similar [15]. Before the Velvet Revolution, Manovich asserts traditional lens-based recording offered little to no manipulability over the dimensions and fixed visual content per frame [16]. The limitations of temporal space quickly became supplemented with composition based production, the spatial dimension (see Fig 2.2). The reduction of software and hardware costs and convenient integration of compositing led way to participatory practices with independent artists, educators, and companies. Further, the flexibility of the After Effects interface presented opportunities for designers to adjust canvas dimensions, import distinct media, independently arrange and access elements, apply filters, and manage transparency. The

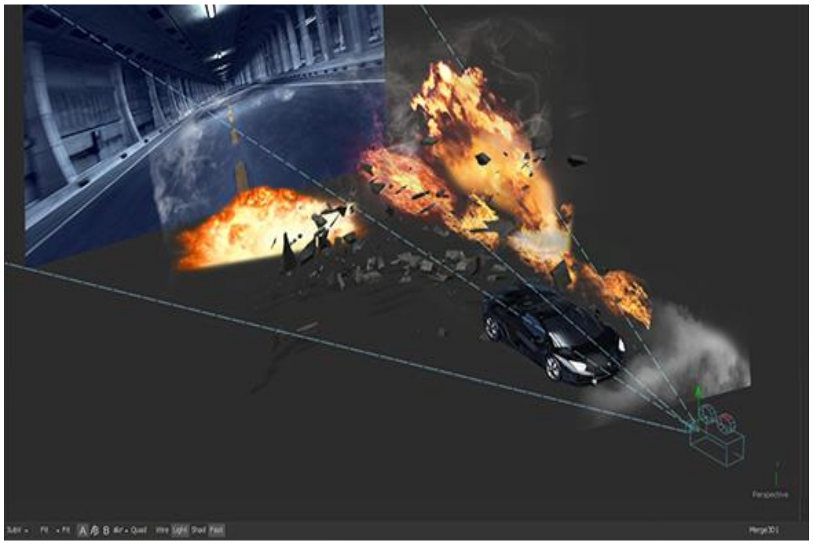


Figure 2.3 3D Compositing

rendered output was no longer dealing with time-based linear storytelling, but of the intentional arrangement of visual elements in space. As convenient and original as the 2D compositing period of development was, during the year 2000 3D compositing forms of production came to fruition [16]. Skills, tools, and techniques originating from 2D compositing quickly translated into 3D Cartesian coordinate space (see Fig 2.3). Notable advantages of 3D space deform, animate, independently access 3D objects that were only imagined and objects mimicking reality, cameras could be animated and manipulated to represent the three-dimensional structure of the world with automatic perspective, and both 2D and 3D elements could meet and be presented within 3D Cartesian space.

The new shift in production tools for Machinima was greatly influenced by the affordability, accessibility, and manipulability of content offered during the Velvet Revolution. For Machinima, recording in-game footage using non-linear editing programs offered machinimators the ability to capture, edit, arrange, and distribute movies in digital format. The flexibility to distribute Machinima films as digital video formats was a powerful transition, allowing access to diverse audiences made up of gamers and non-gamers. However, Machinima moving toward a film format (video capture to screen capture), as Nitsche illustrates, has introduced ramifications, shifting the significance of Machinima from “the recording of the event (video capture) to the recording of a viewpoint to the event (screen capture) [22].” What this means is the shift from video capture to screen capture caused Machinima to lose its value of using a code-based replay engine in post-production, forcing machinimators to repurpose the use of a real-time engine as *only* a production environment, not utilizing all of the engines capabilities; that is not to say Machinima was not successful during this period. In fact, Roster Teeth’s Red vs. Blue comedy series ran between 2003 and 2007, summing up a total of one-hundred episodes; the series exemplified the capabilities of this new mode of production. This shift, although introduced technological shifts from using the game language (coding) to film language (screen capture) in post-production, was accepted as the new norm.

2.3 Machinima's Fragmented Participatory Culture

Following the production shift from video capture to screen capture, new titles such as *The Sims 2* (2004) and Lionhead's *The Movies* (2005) accelerated the popularity of Machinima filmmaking (see Fig. 2.4 and 2.5). The release of *Sims 2* gave way to an increase of Machinima productions, summing up a total of five thousand films between 1996 and 2005; three thousand out of the five thousand films were created with *The Sims 2* [13]. *The Movies*, on the other hand, significantly increased the number of Machinima films produced, summing up to one-hundred thousand in the span of a year (2005-2006). *The Movies*, however, was a game where the player's sole purpose is to create a movie in-game. Players were tasked with editing and arranging movie clips within a game environment to produce a short film within three months (three months in-game). Thus, the very essence of *The Movies* was centered on filmmaking in-game. This form of production, however, raised concerns as to whether films for these games were in fact Machinima. The reason being, games like *The Sims 2* and *The Movies* made Machinima production too easy, according to seasoned machinimators. Pre-2004 Machinima communities and veteran machinimators "felt the essence of Machinima was to show prowess at pushing a game engine beyond the limits intended by its developers [9]." This perspective, Kellan illustrates, is from Machinima community leaders attempting to cling to core values of what was once considered Machinima. In any case, the new production method of Machinima formed

Figure 2.4 *The Sims 2*

Figure 2.5 *The Movies*

newer communities, ranging from seasoned machinimators whose core values resided with using all components of a real-time engine, to machinimators creating films using non-linear editing programs and a real-time engine as a production tool.

2.4 Real-Time Engines for Machinima Production

Speaking from various community perspectives on Machinima, there are advantages and disadvantages of using a real-time engine as a production environment *and* post-production tool. One advantage Salen articulates from using a real-time engine is the opportunity to elicit two kinds of play. "They are, systematized objects, bound by the game's interactive structure and underlying code, but at the same time they are radically free, offering users a unique space in which to perform and play with both narrative and representational codes [24]." Salen's view on this point refers to the visual representation and coding functionality of 3D objects inside a game engine. An example illustrating the power of these two types of play is Unreal 4's *Open World Kite* demo. The purpose of the demo was to present the power of Unreal Engine 4, producing a high quality real-time animation aimed to compete with pre-rendered animated films. More specifically, the two kinds of play Salen illustrates emerge from the artificial intelligence implemented for the herd of deer in *Open World Kite*. The deer not only visually appealed to audiences, but their behavior and animation were influenced by code. To illustrate, the deer were programmed to animate differently depending on the location of the virtual camera; the virtual camera functioned as a virtual predator. If the virtual camera was far enough away from a herd of deer, the deer would animate in a calm, natural way (see Fig. 2.6). If the camera came in close proximity to the deer, the herd would animate scattering, finding other potential routes to flee from the predator, meanwhile analyzing the existing environment to ensure they follow a path that does not lead to falling off a cliff (see Fig. 2.7).

To put the two kinds of play in simpler terms, a rock within a game engine can be viewed and depicted as an ordinary lifeless rock, but fuse code with the rock and an expression emerges, forming a variable the player is able to interact with, thereby creating the probability of expected and unexpected variables to arise during filmmaking. The variable can further be accessed, edited, and defined to perform a function. A function may include the rock turning into a rock monster, breaking, or changing colors. Another advantage of using a real-time engine Kirschner defines is the instant feedback a real-time engine provides. Instant feedback of the engine refers to the real-time functionality allowing filmmakers, artists, and designers the ability to shift and view creative decisions on the fly. The final image is therefore rendered at all times, even when changes are made such as applying lens filters, adjusting intensity and color of lighting

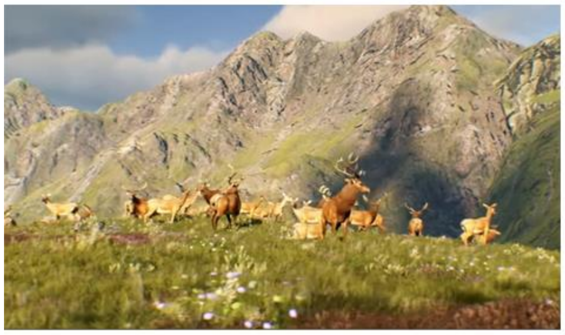


Figure 2.6 Unreal's Open World Kite: Distant Virtual Camera Predator

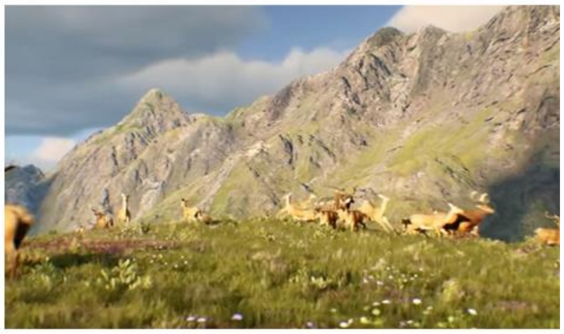


Figure 2.7 Unreal's Open World Kite: Nearing Virtual Camera Predator

conditions, and animating camera movements, among others. The flexibility and complexity of a real-time game engine for developing Machinima films does not come without its disadvantages. Kirschner conversely points out using a real-time engine requires computational compromises that often limit the overall visual quality of a film, lagging behind visual quality produced with pre-rendered animations [13]. Although the matter is subject for debate, as Unreal's *Open World Kite* demo proved a real-time engines visual prowess over pre-rendered content, disadvantages of using a game engine are still in effect. In terms of computational compromises, designers must strategically compress textures, approximate and reduce shadow resolutions, compress animations, and be conscientious of scene memory limitations and performances. The reason for this is because the game engine calculates all resources such as lights, geometry, materials, textures, animations, and coding in real-time. Placing hundreds of complex 3D objects with high polygonal counts, dynamic lights calculating shadows based on moving objects, and high resolution textures can lead to performance drop and memory peak. Thus, to achieve the visual quality presented in Unreal’s *Open World Kite* demo, a designer must be knowledgeable in many areas of production inside a game engine. This brings us to a second disadvantage of Machinima production using a real-time engine, the steep learning curve required to produce content (see Fig. 2.8). Machinima is after all a medium developed from gamers, for gamers [9]. Consequently, many gamers not familiar with asset creation pipeline will not be able to comprehend the complexity involved with creating Machinima content, let alone develop it. Kirschner states the process of generating content such as 3D models, animations, and textures for Machinima leads to a very tedious pipeline. 3D models must first be created using an out-of-game resource such as 3ds Max, Maya, or Zbrush, among others; creating textures for game assets (3D models) require a photo-imaging software such as Photoshop, and animations require complex rigging, skin weight painting, and time-consuming key-framed animations (or motion capture data cleanup). Once all the assets are completed they must be exported from their prospective software application and

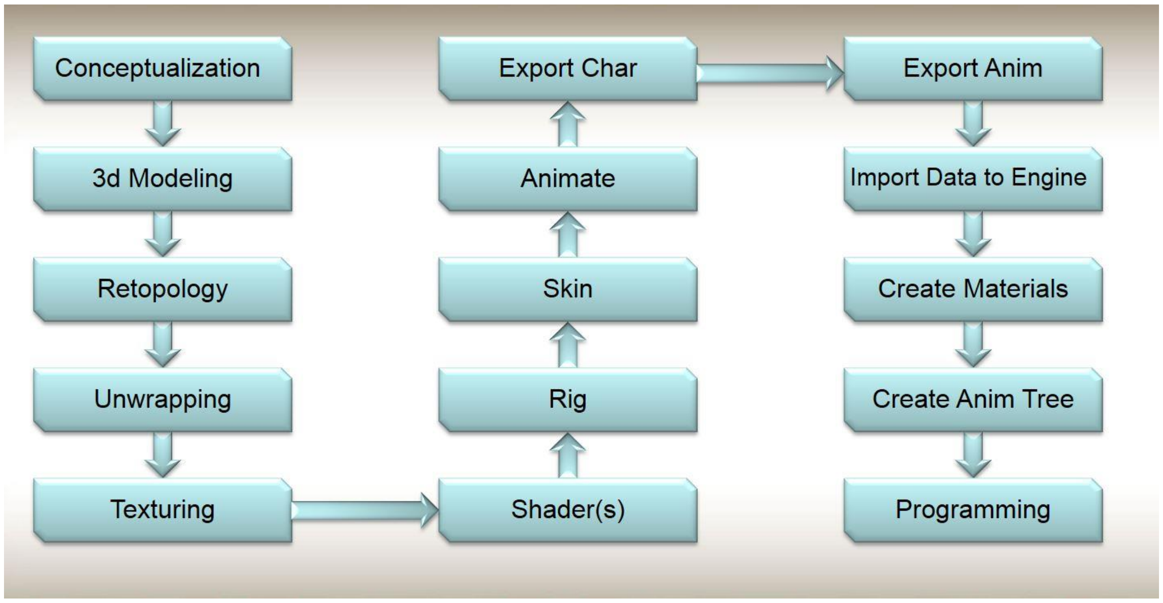


Figure 2.8 Game Production Pipeline

into the real-time engine, where assets will undergo extensive editing in order to be usable. If machinimators require a slight change in animation, they will have to endure the exporting and importing process of external packages due to the limited manipulability and tools available inside a real-time engine to tweak animations. Although the importing and exporting process may seem time-consuming, the benefit of viewing and altering finalized content in real-time, when compared to pre-rendered content, outweighs the time-consuming need to render complex scenes that can span for minutes, hours, and even days [13]. Nevertheless, the steep learning curve involved with producing original Machinima content is a lengthy process that requires newcomers to learn a whole new range of skills with asset creation pipelines, which often prevents machinimators from creating unique aesthetics and narratives. Because of the steep learning curve, seasoned advocates question the effectiveness of using a game engine as a platform for producing Machinima, and whether machinimators required new sets of tools and software to generate Machinima content more easily.

2.4.1 The Ideal Machinima Studio

A 3D game engine is specialized software designed to perform very specific tasks, game related. Since Machinima is a moviemaking hybrid, Kirschner believes filmmaking tools would greatly assist machinimators in designing, editing, and producing content. Kelland agrees, stating new Machinima tools are beginning to surface that will simplify Machinima production [9]. The ideal software would offer machinimators a "real-time approach of Machinima filmmaking in all other aspects of 3D animation production [13]. Kirschner expands on the ideal software for Machinima, stating machinimators require a rapid workflow for asset building, animation, camera setup, character directing, and story scripting. An example of this Machinima tool is the real-time 3D application called *Moviesandbox* (see Fig. 2.9). In *Moviesandbox* machinimators are able to

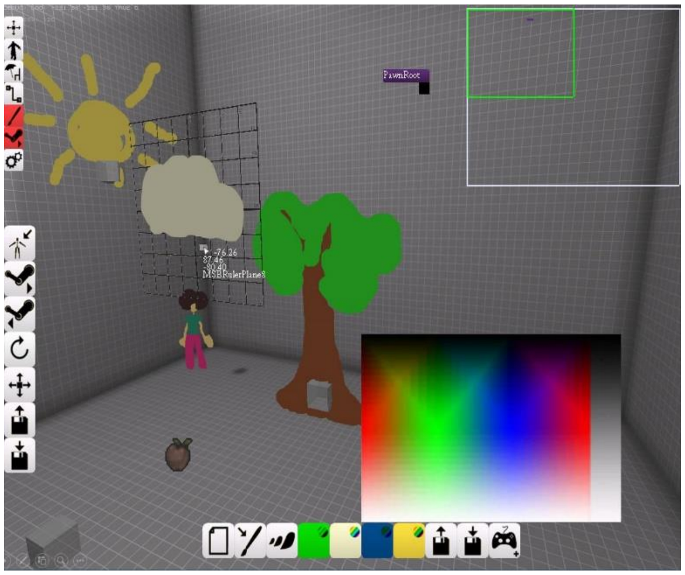


Figure 2.9 *Moviesandbox*: The Ideal Machinima Studio

quickly design, edit, and produce Machinima. The simplistic GUI interface and dedicated tools offer fast prototyping methods for designing props, characters, setting up lights, camera movements, animating, and storytelling. However, this method of Machinima production gave rise to yet another community, expanding the underlying meaning and process of generating Machinima films. The communities were then split amongst seasoned machinimators whose core values of Machinima resided solely inside a video game engine for production and post production, machinimators fusing a video game engine (production environment) and screen capture (post-production) technology, and machinimators outsourcing to a dedicated Machinima real-time engine for fast prototyping and production. Regardless of how and where Machinima was created, the new medium found itself under-exceeding in its potential to alter how animation and film were digitally made [23].

2.5 Why Machinima Doesn’t Seem to Want to Grow Up?

Paul Marino, one of Machinima’s seasoned advocates states “the promise of filmmaking within a virtual space still needs to be fully realized [17].” Marino explains the visual quality and technical advances of game engines and technology have drastically evolved, but the creativity involved for producing stories within a game engine has not met its potential. Salen expands on this point, stating Machinima productions lack a fundamental balance between code and visuals, a valuable component to the origins of Machinima. Salen continues to emphasize that “Machinima doesn’t seem to want to grow up, that game boys are still making silly little films about, well, game boys [13].” What Salen suggests is perhaps the time has arrived for Machinima to expand its target audience appeal to include non-gamers seeking exploratory forms of storytelling through Machinima, rather than exclusively targeting and developing for a narrow gaming audience developing content from games, about games. After all, if there was one area Machinima excelled in it would be promoting convergence culture, the way media is produced and consumed simultaneously from the top down and bottom up [5]. Because Machinima originated as an artistic medium targeting gamers, according to Kelland, developers have created Machinima tools dedicated to *gamers*, offering simple integration, manipulation, and implementation of content (i.e. *Moviesandbox*). Salen further explains the reason for developing simpler tools for Machinima production is directly influenced by the steep learning curve required to build films inside a game engine. Once the mode of production shift between video capture and screen capture took place, however, Machinima was no longer viewed and shared exclusively within gaming communities. The inclusion of participatory practices involving gamers and non-gamers eventually impacted the way Machinima was produced, unintentionally fragmenting various communities with their own distinct understanding of what constitutes Machinima, and how films were produced.

2.6 Enriching Machinima with Modern Input Devices

Machinima as a hybrid between cinematic production, games, live performance, and puppeteering of 3D characters held high hopes for offering new systems of production that would revolutionize film [23]. Even though Machinima has not revolutionized filmic productions, the potential to inspire and facilitate change remains optimistic. Designers and researchers have been searching for more efficient ways to integrate physical and digital spaces, an area in which Machinima can make a contribution [13]. Mazalek suggests live performance and cinematic production is the heart of Machinima [19].” Mazalek is not wrong, as live performance was seen during the video capture era of Machinima, where players recorded in-game performance to show off their abilities to other gamers seeking to improve their own skills. Cinematic production was more prominent during the screen capture phase where machinimators used out-of-game resources such as FRAPS, After Effects, and Adobe Premier to tell linear stories. Mazalek builds on this point, suggesting that fusing tangible user interfaces (TUI) and ubiquitous computing technology would enrich Machinima production (see Fig. 2.10).

> "From ubiquitous computing research, Machinima can derive the importance of providing transparent technological tools, where the interface disappears in the face of the task performed by its user. Machinima can also benefit from interfaces that make use of human manual dexterity to provide expression in a virtual space and that offer immediate feedback to our actions, two central notions of tangible computing [13]."

Mazalek emphasizes the importance of using modern technology to tell unique Machinima stories by means of live performance. Current forms of input devices such as gamepads, keyboards, mice, and joysticks offer limited forms of expressive qualities for developers to explore when creating Machinima [20]. Using technological tools (i.e. head mounted displays and eye tracking devices) as input devices to interact with digital information from a physical

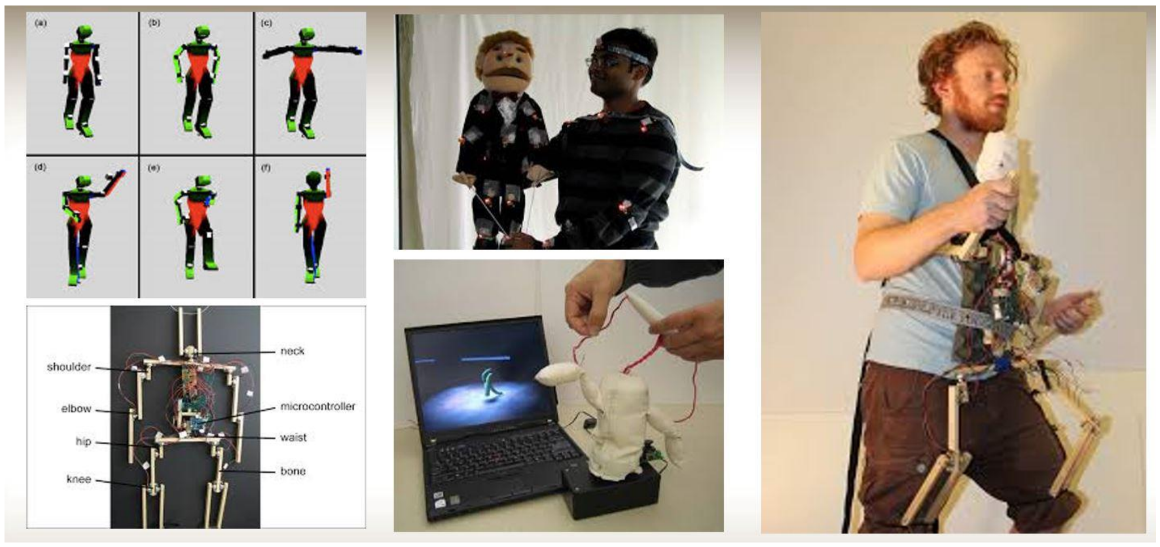


Figure 2.10 Fusion of Tangle User Interfaces and Ubiquitous Technology

environment, coupled with digital interactive storytelling inside a game engine will pave a new avenue for machinimators to explore collaborative alternatives to create theatrical stories. Although the success of this form of Machinima production cannot be empirically evaluated in its current state, Mizuko believes "the promises and pitfalls of certain technological forms are realized only through active and ongoing struggle over their creation, uptake, and revision [13]." In other words, until someone attempts to break the barrier and fuses Machinima, coding, modern technological interfaces, and live performance, no progress will be made to individualize Machinima as a distinct medium.

Hugh Hancock of Strange Company and ILL Clan were the first two companies to use screen capture in Machinima production, marking an important point in Machinima history that gave way to the development shift from video capture to screen capture (see Fig. 2.11 and 2.12).

Figure 2.11 *Eschaton: Darkening Twilight*

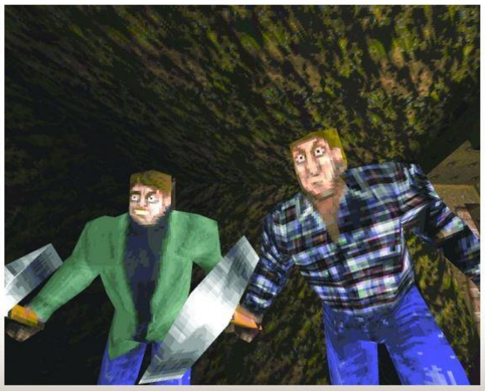


Figure 2.12 *Apartment Huntin'*

Because Machinima builds on previous traditions including game design, film, live performance, and puppeteering, it is only natural for designers to begin fusing modern technology from each field. Fusing different technology, techniques, and expressions is nothing new in our society. As Knobel point out our culture is an "endless remix of hybridization [10]." Knobel aims to point out is, our culture is constantly mixing and remixing different media, whether in the form of compiling different sound tracks, creating photo collages, and creative writing, among others. The learning structure of school is an example of endless hybridization. If a student is required to write a paper contrasting two author views and evaluating their views with your own words, there will be some form of personal evaluation to what has already been written. That being said, in writing, an individual is in fact hybridizing pre-existing content and rephrasing the significance to get the point across. This form of hybridization does not seem to be translating in Machinima. The advancement of technology in varying disciplines over the past decade has questioned whether Machinima has not matured because there is a significant technological lag. Machinimators are still using traditional keyboard, mice, and game pads to puppeteer characters, rather than using the portrayal of Machinima as a hybrid media to their advantage. Head mounted displays and eye tracking technology are prime examples of technology that can be used for Machinima productions which have not been explored. Although head mounted displays and eye tracking devices are separate functional entities and individually experienced, they have shown remarkable interaction capabilities introducing expressive viewing and ubiquitous computing possibilities within virtual reality.

2.6.1 Eye Tracking and Virtual Reality

Head mounted displays such as the Oculus Rift and eye tracking devices such as Tobii EyeX are two technological innovations offering unique possibilities to produce Machinima content by means of live performance. To add, head mounted displays and eye tracking satisfy two important descriptors of virtual reality, leading to increased engagement and strong senses of presence, an attribute Machinima can greatly benefit from. That is, vividness and interactivity [25]. Vividness Seibert illustrates is how technology is able to convincingly simulate multiple senses at once. Head mounted displays are an example of this device, engaging visual and auditory receptors (see Fig. 2.13). To put it more vividly, Buczek describes virtual reality as a surrounding space that sensually involves the viewer, where the camera movement is referred to as your own [2]. Wearing a head mounted display, this gesture can be unexpectedly intense and can even physiologically affect a user's sense of balance, commonly leading to what is termed simulation sickness (i.e. cybersickness). Simulation sickness is classified as a subset of motion sickness where users experience headaches, nausea, and in extreme cases tunnel vision. The causal variables inducing simulation sickness is currently undergoing extensive research. However, the drawback of simulation sickness elicited from head mounted displays did not halt its transformation. The use of head mounted displays spread to various disciplines including education, human computer interaction, medical simulations, army simulations, 360 videos, museums, augmented reality, video games, and mobile technology. One example of possible innovations deriving from the ability to interactively view 360 degrees inside a virtual space is Glen Kean's Duet animation. The hand drawn sixty frame animation depicts an unwinding story of a baby boy and baby girl taking gradual steps into a romantic involvement into adulthood. Kean advantageously made use of the spatial dimension to tell a story in 3D space, allowing viewers the freedom to at any time pursue their favorite character and watch their story unfold by simply moving your head. Interactivity, the second descriptor of virtual reality, refers to the extent a user is able to influence the virtual environment in which they reside [25]. Influencing a virtual environment may come in the form of modifying an object shape, activating scripted events, and moving virtual characters, among others. An example of existing technology capable of influencing virtual environments is eye tracking. Eye tracking has gradually evolved over the years, setting forth possibilities to construct responsive interfaces by means of tracking eye movement. These attributes can further be independently accessed and programmed accordingly, offering many interaction opportunities. Eye tracking technology, similar to head mounted displays have led companies, designers, educators, artists, writers, editors, and researchers to analyze eye tracking patterns in an attempt to understand why people

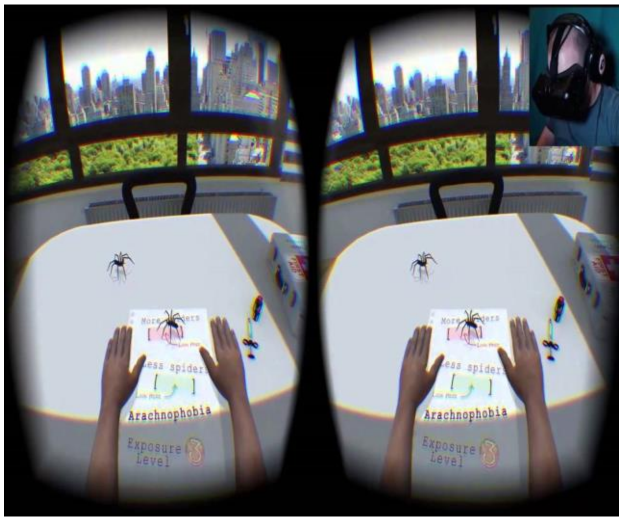


Figure 2.13 Virtual Reality

look at content and for how long. Heatmaps are a common form of eye tracking data collection, a graphical data representing values based on varying intensity. This data can then assist companies develop more appealing and accessible interface designs, define viewing patterns to strategically place advertisements, develop a language for human computer interaction, experience more immersive virtual reality experiences, and naturally enhance gameplay mechanics. Tobii EyeX has already infiltrated AAA games, enhancing immersion and adding transparent layers of natural control. *Assassin's Creed: Rogue* is an example of AAA title which has incorporated eye tracking technology to allow for more immersive, hands-free controls (see Fig. 2.14). Players would no longer require joysticks to rotate the camera view. Using eye tracking systems, the game window or frame from which the viewer sees the world from the characters' perspective would automatically adjust to the location of gaze, leading to greater immersion and more natural forms navigation. *Assassin's Creed: Rogue* players positively responded with this innovative and unobtrusive form of gameplay. Although this new form of human computer interaction shows great promise with future interfaces, much research is needed to determine a suitable language for its uses [18]. Gamers and machinimators can certainly expect eye tracking software to greatly influence future designs in storytelling and all areas of human computer interaction. Eye tracking is a powerful unobtrusive tool requiring proper fundamental integration to improve in game performance and transparency within the virtual world, potentially leading to greater levels of immersion.

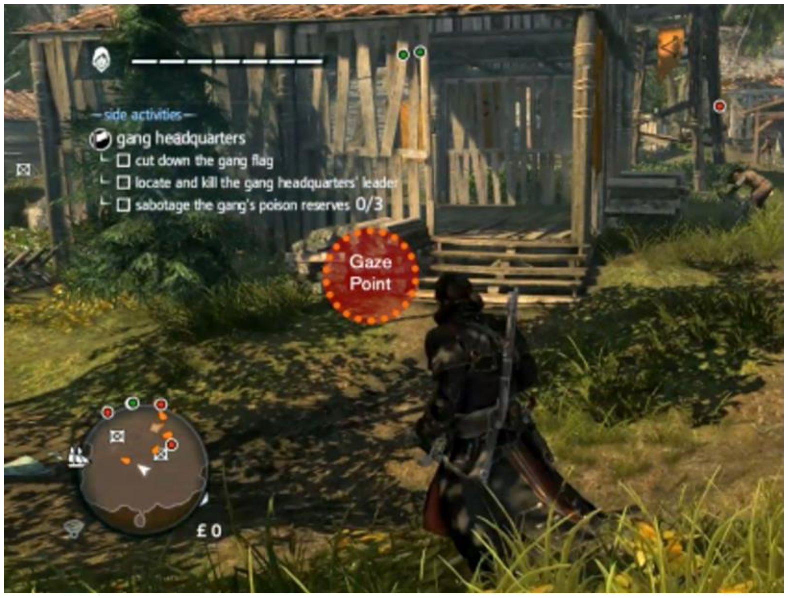

side activities
gang headquarters
cut down the gang flag
locate and kill the gang headquarters' leader
sabotage the gang's poison reserves 0/3
Gaze Point
£0

Figure 2.14 *Assassin's Creed: Rogue* Eye Tracking Integration

3. System Design

3.1 System Requirements

Machinima in the past decade has relied on input devices with limited expressive qualities to puppeteer characters and record in-game live performances, using a gamepad, keyboard, and mouse to produce content [17], [24]. While the use of analog joysticks, buttons, and triggers are fundamentally the most widely accepted practice for creating Machinima films the way we play games, our system deviates from these practices. Instead, our system sheds light on Machinima's potential for *anymation,* by incorporating a seamless form of human-computer interaction for real-time performance and capture (i.e. eye tracking), thereby expanding Machinima production tools, techniques, and possibilities. Conveniently, Unreal Engine 4 and its active community present the opportunity to marry previously distinct technologies, which made possible the creation of this intuitive system.

While the above outlines stagnated input devices habitually used in Machinima production, our system perseveres from the constraint of using games as platforms for Machinima filmmaking, which quickly become outdated. Instead, our system takes the form of a plugin within Unreal Engine 4, which serves as a convenient jumping off point enabling users to select from a list of prebuilt mechanics to begin producing Machinima (see Fig. 3.1). As a plugin, our system can be integrated, with a bit of knowledge, into any Machinima project at any phase in production, (so long as it is within the confines of

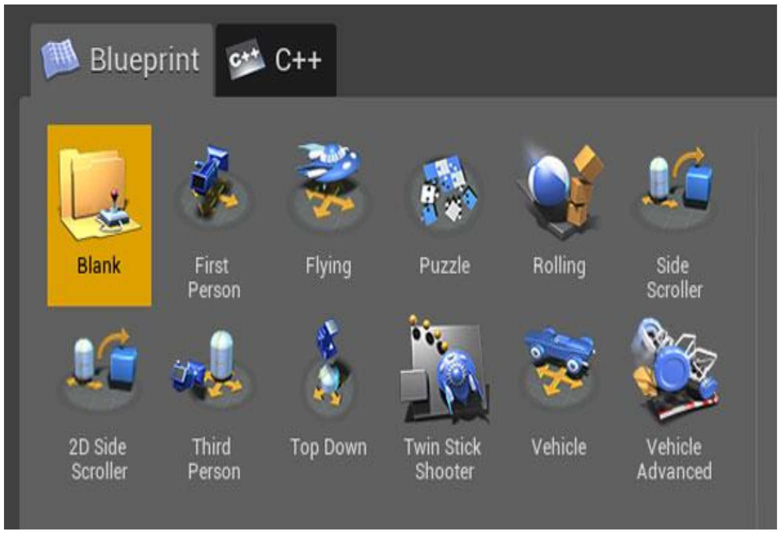


Figure 3.1 Unreal Engine 4: List of Game Templates

Unreal Engine 4). As a plugin, our system strives to remedy Machinima's constraint of using games with dedicated game hardware, enabling our system to evolve with the Unreal community by means of system updates. In doing so, a playground of possibility emerges, introducing experimental opportunities that is applicable to producing and capturing scenarios for cinematography, including scene development, artificial intelligence, shooting, choreographing, directing, and collaboration.

Our system exemplifies this experimental opportunity by providing a transparent, intuitive, flexible, and collaborative user interface. However, to enable collaborative Machinima production, a networking infrastructure using two Windows desktops is necessary in order to: (a) enable multiple users to interact and collaborate within a shared, virtual space, and (b) optimally render a stereoscopic output for the Oculus Rift DK2 head mounted display and monoscopic output for the Tobii EyeX eye tracker (see Fig. 3.2). In regard to optimization, the Oculus Rift DK2, unlike the eye tracker, requires a special render setting that duplicates the viewer's in-game

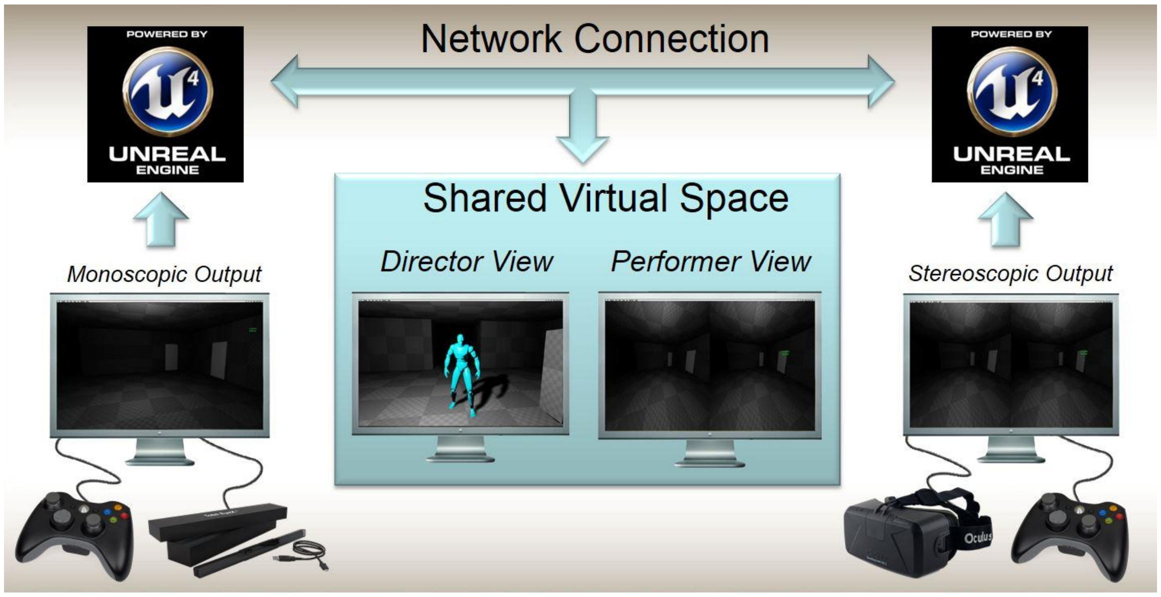


Figure 3.2 Networking Infrastructure: Monoscopic and Stereoscopic Output

camera at slightly different angles to create an impression of depth- stereo. This setting makes possible the use of virtual reality within Unreal Engine 4, but does not come without its drawback. Because Unreal Engine 4 calculates complex lighting, geometry, materials, animations, textures, and post-processing effects in real-time, a computer with strong graphical processing power is required. Without a *high* performance graphics card, the Oculus Rift DK2 operator using our system can experience significant latency. With increased latency, users are more susceptible to experience simulation sickness [25]. Add three game windows (two for stereoscopic and one for monoscopic output) rendered simultaneously to a single computer, and the system usability will decrease significantly, countering the system plugin's intuitive design. Thus, for maximum in-game performance for the Oculus Rift DK2 operator, two separate desktop computers are used to maintain Oculus's recommended 75 frames per second (FPS) playback.

3.2 System Architecture

Leveraging Unreal Engine 4 with affordable modern input devices, it is possible to create a collaborative real-time virtual cinematography tool. The culmination of game and production technologies, which led to the development of our system, must be used synchronously to effectively make use of its functionalities.

3.2.1 Game Engine

The system plugin is reliant on using Epic Games' Unreal Engine 4 version 4.10. Unreal Engine 4 is a free, robust game engine contributing to pushing games such as Hellblade (view Fig. 3.5) and cinematic content such as Open World Kite (view Fig. 3.4) to newer levels of high visual fidelity. The gap between the quality of game production and cinematic production continue to narrow.

Figure 3.3 Unreal Engine 4

Figure 3.4 Unreal's Open World Kite

Unlike pre-rendered content, game engines provide many attributes, including real-time feedback of iterative design, the benefit of seeing the final look at all times, and the ability to calculate complex models, scenes, lighting, animations, and textures in real-time. In addition, Unreal has contributed to five million dollars in Unreal Dev Grants given to individuals and teams to incentivize the production of better content, tools, and learning materials for the Unreal community, contains 1.5 million dedicated developers, provides extensive support, resources, and documentation, and actively integrates third-party developer plugins with engine releases, which has contributed to the successful culmination of our system. Unreal's active involvement in pushing the game engine's limitations and encouraging participatory culture exemplify the validity of the engine's recurring extensibility, and thus why it is selected as the core development platform.

Figure 3.5 Hellblade

In addition to the above outlined attributes of using Unreal Engine as the selected platform for our system, the game engine is also selected because of its intuitive visual scripting node-

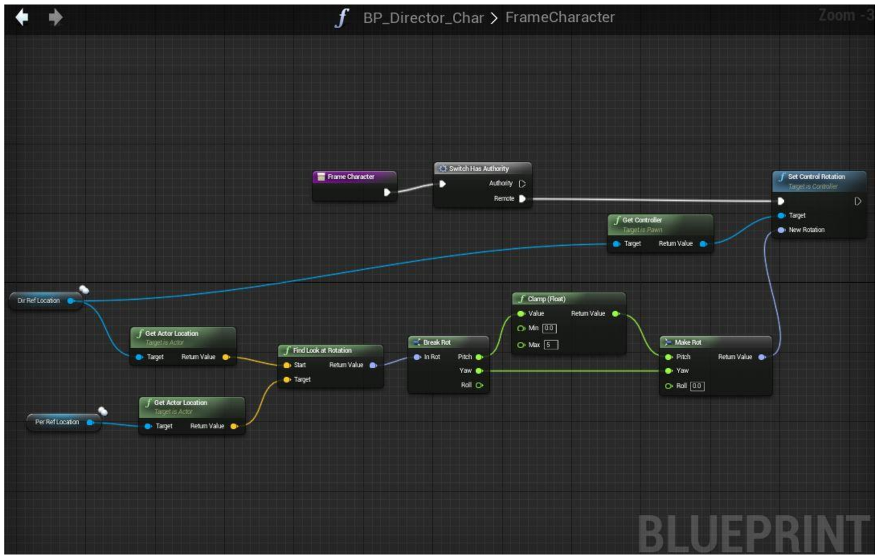


Figure 3.6 Unreal Engine 4 Visual Scripting Editor: Blueprints

based editor referred to as blueprints (view Fig. 3.6). The visual scripting editor is an alternative programming method that enables users to program elements graphically using visual expressions and symbols, rather than textually. Blueprint functionalities lower the technical learning curve for programming, making it an ideal platform for Machinima newcomers who may not possess the wider knowledge base required for programming using traditional text-based platforms such as Microsoft Visual Studio. Since text-based syntax is less of a constraint in Unreal Engine 4, newcomers are able to quickly learn, modify, and expand our system to cater their needs.

3.2.2 Input Devices

Eye Tracking

Tobii EyeX's eye tracker offers consumers, at an affordable price, natural human-computer interaction capabilities using eye-gaze. The eye tracker, which is integrated into our system, uses near-infrared light (NIR) to track eye movements and gaze point of the user (view Fig. 3.7). Its EyeX Engine works similar to an Operating System extension, repeatedly reacting to a user's

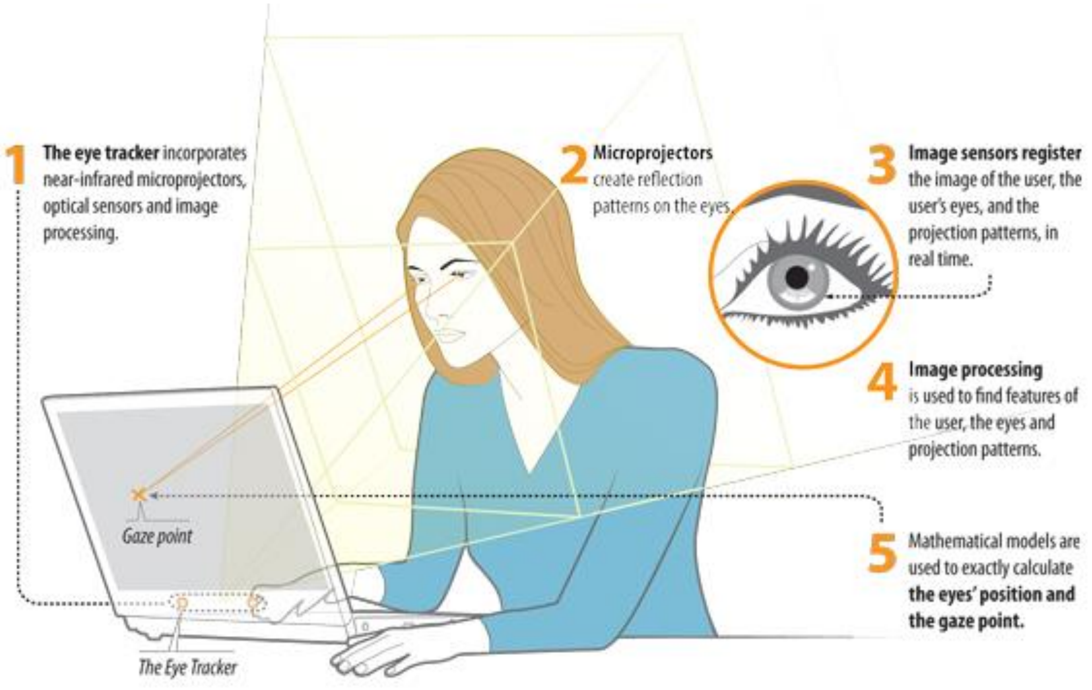

1 The eye tracker incorporates near-infrared microprojectors, optical sensors and image processing.
2 Microprojectors create reflection patterns on the eyes.
3 Image sensors register the image of the user, the user's eyes, and the projection patterns, in real time.
4 Image processing is used to find features of the user, the eyes and projection patterns.
5 Mathematical models are used to exactly calculate the eyes' position and the gaze point.
Gaze point
The Eye Tracker

Figure 3.7 Eye Tracking

eyes, sending messages to the system, interpreting the data by mediating between multiple applications, and together forms a built-in heuristic that streams and filters data to continually determine the user's location of gaze. In other words, the EyeX Controller calculates the user's gaze point coordinates, then the EyeX Engine receives and transforms the screen coordinates to pixel coordinates on screen. To accurately transmit the eye-gaze coordinates, a brief calibration setup is required. Then, users are able to experience the EyeX Engine API (view Fig. 3.8) within Unreal Engine 4. The way in which the EyeX Engine API works with our system can be divided into three categories: streams, states, and behaviors.

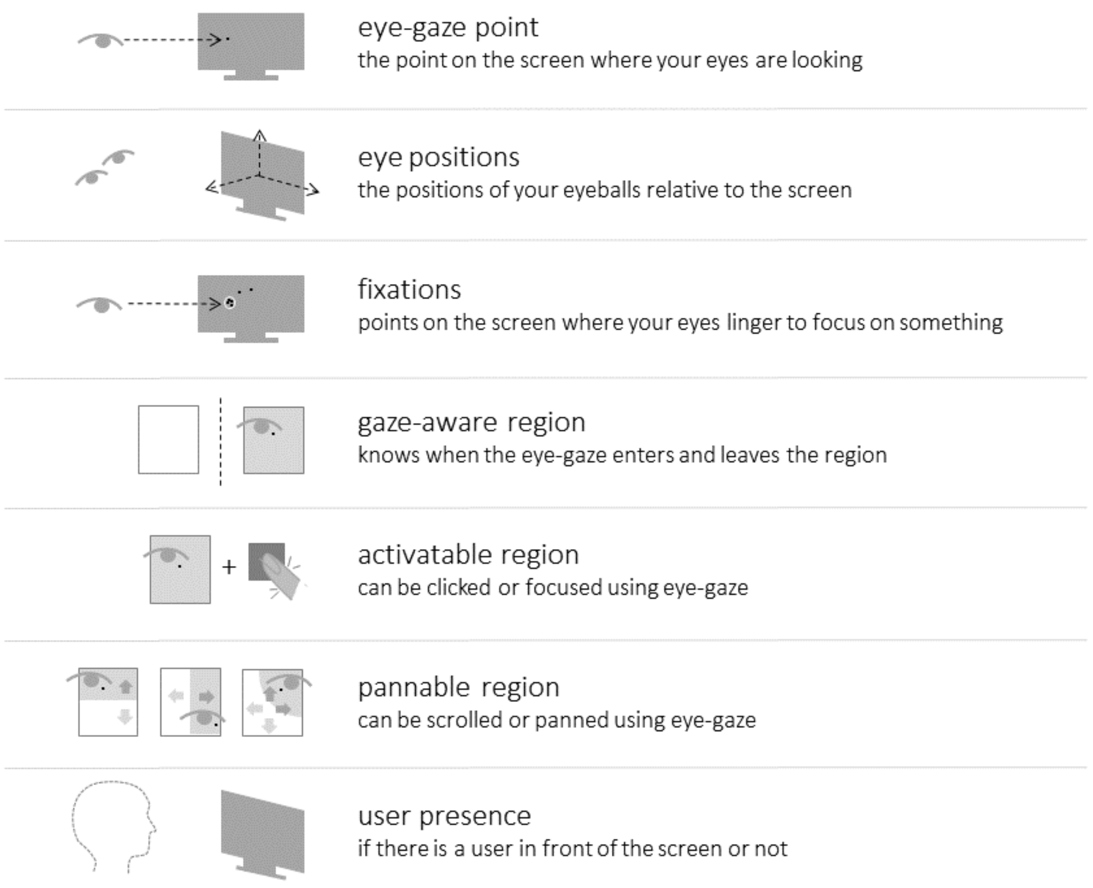
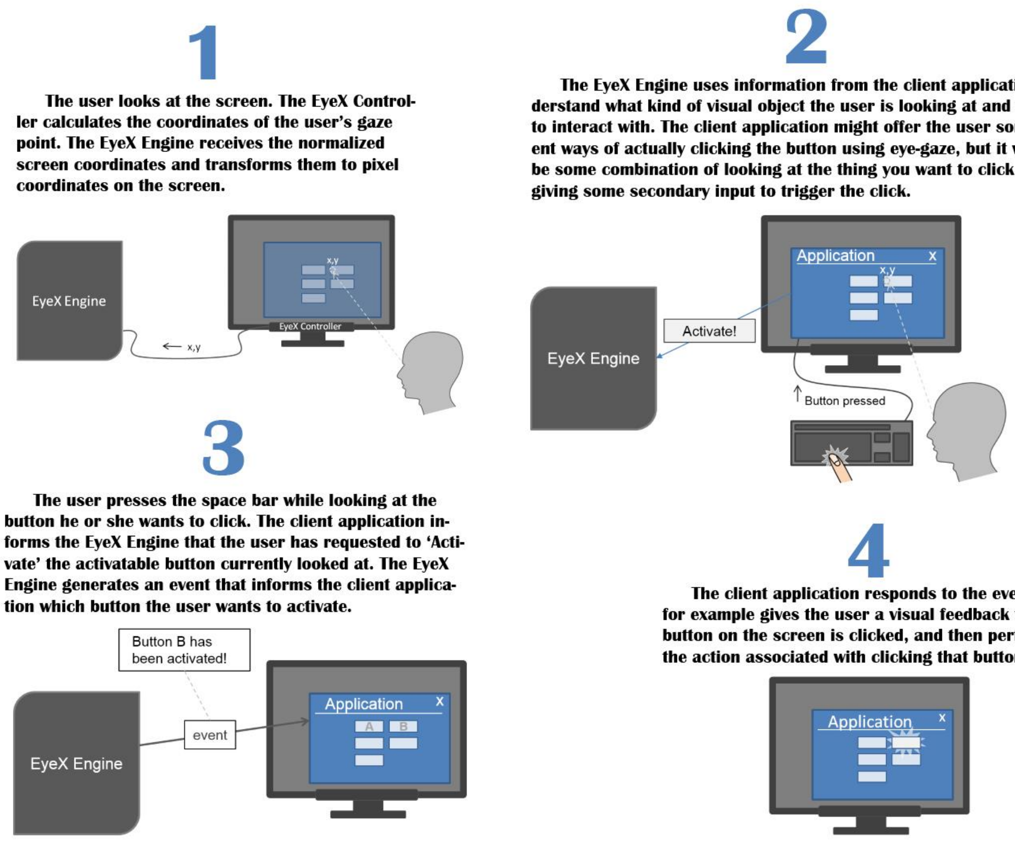

eye-gaze point
the point on the screen where your eyes are looking
eye positions
the positions of your eyeballs relative to the screen
fixations
points on the screen where your eyes linger to focus on something
gaze-aware region
knows when the eye-gaze enters and leaves the region
activatable region
can be clicked or focused using eye-gaze
pannable region
can be scrolled or panned using eye-gaze
user presence
if there is a user in front of the screen or not

Figure 3.8 EyeX Engine Functionality

Streams: Streams contain smoothly filtered eye gaze data that is transformed into a dedicated coordinate system. That is, *gaze point*, *eye positions*, and *fixations*. Gaze point is the place on screen where the user's eyes are looking at, eye positions are the location of the user's eyes (in millimeters) relative to the distance of the physical eye tracker, and fixations are locations where a user's eye linger to focus (view Fig. 3.8).

States: States track the current state of the EyeX system. In other words, it tracks the dynamic state of the user. These are, *user presence* and *gaze tracking*. User presence tracks to see if a user is in front of the eye tracker or not, and gaze tracking relates to whether a user's gaze is being tracked (view Fig. 3.8).

Behaviors: Behaviors relate to more complex types of interactions, such as scrolling and clicking on a screen. Interaction behaviors in this scenario require either mapped activation regions on screen or supplement the activation region with another form of input (i.e. button on a gamepad, keyboard, mouse, and so on) to confirm interaction (view Fig. 3.9).


1

The user looks at the screen. The EyeX Controller calculates the coordinates of the user's gaze point. The EyeX Engine receives the normalized screen coordinates and transforms them to pixel coordinates on the screen.

2

The EyeX Engine uses information from the client application to understand what kind of visual object the user is looking at and might want to interact with. The client application might offer the user some different ways of actually clicking the button using eye-gaze, but it will always be some combination of looking at the thing you want to click, and then giving some secondary input to trigger the click.

3

The user presses the space bar while looking at the button he or she wants to click. The client application informs the EyeX Engine that the user has requested to 'Activate' the activatable button currently looked at. The EyeX Engine generates an event that informs the client application which button the user wants to activate.

4

The client application responds to the event and for example gives the user a visual feedback that the button on the screen is clicked, and then performs the action associated with clicking that button.

Figure 3.9 EyeX Engine: Behavior API

What isolates Tobii EyeX from the consumer market is two-fold: (a) the eye tracker is the first of its kind to be used in games such as *Assassin's Creed Rogue* and *The Division*, among others, (b) and the eye tracker provides a readily accessible third-party plugin for use within Unreal Engine 4, enabling users to experiment with its unique interaction capabilities. For example, eye tracking is being used in *The Division* as a way to aim and shoot a weapon in the direction of gaze. In *Assassin's Creed: Rogue*, the eye tracker is used as a way to move the character's camera viewpoint to the location of gaze, highlight virtual objects, reveal information, and so on. Because such an eye tracking system is relatively new, there is a growing body of research as to its potential application, and so this technology *is* selected as central for experimentation as a virtual cinematography tool.

Gamepad

Eye tracking will work in tandem with a wired twin-stick Xbox 360 gamepad (see Fig. 3.10) for greater precision of interactions. As mentioned above regarding Tobii's EyeX Engine's API, complex interaction behaviors such as clicking and scrolling for eye tracking require either an activation region or an activation region supplemented with an input device to confirm an interaction. For our system, initial explorations exclusively used eye tracking with activation regions to interact with the virtual environment. This method of interaction introduced significant hardships, enabling users to - at all times- interact with their surrounding virtual space, intentionally and unintentionally. This does not make for an intuitive system. Assuming the user is gazing in a particular location on screen does not imply the need for interaction. As a result, activation regions are modified to include the most widely accepted input device in Machinima production, a gamepad (Xbox 360 controller).

Gamepads enable sophisticated control via analog stick, buttons, and pressured triggers. They provide a natural user interface that is widely supported in games. Yet, with analog inputs comes a great burden. There are only so many buttons that can perform only so many functions. However, that is not the case within our system. The gamepad works in tandem with eye tracking, which increases the limited number of functionalities each button is able to perform. Reason being, the functionality of each button is driven by what the user is gazing toward, opening up fertile ground for interaction possibilities from analog input devices.

Figure 3.10 Xbox 360 Gamepad

Oculus Rift DK2

The Oculus Rift DK2 (see Fig. 3.11) is an affordable head mounted display compatible with Unreal Engine 4. It is important to note the Oculus Rift DK2 will undoubtedly serve a greater purpose in future work of real-time animation capture and editing. In the system plugin's current state, the Oculus Rift DK2 serves as proof that the integration between these two distinct technologies is possible within Unreal Engine 4.

At the time of this writing, Unreal Engine 4 contains a prebuilt plugin enabling the use of Oculus Rift DK2 head mounted display to be functional and its parameters editable. The Oculus Rift DK2 requires SDK and Runtime 0.4.0 to be installed for compatibility with our system (Unreal version 4.10.4). In doing so, upon loading the game build, a stereoscopic display immediately takes effect. While operating the Oculus Rift DK2, the positional tracker camera, supplemented with the head mounted display, tracks the position of the Oculus Rift in physical

Figure 3.11 Oculus Rift DK2

space and translates any head movements performed by the operator into the virtual space.

3.2.3 Networking

Our system relies on networking to establish collaborative play and bridge the Oculus Rift DK2 and Tobii EyeX eye tracking. Tobii released a plugin for the eye tracker to work in conjunction with Unreal Engine 4 titled *EyeXforUE4*. The plugin is used in our system across a server via UE4's server framework. Using *EyeXforUE4*, the eye tracker, after calibration, streams across a server via IP connection.

Oculus Rift DK2 also has an SDK and plugin that is integrated with Unreal Engine 4's server framework via IP connection. This makes possible the rendered output of stereoscopic and monoscopic displays within a shared virtual space.

3.2.4 System Rationales

The above articulates hardware and software which make up the backbone of our system plugin. Through a build of the game, two or more users are able to connect via network and collaboratively produce Machinima.

Our system design, in its prototype state, is limited to two roles, the *Director* and *Performer*. The selected roles are a result of carefully thought out functionalities that would best showcase our system's usability; by no means are these two roles the limitations and capabilities of our system. In fact, more than two operators are able to collaborate within the same virtual space, whether undertaking the *Director* or *Performer* role. Because Machinima production is a collaborative effort, our system offers a readily accessible collaborative platform whereby team members, whose locations may vary from local to distant, can establish choreography over a network (see Fig. 3.2).

Upon running a standalone build of the game, a scene populated with custom pre-made assets are loaded for rapid prototyping. Custom assets are made to provide machinimators a readily accessible filmmaking environment for Machinima production. Although the custom scene is pre-designed, users of our system have access to every visual component; meaning, machinimators have the ability to edit the scene- add custom assets, rearrange the scene elements, alter the visual aesthetic, change lighting- to cater their needs during production (view Fig. 3.12).

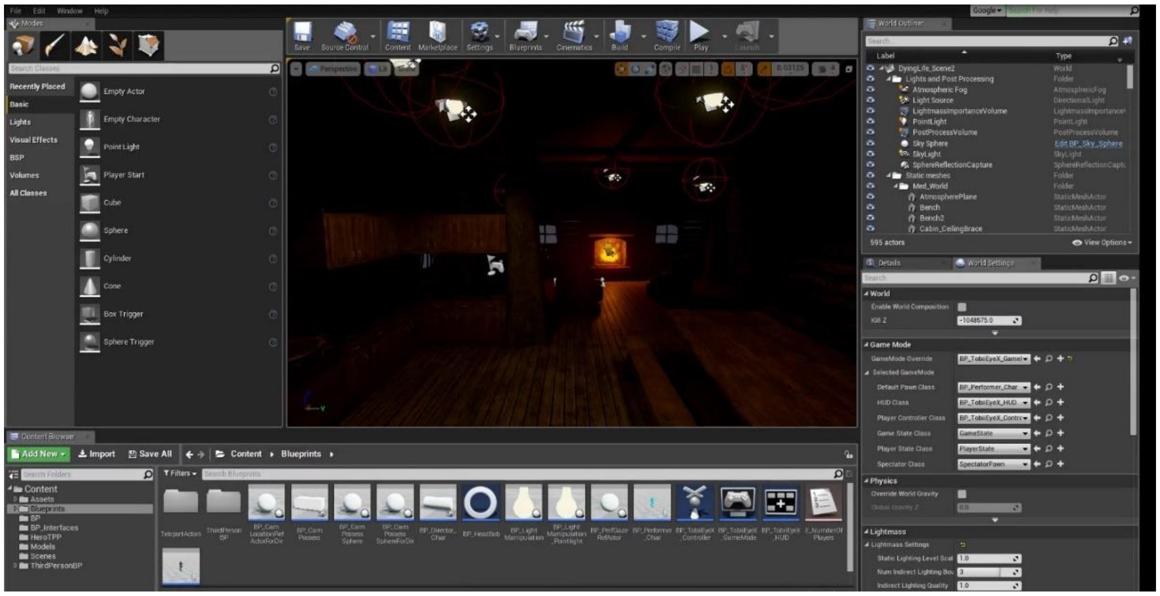


Figure 3.12 Unreal Engine 4 Interface

Developing a democratized, affordable, collaborative, intuitive, and low-entry barrier system are core features of our plugin. In the interest of democratization, our system plugin is distributed as open source, and is integrated within a robust game engine that is free to use. What this means is, our system is designed in such a way that consumers can employ personal computers as platforms for creation of game-generated cinema. And because the hardware and software used in our system is affordable and accessible, users have the opportunity to individually or collaboratively produce sophisticated cinematic productions from the comfort of their own homes, with only a fraction of the cost when compared to equipment used in high end cinema productions. For low-entry barrier, our system focuses on establishing collaborative interactions and basic camera motions during runtime. Efforts are made to limit the need to exit the game engine, which would require newcomers to learn how to navigate within Unreal’s content-rich user interface (view Fig. 3.12). Instead, our system is designed to be used during runtime in order to focus what is important, that is real-time directing, rehearsing, and performance capture. Although the system is designed as *plug-and-play*, the nature of the system as a plugin adds the flexibility and modularity to customize and expand its functionality, tweak settings based on user preferences, and integrate distinct technologies (i.e. eye tracking, virtual reality, Perception Neuron’s real-time motion capture solution).

As a *plug-and-play* system, our plugin offers a highly accessible jumping off point for consumers to begin creating animated films, untethered to the tediousness of learning complex 3D application workflows and non-linear editing programs. Although this system, in its current state, temporarily imposes limitations to the types of narratives and visuals, its function serves to provide an alternative means for real-time animated film production, using a game engine as a filmmaking platform for production *and* post production- a key component of Machinima’s origin.

3.3 Interaction Design

As outlined above, the featured roles in our system include the *Director* and *Performer* (view Fig. 3.13). The *Director* operates an invisible floating camera and uses gamepad to navigate within the virtual space. Using the Tobii EyeX eye tracker (recognized by EyeX Engine's API and accuracy registered by completing a simple calibration step), the *Director* is able to trigger in-game events using eye gaze; the *Director* also vocally choreographs performances in real-time. In contrast, the *Performer* puppeteers an avatar that is visible in-game, using a gamepad and Oculus Rift DK2. The sole purpose of the *Performer* is to perform in-game. For the Oculus Rift DK2, no calibration is required; although, the user must position the head mounted display approximately five feet away from the supplemented positional tracker for maximum positional accuracy. In regard to gamepads, both the *Director* and *Performer* use the gamepad controller, which contain pre-defined mechanics. The mechanics differ from *Performer* to *Director*, meaning the *Director* is able to possess director-related mechanics, including creating/cutting to/deleting

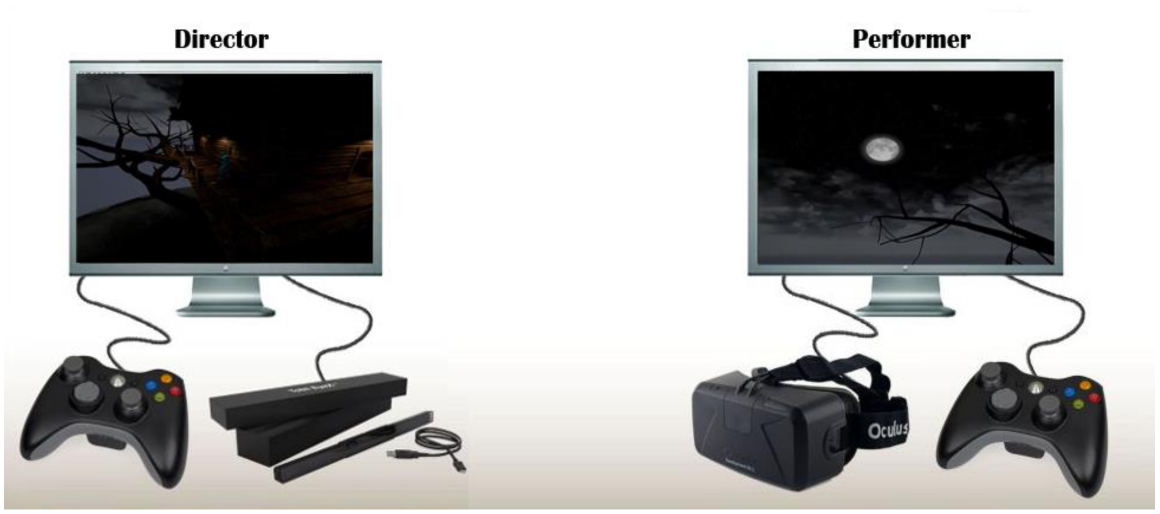


Figure 3.13 Director and Performer

cameras, performing camera related motions, target lock-on, collaborative item swapping, staging, and light manipulation. The *Performer*, on the other hand, has performance related functions to puppeteer an avatar, including walking, running, jumping, and opening a door. The following section details the interaction design for both the *Director* and *Performer*.

3.3.1 The Director

Due to the system's usability as a virtual cinematography tool, camera functionalities are emphasized to showcase eye tracking possibilities and uses within a collaborative virtual filmmaking environment. The following section overviews the core mechanics of the Director: camera navigation, target locking, camera spawn/possess/delete, collaborative item swapping, lighting manipulation, and staging.

Camera Navigation

For intuitive Machinima filmmaking to be both effective and efficient, flexibility of camera navigation is required in order to maintain constant view of the focal point. Thus, in our system, the *Director* is able to maneuver freely within a virtual space (see Fig. 3.14). By tilting the left analog joystick on the gamepad, the *Director* is able to move the virtual camera's position in 3D

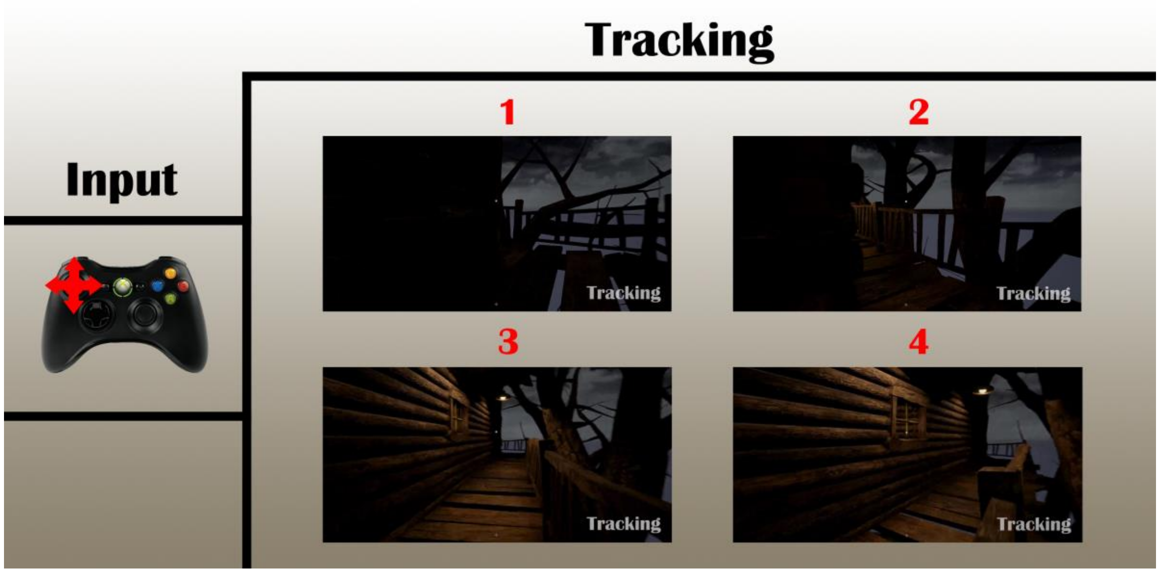


Figure 3.14 Camera Tracking

Cartesian space (i.e. tilting the joystick forward will move the virtual camera forward). The further the joystick is tilted, the faster the camera moves. To compensate for newcomers who may not possess the experience and dexterity of using a gamepad to dynamically center the camera's viewpoint during active performances, gradual movement interpolations are implemented. What this means is, the act of pressing forward on the joystick does not translate immediately to the virtual camera; the result is gradual acceleration when tilting the joystick in any direction, and gradual deceleration when the joystick abruptly defaults to its upright position. Without this mechanic, jittery camera movement are certain to ensue. This mechanic remedies the possibility of sharp, unnatural camera movements, more forgiving to new users.

In addition, tilting the right joystick, the *Director* is able to tilt (view Fig. 3.15) and pan (view Fig. 3.16) the virtual camera. As with the left joystick, the further the joystick is tilted in a particular direction, the more intense the action is. Again, because newcomers may not have enough experience using gamepad controllers for virtual cinematography, a target lock is implemented (view target lock-on section below) to reduce the chances of tearing- which can occur when inexperienced operators pan a virtual camera too fast.

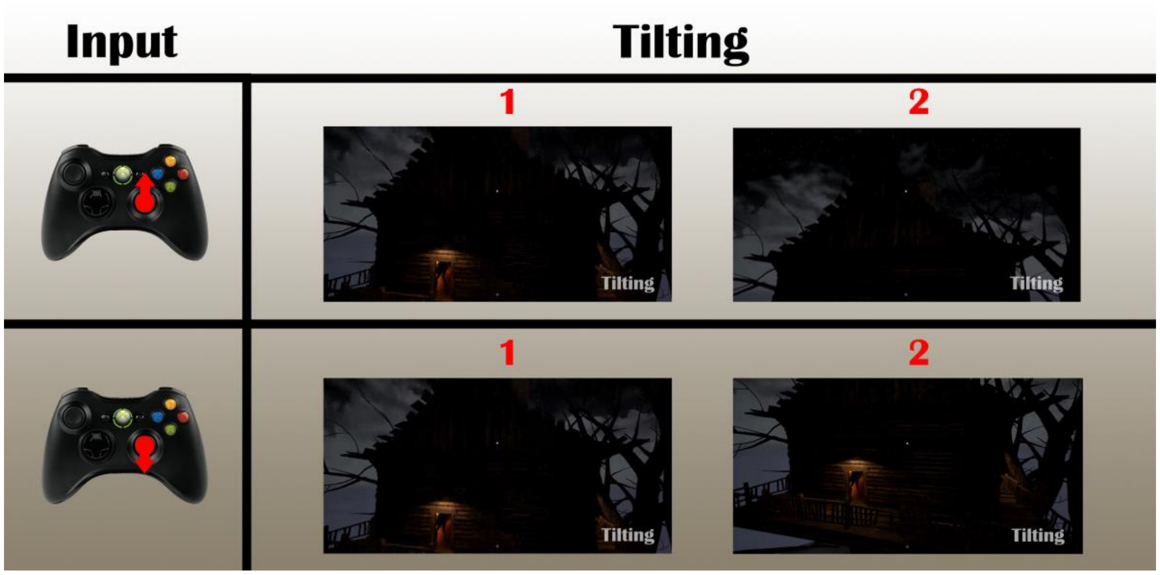


Figure 3.15 Camera Tilting

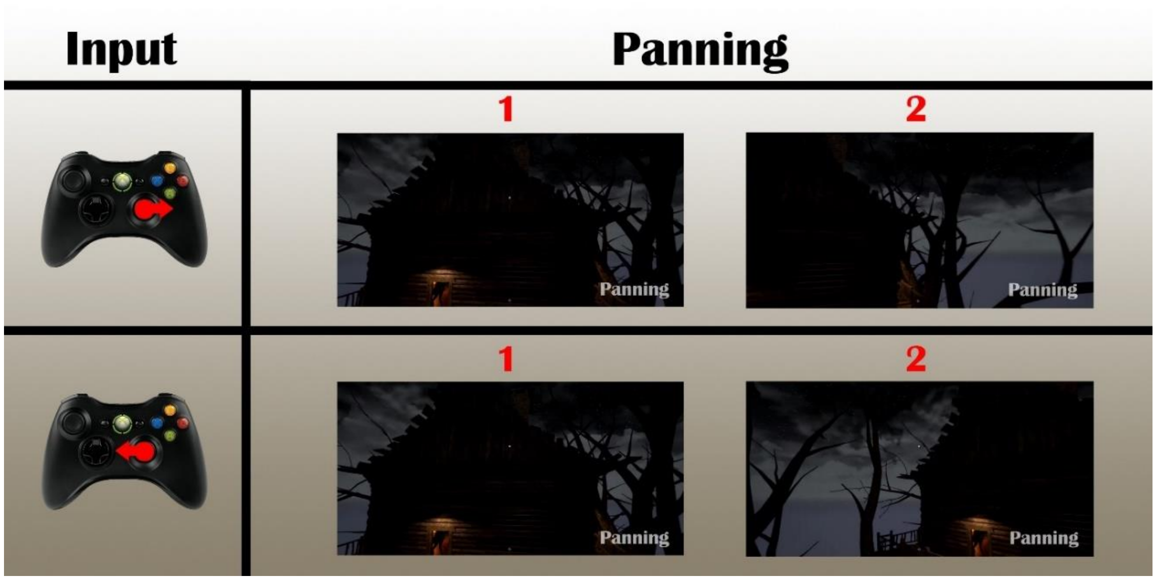


Figure 3.16 Camera Panning

In addition to moving, tilting, and panning the camera, a crane shot mechanic is implemented for navigational flexibility (see Fig. 3.17). Currently, with the use of the left joystick controller,

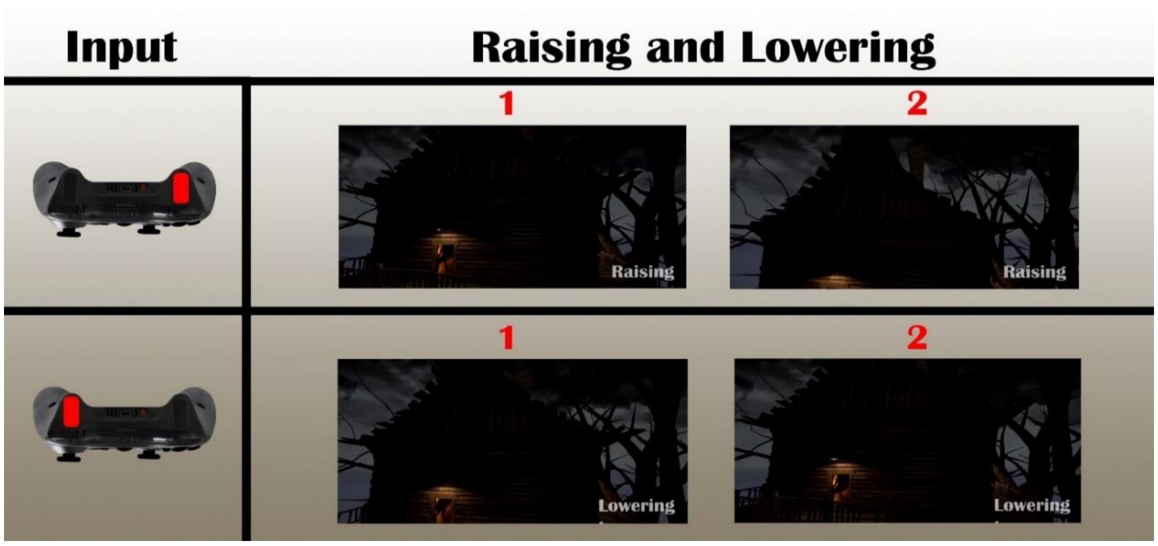


Figure 3.17 Camera Raising and Lowering

the *Director* is able only to move the camera forward, backward, left, and right relative to its forward vector. To raise the camera, the *Director* has to use the right joystick to orient the camera at an upward angle, then tilt the left joystick forward. Such manual control convolutions do not make for an intuitive system. To supplement that constraint, the left and right triggers on the gamepad are used to raise and lower the camera. As with the left and right joysticks, the further the triggers are pressed, the more aggressive the action is. Holding the right trigger on the gamepad raises the camera, while holding the left trigger lowers the camera. The left and right triggers can also be used simultaneously with all other navigation functions for more natural, intuitive control.

Target locking

Apart from navigating using the gamepad joysticks and triggers, the *Director* is able to target-lock on the *Performer* (see Fig. 3.18). In the event of dynamic scenes where the *Director* is required to actively use the gamepad joysticks and triggers to maintain focal view of the Performer, for experienced machinimators it may be easy. However, for newcomers it can be hard to adapt since they lack a familiar dexterity. To make the system a bit more accessible, the *Director* in our system can opt out of manually orienting the camera toward the performer by simply pressing the left bumper on the gamepad. Upon pressing the left bumper, the camera orientation (forward vector) gradually interpolates toward the performers position, maintaining a centered camera view on the performer. While locked-on, the performer is able to use other navigation functions to freely navigate. And because the orientation of the camera interpolates

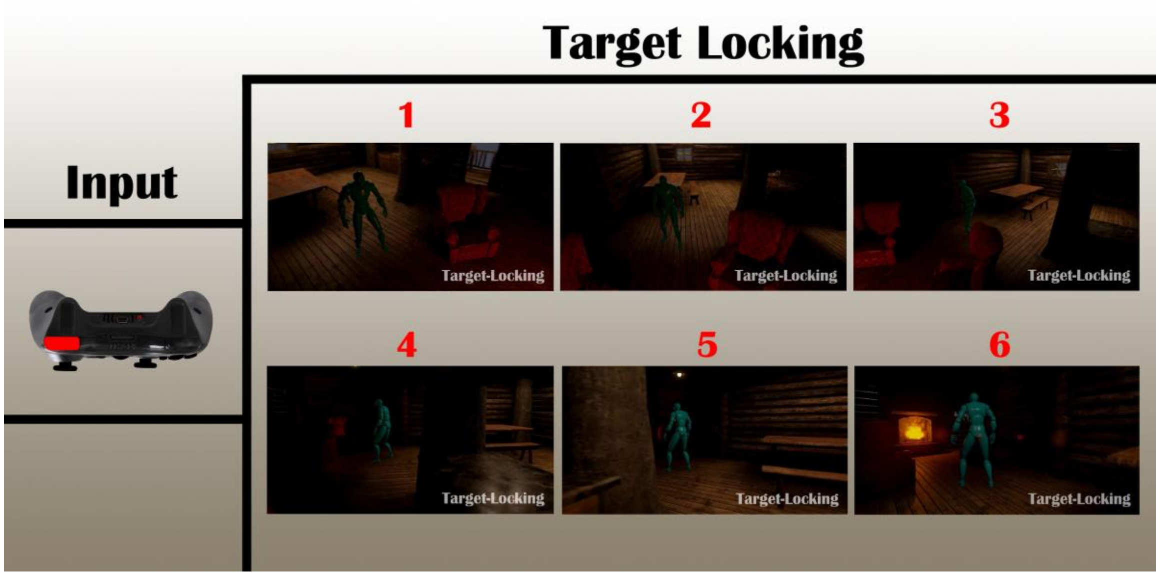


Figure 3.18 Target Locking

toward the position of the performer every frame, any sudden movements by the performer- such as jumping, abruptly changing directions, and so on- will translate smoothly. As with gradual acceleration and deceleration, this function remedies the issue involved with sharp, mechanical, unnatural camera motions.

Camera Spawn/Possess/Delete

Typically, in Machinima production, to achieve multiple takes of a scene, a single camera is used. The performance is recorded in either one or multiple takes, and then in post-production work the take is split into various cuts. In contrast, some Machinima workflows involve placing a virtual camera in the scene, running a build of the game to get a view of where the camera is facing, exiting the engine to make the necessary adjustments for an acceptable angle, key framing the camera movement to match the performance, and so on. This tedious process continues with as many cameras as needed in order to capture the entire performance. For even more sophisticated performances, multiple cameras are programmed to activate- or cut- after a triggered event, typically based on an avatar's location within the virtual space. That being said, the process to which camera cuts are established within a real-time engine can be tedious, thereby calling a need for an intuitive system.

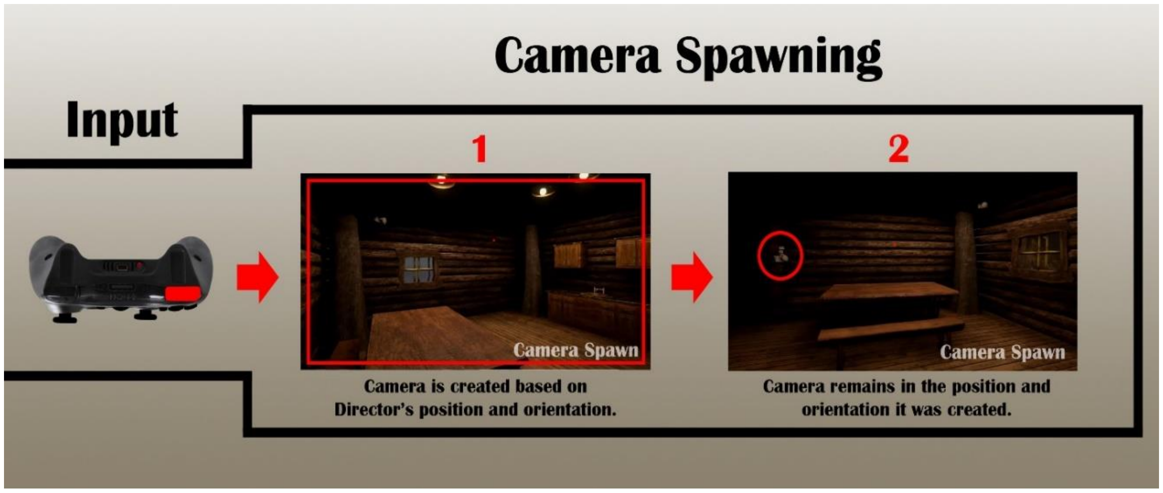

Camera Spawning
Input
1
2
Camera Spawn
Camera Spawn
Camera is created based on Director's position and orientation.
Camera remains in the position and orientation it was created.

Figure 3.19 Camera Spawning

Our unique system addresses this tedious workflow of having to repeatedly enter and exit the game engine to add cameras, see the result of minor actions such as rotating, positioning, and key framing, or program multiple camera functionalities. With our system, the *Director* is able to spawn up to five cameras in the environment by pressing the right bumper on the gamepad controller (see Fig. 3.19). By default, the camera in use is set to a value of one (maximum value of five). Each camera that is spawned is contingent on the current value, so a value of two is associated with camera two, and so on (see Fig. 3.20). To add values, the *Director* must press *up* on the directional pad. To subtract values, the *Director* must press *down* on the directional pad. This method of spawning cameras enables the *Director* to have independent control of each

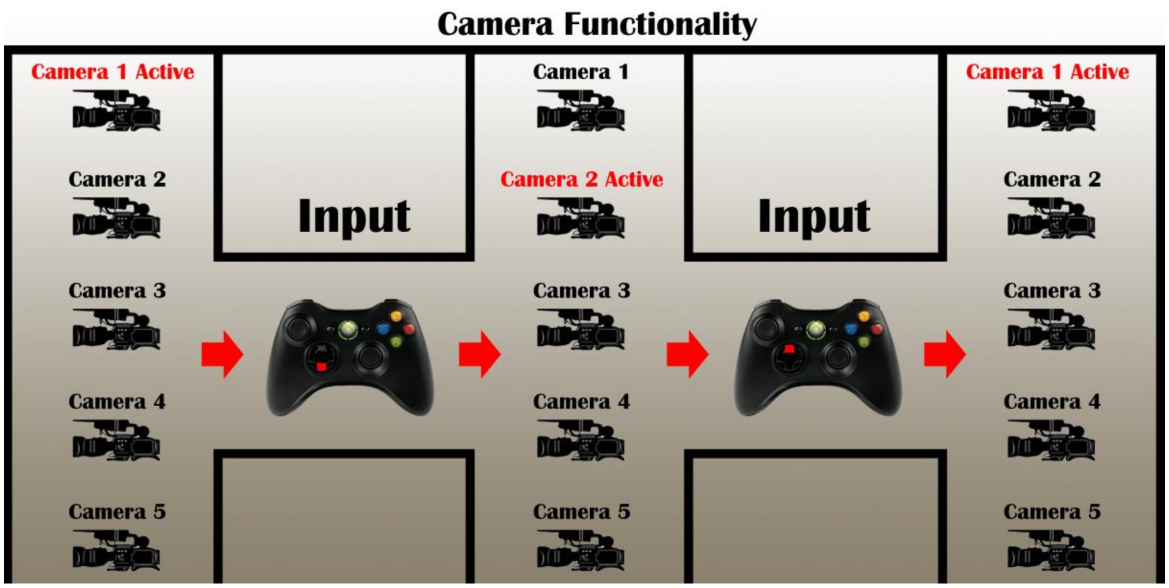


Figure 3.20 Camera Functionality

camera. When a camera is spawned, it is oriented and positioned precisely where the *Director* is located and looking when the right bumper is pressed. To reposition a camera, the *Director* can either delete and re-spawn the camera, or press the right bumper. This function remedies the need to exit the game engine to see what is in the camera's viewpoint, because what is in focal view is based on where the *Director* is positioned and facing before a new camera is created. This facilitates quick and painless set up of desired shots and angles.

In addition to spawning five cameras, a unique attribute is offered to ease the technical hurdle of cutting between shots (cameras). There are two ways: (a) Using eye tracking with a gamepad, and (b) pressing the back button on the gamepad (see Fig. 3.21). It is important to note that the

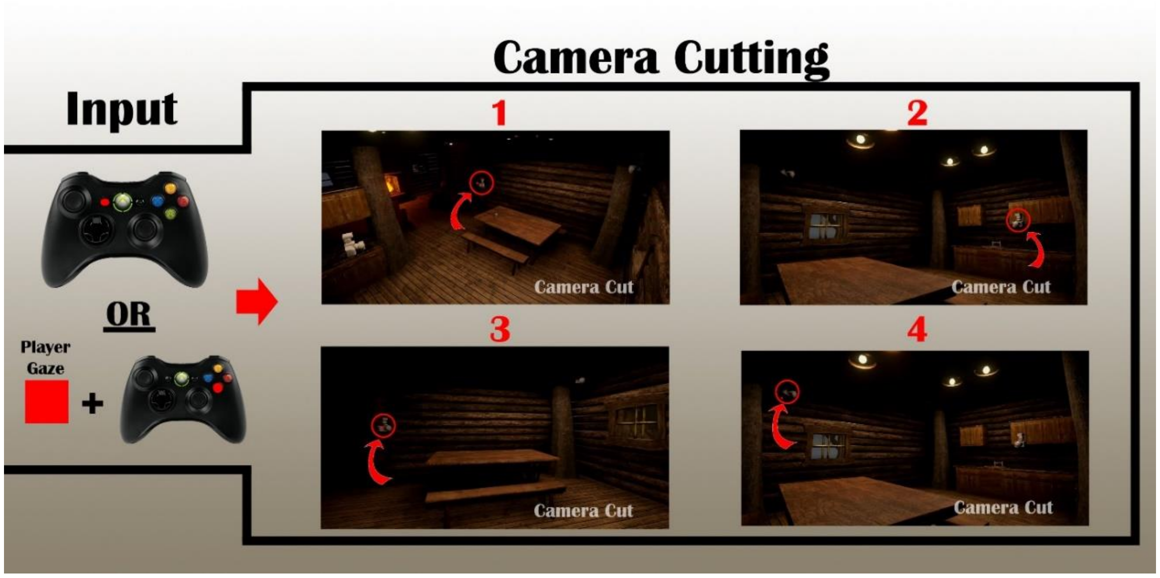


Figure 3.21 Camera Cutting

Director is able to take control of the any of the five cameras, so long as they exist (previously spawned) in-game. In relation to the first method, the EyeX Engine API is programmable in Unreal Engine 4, which allows us to track where the user is gazing on screen. As the user gazes in a particular direction, we can specify the types of interactions that follow. In our system, to straight cut between cameras, the *Director* must gaze toward the visible camera and press the bottom face button on the controller. Immediately the *Director* is positioned and oriented to the gazed upon camera view (similar to staging mechanic below); and has the ability to freely maneuver within the virtual space. At any point in time, the *Director* can re-take control of any cameras by performing the same function as outlined above. In relation to the second method, the *Director* must press the back button on the gamepad to cut to a camera. With this method, the *Director* only cuts to the camera associated with the current value (one to five). If the value is set to four when the back button is pressed, the *Director* is positioned to the location and orientation of the fourth camera- so long as it exists. This mechanic removes the need to exit the game engine to program straight cuts and the need to split long sequences using nonlinear editing programs such as After Effects. Uniquely, cutting between camera's in real-time offers a distinct attribute that includes basic post-production work during real-time production.

With the advent of spawning and cutting between cameras, our system offers yet another intuitive feature. As with spawning cameras, the *Director* is able to affect the deletion of a camera by gazing toward it and pressing the left face button (see Fig. 3.22). This functionality offers an iterative workflow, allowing the *Director* to position and reposition cameras as many times as needed. With enough experience, *Directors* can spawn, reposition, and delete multiple cameras in one sequence, offering a unique playground for real-time *performance* capture.

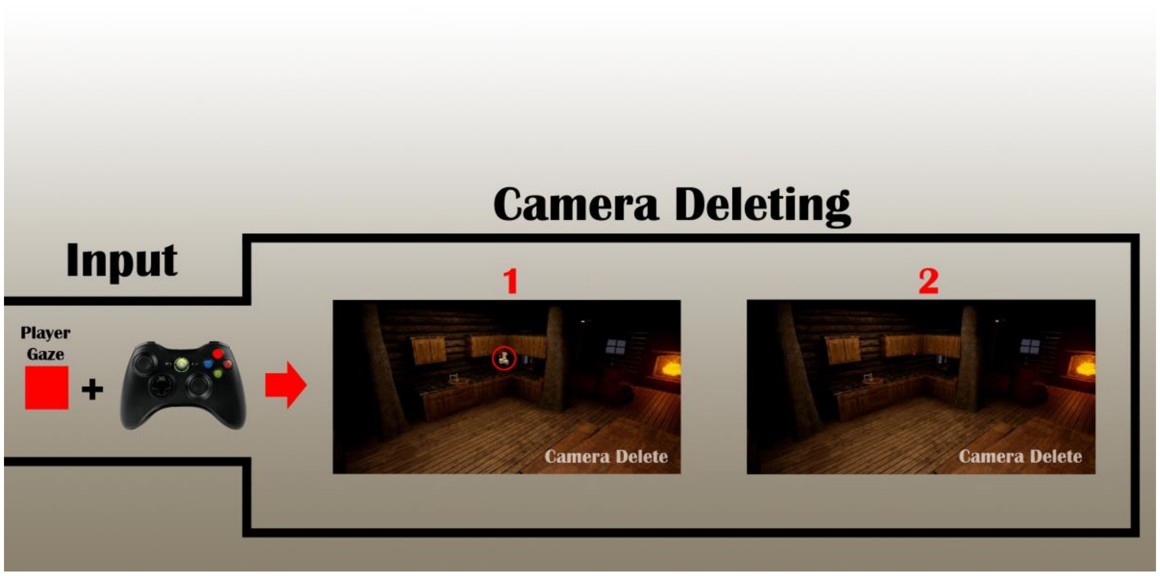


Figure 3.22 Camera Deleting

Collaborative Item Swapping

Machinima films are typically made in teams with people holding various roles, be it director, lighting artist, camera operator, and editor. Because collaboration is a crucial component for production in cinematography, our system, as proof of concept, provides a basic collaborative structure unique to Machinima production.

In our system, the *Director* is able to rapidly assist the *Performer* during a performance by swapping in equipment. Because our system is in its prototype stage, a flashlight is selected as the only piece of auxiliary equipment, used to exemplify a new form of collaborative interaction. A flashlight is selected because it contains built-in functionality for the *Performer* to augment

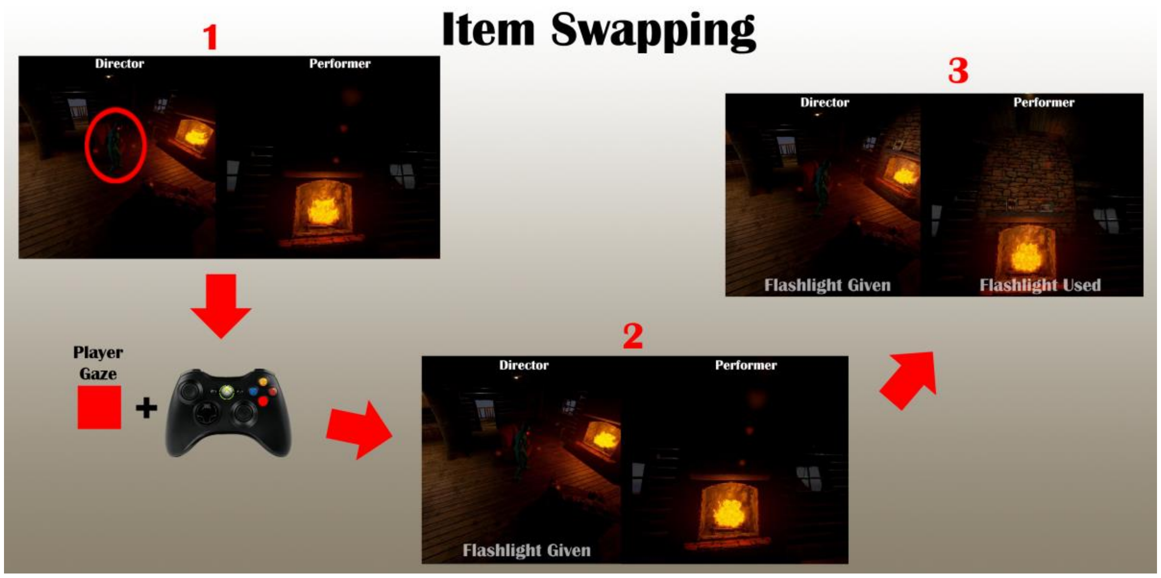


Figure 3.23 Item Swapping

movement throughout the game space (view *Item Possession* section below for additional details). To gift the flashlight, the *Director* must gaze in the direction of the *Performer* and press the bottom face button on the gamepad (see Fig. 3.23). Instantly, the *Performer* will have access to equipping and interacting with the flashlight.

The purpose for this *item swapping* interaction derives from the limited functionality of a gamepad. In our system, the *Performer* uses a gamepad to puppeteer the avatar in-game. If the Performer was also responsible for selecting and equipping items, problems would ensue. Reason being, there are only so many combinations to select and equip items using a gamepad before a complicated user interface is required. More to the point, sorting through the user interface while the performance is active will disrupt the flow, causing unwanted ripples that would require additional editing using non-linear editing programs. Thus, the *Director* is selected for the role, removing the cluttering UI component for the *Performer*. In turn, this mechanic forms a powerful interdependent relationship between *Director* and *Performer*, a unique collaborative attribute exclusive to real-time production.

Lighting Manipulation

Lighting is an important component to theatrical design, harnessing a power that influence how viewers see and respond to unfolding narratives. More so, they function to effectively create an emotional impact. As a result, our system enables the *Director* to, in real-time, adjust lighting attributes, including intensity and attenuation (see Fig. 3.24). To manipulate the light intensity, the *Director* must gaze in the direction of the light source by either holding *up* or *down* using the directional pad. Holding *up* gradually increase the light intensity, while holding *down* decreases the intensity. To control the light's attenuation, the *Director* must gaze toward the light source and hold either *left* or *right* on the directional pad. Holding *left* tapers the light attenuation, while holding *right* widens the light attenuation. In addition, while gazing in the direction of the light source, the *Director* is able to toggle the light on and off by pressing the left face button on the gamepad.

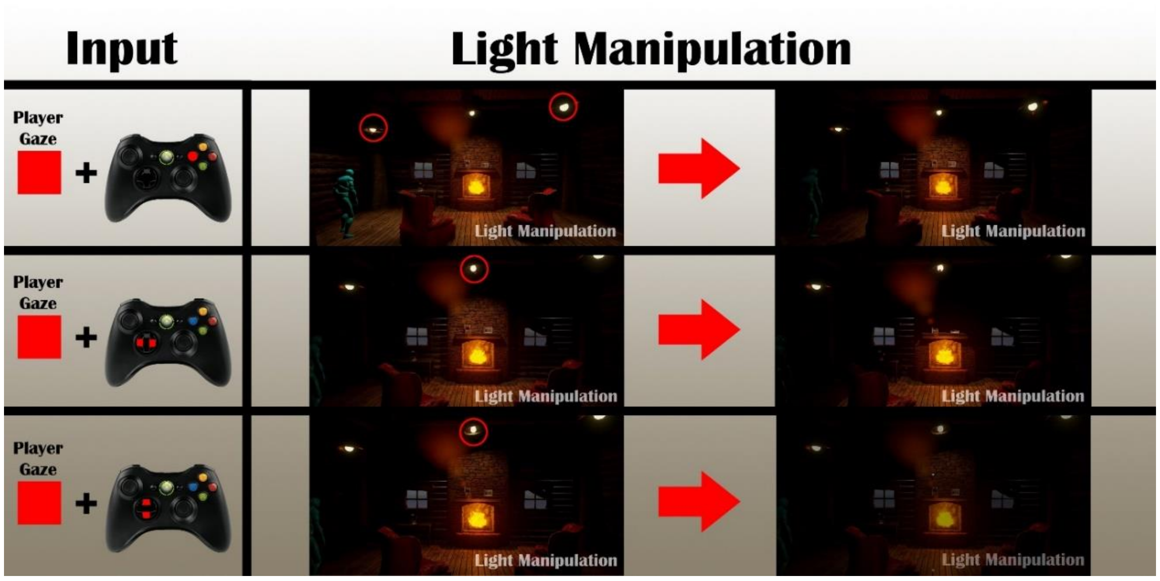


Figure 3.24 Light Manipulation

Staging

Staging is an important process during filmmaking that spans from designing, selecting modifying, adapting, and blocking out performance spaces, structures, and performers. In our system, a basic staging mechanic is implemented for the *Director* to quickly set the *default* stage for performances (see Fig. 3.25). Pressing the right face button on the gamepad spawns a staging object, providing a visual indicator of where the director will be positioned and oriented when

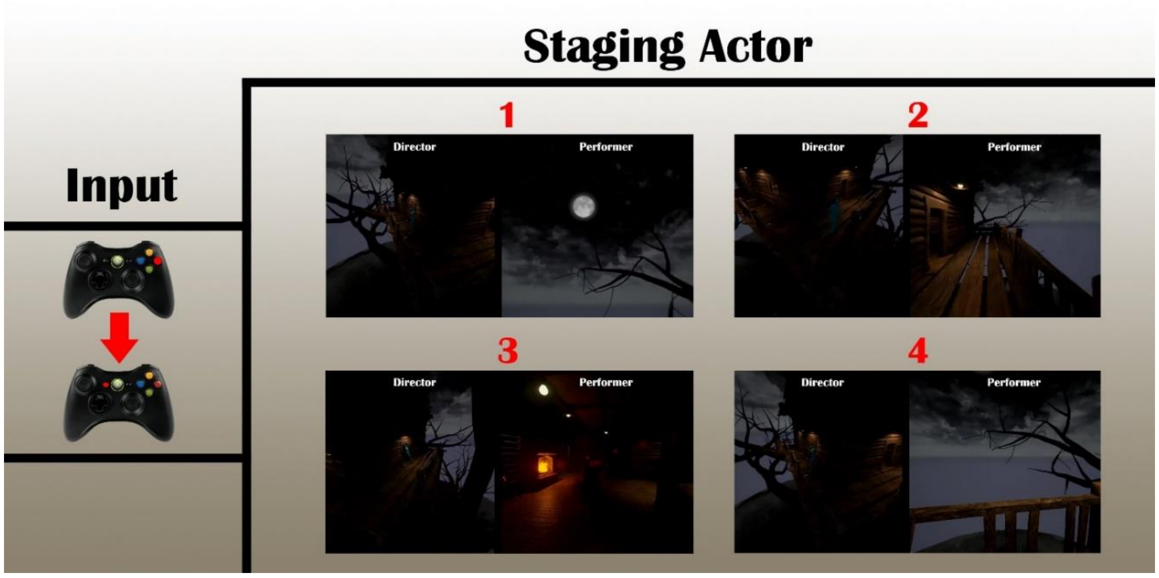


Figure 3.25 Staging Actor

activating the *default* stage; spawning the staging sphere stores the *Director's* position and orientation in 3D space. Upon the *Director* pressing the *back* button on the gamepad, the *default* stage mechanic activates, relocating the *Director* to the location of the staging sphere. This mechanic is exclusive to real-time production, providing machinimators a fast, consistent, and convenient method for capturing iterative rehearsals.

3.3.2 The Performer

The *Performer*, in contrast to the *Director*, control system contains limited functionality and is a temporary solution for the grander vision of real-time animation capture and editing. Thus, the following section briefly overviews the performer’s mechanics: navigation, item possession, and scene interaction.

Navigation

In Machinima production, a gamepad is used to puppeteer avatars in-game. Our system builds on this tradition, enabling the *Performer* to use a gamepad as a steering device to drive character movements. By tilting the left joystick, the *Performer* has the ability to move the virtual avatar (see Fig. 3.26). Using the right joystick, the *Performer* can rotate the first-person camera view

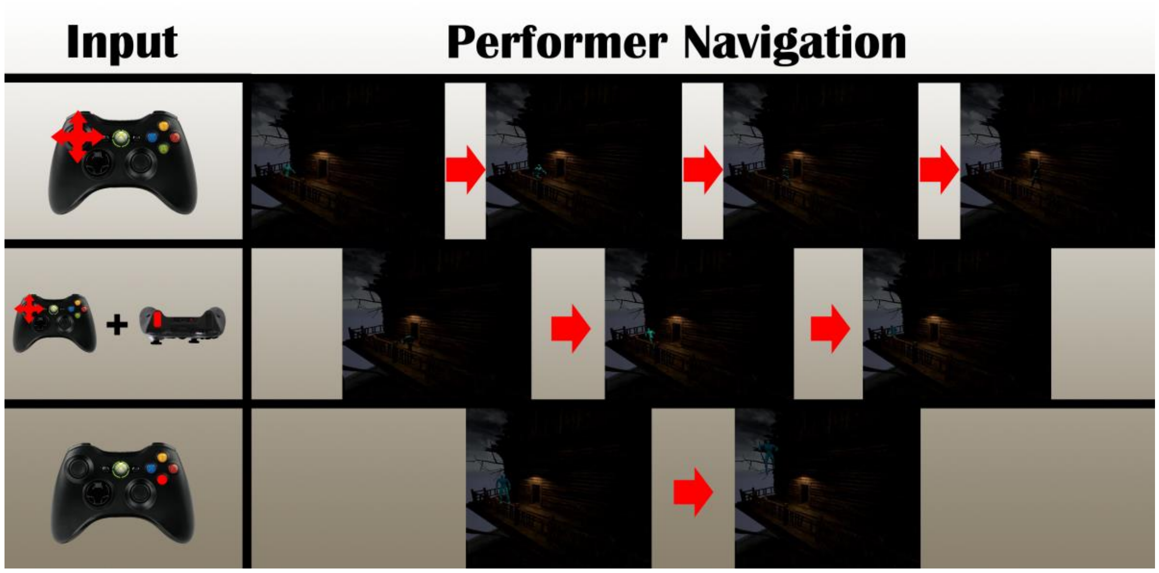


Figure 3.26 Performer Navigation

(see Fig. 3.26). Similar to the camera navigation functionality for the *Director*, the further a joystick is tilted, the more aggressive the action is. Rotation of the avatar can also be achieved by rotating the head, since the Oculus Rift DK2 translates any head movements to a virtual avatar.

In addition to moving and rotating the first-person view, users are able to use the gamepad to make the avatar jump and run. To cause the avatar to jump, the *Performer* must press the bottom face button on the gamepad (see Fig. 3.26). To cause the avatar run, the *Performer* must press and hold the left trigger. (see Fig. 3.26). These navigation functions are standard in Machinima production in order to effectively puppeteer a virtual avatar using a gamepad controller. In future work, the avatar movement will be driven in real-time by a synthespian actor using motion capture technology. In doing so, machinimators will have the ability to capture performances, in real-time, of virtual avatars effectively portraying bodily, facial, and emotional expression.

Item Possession

Aforementioned in the Director's *Collaborative Item Swapping Mechanic*, the *Director* is able to quickly assist the *Performer* during a performance by swapping in a flashlight (view Fig. 3.23). Upon access to the given flashlight, the *Performer* has the ability to toggle the flashlight on and off by pressing *up* on the directional pad. This mechanic, although limited to one item, serves to exemplify a collaborative possibility between *Director* and *Performer*. For additional information of its forthcoming application, navigate to *Future Work* section below.

Scene Interaction

A component of Machinima filmmaking is the ability for *Performers* to interact with their surrounding virtual environment. Our system, in its prototype state, contains a basic custom scene for the *Performer* to explore using a gamepad and head mounted display. Within the scene is a log cabin, trees, and basic furniture. Gameplay begins by spawning the Performer within the log cabin. At any point in time, the *Performer* is able to interact with a door, which opens upon

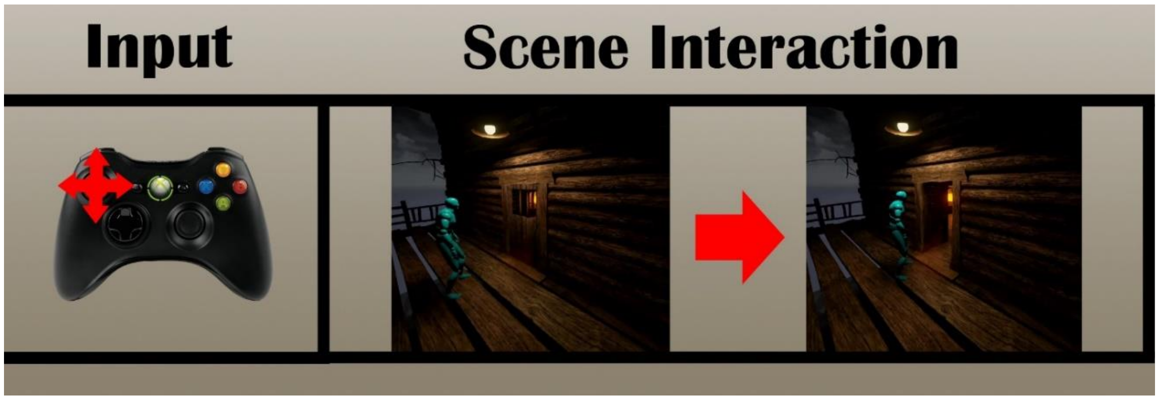


Figure 3.27 Performer Scene Interaction

collision (see Fig. 3.27). Upon opening the door, the *Performer* has access to the outside world, where the user can roam freely.

4. Methodology

The following investigates the exploration of producing an Unreal Engine 4 system plugin integrating eye tracking and virtual reality for collaborative Machinima filmmaking. Our system exploits new in-game techniques and possibilities for machinimation scene capture, emphasizing the use of eye tracking as a virtual cinematography tool to increase interactivity and intuitiveness. And because Machinima is a complex medium with diverse participatory cultures, our system offers a low-entrance barrier with a modular and flexible component, enabling machinimators of all skill levels to quickly and easily establish basic camera motions and collaborative interactions. Although there is no single-purpose solution for making Machinima more accessible to newcomers, our system – in its prototypical form- is a step in that direction. It serves as a precursor to the larger idea of using game engines as production environments *and* post production tools for real-time virtual production, choreographing, directing, and editing. In this chapter, we will discuss the user study design, system's architecture, explore its interaction design, dive into its practical applications, overview its limitations, and close with future development.

4.1 Approach

Playtesting is a process to gain insight as to whether a game, system, or application is achieving the desired vision for participants to experience. It is the designer who must advocate for participants and at all times keep them in mind during the process. Throughout the development of our system, 15 informal playtesting sessions were held, which were used to: (1) inform the design of the system, (2) address problems arising, (3) determine the system's usability, and (4) evaluate its ease of use. Revisions to the system were addressed throughout its development after every two rounds of play (each round includes two people); if the majority of focus groups disliked a feature, it would either be removed or improved based on iterative feedback.

Playtesters were recruited from Drexel University and NYU’s Game Center during Playtest Thursdays. Playtest Thursday is a weekly NYU event where developers gather to receive feedback for their prototype games (of various sorts) and technology, among other things. This event gave us the accessibility and flexibility to recruit people with various backgrounds, since the event was open to the public. For recruitment, flyers were made detailing playtesting dates, which were posted around campus such as the Recreational Center, Hagerty Library, Center City Library, Hahnemann Hospital, the Science Center, Stiles Hall dorm rooms, and NYU. The majority of playtests were hosted at *home* and at NYU’s Game Center.

Group demographics included undergraduate and graduate students ages 18 and up. Targeted individuals were split in three categories: (a) newcomers to virtual cinematography, (b) artists/programmers of varying skill levels, and (3) people with experience producing at least one Machinima film. Newcomers were considered participants with *little* to no experience using eye tracking, virtual reality, creating animated films, and playing games using a gamepad; *little*, meaning less than an hour or so of exposure per week to the aforementioned categorical conditions. Artists and programmers included participants that were technologically savvy, possessing knowledge of playing and creating games. Artists and programmers in this category ranged from beginners with less than a year of experience, to industry veterans with fifteen-plus years of game production involvement. Experienced machinimators were classified as individuals with at least one publically distributed Machinima film - whether distributed via YouTube, Vimeo, or Netflix, among others; the tools and techniques employed in Machinima production, so long as they fit within the confines of real-time filmmaking, did not influence the validity of machinimator’s *experience*. Valid forms of techniques to produce Machinima included, but are not limited to: (1) Using games for production and post-production, (2) using gamepads to puppeteer avatars, (2) using games as a production environment and non-linear editing programs to edit and distribute content, and (3) using dedicated Machinima software such as *Moviesandbox*. With these three focus groups, we were able to evaluate our system in varying perspectives, including how it is used by veterans as a virtual cinematography tool, the accessibility and ease of use, and the usability for collaborative performance capture.

For each playtest session, two players were needed (one person as the *Performer* and one person as the *Director*). Players selected for collaborative playtesting were required to have similar, if not identical, background and experience producing Machinima, playing games, and creating games. In other words, newcomers playtested with newcomers, machinimators with machinimators, and artists/programmers with artists/programmers. This method of playtesting was very important, because it ensured the feedback received was exclusive to each group demographic. In so doing, we were able to observe, document, and receive feedback of how our system is used from people with different backgrounds, which have varying expectations of what constitutes virtual cinematography. Because our system is an alternative filmmaking practice that strays from traditional cinematography practices, it was essential to evaluate our system from different perspectives. Newcomers with marginal knowledge of playing games and creating Machinima were selected because they lacked knowledge of *tried-and-true* traditional cinema language and methods. This enabled them to provide fresh feedback of its usability from a consumer's perception, not a conditioned, biased cinematographer. Artists and programmers were selected to provide a more technical standpoint of how our system functionality can be improved and expanded upon. Machinimators were selected with the goal to gain insight of filmmaking techniques they have discovered while using games as filmmaking platforms. Further, they we were selected in order for us to observe and evaluate their strategies to exploit our alternative filmmaking tool, and how it fits in with their conditioned methods of producing Machinima.

The focus groups, upon arriving, were prefaced with the following: "Thank you for coming, today you will use a real-time virtual cinematography tool by which a performer and director collaborate within a shared virtual space to produce Machinima. Please use the existing tool, in any way you like, to produce Machinima. If you have any questions regarding functionality, ask away. There will be a discussion of the experience following the play session."

Upon articulating the introduction, playtesters were asked if they have any questions before the play session begins. If so, questions were answered, then the play session begins. All playtest sessions were timed using a stopwatch, lasting fifteen minutes; each participants, after eight to ten minutes of playtesting, were asked to swap roles. During the play session, participants were asked to think out loud, so uncertainties presented with their choices are heard, naturally. For example, hearing participants say "I think we can destroy that, maybe not," helped identify components of the system needing improvement, an important element that was brought up during the discussion of the game experience. Throughout the play session, data from participants was collected, including their collaborative tendencies, interaction, adaptability, leadership, strategy, performance, and reaction. Intervention during the play session was avoided at all costs, unless the participants were stuck during gameplay or have been quiet for an extended period of time.

When the play session phase is complete the discussion of the game experience commences, lasting approximately ten minutes. Participants were informally interviewed together to discuss and evaluate the system's collaborative attributes, usability, and intuitiveness. When the discussion of the play experience is completed, the playtesting session ends.

5. Results

As outlined in *Approach*, a series of informal playtests were hosted to get a general consensus of where our system stands as an alternative filmmaking tool. Targeted groups included newcomers to virtual cinematography, artists/programmers of varying skill levels, and people with experience producing Machinima. With these three focus groups, we were able to evaluate our system from varying perspectives, which was used to: (a) inform the design of the system (b) and evaluate the system's collaborative attributes, usability, and intuitiveness.

The following section details, *individually*, each group's evaluation of the system's collaborative attributes, usability, and intuitiveness. We then close with external observations and analysis of the results.

5.1 Group Evaluations

5.1.1 Newcomers

Collaboration

	Newcomers
Performed as Expected	7
Did Not Perform as Expected	2
% Performed as Expected	78%
% Did Not Perform as Expected	22%

Figure 5.1 Newcomers' Collaboration Results

Nine of thirty participants were newcomers to Machinima. 78% of them believed our system adequately embraced collaborative interactions, stating "If users used the tool correctly, they would quickly realize collaboration was essential in order to capture the desired take." Although, concern was expressed regarding the prototype's limitation of *Performer-to-Director* interactions. Meanwhile, 22% disagreed of the system's usability as a collaborative platform. According to their evaluation, "In the prototypes current state, the *Performer* lacks the mechanics to enable collaborative play."

Usability

	Newcomers
Performed as Expected	3
Did Not Perform as Expected	6
% Performed as Expected	33%
% Did Not Perform as Expected	67%

Figure 5.2 Newcomers' Usability Results

For the usability of our system, 33% used it as a virtual cinematography tool. With these individuals, they were able to instantly grasp the system and use the *Director* and *Performer* functions to rehearse and mock-capture performances. However, 67% did not use the system as intended. These individuals simply explored the environment admiring the visuals, using the play-space as a game-space.

Intuitiveness

	Newcomers
Performed as Expected	4
Did Not Perform as Expected	5
% Performed as Expected	44%
% Did Not Perform as Expected	56%

Figure 5.3 Newcomers' Intuitiveness Results

For the system's intuitiveness, 44% agreed to its ease of use. Evaluations suggest eye tracking was an easy method to interact with the interface in-game. Further, the target lock-on served convenient for non-gamers to keep the performer in camera focus. In contrast, 56% believed the system was not intuitive. Some expressed their frustration using a gamepad to maneuver a camera, while others experienced difficulty learning and performing the mechanics. Common feedback attributed to the system's lack of intuitiveness derived from the absence of a tutorial, legends screen, and frequent eye tracking recalibration.

5.1.2 Artists and Programmers

Collaboration

	Artists and Programmers
Performed as Expected	6
Did Not Perform as Expected	4
% Performed as Expected	60%
% Did Not Perform as Expected	40%

Figure 5.4 Artists' and Programmers' Collaboration Results

Ten of thirty participants were made up of artists and programmers exhibiting knowledge in game production pipelines. 60% of artists and programmers expressed positive attitudes toward the system's collaborative attributes. This group was particularly keen on working together to stage scenes. Feedback received suggests collaboration is naturally tethered to the system's design, even though the *Performer* can't see the *Director*. In contrast, 40% of participants neglected to engage in collaborative play, stating the Performer is limited to opening doors, walking, running, and jumping. They further added, the Performer's pre-made animations are a significant constraint. They believe *no one* will express empathy for a character if the only emotional expression exhibited is robotically uncanny.

Usability

	Artists and Programmers
Performed as Expected	4
Did Not Perform as Expected	6
% Performed as Expected	40%
% Did Not Perform as Expected	60%

Figure 5.5 Artists' and Programmers' Usability Results

In regard to the system's usability, 40% approved of its use as a virtual cinematography tool. Evaluations suggest the tool is unique, possessing great potential that can get the Machinima community involved. Further, they mentioned the system, in its current state, is a convenient jumping off point for blocking out shots and basic previsualization work. Dissimilarity, 60% of participants rejected the system's use as a virtual cinematography tool. Evaluations received suggest the *Performer* was not usable due to the lack of *meaningful* functionality. Similar to newcomers, a common trend was using the system as a game-space, not a real-time filmmaking tool.

Intuitiveness

	Artists and Programmers
Performed as Expected	6
Did Not Perform as Expected	4
% Performed as Expected	60%
% Did Not Perform as Expected	40%

Figure 5.6 Artists' and Programmers' Intuitiveness Results

As for the system's intuitiveness, 60% of participants were able to quickly learn and adapt to the control systems. Participants expressed interest in using our tool in their own project, stating our system is an ideal solution for their shooting scenarios. According to other evaluations, participants expressed satisfaction with the eye tracking control setup; that is, to interact with an object, the user must look toward it, then press a button on the gamepad. Participants also suggest our tool is easy to use, offering a natural, unobtrusive, and intuitive method for interacting within a virtual space. In opposition, 40% of participants experienced difficulty getting used to the control systems. Common reasons included the absence of a tutorial screen, having to repeatedly recalibrate the eye tracker, and the time consuming efforts that would be needed to add custom characters. Other feedback received touched base with eye tracking's *sparing* use. In other words, participants did not seem in favor of supplementing eye tracking with a gamepad, they would rather see eye tracking being used *more* exclusively.

5.1.3 Machinimators

Collaboration

	Machinimators
Performed as Expected	9
Did Not Perform as Expected	2
% Performed as Expected	82%
% Did Not Perform as Expected	18%

Figure 5.7 Machinimators' Collaboration Results

Eleven out of thirty participants were experienced machinimators from New York City (NYC). 82% attributed to collaboration coming natural to them while using our system. Feedback received suggests the *Performer* does not need to interact with the *Director* in-game to be a collaborative experience. They continued, advocating the relationship between *Director* and *Performer* should remain transparent in-game. Because the *Director* role in our system serves as the "director" *and* camera operator, the *Performer* should be primarily focused on *how* and *where* to perform (in other cases, the addition of *who* to perform with); *Performer-to-Director* interactions should not translate in-game, it should be established vocally during rehearsal. Contrarily, 18% of experienced machinimators were not convinced of the system's collaborative capabilities. Collectively, this group conveyed their distaste for the *Performer's* limited interactions. Common feedback was the need for the *Performer* to see the *Director*. In doing so, the *Director* would be able to *quickly* guide the *Performer*, establishing clear parameters as to where the Performer should be positioned during each take, path of travel, and where triggered events will occur.

Usability

	Machinimators
Performed as Expected	9
Did Not Perform as Expected	2
% Performed as Expected	82%
% Did Not Perform as Expected	18%

Figure 5.8 Machinimators' Usability Results

On behalf of the system's usability, 82% were optimistic of its applicability. Each participant used the system as intended, staging and directing performances in real-time. Positive feedback was attributed to spawning and cutting between cameras, they thought "it was a clever way to include post production work during production." In opposition, 18% experienced difficulty using the system as a virtual cinematography tool. More specifically, feedback received dealt with concerns as to how machinimators would edit performances (using nonlinear editing programs) which already have camera cuts integrated. The ability to record a full-fledged sequence offers the flexibility for machinimators to split, extend, and/or shorten the *duration* of sequences (or *clips*). If performances are captured with camera cuts already integrated, there is less flexibility to manipulate sequences using non-linear editing programs.

Intuitiveness

	Machinimators
Performed as Expected	8
Did Not Perform as Expected	3
% Performed as Expected	73%
% Did Not Perform as Expected	27%

Figure 5.9 Machinimators' Intuitiveness Results

Relating to the system's intuitiveness, 73% of participants handled the system with ease. According to feedback, "The target lock-on felt natural and appealing. The fact that it smoothly trails behind the target was a convenient method to maintain the performer in camera focus." Other constructive comments included eye tracking's fast and intuitive response to interacting with the interface, as well as the convenience of operating lights in real-time. In contrast, 27% of experienced machinimators struggled to use the system. Their evaluation suggests the absence of a tutorial and/or HUD screen makes it difficult to use the system. According to this group, because the gamepad's functionality changes based on a user's gaze, it can be difficult to anticipate the interaction that ensues.

5.2 External Observations and Analysis

Collaboration

	Performed as Expected	**Did Not Perform as Expected**
Collaboration	22	8
Total Percentage	73%	27%

Figure 5.10 All Groups' Collaboration Results

To this point, we were able to receive feedback of the system's collaborative attributes, usability, and intuitiveness. Upon evaluating feedback and responses from all group participants, an average of 73% agreed collaboration was embraced and effectively used in our system. 27% did not align with the system's usability as a collaborative platform. Interestingly, we were able to see significant commonalities apropos of newcomers and advanced machinimators tackling collaborative play. All three groups were quick to work together and mock-capture performances. A significant pattern emerged: the participant taking the role of *Director* had a tendency to authoritatively direct performances, frequently telling the *Performer* what to do, where to go, and how fast to perform an action. Interestingly, rather than *Director* and *Performer* cooperating in unison, the collaboration seemed to be one-sided, with the *Director* holding the alpha role. Because of this, 27% of participants believed collaboration was limited, if existent. To them, verbally communicating was not a sufficient form of collaboration.

Usability

	Performed as Expected	**Did Not Perform as Expected**
Usability	16	14
Total Percentage	53%	47%

Figure 5.11 All Groups' Usability Results

As for the usability, an average of 53% across all groups used the system as a virtual cinematography tool, while 47% did not. A common trend was apparent between newcomers, artists/programmers, and advanced machinimators. The less experience with game production pipelines and producing Machinima, the less likely participants were to use the system as a filmmaking tool. During playtests, newcomers showcased confusion as to the purpose of camera spawning, swapping, staging, and target lock-on. Whether their lack of experience in producing animated films attributes to the confusion of what *controls* are needed is unclear. Nevertheless, newcomers often struggled with the system's usability as a virtual cinematography tool. Experienced machinimators, on the other hand, were apt in using all the system's functionality, quickly making use of camera swapping, and target lock-on to cater their needs during mock-rehearsals needs.

Intuitiveness

	Performed as Expected	**Did Not Perform as Expected**
Intuitiveness	18	12
Total Percentage	60%	40%

Figure 5.12 All Groups' Intuitiveness Results

Toward the system's intuitiveness, 60% of all participants approved of its ease of use, while 40% did not. During playtests, newcomers struggled to gain traction on how to use the system, often *giving up* and exploring the virtual space before experiencing the system's full-fledge functionalities. More significant was the difficulty newcomers experienced using a gamepad to puppeteer characters; most newcomers testing our system did not play games extensively, and so looking down on the gamepad to press the appropriate button was a recurring action. Further, newcomers exhibited *quirky* motions while using a joystick to navigate, frequently stating it was difficult for them to pilot the joystick using their thumbs. On the other hand, similar to newcomers, machinimators experienced difficulty in understanding and using the control systems at first. The absence of a tutorial and help screen was frequently regarded as the perpetrator. Nonetheless, once familiar with the control systems, machinimators were able to quickly stage scenes and use camera swapping techniques during mock-performances.

6. Conclusion

6.1 Application and Contribution

Our system is a virtual cinematography tool by which a *Performer* and *Director* collaborate within a shared virtual space. By leveraging consumer affordable technology, we aim to create an intuitive, flexible, modular, and low-entry barrier system for filmmakers, regardless of experience. Our tool is democratized in such a way that people, with their own personal computers, can begin creating their very own cinematic films entirely within a virtual environment. Further, our system strives to use a real-time engine as a production environment *and* post production tool, a characteristic of Machinima's origin that distinguishes itself from traditional cinematography practices. In doing so, we present fertile ground for programming possibilities whereby distinct technologies, synthespians, and artificial intelligence can mutually cooperate within a shared virtual space.

In the next section, we explore our system's potential application as a plugin, including basic staging, rehearsing, and performance capture. We then articulate how our system contributes to enriching Machinima, which differentiates itself from existing cinematography solutions. Finally, we close with how our system's real-time collaborative structure can be applied to various fields- outside the realm of games- in order to achieve greater efficacy, flexibility, and even safety.

6.1.1 Real-time Virtual Staging, Rehearsing, and Performance Capture

Our system is an easy to use virtual filmmaking alternative that enables users to quickly stage, choreograph, rehearse, and capture performances, in real-time. The following articulates the *intended* process by which amateurs and experienced cinematographers can go about using our system to compose their desired scenes.

Users begin by loading either Unreal Engine 4 editor or running a pre-baked build of the game. Upon calibrating the eye tracker, connecting the gamepads, and connecting the Oculus Rift

DK2, users are able to adjust the scene's lighting conditions, add and manipulate cameras, stage the scene, and choreograph performances, iteratively. If the tool is loaded within Unreal Engine 4 editor, users are able to manipulate the scene, add assets, customize character functionalities, and so on to cater the needs of the shot.

Once satisfied with the scene layout, the *Director* and *Performer* can utilize their gamepads to freely navigate within the scene. The *Director* at this time can place, delete, and move cameras to achieve desired camera angles for each take; this permits the *Director* to compose and frame shots real-time. When the camera is framed to the *Director's* satisfaction, a staging actor is spawned. The staging actor is used as a quick way for the *Director* and *Performer* to revert back to their starting positions for iterative performance capture. Upon placing a retrievable camera and staging actor, the *Director* is able to perform multiple dynamic actions/cuts in one sequence.

Using the gamepad, the *Director* has access to various dynamic camera functions to aid during performance capture, including tracking, panning, tilting, raising, lowering, and target locking. After performances are fluently enacted, cinematographers can record their performances as many times as needed using their preferred screen capture software. In the prototype's current state, performance capture is not addressed; although, future development will address recording performances in *demo* format, an unusually opportunistic functionality genetic to real-time game spaces.

6.1.2 To Move Forward, We Must First Think Back

T*o move forward* in enriching Machinima as a distinct medium, *we must first think back* to its origins. Aforementioned in the *Background* section, Machinima's origins can be traced to the Video Capture Period of Development, wherein gaming performances were captured in demo format. The tools used for creating, editing, storing, and displaying narrative performances resided exclusively within the game engine. This meant machinimators used a game engine as a production environment *and* post production tool. However, with the release of *Tritin's Quad*

God, new post-production techniques emerged, which introduced ramifications that shifted the significance of Machinima. Machinima then no longer became exclusive to machinimators using the game engine as a production environment and post-production tool. Instead, the arrival of new post-production techniques- using non-linear editing programs and screen capture software- *encouraged* machinimators to experiment with new methods of producing Machinima. In so doing, the game engine was repurposed by machinimators as *only* a production environment, thereby omitting all of the game engines capabilities as a post-production tool. While this shift expanded participatory practices and offered machinimators a convenient and flexible way to edit, arrange, and distribute films in various digital format, it was no more than 2D post-production techniques borrowed from cinematic practices. The belief that this shift was the first evolutionary step toward Machinima revolutionizing new systems of filmic production was nothing more than Machinima mimicking the *film language,* particularly in post-production. And therein was the problem. Machinima was birthed within a 3D compositing space, so why is its three-dimensionality being degraded to 2D just to compensate for cinematography's need for non-linear editing, which is a 2D compositing space? If you recall, according to Manovich:

> "Before the Velvet Revolution, traditional lens-based recording offered little to no manipulability over the dimensions and fixed visual content per frame. The limitations of temporal space quickly became supplemented with composition based production, the spatial dimension. The reduction of software and hardware costs and convenient integration of compositing led way to participatory practices with independent artists, educators, and companies. Further, the flexibility of the After Effects interface presented opportunities for designers to adjust canvas dimensions, import distinct media, independently arrange and access elements, apply filters, and manage transparency. The rendered output was no longer dealing with time-based linear storytelling, but of the intentional arrangement of visual elements in space. As convenient and original as the 2D compositing period of development was, during the year 2000 3D compositing forms of

> production came to fruition [16]. Skills, tools, and techniques originating from 2D compositing quickly translated into 3D Cartesian coordinate space."

If the skills, tools, and techniques originating from 2D compositing have truly translated into 3D compositing, why has real-time production within a 3D compositing space been limited to previsualization work for films and performance capture for Machinima? More to the point, what is it about 3D compositing spaces that fails, or lacks thereof, to uphold a firm foundation for its use as a production environment *and* post-production tool? The answer to these questions is not readily clear, but it may be safe to assume developers are beginning to exploit real-time possibilities that exist in the realm of 3D compositing. For instance, the Unreal team recently announced the release of Sequencer, an in-game cinematic tool developed by Ninja Theory.

Sequencer is a real-time collaborative production and editing solution that is based on traditional shot-based workflows, but in 3D space. Unlike 2D compositing shot-based workflows, Sequencer:

> "Let's users scrub through *three-dimensional* scenes as if they were shots in a movie editing program like Final Cut Pro. Even as Sulua's character stood frozen in place, dynamic animated effects like wind and fire continued as the editor played with camera angles. Sequencer makes it trivial to copy shots and add new elements, swapping them in and out to see how the different "takes" look in quick iteration. The shot-based workflow should also allow multiple people to collaborate on scenes easily, Epic said."

Sequencer enables the director to have iterative control over scenes, including lighting, cameras, and VFX. Further, users can select from a variety of physically based cameras, use an object picker for animating depth of field, key frame cameras, edit and arrange shots in real-time, customize camera attributes such as adding camera shakes after the camera is animated, record and overlay various demo performances, add VFX to any *scene* at any moment in time (which allow for creative decisions to be made on the fly), and much more. And because production is driven in real-time, there is no need for green screens. Rather, the performance is tuned based on the scene they inhabit, which adds an additional layer of realism; performers are now able to act upon on where they are and what they see, not where they are perceived to be and conditioned to see. The significance of the tool spans beyond the end goal of previsualization, and although its application is vaguely unclear, it provides a platform by which anyone can create sophisticated productions entirely within a virtual space, using virtual humans.

Sequencer has made a significant leap toward exploiting real-time production techniques and possibilities, but just because the tool has incorporated many of 2D compositing's shot-based film techniques into a real-time 3D space, it does not mean the 3D compositing space has truly evolved. As with Machinima *seemingly* evolving because it borrowed techniques from the film language, the real-time 3D compositing space is following a similar path. What the 3D compositing space requires are new design solutions The process by which camera motions are created and custom avatars are animated in real-time are still too complex, requiring a steep learning curve to effectively make use of Sequencer. This is where our system fits in, a unique, easy to use alternative by which people can collaborate to easily and affordably direct, stage, and perform within a shared virtual space.

Our system, in its current prototype, is an alternative filmmaking tool for the production of real-time filmmaking. Unlike traditional cinematic practices whereby sequences are taken into 2D compositing programs for final editing, our system reserves the use of a game engine as a production environment and post-production tool, similar to Sequencer. And unlike Sequencer, our system is designed for anyone, regardless of cinematic production experienced, to quickly, easily, and intuitively stage scenes, direct performances, and perform rehearsals, in real-time. Upon implementing our system, an unexpected turn of events occurred. To illustrate, the *Director* in our system initially served to direct performances, but what does that mean? One thing is to understand the director role, another is to translate its significance within a virtual space. The question became, "How and what mechanics would be needed for the director to have complete control over the virtual space, including directing performances, setting up camera shots, staging scenes, producing camera motions, manipulating lighting conditions, collaboratively interacting with the performer, and in sum have significant impact on the overall cinematic delivery?" The answer to that question is exemplified in our system, where the *Director* is required to take on multiple roles in order to satisfy the aforementioned duties. These roles include: the choreographer, camera operator, and game master. As a choreographer, the *Director* was able to collaboratively direct and rehearse performances, in real-time. As the camera operator, the *Director* was able to select and establish appropriate camera angles for each given shot, actively *performing* to capture performances, using various camera navigation techniques for smooth, natural movement/transitions. As game master, a unique role exclusive to games, the *Director* is able to create, control, and enforce anything that exists within the virtual space, in real-time- from light manipulation to performance capture. The unique functionality of the *Director* serving three completely different roles showcases how the role of director can quickly evolve and/or translate when a real-time play space is involved. The ability to composite, frame shots, establish camera cuts, perform crane shots, arc shots, and use real-time technology to drive believable animations just scratches the surface for what is possible when real-time engines are used in production and post-production. And although real-time camera cutting may not seem like an efficient way to edit sequences in shot-based workflows, it does not signify it's incorrect; rather, it is an exploratory alternative to production showcasing real-time collaborative techniques and possibilities, which is not bound by 2D compositing practices.

6.1.3 Real-time Efficiency, Flexibility, and Safety

Our system's contribution is primarily focused on real-time virtual filmmaking practices, but is not limited to cinematography. The real-time collaborative interactivity our system builds from is applicable to other distinct fields and personnel, including the military, athletes, construction, the handicapped, entertainment businesses.

To illustrate, for military personnel, our system can help tailor breach training and bomb disarming exercises, whereby soldiers learn to collaborate, communicate, and make decisions in a variety of circumstances, places, and obstacles; and because these interactions would take place within a virtual space, it provides a safe working environment to help condition military personnel to respond appropriately in the case of life-threatening situations. In addition, performances within a virtual space can help improve an athlete's performance. For example, in the case where an athlete is honing his technique for a 100-meter hurdle race, if his performance is captured while wearing motion capture technology, coaches will have the ability to view the actual performance within a virtual space. The advantages of this is, coaches will be able to pinpoint, using a variety of virtual cameras that represent the three dimensional structure of the world, when and where the athlete’s form can be improved. And because the performance is data that is accessible within the virtual space, coaches can tune the existing performance to show, visually side by side, how and at what point the athlete’s motion diverts from an optimal sequence. Construction can also benefit from the real-time feedback of virtual environments and collaborative play. With the advent of 3D scanning and photogrammetry, our physical spaces can be replicated accurately within a virtual space. Upon replicating the physical space into a real-time virtual space, construction workers can wear motion capture technology to have their physical location translated within the virtual space. And because their location is captured within an exact replica of their physical working conditions, everyone’s position can be pinpointed at all times; this would come in handy in the case of a working accident, detailing when, how, and where to search for the injured or missing individual. Our system further contributes people who are handicapped, particular with motor trauma. Because our system uses eye tracking, with a bit of tweaking, it can provide a unique platform for the disabled to perform human computer interactions; no longer are they limited to the boundaries of analog input. Lastly, entertainment businesses can benefit from our system by uniting working conditions within a real-time, shared virtual space. In this scenario, all employees have their very own physical and virtual working space. However, at any given time, the supervisor can enter your virtual space, with authentication, to follow up on the status of a given assignment, provide constructive criticism, and even direct your virtual space to other virtual spaces for rapid, iterative, and collaborative productivity. This real-time interaction and collaboration can provide an efficient platform by which supervisors and art directors can see what you are working on, while you are working. Even more, because the shared virtual space is networked, multiple individuals can enter and work together within a grander virtual space, providing everyone with a visual representation of the final delivery, at all times.

As articulated above, our system can be tailored a number of applications, fields, and personnel. Its unique structure presents a firm foundation for experimental avenues, offering unique possibilities to design, collaborate, interact, present, and showcase within a real-time 3D compositing space.

6.2 Known Limitations & Future Work

Our system, in prototype form, demonstrates an alternative filmmaking platform within a real-time 3D space. While the above outlines the system's current capabilities as a virtual cinematography tool, by no means does it showcase all its possibilities. There are, however, existing limitations and constraints regarding our system and the input devices used.

In this section, we will discuss our system's constraints and limitations regarding eye tracking, real-time performance capture, editing, and collaboration. We close with future development.

6.2.1 Eye Tracking

Some of the more cumbersome constraints encountered dealt with the current technological state of the Tobii EyeX eye tracker. For instance, the eye tracker was unable to take into account a person’s positional offset after calibration. This limitation is quite tedious as it typically results in having to repeatedly calibrate to compensate for accidental positional offsets. And because a dedicated stand is not supplemented with the eye tracker to increase its stability, recalibrating becomes a more persistent hindrance.

A further constraint was with the eye tracker’s inability to differentiate between left and right eye. Although this may not seem like a significant limitation, it did present challenges that limited its usability. Reason being, there are only so many ways to interact with an interface using our eyes- closing eyes, gazing, and fixating. Because Tobii’s API excluded left and right eye interactions, a gamepad was necessary to supplement for eye tracking’s limited functionalities.

Another limitation was the eye tracker’s tracking distance. For optimal performance, participants were required to maintain a fixed distance between one to four feet from the eye tracker. Diverting from these distances after calibration resulted in frequent loss of eye gaze. Lastly, because the eye tracker relies on pupil reflectance to track the position of the eye’s, ambient lighting was required in order to maintain consistent eye capture.

3.7.3 Real-time Capture and Editing

With every Unreal Engine 4 update comes expansion of tools, greater optimization, handling of graphical processes, and bug fixes. Consequently, our version of Unreal Engine 4 (4.10.4) contains a demo record and playback bug, which is a handy tool whereby *Directors* can capture, store, and replay performances. Currently, our system is able to capture performances in real-time from the perspective of the player, but offers no way to edit and output the sequence from within the game engine. To capture and edit performances, machinimators will have to endure capturing the performance using screen capture software (i.e. Fraps), then using a 2D compositing program (i.e. After Effects) to edit the sequence.

With the release of Unreal Engine 4.12, the aforementioned cinematic tool - Sequencer - will be readily available. However, Tobii's Unreal Engine plugin is lagging behind two major Unreal Engine releases, preventing our system from making use of Sequencer's 3D compositing shot-based workflow to capture, edit, and output performances.

3.7.4 Collaboration

In its prototype form, our system contains several collaborative limitations. One being, the *Performer* cannot see the *Director*, nor the spawned cameras in-game. This constraint isolates the *Performer* from the *Director*, which is an important interpersonal relationship required when choreographing and rehearsing performances. Although it is not necessary for the *Performer* to see the *Director*, it can increase clarity of commands. An example case scenario is: "Ok, do you see where I am? You need to look in this direction upon exiting the cabin. I will cut to a camera close-up, and when you hear the sound of a gun, I need you move to immediately move in this direction, then briefly look down in shock and awe."

Another limitation deals with how communication is established. Our system does not address sound input/output through networked play. This means either: (a) the *Performer* and *Director* will have to remain in close physical proximity to communicate, (b) or the *Director* and *Performer* will have to rely on voice call services such as Skype or TeamSpeak (among others) to communicate and collaboratively produce Machinima.

3.7.4 Future Development

Our system is a flexible tool that upholds a firm foundation for expansion and experimentation. As a result, our system can be expanded upon to include the following:

***<u>MachinaRoles</u>*:** Currently, our system supports two roles, *Director* and *Performer*. Each of these roles have specialized functions relating to their duties during film production. However, future development will address adding other specialized roles (MachinaRoles), including scriptwriters, lighting artists, special effects artists, set dressers, sound technicians, post production editors, animators, synthespians, texture artists, and so on. These specialized roles can even extend beyond traditional film roles by incorporating the game space. For example, various roles can include a Game Master that controls the rules of the game, a role involving spawning and manipulating artificial intelligence during a performance, an optimizer that is able to use eye tracking and another input devices to maintain optimal computer performance; optimizer's location in-game can attribute to a change in levels of detail for mesh optimization, reduction of shadow quality if camera's are out of proximity, texture de-resolution, discarding animations out of focal view, and so on. Our system can be expanded upon to include all of the aforementioned roles within a shared virtual space. And because Machinima is traditionally made within game engines, which run on software, there exists almost limitless flexibility in what roles can be integrated to inhabit and influence the real-time virtual space.

MachinaMarkers: As aforementioned, "Because production is driven in real-time, there is no need for green screens. Rather, the performance is tuned based on the scene they inhabit, which adds an additional layer of realism; performers are now able to act upon on where they are and what they see, not where they are perceived to be and conditioned to see." Building on this point, our system seeks to exploit a new method by which performances can be collaboratively choreographed, in real-time.

MachinaMarkers exemplifies this potential solution, which will serve as a guide to assist performers flawlessly execute a sequence of actions within a pre-scripted virtual space. MachinaMarkers will be symbols only visible to *Performers* during a performance, which can be toggled off at any point in time. Each MachinaMarker will serve a purpose. We consider the following types of MachinaMarkers:

Event Marker: A marker that performers must reach in order to trigger the next film sequence and/or end take. Upon in-game collision with this marker, pre-scripted events are triggered, be it explosions, particles, spawning artificial intelligence which leads to an action scene, lights turning off, jets flying by, activating pre-scripted dialogue, activating timed markers, execute markers, and so on.

Pathway Marker: A marker that visually guides the performer where to move next, be it running, walking, running, crawling, taking cover, and so on. This marker ensures the performer is performing and following the intended path laid out by the director.

Execute Marker: A collaborative marker detailing when performers should execute an action. Types of actions include interrupting a dialogue, storming into a room, arming a missile, dodging a bullet, and so on.

Emotional Marker: A marker that reminds performers when and how intensely they should express emotion. This can serve beneficial as director and performers are transparently and actively communicating during the performance.

Timed Marker: This marker gives a visual representation of when a performer should execute an action, end an action, reach a destination, or respond to an action.

Action/Take Marker: This marker communicates to performers when a take begins and when it ends.

Wing It Marker: A marker giving performers the freedom to execute actions at will during a performance.

MachinaMarkers may be designed for rehearsals to give performers a greater sense of what they should be doing, how, when, and where. Because performers inhabit a virtual space (and not the other way around), a new language may be appropriate to effectively make use of real-time game spaces.

Camera Eye Gaze Depth of Field: Future development may incorporate depth of field based on a user's location of gaze. This can be used to drive attention to focal points in a take, a technique often employed to guide the viewer's attention.

Sophisticated Restaging Mechanic: In traditional film practices, several takes of the same performance is often necessary to achieve the envisioned sequence. Action scenes, in particular, where objects and furniture break require cleanup crews to reset the stage. In our system, performances reside within a virtual space. What this means is, with this mechanic, an action scene within a virtual space can be restaged with a simple press of a button. With this planned staging mechanic, the entire virtual world will be reset, including animations, lighting, actors, camera positions, and so on. This provides a convenient way to capture iterative performances.

Demo Capture and Real-Time Scene Manipulation: Planned future work is to capture performances in demo format. In so doing, performances can be stored, accessed, manipulated, and overlaid with other demo performances. This proves useful because performances captured in demo format retain their 3D representation. This means stored performances can be recaptured at any time using a variety of camera angles, camera movement techniques, lighting conditions, environments, and so on without having to retake a performance.

Real-Time Virtual Collaborative Production: Another planned addition to our system is to converge cinematic production into a collaborative, shared networked space. In this scenario, the entire workforce will be interconnected virtually by means of digital workspaces, working in unison using a single Git repository throughout production. This means the final output of the project will visible at all times during production. Further, because the entire workforce shares a networked space, directors and producers can digitally infiltrate any department workspace to *see* their progression and provide iterative feedback, all while production is active.

Perception Neuron and Sequencer Integration: Real-time animation is a unique hybrid that has infiltrated cinematic production, showing great promise of future endeavors. As a result, for future development, the *Perception Neuron Mocap* will be integrated with Sequencer. Perception Neuron is a real-time motion capture solution that translates human body movements to a virtual avatar, in real-time. By coupling this technology with Unreal's Sequencer, performances can be captured and manipulated in a variety of ways; this includes adjusting animation playback speed (or specific actions), splitting animation sequences, creating blend-able morph targets, and export real-time captured animations into 3D software application packages for additional editing, among others.

6.3 End Remarks

Over the past few years, game engines have made a tremendous impact in the film industry, particularly in previsualization studies. Attention has been directed to their affordability, technical quality, and scalability in cinematic pipelines. As a result, game engines have been regarded as "posing a potential threat to the tried and true methods of the movie industry [11]." Whether these claims will come to fruition is up for debate, but it is clear game engines possess fertile ground to speed production workflows, offering untapped possibilities for 3D compositing.

In the creation of our system, we aimed to demonstrate these untapped possibilities and to exploit new collaborative in-game interaction techniques and possibilities for machinimation scene capture. By leveraging consumer affordable technology, we were able to develop a flexible, modular, and low barrier entry platform with fresh control mechanisms that exemplify cinematic production alternatives within a real-time virtual 3D space. We draw upon the idea of democratizing cinematic production, offering an affordable method by which amateurs and small production teams can produce passive, exploratory, and interactive viewing content from the comfort of their own homes. Our goal in this thesis is to involve more people in using a real-time engine as a media production environment *and* post production tool, a characteristic of Machinima's origin that distinguishes itself from traditional cinematography practices. In so doing, we aim to exploit a playground of programming possibilities whereby synthespians and artificial intelligence can mutually cooperate within a shared virtual space.

With the advent of 360-degree video recording, playback, and viewing, new production methods and techniques are shaping film. The reason for this is, with 360-degree cinematic content, the director loses control of what viewers see, struggling to direct the audience's attention in a space where they form their own stories. In the same way 360 degree immersive films call for new production and compositional techniques to guide the viewing experience, real-time 3D spaces require a new language. In virtual filmmaking, camera movement lags, cue delays, uncanny artificial intelligence, and other forms of *perceived* mistakes not only form the composition, but offer clear direction of gaps in knowledge, which can lead to innovative solutions that will push real-time production to its promise of revolutionizing new systems of filmic production. In relation to our system, we do not aim to replicate the tried and true methods of the film language. Rather, *Directors* and *Performers* are encouraged to experiment with what is possible within a real-time virtual 3D pace, using the structure of games to influence the film language- whether experimenting with different shot compositions, integrating distinct technology, restaging a destroyed scene with a press of a button, integrate artificial intelligence that behaves differently in each performance sequence, involve multiple camera operators with distinct collaborative roles (i.e. MachinaRoles), capture and edit demo performances, and so on. In the end, if we don't exploit the opportunities presented with real-time 3D spaces, we may miss out on technological advancement that can change the way we create, view, and experience films. "The promises and pitfalls of certain technological forms are realized only through active and ongoing struggle over their creation, uptake, and revision [13]."

References

[1] J. D. Bolter and R. A. Grusin, *Remediation understanding new media*, 1st Edition ed.: The MIT Press, 2000.

[2] I. Buczek, "The aesthetic of immersion in the immersive dome environment (IDE): Stepping between the real and the virtual worlds for further self-constitution?," *Technoetic Arts,* vol. 10, pp. 3-10, 2012.

[3] F. R. Dellario, "The Future of Machinima as a Professional Animation Resource and its Growth as Real-Time Animation in Virtual Worlds," *Journal of Visual Culture,* vol. 10, pp. 89-92, 2011.

[4] T. Fullerton, *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*: Taylor & Francis, 2008.

[5] H. Jenkins, *Convergence culture: where old and new media collide*. New York U6 - eBook: New York University Press, 2006.

[6] H. Jenkins, S. Ford, and J. Green, *Spreadable Media : Creating Value and Meaning in a Networked Culture*. New York: NYU Press, 2013.

[7] J. J. Jerald, "Scene-motion- and latency-perception thresholds for head-mounted displays," Dissertation/Thesis, ProQuest, UMI Dissertations Publishing, 2009.

[8] P. Johnson and D. Pettit, *Machinima: the art and practice of virtual filmmaking*. Jefferson, N.C: McFarland & Company, Inc., Publishers, 2012.

[9] M. Kelland, D. Morris, and D. Lloyd, *Machinima: Making Animated Movies in 3D Virtual Environments*: Ilex, 2005.

[10] M. Knobel and C. Lankshear, "Remix: The Art and Craft of Endless Hybridization," *Journal of Adolescent & Adult Literacy,* vol. 52, pp. 22-33, 2008.

[11] H. Lowood, "High-performance play: The making of machinima," *Journal of Media Practice,* vol. 7, p. 25, 2006.

[12] H. Lowood, "Found technology: Players as innovators in the making of Machinima," *Digital Media*, 2008.

[13] H. Lowood, M. Nitsche, and I. ebrary, *The machinima reader*. Cambridge, Mass: MIT Press, 2011.

[14] L. Manovich, *The language of new media*. Cambridge, Mass: MIT Press, 2001.

[15] L. Manovich, "Image Future," *Animation,* vol. 1, pp. 25-44, 2006.

[16] L. Manovich, *Software Takes Command*: Bloomsbury Academic, 2013.

[17] P. Marino, *3D Game-based Filmmaking: The Art of Machinima*: Paraglyph Press, 2004.

[18] N. H. Mat Zain, F. H. Abdul Razak, A. Jaafar, and M. F. Zulkipli, "Eye Tracking in Educational Games Environment: Evaluating User Interface Design through Eye Tracking Patterns." vol. 7067, ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 64-73.

[19] A. Mazalek and G. Davenport, "A tangible platform for documenting experiences and sharing multimedia stories," presented at the Proceedings of the 2003 ACM SIGMM workshop on Experiential telepresence, Berkeley, California, 2003.

[20] A. Mazalek and M. Nitsche, "Tangible interfaces for real-time 3D virtual environments," presented at the Proceedings of the international conference on Advances in computer entertainment technology, Salzburg, Austria, 2007.

[21] M. S. Meadows, *Pause & Effect: The Art of Interactive Narrative*: Pearson Education, 2002.

[22] M. Nitsche, "A Look Back at Machinima's Potential," *JOURNAL OF VISUAL CULTURE,* vol. 10, pp. 13-18, 2011.

[23] K. Salen, "Arrested Development: Why Machinima Can’t (or Shouldn’t) Grow Up," ed: The MIT Press, 2011.

[24] K. Salen and E. Zimmerman, *Rules of Play: Game Design Fundamentals*: The MIT Press, 2003.

[25] J. Seibert, "An Exploratory Study on Virtual Reality Head Mounted Displays and Their Impact on Player Presence," Dissertation/Thesis, ProQuest, UMI Dissertations Publishing, 2014.

[26] G. J. Winters, J. Zhu, A. Drexel University. College of Media, and Design, "Developing Visual Narrative: Designing Structural Composition Principles to Guide Player's Attention in Adventure Games," Dissertation/Thesis, 2013.

Appendix A.

Definition of Terms

Asymmetrical Gameplay: When two players have separate experiences as they play a video game together. For instance, while the player wearing a head mounted display may see an ordinary painting on a wall, the display for the other player may reveal a safe requiring interaction in order to proceed through the narrative.

Blueprints: A visual scripting node-based interface in Unreal Engine 4 used to create gameplay elements.

Digital Interactive Narrative: A form of storytelling allowing users to influence the unfolding path of a narrative. Unlike a linear story experience where users passively view a sequence of events, a digital interactive narrative requires users to actively participate by spatially navigating in a game environment in order to progress through the story.

Distinct technologies: The mechanical and functional differences between a head mounted display and eye tracking system. A head mounted display is designed to be worn and experienced by a *single* individual, while the eye tracking system offers the flexibility for participation with *multiple* individuals.

Eye Tracking: A device used to measure eye position, movement, gaze, and pupil size in real-time. An eye tracking device will be used as an input device to spatially navigate inside a 3D virtual environment.

FRAPS: A real-time screen capture application released in 1999, enabled players to record gameplay outside the game engine.

Head Mounted Display: Piece of hardware that can be placed over a user's eyes, placing the viewer in a virtual 3D environment.

Immersive: The *ability* for two participants to actively collaborate using distinct technology while- both physically present but one telepresent- maintaining high involvement, focus, and transparency during gameplay.

Limited Ocular Impairment: People who are able to clearly distinguish between objects and colors two to three feet from a computer screen.

Machinima: In this paper I use The Academy of Machinima Arts and The Academy of Motion Picture Arts and Sciences definition of Machinima that is, "the art of making animated films within a real-time virtual 3D environment."

Machinimators: Any creator of Machinima content.

Machinimation: The process of recording animation in-game, in real-time.

Participatory experience: The *act* of collaboration between two participants using distinct technology to successfully unfold a digital interactive narrative.

Synthespian: An actor wearing motion and facial capture technology to drive a computer-generated three-dimensional human character, designed to simulate a lifelike performance on film.

Tangible User Interface (TUI): A transparent controller interface that allows a person to use human manual dexterity to directly interact with a real-time game engine from their physical environment.

Telepresent: The sensation of "actually" being there in a virtual 3D environment.