NARRATIVE STORYTELLING IN VR THROUGH GAMING

by

Brycon Andrew Carpenter

Honors Thesis

Appalachian State University

Submitted to the Department of Computer Science

in partial fulfillment of the requirements for the degree of

Bachelor of Science

May 2017

APPROVED BY:

_____

Dee Parks, Ph. D., Thesis Director

_____

Scott Rice, Second Reader

_____

Dee Parks, Ph. D., Departmental Honors Director

_____

James Wilkes, Ph.D., Chair, Computer Science

ABSTRACT

Brycon Carpenter: Narrative Storytelling in VR Through Gaming

(Under the direction of Dr. Dee Parks)

Gaming has consistently been acknowledged as a storytelling medium for its signature ability to provide user interaction. As virtual reality becomes a more prominent gaming environment, it will be expected to carry on the tradition of storytelling. Currently, virtual reality is in a state of infancy, where products offer little sophistication and serve as amusements rather than storytellers. As this changeover takes effect, there are certain discrepancies which will hinder a transition from flat screen gaming to virtual reality gaming. In order to create a successful narrative driven virtual reality game, these discrepancies much be addressed in a meaningful way This thesis will also experiment with the aforementioned differences by developing a virtual reality game in Unreal Engine 4.

Contents

Introduction 5
Evolution of Videogame Narratives 7
2.1 Early Video Games 7
2.2 Rise in Narrative Games 10
Narrative Storytelling in Virtual Reality 14
3.1 Importance of Storytelling in Gaming 14
3.3 Problems to Address Transitioning to VR 19
Sword Quest II: Concept and Components 28
4.1 Conception of Sword Quest II 28
4.2 Oculus Rift Development Kit 2 30
4.3 Unreal Engine 4 33
Sword Quest II: Development Log 37
5.1 Initial Research and Brainstorming 37
5.2 Design Process 38
5.3 The Animation Process: Designing the Protagonist 40
5.4 Acquiring Other Assets 43
5.5 Landscape Generation: Creating the Forest Level 44
5.6 Game Mechanic Systems 45
5.7 Sound Assets 47
Conclusion 49
6.1 Future Expansion 50
6.2 Speculation on the Future of Gaming 51
Asset Attribution List 53
Bibliography 54

Chapter 1

Introduction

This undergraduate honors thesis project presents an analysis on the current state of virtual reality (VR) technology and game development, specifically discussing how this new medium directly impacts traditional means of storytelling in gaming. After discussing the disparities between narrative techniques, this thesis attempts to synthesize researched solutions into a VR game. Throughout the thesis, the evolution of gaming technologies, and VR is discussed with focus being on narrative complexity.

Chapter 2 discusses a brief background of the past and present state of gaming narrative storytelling, examining its early applications as well as showing how it has changed over time. Furthermore, this chapter shows how the earliest forms of gaming relate to the earliest forms of motion pictures, suggesting a possible parallel. This chapter concludes by providing an example of a true, complex gaming narrative story.

Chapter 3 examines narrative storytelling in virtual reality. Beginning by highlighting how gaming differs from other storytelling mediums with its capability for interactivity, it then details how to successfully provide a strong interactive story. The main focus of this chapter is detailing what discrepancies need to be addressed in relation to moving narrative storytelling to a VR environment, suggesting some possible solutions.

Chapter 4 transitions to discussing the product of this thesis, a VR capable game written in Unreal Engine 4, which seeks to provide a primary narrative. Specifically, this chapter highlights the conception of the product, detailing the origins of the name. This section then moves onto describing the components used in creating the game, namely the Oculus Rift DK2 and the Unreal Engine 4. Each component is defined and briefly explained in a general overview.

Chapter 5 serves as a development documentation for the product. It details the initial research and brainstorming that led to the design of the game. Afterwards, various processes that went into making the game are documented in brief detail, including animation with graphical software, searching for usable assets, landscape generation, game mechanic systems, and sound assets. Examples will be given where applicable.

Chapter 6, summarizes the conclusions drawn from this thesis, as well as suggests areas for future development and speculates on the future of gaming.

There is one appendix. Appendix A B provides an asset attribution list which gives full credit to the creators of each asset for their contribution. The thesis writer takes no credit for these free license assets.

Chapter 2

Evolution of Videogame Narratives

2.1 Early Video Games

Gaming, similar to the medium of motion pictures, began as a pure novelty, expressing no intention or elements of narrative storytelling (e.g. plot, complex characters, character development, etc.). The earliest acknowledged electronic game traces back to 1947 in the form of a "cathode-ray amusement device," an interactive game that allowed the player to simulate controlling an artillery shell arcing towards a target, visualized as a dot of light on a screen. As the name "amusement device" implies, the game had no purpose other than quick and meaningless entertainment. Motion pictures had a similar beginning of a frivolous existence, with the public lined up to pay a nickel to watch Thomas Edison's kinetoscope record of a man sneezing in *Fred Ott's Sneeze*, the first copyrighted motion picture in the United States. Eventually dubbed the "cinema of attractions," films in the early era from 1894-1907, which targeted audiences based on recordings of dances and idiosyncratic feats, never imagined expressing a story.

It would take decades before video games became a true consumer item in the early 1970s, first in the form of arcade games. In 1972, Atari released *Pong*, a virtual table tennis

game for two players. The game received unprecedented success in the coin-operated amusement industry and inspired numerous variants of ball-and-paddle video games, causing a market saturation. To combat this saturation, large companies like Atari and Midway turned to new genres like racing games, one-on-one dueling games, and target shooting games. However, the market for video games was stalled by the emergence of solid state pinball, pinball utilizing electronic technologies. It was not until 1979, when Midway released *Space Invaders*, that the market began to surge once again in favor of video games. *Space Invaders*, as the large commercial success that is was, pioneered many important arcade game concepts, including basing playability on lives instead of a timer or score, gaining lives based on accumulating points, and the tracking of high scores, inspiring future arcade successes such as *Galaga* (1981). Shortly afterwards, Namco released *Pac-Man* in 1980, and this advanced gaming to begin focusing on an identifiable character – other popular games of this type include *Ms. Pac-Man* (1982), *Donkey Kong* (1981), and *Q*bert* (1982). Because of these broadly acclaimed successes, the period from 1978 – 1982 is largely considered to be the "golden age of arcade video games," and, while these games still have either an extremely primitive or nonexistent approach to storytelling, they are an important step in the evolution of video games.

As arcade monsters began to dominate the public interest in gaming, and as a result of the advancements in integrated circuits, demand for home video game consoles escalated by the middle of the 1970s. One of the earliest attempts at a home console was the Magnavox Odyssey, released in 1972, and the result did not interest the public. The technology of the first Odyssey had extremely limited technology, so when large-scale integration microchips became cheap enough to incorporate in a consumer product, Magnavox re-released the console multiple time, still piggy-backing off the paddle-and-ball phenomena. Atari shortly followed suit with their own

release of the Home Pong system. First generation home console systems provided limited capability or creativity, and, as a result, had limited commercial success.

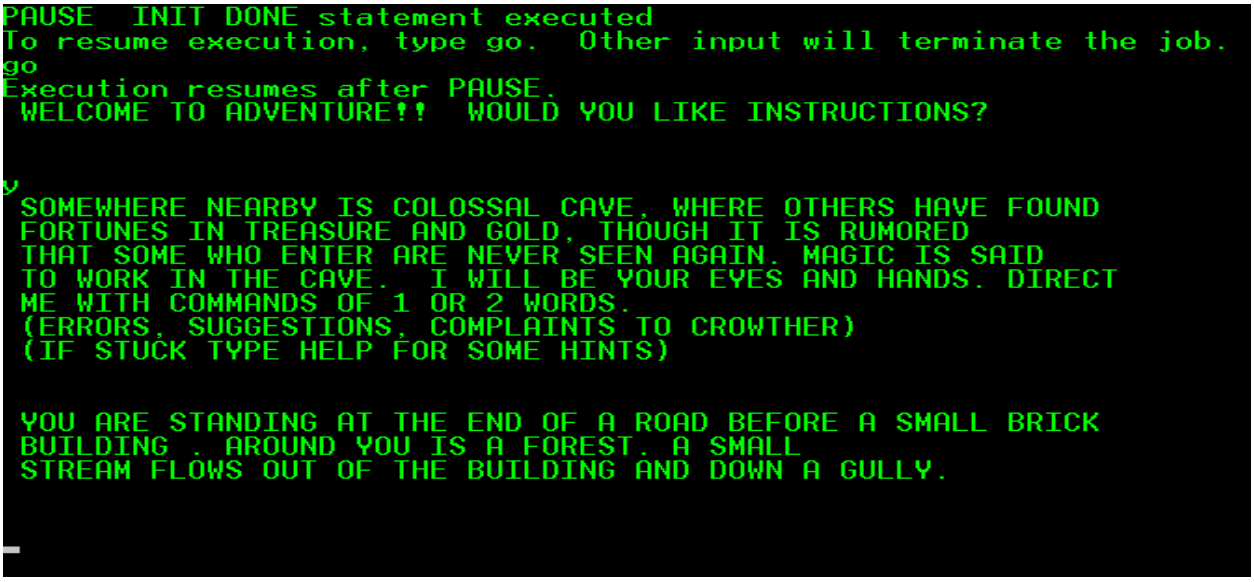

Figure 2.1: A screenshot from the 1976 text adventure game Adventure. This genre was one of the first in gaming to incorporate narrative storytelling.

The last platform for early video games was the mainframe computer, which was admittedly unsuccessful in the 1960s, due to a scarcity of computer resources and a lack of programmers interested in developing entertainment products. However, this platform became much more accessible by the 1970s, following the adoption of BASIC and C high-level programming languages. Unfortunately, most mainframe computers relied on teletypes, which was limited to text display, rather than monitors, ruling out porting of most arcade games to home computer. Instead, this environment cultivated a genre of game, which is arguably the forefather to narrative gaming: text-based games. Perhaps the most successful game of the period was *Colossal Cave Adventure* (1976), also simply known as *Adventure*, pictured in Figure 2.1. The game's creator Will Crowther was inspired by his passion for caving and concepts from the newly released tabletop role-playing game *Dungeons and Dragons*. The game was expanded the following year with content emphasizing the high fantasy of J.R.R. Tolkien. *Adventure* pioneered

a genre of inventory-based puzzle solving and exploring, and its focus on storytelling over fast-paced action would reverberate through gaming history.

2.2 Rise in Narrative Games

As games transitioned from an arcade style to home console setting, the industry began to see a bump in narrative capabilities. Games like *Super Mario Brothers* (1985) and *Castlevania* (1986) exhibited simplistic but slightly more evolved forms of storytelling than there arcade predecessors. These games inherited the quality of a central identifiable protagonist, from games like *Pac-Man*, but put them in a situation where there was a central conflict that needed to be overcome. Whether having Mario save Princess Toadstool (Peach) or using Simon Belmont to defeat the vampire Dracula, the player had a narrative they were following while playing through a game. The end of the game was no longer contingent on "how long can I avoid these ghosts chasing me" or "how long until I use my last space ship life." Instead, the player had an ultimate goal to work towards that fell within the confines of that game's reality. While still utterly simplistic and heavily reliant on gameplay mechanics, these were an important step in developing narrative-driven games.

Expanding on the success of *Adventure*, another, more narrative intense variant emerged named *Zork* (1980). What separated *Zork* from many other text-adventure games was both its attention to storytelling as well as its sophisticated text parsing mechanisms. The game recognized more of what the player wanted to do, understanding some prepositions and conjunctions and using a more descriptive language to create atmosphere. *Zork* challenged the player to think creatively when solving puzzles and allowed the player to explore the world in a non-linear way. Unlike side-scroll games (like *Super Mario Brothers* and *Castlevania*) where

players could only move forward, players in *Zork* could explore the game's locations in any order they liked. The game has a central motivator for its actions that are strictly reliant on the narrative: the grue.

(In Game Text)

> what is a grue?

The grue is a sinister, lurking presence in the dark places of the earth. Its favorite diet is adventurers, but its insatiable appetite is tempered by its fear of light. No grue has ever been seen by the light of day, and few have survived its fearsome jaws to tell the tale.

With a vague description, the game invokes the imagination of the player and provides a mechanism to motivate the player to participate in the narrative, such as in the action of finding a light source: "It is pitch black. You are likely to be eaten by a grue." By using a construct in the story to motivate a player, instead of just directly telling them to do so, the game creates a better sense of immersion and a stronger narrative.

Between the late 1980s and early 1990s, game design shifted to more emphasize narrative. With the increasing capability of home consoles such as the Super Nintendo, leading into the PlayStation, technology could visually create an immersive, even 3D, environment. Without having to heavily rely on the player's imagination, games could now reach a wider audience, including people with limited imagination. One series that is a great example of this would be the *Final Fantasy* series, a Japanese role-playing series, with a reputation for large, expansive universes with complex characters and intriguing lore. Each installment presents an entirely new reality, with new characters that make a significant effort to stray away from clichés. Beginning with *Final Fantasy* in 1987 and continuing to the present day with *Final Fantasy XV* (2016), these games have continued to challenge the bounds of narrative storytelling,

even while sporting some of the most innovative mechanics in gaming history. Among the fans, it is commonly agreed that *Final Fantasy VII* (1997) might have been the most effective at doing so, with up to 200 hours of game play, featuring some of the most well-known characters in gaming. This could also have some basis in that it was the first 3D environment *Final Fantasy* title.

To establish the complexity of narrative storytelling in the *Final Fantasy* series, I believe it is appropriate to provide, as example, the overall narrative of my favorite title, *Final Fantasy X* (2001). It is also important to note that this was the first narrative-driven game that I played, so it is clear why this experience made such a lasting impact on me. The story revolves around the protagonist Tidus, a star Blitzball player – an underwater sport – whose home city of Zanarkand is attacked by an immense creature known as Sin. In the aftermath of the attack, Tidus finds himself carried away to a world known as Spira, which he has no knowledge of, and he stumbles around confused as everyone he asks about getting home tells him Zanarkand was destroyed 1000 years ago by Sin. The lore of the world states that Zanarkand was once a large machina city, controlled by large machines that people became dependent on to accomplish everyday tasks. As a result of this hubris, Sin appeared and laid waste to the metropolis and has since plagued the world as a reminder to atone for sins. To aid the process of purification, Summoners, priests/priestesses who are trained in battle, must perform a pilgrimage to the different temples in Spira in order to prepare themselves for the final battle with Sin, where they must give their lives to provide several years of Calm, when Sin is in a process of rejuvenation. Tidus becomes a part of a traveling Summoner's party, and along the way the game addresses issues such as racism, religious fanaticism, arranged marriage, machine dependence, afterlife, and existentialism, all the while learning about the history of the Machina Wars, the Crusaders, and the Al-Bhed, Guado,

and Ronzo people. From this short examination of the narrative of *Final Fantasy X*, it is clear that storytelling in games has evolved significantly from their simplistic representation in the 1980s.

Chapter 3

Narrative Storytelling in Virtual Reality

3.1 Importance of Storytelling in Gaming

Storytelling has always been a universal feature in every culture and country [1]. It's how humanity has recorded history, inspired revolution, and driven the imagination and comfort of innumerable people. Brian Boyd, author of *On the Origin of Stories: Evolution, Cognition, and Fiction*, argues that "stories are a kind of cognitive play, a stimulus and training for a lively mind." Rather than just being a trivial pastime, storytelling embodies innovation, being more of a tool than a spectacle. Bringing together key elements of intelligence, cooperation, pattern-seeking, alliance-making, and the understanding that other beings have beliefs and knowledge, stories make us stronger and more effective as a species.

It should not be confused: storytelling is a multi-disciplinary art form that transcends the bounds of every medium of art. Originally stories were told visually, through cave drawings, and this slowly transitioned to oral tradition, communicating with others by word of mouth. Continuing to evolve, humanity embedded stories in music and dance, with melodic flow and body motion conveying a narrative meaning. In contemporary times, storytelling has continued to pervade every medium, such as photography, motion pictures, television, radio, mobile

technologies, and social media. Knowing this, it comes as no surprise that gaming has become a prominent medium of storytelling as well – in fact, it might currently be storytelling's most powerful form. Chapter 2 of this thesis discusses the evolution of gaming to the modern day, and we can see a parallel trend in the increasing power of storytelling in this medium. Coming from humble beginnings as a sideshow spectacle with games like *Pong* (1972), gaming has transformed into a powerful medium with expansive universes, complex characterizations, and intricate plots, such as *The Witcher 3: Wild Hunt,* which packed in over 150 hours of narrative driven gameplay, discounting expansion packs.

However, there are at least two key difference which separate gaming from other current mediums of storytelling: interactivity and uncertainty. No other medium to date gives the listener the ability to actively participate in the narrative, to actually become the character they are being told about. In narrative-driven games, the narrative cannot progress until the player provides their input. It is true that the story is orchestrated by the developers, but it is up to the player to allow that story to be told. The uncertainty, that a game's story is undetermined until the gamer plays their part, combined with the knowledge that the player has influence over the story, maintains engagement like no other medium [2].

It must also be acknowledged that not all video games are purposed for storytelling. Similar to music – with the genre of elevator music – and film – with films like *Mothlight* (1963), the result of smashing moth wings, flower petals, and blades of grass onto film strip – video games often have products that offer no story and were never meant to. Actually, these plotless video games can usually be identified by their genre. For example, most fighting games, like the series of *Street Fighter*, *Mortal Kombat, Super Smash Brothers*, *Soul Caliber*, and *Tekken,* are entirely produced and successful based on their gameplay mechanics – rooted in

their origins as arcade games. Similarly, with the rise of the online multiplayer experience, many first-person shooter (FPS) games have emerged that solely market themselves on the multiplayer aspect. These FPS games are nothing more than arenas for players to fight one another. Successful examples of multiplayer based FPS games include the series of *Team Fortress 2*, *Counter Strike*, *Call of Duty*, *Overwatch*, and *Battlefield*. As these games have progressed, with more installations into the series, developers have begun including what has become known as "excuse plots." These are essentially shallow, clichéd, cobbled together stories that are merely included so the game can boast another sellable feature. As such, these types of games are excluded when this thesis discusses narrative-driven games.

3.2 Components of Interactive Storytelling

For video games to successfully harness their ability for interactivity and not misuse it, there are certain criteria which should be followed at the game design stage. Most of these criteria are steeped in engaging the player and stimulating immersion. The former of these is obvious, and applicable for almost every medium of storytelling. If the listener has no interest or investment in the story, then the story will have no prominent effect and might not even be heard. To maximize the effect of interactive storytelling, all of the following components need to be in place. Dropping even one of them will negatively affect the narrative [3].

Firstly, the games must be built primarily around a central story. To not completely kill an immersive link, these central narratives need to be somewhat tangible, containing relatable characters and settings with a clear-cut drama. If, instead, the game is built around the idea of a gameplay mechanic, like swinging swords, shooting targets, or fist fighting, the story will lose legitimacy. The game could still feature these mechanics, but they cannot be the core focus of the

experience. Often games with weak stories have not heeded this warning, and many of the problems listed below stem from this lack of such attention.

Just as it is possible to completely neglect telling a story, it is also possible to go completely the opposite direction and overwhelm the player with story and no gameplay. No other medium can reasonably compete with videogaming's facility for interactivity, but it is also not uncommon for videogames to themselves forget this fundamental concept. It is becoming increasingly more of a tendency to run long cinematic cutscenes, showcasing top of the line graphical rendering abilities, but games like this lose all interaction with the player. The bulk of the experience must involve interaction, playing the game. Experimentation with this principle has produced a genre of video games that are essentially interactive movies. Similar to *Choose Your Own Adventure* books from the 1980s, games like Telltale's *The Walking Dead* play out a movie, forcing players to make crucial decisions in stressful situations. However, to create successful interactive narrative games, you need a steady stream of active play, otherwise the player will distance themselves from the experience, breaking any immersive or engaging experience.

At the same time, interaction cannot be simply implemented for the need of having interactivity; any interaction between the game and the player needs to take place within the context of the narrative. Gameplay cannot be of no consequence to the story; the two must exist for the same purpose. Players need to feel as if they are an active part of the story and not a passive observer. There must be the agency of the player in the interaction opportunities; it must be foundational to the narrative and not just something to fill the time while waiting for the next cutscene. To this extent, the players should be able to understand their role in the story from their actions alone. For example, if the player is supposed to be a pirate, then this should be evident

from the gameplay, sailing ships and looting cargo. Should a game's theme not be apparent from the interaction alone, then the story has not been properly told through gameplay.

Commonly, a game's interaction forms a system of mechanics that a player attempts to master through repetitive practice of the same actions. Unfortunately, repeated actions will diminish a game's story, especially if the length of these repetitive sessions far outlasts the pacing of the story. Continuing to beat the player down with repetitive actions will also dull a player's imaginative senses to comprehend the story. If the player cannot become engrossed in the narrative, they develop a mechanical view of the experience, and the story falls flat on its face. Mechanics do not necessarily need to constantly be changing, but their application should be varied as the game progresses. For example, in the game *Limbo* (2010) the only gameplay involves platforming and puzzle solving, but, throughout the game it takes these skills to thematically diverse areas, where players can test these skills against varying environments.

Not only does the game need to keep its focus on the story, but the game also needs to keep the player's focus on the story. Challenges and puzzles should not consume the entire focus of the game. If a player becomes stuck for a long period of time, they begin focusing on the underlying game mechanics and systems to understand their situation, if not all together just giving up, degrading the narrative experience. There are many progression blocks that can keep a player rooted, like complexity in puzzle design, obscurity in navigation, or skill-intensive sections. That does not mean that games should not have these challenging sections, but they cannot be allowed to diminish the urgency of the narrative aspect. If a section in the game would consistently deter player immersion into the narrative, it is worth considering altering or removing that section.

These components are by no means an indicator of the quality of the game. A game that meets none of the requirements could still be an excellent game, but it cannot claim to have a fully interactive narrative at the heart of its focus. For example, let us take a recent game with some significant claim to narrative storytelling, *Bioshock Infinite* (2013). The game focuses on a narrative driven story to infiltrate the floating air city of Columbia, extract the prisoner Elizabeth, and escape amidst the struggles of warring factions. The game intricately addresses themes of racism, religion, and an ideological society, while simultaneously examining the concept of reality through tears in the space-time continuum. The story was well composed, and a welcome change amongst the shift to multiplayer intensive games at the time, but it still falls short of a couple of the components previously discussed. Elements 4 and 5 (repetition and progression blocks) plague the game at certain points; the FPS gameplay offers little excitement and it is likely the player will feel removed from the story amongst the tedious shootouts. Despite the massive scaling in pacing of the story, the actions do not follow suit, making it a victim of lack of element 3 (story-related actions). Still the game is a strong contender as one of the best games in years, and the narrative remains strongly defined, but these shortcomings likely reveal a lackluster passion for commitment to the story. Knowing the game's history, it is easy to see that development of the game was hindered by an ever-changing release date and an assimilation into the previous *Bioshock* installments.

3.3 Problems to Address Transitioning to VR

As new VR technologies emerge, gaming will be the frontier development zone for virtual reality storytelling. Despite advancements in 360-degree video and virtual reality films, like Oculus Story Studio's *Henry* (2015), VR gaming is experiencing the largest growth in VR

products of any other medium. However, gaming has only been a prominent storytelling environment for around the last 40 years, and, unfortunately, many of the techniques that have been adapted for interactive storytelling will not transfer to a virtual environment. This produces many discrepancies in the way video games have been telling stories on a flat screen, as opposed to how they will need to be told in a virtual setting.

One of the most obvious and discouraging discrepancies, which I must discuss first before examining storytelling elements, is a limitation of the current technology in use: motion detection. This has been the same problem since the 1995 Nintendo Virtual Boy, a 32-bit table-top virtual reality console. Granted, the current head-mounted displays (HMD) of the Oculus and HTC Vive allow for more head motion, including 360° vision and a new sense of depth. However, these HMDs do not allow for full range of movement, and, while you can look around, you cannot truly inhabit the virtual space. Room-scale VR has been an attempt from developers to work within the confines of a pre-calibrated space, but the general premise still confines you to believe in an expansive virtual world from a confined space in the natural world. Another attempt to combat the confines of space is allowing your character in game to warp/teleport around. *Minecraft* (2011) has taken this approach, by allowing you to point to a position and warp to it. However, these are mere bandages and are by no means a permanent solution. It is not feasible to believe that games, which have recently followed a trend of massive open-world exploration in outrageously successful titles like *The Elder Scrolls V: Skyrim* (2011) and *Grand Theft Auto V* (2015), could ever be restricted to small range of motion.

Now that the elephant in the room has been discussed, this thesis can begin to examine more abstract concepts of storytelling, beginning with discussion of perspective. Historically, games are usually developed for any combination of three different perspectives: first-person,

third-person, and omniscient. First-person (e.g. *Halo: Combat Evolved* (2001)) and third-person (e.g. *Gears of War* (2006)) games are widely known and recognizable. What might be more difficult to think of would be omniscient perspective games. Usually these appear in the form of strategy games, where the player takes on a god-like role (e.g. *Sid Meier's Civilization* (1991) and *Starcraft* (1998)).

There is a fundamental issue here when relating perspective to VR games: does every game in VR need to be in first person perspective? Intuitively that seems like an obvious answer: yes. How can an immersive world be created in a narrative if you are not seeing through the eyes of the protagonist? Hovering just behind the protagonist offers no immersive experience. But if gaming is to inevitably transfer to a VR environment, does that mean that we abandon the success of current third-person perspective games? Games like *Grand Theft Auto*, *Mass Effect*, *The Legend of Zelda*, etc. would be forced to a first-person perspective, and for many this would completely ruin the game. On top of this, this type of change may also fundamentally impact gameplay mechanics. Performing spins and rolls from a first-person perspective would be absolutely sickening, especially for those who already suffer from motion sickness. Currently, there are experiments with developing viable third-person games on VR. One of the most prominent is *Chronos* (2016) a game with similar style to *The Legend of Zelda*'s puzzle-based dungeons. However, it is undeniable that first-person perspective is significantly more immersive than third-person, allowing you to become the protagonist rather than just observe them.

After examining perspective, we must also consider the narrative relationship between story and plot; that is, the distinction between what happens and how it happens. In cinematic terms, as in gaming, there are events that occur within the context of the story that are not seen in the plot. Take, for example, the famous scene from *Raiders of the Lost Ark* (1981), where Dr.

Jones flies from his college in the United States to the snowy mountains of Nepal. The story of such an event involves all the steps that he would need to make this flight: buying a plane ticket, checking luggage, boarding the plane, etc. The plot, on the other hand, involves the superposition of a world map with an arrow to symbolize his journey. This is an important distinction to be made; the viewer would have no interest in watching events that have nothing to do with the protagonist's ultimate goal.

The alternative to a sharp distinction between story and plot involves the situation where story and plot are identical in length, and this is how VR naturally lends itself. We experience life in real-time, so for an environment to be truly immersive it must also be experienced in real-time. This means that VR narrative games should not have large inexplicable cuts in time, a technique which has often been used in modern gaming, especially action-intensive games. Consider a game like *Call of Duty* or *Battlefield* with campaign missions that oscillate between different countries all over the world. In a narrative where story equals plot, we would need to be on the plane ride between each location, experience full briefing and debriefing segments, as well as equipment checks and routine operations. Of course, this will in all likelihood never happen (particularly at the request of this thesis writer), but on a technicality, this does present a weakness in full immersion. Hopefully, this is one weakness that developers and gamers alike are willing to overlook.

Another feature that flat-screen games have become increasingly reliant on but that will not transfer to a VR world is the use, or rather overuse, of user interfaces (UI). For almost all gaming experiences, it is not sufficient for the user to just rely on the people and objects in the world to have full interaction. Instead, games introduce onscreen widgets to provide information or facilitate a vehicle for interaction. These include, but are not limited to: health bars, inventory

menus, settings menus, time, weapon ammunition, mini-maps, reticles, objective messages, etc. On-screen widgets have become paramount to the gaming experience, as increased potential of UI gives way to many new gameplay features. However, these on-screen widgets really have no place in a virtual reality world. Not only do they remove an aspect of immersion, in some instances they are just impossible to implement. The common technique for UI widgets is to anchor them to corners of the screen; with VR headsets, you cannot even see the corner of the screen, and even if you could they would distort the view. However, just because the go-to solution of slapping text on screen might not be the answer, the gaming industry cannot simply discard the idea of user interfaces – they have proven invaluable to the experience. Instead, there needs to be solution that can encapsulate the concepts of these on-screen widgets.

To create the perfect versions of these 3D widgets, which optimize functionality and immersion, they should meet certain criteria. Other than technical specifications for ideal placement for the best user experience, the most important criterion pertains to the diegetic attribute of the widget, whether it is an element of the world and, preferably, has narrative purpose. A non-diegetic UI does not exist within the game world; they usually just represent concepts, like health and energy, and there is no thought that the character actually sees this element. What a VR world desires is a diegetic element, one that exists in the game world and the character purposefully interacts with, and the optimal type of these would be narratively relevant. While a giant square menu that spawns in the world can be considered a diegetic element because it physically exists in that game world, it does not have any narrative relevance, as it is not typically plausible to have giant menus spawning randomly. Instead gaming should look more to things like a character's wristwatch to display heartbeat as an indication of health.

To illustrate the concepts explained above, this thesis will provide some historical examples of good gaming UIs. Ironically, some of the best examples of these UI exist in flat screen games. Figure 3.1 shows the Pip-Boy interface from *Fallout 3*, an on-arm device that is the primary conduit for information about the player's character, area, etc. The Pip-Boy is completely diegetic; many characters reference the Pip-Boy product and the character's animation pulls the device to the foreground when the "menu" is activated. The Pip-Boy also serves as a diegetic device for many other functionalities of the game, including a Geiger counter, radio, and flash light. Another great example of good diegetic UI comes from *Dead Space*'s Resource Integration Gear (RIG) interface, shown in Figure 3.2. Perhaps the strongest proponent of this UI is its subtlety, providing a more immersive environment. The player's suit provides a spine-mounted display that indicates the player's health as well as a scapular monitor that shows the player's stasis capacity (an element of the game equivalent to an energy meter). Furthermore, the RIG provides other diegetic functions such as the "flat-line" noise on character death, holographic projectors to create a 2D display for the character, a linear display guidance system, and biometric locking capabilities (explaining how players cannot access certain areas of the world until certain points in the game). These are phenomenal examples of the innovative types of UI that will be needed in a virtual world.

Figure 3.1. The Pip-Boy 3000 interface from Bethesda's *Fallout 3*.

Figure 3.2. The RIG interface from EA's *Dead Space*. The spinal bar indicates health and the right scapular crescent shows the player's stasis meter.

On the topic of UI challenges, subtitles are another critical component in gaming, that have been encompassed into nearly every game. Originally implemented for use on television, these have been used primarily to provide accessibility of content to the hearing impaired. However, a large percentage of usage occurs for people that can hear. Especially among gaming, many use subtitles as a way to visually follow along with dialogue, creating a more cohesive experience. Especially in a procedurally generated gaming world, where dialogue might potentially be obscured by irrelevant sound effects or inopportune audio mixing, subtitles serve to provide clarity.

But how can this feature be implemented successfully in a virtual environment? On flat-screen games, these subtitles can be perfectly placed in the margins of the screen that might otherwise go unused. However, VR games have no such margins, so subtitles would need to be placed in the world, as previously discussed diegetic elements. This results in small transparent widgets floating around the player with text, and they must face the subtitles to be able to read them, and on a VR platform this likely means completely missing the central content. Certainly, this is not a favorable experience, but developers have little choice; they must provide this feature to allow access for the hearing-impaired. One interesting thought might be to create a game with the dialogue spoken solely through sign language, although the technical difficulty of this would be high, and the market for such a game would likely be insufficient to dedicate resources.

There are also some smaller concepts to consider that might just be for pure speculation. For instance, how will the VR experience impact death mechanics? Someone who has played games for most of their life has likely become desensitized to dying in games, but it will likely become a completely different experience in VR. Aside from being a psychological challenge, it

is also technically challenging. The go-to for dying in first person would be a fade-to-black transition, but players will eventually demand more creative death sequences, like a ragdoll effect from explosions. Developers have expressed that even experiences like effects from flash grenades and stun grenades have had to be toned down, to not make the player uncomfortable. It is no longer a computer game on your screen in front of you, it is an entire experience, and what was once a thrilling experience now becomes unpleasant [4].

Chapter 4

Sword Quest II: Concept and Components

4.1 Conception of *Sword Quest II*

As previously stated, the goal of this thesis was to produce a narrative-driven VR game using techniques identified through research. Immediately, this involved constructing a narrative around which to focus the game. Jumping to the forefront of my mind, was a game I developed for my senior project in high school: *Last Quest* (Figure 4.1). At the time, I had very limited experience with programming, having only a single AP Computer Science course in my repertoire, so I challenged myself to write a text-based adventure game in the Java programming language. The narrative of the game ended on a cliffhanger, due to time constraints and the complexity of the scope of the project.

Appropriately, when I found myself in need of a narrative for this undergraduate thesis, I jumped at the opportunity to pursue a sequel to *Last Quest*. The evolution from text-based adventure to virtual reality with the backing of an industry-grade development engine highlighted an ironic full-circle in my education. Furthermore, my choice to produce a sequel also satirized the tendency of the current video game industry to rely heavily on sequels and spin-offs, rather than producing original content.

There was some further blatant irony and allusion in the naming of the game, which may not be clear at first. Obviously, with the first installment being named *Last Quest*, the expected name for a sequel would be "Last Quest II," if a conventional enumerated naming scheme is followed. When *Last Quest* was run through its beta phase, the testers and their friends consistently referred to the game as "Sword Quest" – an effect of the large ASCII art sword on the title screen (Figure 4.1). From then on, the game was affectionately dubbed "Sword Quest," with no mention of the original title. When the conception of a sequel game began, the title was instantly set to be *Sword Quest II*, as an homage to the misnomer.

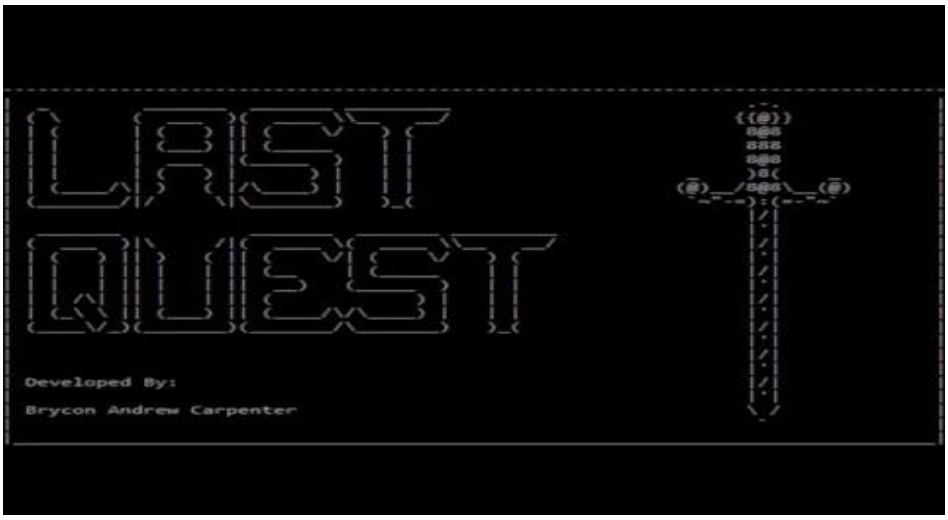

Figure 4.1: The title screen of my text adventure game *Last Quest*, developed in Java for my high school senior project. The source code is available at: https://github.com/Bryconc/Last-Quest-Text-Game

Thus, this confusing name choice further parallels a few historical game naming convention headaches. One of the most infamous naming *faux pas* was in the localization of the *Final Fantasy* franchise. When the original *Final Fantasy* was released in Japan, the US received its localization under the same name. Japan's *Final Fantasy II* and *Final Fantasy III* did not release in the US, and when *Final Fantasy IV* was released in Japan, the US released it under the title *Final Fantasy II*, as it was the second installment of the *Final Fantasy* series to get a US release.

Continuing with this trend, when Japan released *Final Fantasy V*, the US again did not receive a localized version, so when *Final Fantasy VI* released in Japan, the US released the game as *Final Fantasy III*. Then the development company SquareSoft switched development from Nintendo's NES and SNES to Sony's PlayStation, and they released *Final Fantasy VII* with the same title in both the US and Japan and have done so with each subsequent release (visualized in Table 4.1). This naming error has since been corrected, with localized re-releases of each game in the series, but for years US players were confused about what happened to installments IV, V, and VI, when they should have questioned what happened to installments II, III, and V.

Installment Number	Initial Japanese Release	Initial US Release
1	*Final Fantasy (1987)*	*Final Fantasy (1990)*
2	*Final Fantasy II (1988)*	X
3	*Final Fantasy III (1990)*	X
4	*Final Fantasy IV (1991)*	*Final Fantasy II (1991)*
5	*Final Fantasy V (1992)*	X
6	*Final Fantasy VI (1994)*	*Final Fantasy III (1994)*
7	*Final Fantasy VII (1997)*	*Final Fantasy VII (1997)*

Table 4.1: The original release date of each *Final Fantasy* game along with its original US localization. This chart shows the discrepancy in names across regions.

4.2 Oculus Rift Development Kit 2

This thesis used two primary sources of hardware for completion. A head-mounted display (HMD), Oculus Rift DK2, was used to present the game. In parallel with the HMD a positional tracker device was used to receive user head motions. A tradeoff was made over the course of the project. Originally the project was planned to use Leap Motion technology to read input from the user's hands, but to allow for more mobility and capability in the game, the project instead was developed to receive input from traditional keyboard and mouse.

HMDs can be classified as anything that attaches to your head and presents visuals directly to your eyes, but the more important component in VR is the attention to peripheral vision, or vision from the side. In creating a VR environment, an HMD combines many technologies to produce the best possible experience including display technology, refresh rate, latency, optics and head tracking [5].

Display is perhaps the most obvious part of HMD, as if your face were glued to the lens. Historically, the display has been accomplished with LCD, Liquid Crystal Display, panels, similar to most smartphones and computer monitors. Recently – as with the Oculus Rift DK2 – there has been a switch to OLED (Organic Light Emitting Diode), to allow for better response time and exhibit "low persistence." On an LCD screen, the image is held until a new image replaces it at a rate of 60Hz, 60 times per second. When your head moves, the image will blur, because the image shown will only be valid for a few milliseconds, thus feeding your brain wrong information, causing many users violent motion sickness. The OLEDs instead only show the image for a few milliseconds, afterwards the screen is black until the next image appears (a timing that is too small for the brain to recognize), avoiding a blurring effect. The effect of this low persistence is a razor-sharp image despite shaking your head vigorously [6].

HMDs must also rely on high refresh rates to produce smooth movements and avoid motion blur. Refresh rates refer to the speed at which a display can change its content over a certain period of time. Modern computers typically have a refresh rate of around 60Hz which translates to a maximum frame rate of 60 frames per second (fps) – where a frame is a complete image displayed on a computer. As a benchmark, 60fps would be a minimum needed to obtain fluid motion and eliminate motion blur, but HMDs as high as 120Hz already exist – with the Oculus Rift DK2 clocking in at 90Hz. Just as HMDs seek to maximize refresh rate, in the same fight they need to

contend with latency. Latency, which in general terms means a time between input and output, in relation to HMDs means the time it takes for the picture in the VR world to keep up with head motions. To fool the brain into believing the immersive experience, HMDs require extremely low latencies; an ideal experience usually has latency of 20ms or less [6].

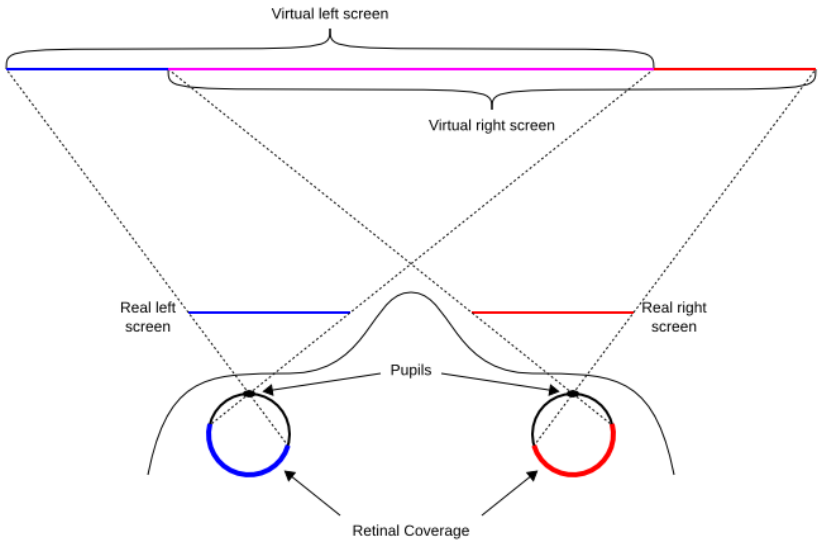

Figure 4.2: Diagram of hypothetical HMD for calibration purposes. The HMD consists of two small real screens mounted directly in front of the user's eyes, and through using optics they create a larger virtual screen at a longer distance away [7].

It is not enough to boast a large size screen and sit very close to get an immersive experience. A single large rectangular screen does not fill a person's entire field of vision. In order to stretch the image to an immersive experience, HMDs utilize optics. Research by the University of South Carolina indicated that virtual reality through HMDs could be achieved if a field of view (FoV) between 90° and 100° was used – Oculus DK2 uses a FoV of 100°. To achieve this, the HMD projects a flat image through a lens, changing it into something that fills the entire field of vision. Since HMDs use a binocular FoV, the image that a person perceives is an overlap of the two virtual images (Figure 4.2). Quality of lenses is also of direct importance to

creating a credible virtual experience, as poor quality lenses will reflect in bad image quality, low clarity, and unwanted distortion.

The final component of HMD that this thesis is going to discuss is head tracking. For HMDs to create visual immersion in a virtual world, they need to record head position in space and transform that into other data using head tracking technology. Advances in smartphone technology have enabled the production of multi-axis accelerometers, devices that can measure acceleration in multiple directions as a vector quantity. These chips, in combination with infrared cameras tracking markings on the HMD, relay data to the computer which can then calculate proper visual presentation based on user head orientation. Some HMDs, like those that utilize smartphones, can only track the direction in which the user is looking; however, a few HMDs, such as the Oculus DK2, have the ability to track along another axis, allowing users to lean in to look more closely at visual surroundings, a key component in full immersion [5].

4.3 Unreal Engine 4

When development on the project began, the immediate decision to make was a choice of game engine. Clearly writing an engine from scratch would be completely beyond the scope of this project, so research was done into current viable options. Some of the leading engines to consider included Unity, Unreal Engine 4, and CryEngine, which all featured Oculus Rift support.

All engines provide great features at no cost, so instead of deciding based on the features themselves, the choice was made based on end product comparison. Unity has historically been known for independent or lesser known developers, with titles like *Kerbal Space Program* and *Rust*. Unreal Engine 4 and CryEngine, on the other hand, are used in more professional

environments and AAA titles (games with highest development budgets and promotion). Unreal Engine 4 has produced *Gears of War 4* and *Street Fighter V*, with much anticipated titles *Final Fantasy VII Remake* and *Kingdom Hearts III*, set for release in late 2017. CryEngine, which is currently at release 5.3 currently boasts games such as *Evolve*, the *Crysis* franchise (well known for their high technical requirements), and *Far Cry*. Between these choices, the Unreal Engine products made the most compelling argument, and one of the goals of the project was to research game development from an industry perspective, so the choice was confirmed.

Unreal Engine 4 is the fourth generation of game engines from developer Epic Games. Beginning with Unreal Engine 1 releasing in 1998, the most recent release 4.15 was made available on 15 February 2017. Unreal Engine is a complete toolkit for building applications in the current age of gaming, with capability to produce a range of products from 2D mobile games to console blockbusters. The framework offers a comprehensive layer of abstraction known as Blueprints, which are entirely visually driven with no necessary knowledge of underlying code. This enables authoring of level, object, and gameplay behaviors, as well as modifying user interface, and adjusting controls for people less experienced with programming. For those needing to fine tune underlying code, Unreal Engine offers full C++ source code access, editable within Microsoft Visual Studio.

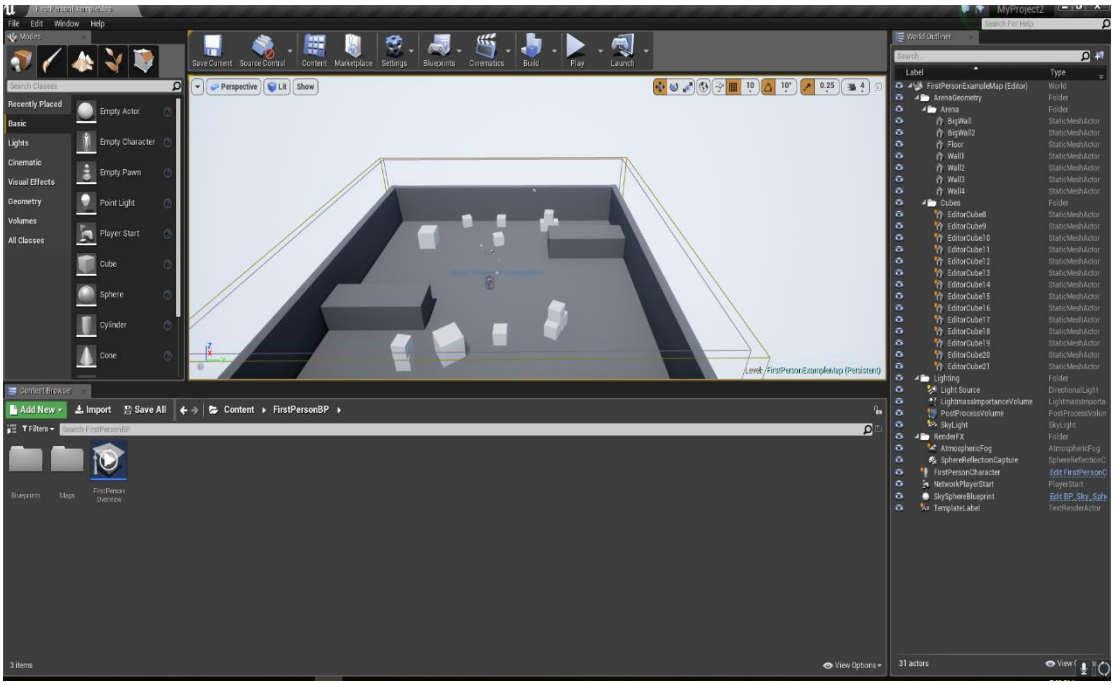

Figure 4.3: A snapshot of the Unreal Engine UI, showcasing the default First Person Perspective skeleton code. Each pane has a completely different purpose composed with submenus of submenus.

While the Unreal Engine has many benefits for simplifying game development, it can also be extremely overwhelming for an individual user. With so many available features, the UI appears crowded, with every menu leading to a submenu with even more options (Figure 4.3). Unless the user has a large monitor with a high resolution, they might have difficulty in viewing all available options, much less understanding them. Fortunately, Epic Games provides full documentation and video tutorials to accommodate the learning curve. Unreal Engine also offers in-application navigational assistance, which is highlighted upon initially opening. It is by no means designed to appeal to a minimalist designer, but once the complexity has been mastered Unreal Engine offers one of the best solutions to game development integration currently available.

One important note to make: do not be fooled by the price tag. Just because the engine is free of charge does not mean developing a game will come on the cheap. While Unreal Engine 4,

made free to everyone in early 2015, has no subscription fee or purchase price, there will not be much game development without assets. Out of the box "starter content" is available with the engine, but this is limited to about 20 textures that are not terribly interesting. To compensate for this, the developer will need to provide their own artwork, models, textures, and sounds – something most developers are not especially inclined to do. Despite potential lack of artistic vision, designing assets is a very time consuming process, time that is better spent designing the game. Knowing this shortcoming, Epic Games has provided a marketplace for designers to post their assets and developers can purchase them, which, while the prices might be viable for a company, an independent developer with next to no budget cannot afford much. This leads to developers seeking assets elsewhere including license-free asset postings on third party sites and potential handouts from artistic friends. This is no fault of the engine itself by any means, but rather a testament to the capital investment required to produce quality games.

Chapter 5

Sword Quest II: Development Log

5.1 Initial Research and Brainstorming

After posing the challenge of developing a narrative-driven VR game, I first questioned the narrative, because, as I discussed in Chapter 3, to have a strong narrative-driven game, the narrative needs to be the primary focus in game development, and nothing else. I could not begin development until I had a plan for the central story. Where the first game left off on a cliffhanger – the protagonist, having just been taught the ways of the world, was left holding his dying father, after the Baron had ransacked his home and taken his mother captive – *Sword Quest II* would pick up with the protagonist leading the remainder of his father's forces to rescue his mother and exact vengeance in his father's honor. Through the course of the main storyline, I wanted to incorporate smaller quests, where the hotheaded protagonist would aid in the personal welfare of his soldiers, humbling himself and learning the weight of the responsibility he must carry. With the scope of the project having such a small timeline, I would not have sufficient time to develop a complex narrative, so these ideas would need to be expressed in a simplistic manner and act as a proof of concept.

Coming from my previous game, a text-based adventure written in Java for play in the Command Prompt/Terminal, I knew I wanted to do something more graphically intensive. I wanted to use a fully developed gaming engine this time around, even if I would have to venture

into some graphic design territory. As previously acknowledged in Chapter 2, the three primary engines I had to choose between included Unity, CryEngine, and Unreal Engine – each had proven themselves to produce quality games; furthermore, each was free to use and had large amounts of support and learning services. From some brief research on each, I determined that Unreal Engine was my best choice, as the resultant products it boasted were well-known to me, and the general overview was that the Unreal Engine developer's environment facilitated faster production due to a higher visual orientation. Where Unity utilized an intensive C# programming structure, Unreal uses a C++ backend, but nearly everything that can be written with a C++ class can be written in an Unreal Blueprint, a visual programming tool. For more discussion on the Unreal Engine development environment, refer to Chapter 2.

5.2 Design Process

Before formally beginning any development on the game, I wanted to have a clear plan in mind. From some initial research into game design, which in itself is an intensive field and highly sought out profession, I learned that the key to any successful game is a rigorous design, typically in the form of a living document called a game design document (GDD). To replicate this design process, I sat down to fill out a GDD of my own; thankfully, I found a template on a Unity forum written by Alec Markarian and Benjamin Stanley that was robust and available for reuse, included in Appendix A. Completing this design document was an arduous task on its own, as the template was extremely thorough, and in some cases redundant, but the intention behind this was to produce quality work. Admittedly, while filling out this document I was extremely overwhelmed and hit a significant creative block, which was quite discouraging. The template desired a detailed account of story, gameplay, assets, mechanics, etc., and I simply did

not have that all together yet. Especially since, considering that I was using an engine that I was completely unfamiliar with and had a limited timescale, I had no way to judge what would be feasible proposal. Because of this, I quickly learned why the GDD was considered a living document; the GDD is consistently subject to change throughout the entire lifespan of the project. As I became more familiar with Unreal, and my creativity expanded, I added more and more to the document, even up until the final development days. It is important to note that as a one-man band game developer, I had complete authoritative power over the design, and I could adjust that design based on my opinion of feasibility. In a larger team environment, this would not be the case, and I could see how this would lead to tension and confrontation if not handled reasonably.

How I initially documented the game design was fairly simplistic. I wanted to have at least one boss battle with Baron von Drasyl, to conclude the narrative of the story. This was the end goal, so I needed to work backwards. I would need a combat system to facilitate the fight between the main characters. A simple combat system would do for this game, just melee combat with swords, nothing like ranged or projectile attacks. I also did not want to worry about blocking attacks or armor protection. Clearly, the fight would need a setting, but I did not want to start the player in the end area of the game, so I wanted there to be a minimum of two levels (willing the development capability), the final level being the Baron's castle. As to allow for a narrative flow, I would need some kind of basic dialogue system, but since I was unsure of my abilities with the engine, I did not specify a need for subtitles or focused cutscenes, although they would be desirable. Then I just needed some kind of objective for the main area, and, with my knowledge from typical RPGs, I wanted to implement some type of gathering skill (e.g. fishing

to gather food to restore health lost from combat). With this primitive outline in mind, I decided to postpone further design documentation until I understood the scope of the project.

5.3 The Animation Process: Designing the Protagonist

Intuitively, one might think that designing the protagonist for a first-person game would be an arbitrary process; on the contrary, I found it to be nearly the most frustrating part. I set this as my first task, because with a character created I could then examine other elements of my game more accurately within the world. For a first-person game, I began searching for a free pair of arms, which could be animated for my game's purposes. This proved to be severely more difficult than I had originally intended, as for some reason arms were a scarce resource, unless I wanted to pay the $50 price tag on the Unreal Marketplace, which was not a viable option over the course of the project. So, I finally decided to try my hand at making my own. Through a significant amount of research, I found an arsenal of graphics programs, which I could use to achieve my purpose – to my dismay there was no singular program for this. And thus, I had to learn a total of three different graphical programs, each completely distinct from the others.

The first graphical program I used was called MakeHuman, which is an open source 3D computer graphics program. Its sole purpose is to produce photorealistic humanoid figures, which can be used in other 3D graphic environments. Admittedly, this was the easiest of the three programs to use, as it had the most user-friendly interface, and my primary concern was just the arms. Furthermore, I was actually somewhat familiar with designing humanoid figures in this style of interface, thanks to my gaming experience. The interface of MakeHuman was very similar to that of *Fallout 4*, *Mass Effect 3*, and *The Elder Scrolls V: Skyrim*, adjusting many minute details to achieve a desired visual effect. A comparison of these interfaces can be seen in

Figure 5.1. MakeHuman also provided a rigging feature that was applied to the humanoid model. In computer animation, the term “rigging” means to create a skeleton for a 3D model so that it can move and be posed for animation. MakeHuman’s rigging functionality was quite simple, since all humanoids essentially have the same skeleton.

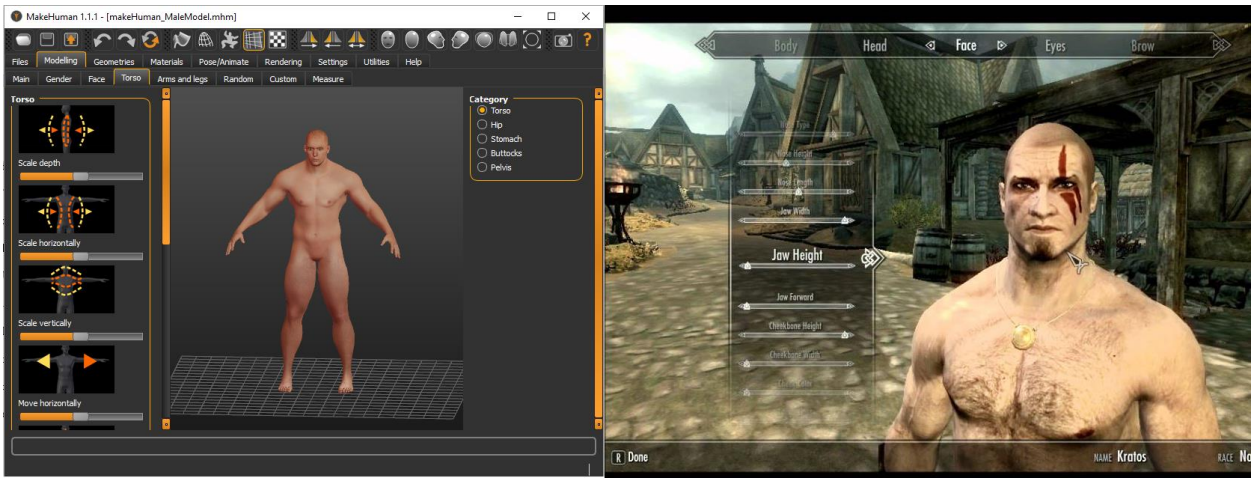

Figure 5.1: A comparison of the MakeHuman design interface and *The Elder Scrolls V: Skyrim* character customization screen. Similarities between the two allowed for easy use.

Now that I had a workable humanoid model, I needed to cut away the unimportant parts so that only the arms would be rendered in game; for this I used the open-source 3D computer graphics software Blender, pictured in Figure 5.2. Unlike MakeHuman, Blender is a multi-purpose utility which can be used for everything from polygonal modeling to animating, all on every type of graphical resource. The interface was nightmare for me, having never seriously used a professional computer graphics program, and, on top of the off-putting UI, the worst part of learning the program was navigating the 3D window. There is nothing intuitive about the navigation; no combination of mouse clicks/scrolls gets you where you want to go. It got to the point where I translated the model to the coordinates of my viewing window just to center the view. The only process I used Blender for was stripping the model for the humanoid down to just a pair of arms, but that was still too much. I wish I had been able to find a tutorial that could

explain even window navigation to me, but all the tutorials I found were completely irrelevant to my needs. I am sure the program is amazing when trained to use it properly.

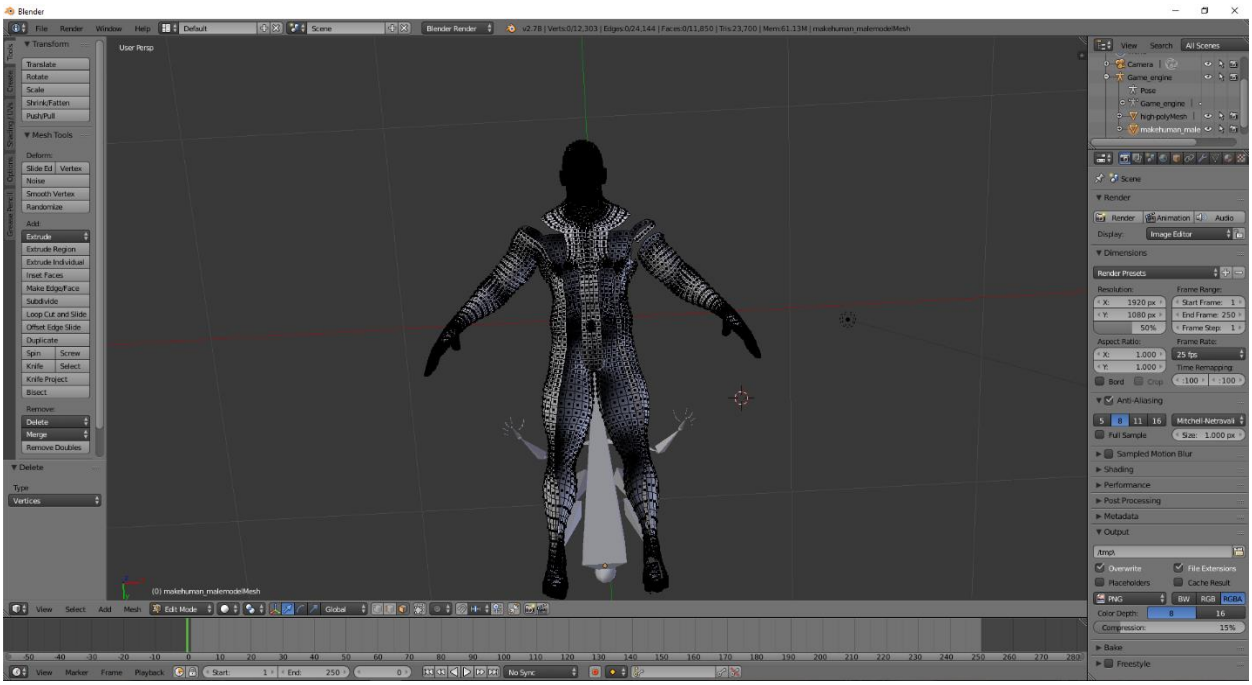

Figure 5.2: The Blender graphics program, during the process of deleting polygons to reduce the model down to a pair of arms.

Finally came the point where I had the pair of rigged arms I had been seeking for so long, and now it was time to animate them for some specific instances. For this I used Autodesk Maya, pictured in Figure 5.3, the first graphics program I needed to use that was not free and open source. Thankfully, they offered a student license that gave me use of the program free of charge. Coming off of Blender this was a welcome change that, while still not intuitive for a non-graphic designer, had plenty of documentation and support that made learning it much simpler – I was thankful to have full control of window navigation again. I specifically used Maya for animation, although it has plenty of other uses as well, similar to Blender. Animation was simple; all I had to do was move the rigged mesh to certain poses and capture keyframes. Then, Maya would calculate the intermediate stages and create smooth motion, which I could import directly into

Unreal Engine 4. For the arms, this involved animations for an idle state, a running state, and some attack animations. Once I had an understanding of the program the animations were as simple as knowing what I wanted them to look like.

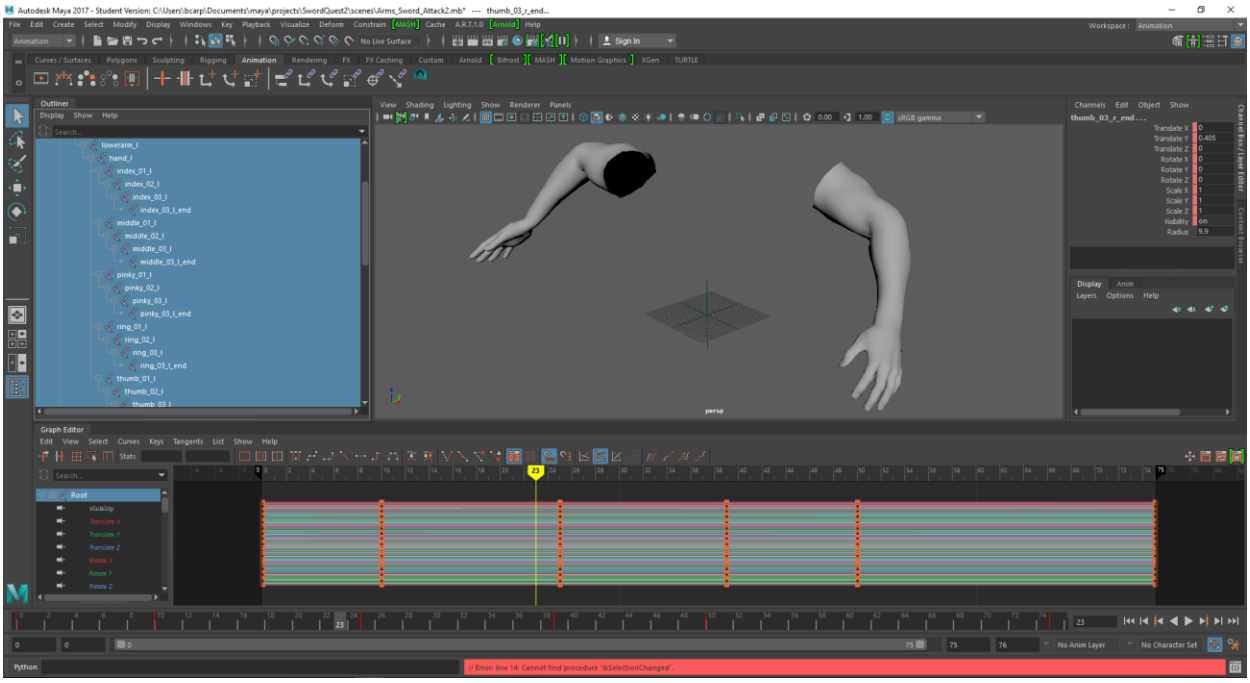

Figure 5.3: The Maya graphics program, used for animation of 3D models. The figure shows the process of making an attack animation. Every bone must be placed distinctly for poses.

5.4 Acquiring Other Assets

At this point I had now experienced the full process of making animated 3D models (i.e. modeling, rigging, and animation), which was extremely time consuming and frustrating, and I could not afford to spend this much time with each and every animated asset, or else I would not have time to actually make the game itself. I needed to find a better solution, one that would not be a large investment of my time. There are two solutions at this point: purchase assets on the Unreal Marketplace or find license free assets online. With the former solution, the availability and quality of assets was a lot higher, but I could not feasibly afford the prices as a one-man

developer student planning on no resale, so my only option was to find assets wherever I could get them for free. Many, many Google searches later I had very little to show; what I did find was not always relevant. Finally, I came across a website called Mixamo. Mixamo is an online 3D character animation platform, which I could quickly and easily integrate into my Unreal Project. Recently partnered with Adobe, the Mixamo service is currently available royalty free for all customers with an Adobe ID, which is free to make as well. These assets are available to be used in commercial products, but that is of no consequence to me, having no plans to commercialize this game. Using this service I was able to acquire model assets for all of my other animated characters; whatever I needed I found some variety of model that could suit my purchase. The platform also had many free animations which could be applied to all characters available through their site, because they all use the same rig (bone) structure.

My last, consistent source for assets came from those generous enough to make tutorials and provide example assets. Between Epic Games and YouTube tutorials, many people have uploaded accommodating assets, which are under a free education license. There are two tutorial series which I followed and borrowed assets heavily from: YouTube users UnrealGaimeDev (quest and dialogue) and Moize Opel (combat). Every single asset that I used is documented in Appendix B, the Asset Attribution List, including single assets that I found from random Google searching.

5.5 Landscape Generation: Creating the Forest Level

For the starting level of *Sword Quest II*, I decided to create a rudimentary forest area from the ground up. To accomplish this, I used Unreal Engine's Landscape feature, a simple tool for building terrain and molding it to the desired effect. After generating a flat square of space, I

applied a material that I created through blending three separate materials, provided in Unreal Engine's starter content: dirt, rock, and grass. Once I had a textured landscape, I used the tool's features of sculpt, smooth, sharpen, and simulate erosion to produce a landscape that was suitable to hold my level, a snapshot of a version of this landscape can be seen in Figure 5.4. On top of this platform I used the Foliage tool to build a forest of trees and place grass and flowers. Using the Foliage tool allows these static meshes to be rendered through hardware instancing, which means that several copies of the same mesh can be made with one draw call, decreasing load of drawing. To fill in the pond which is at the base of the mountain, I placed a floor static mesh across the surface of the pond and textured it with a water shader, turning off collisions to allow the player to fall through the "floor."

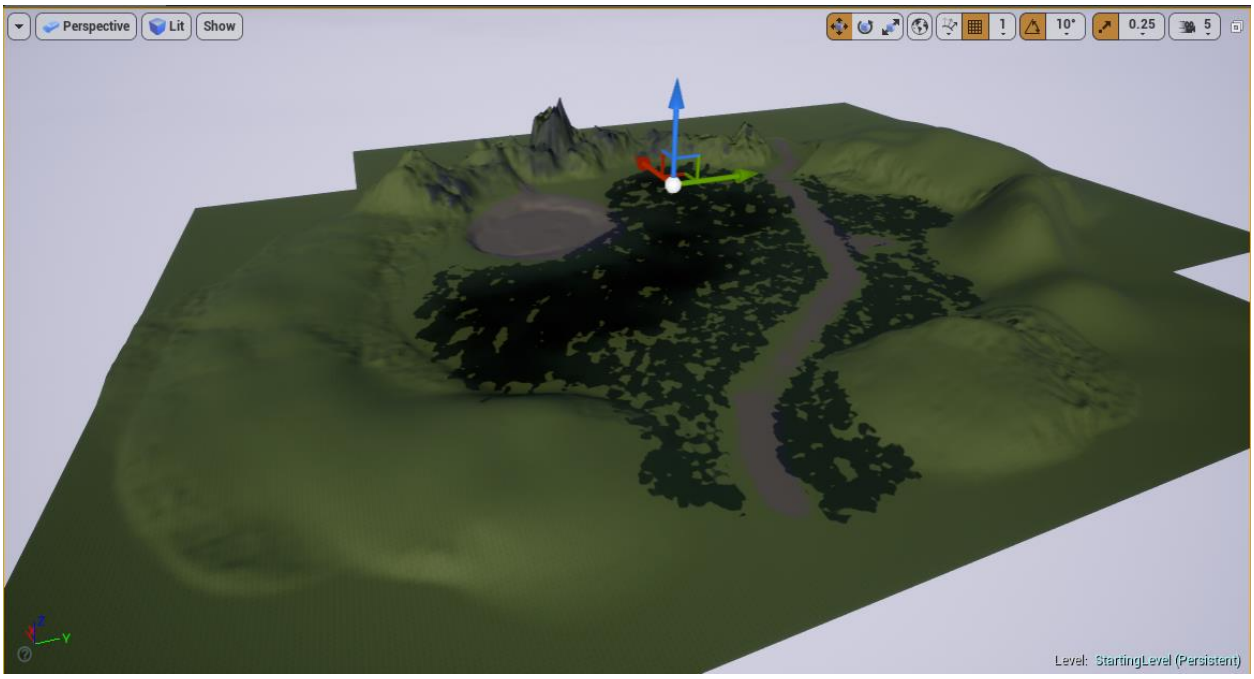

Figure 5.4: The landscape for the forest level after it has been subjected to the Landscape tool.

5.6 Game Mechanic Systems

The heart and soul of all programming for this project was done in developing robust systems which could be easily extended and reused. This made some of the development discouraging to demonstrate, because most actual programming work was done in making invisible systems to manage actions, and not flashy visuals. These main systems include the combat, quest, and dialogue systems, and every system was written entirely in Unreal Blueprints, allowing them to be understood by nonprogrammers.

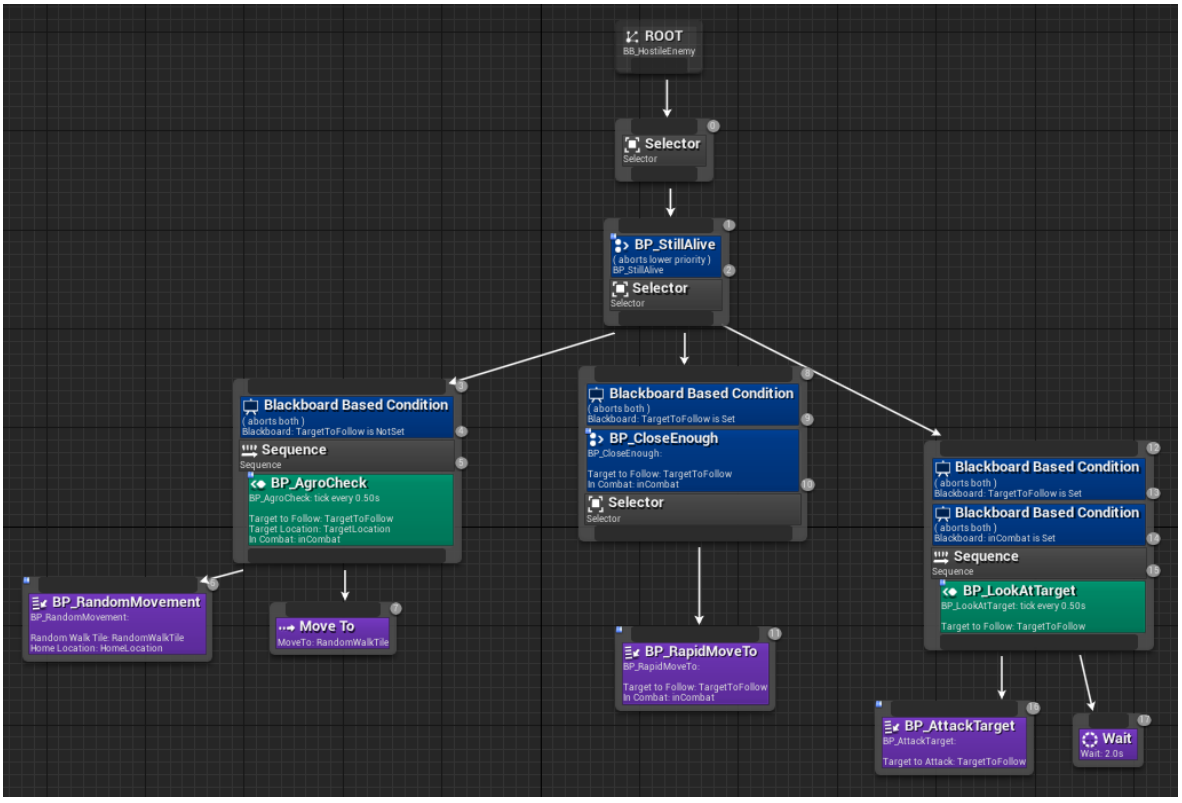

Figure 5.5: The BehaviorTree for my HostileEnemy AI Controller in the combat system. While the AI is alive it alternates between 3 different states: nonaggressive random movement, running to a player when a target is found, and then attacking that player.

As an example, I will examine some components of the combat system. The combat system relied heavily on artificial intelligence (AI) built with blackboards and behavior trees. Blackboards are essentially a knowledge base of the AI shared between different specialist functions. Behavior trees define a specific plan execution, ordering complex tasks composed of

simple tasks, with logic determining if that task should be executed. A sample behavior tree for my HostileEnemy AI Controller is shown in Figure 5.5. Once one of these has been implemented, it can easily be applied to any character, and they will observe this behavior pattern. The specifics of combats (e.g. attack animations, health, damage range, etc.) are defined on that Character Blueprint (e.g. the goblin blueprint). Collision detection drives the combat system; damage is applied when the collision capsule that encapsulates a skeletal mesh overlaps another character. These are event-driven and the engine itself notifies a character when there is an overlap on one of its collision capsules.

5.7 Sound Assets

Visuals were not the only assets I needed to find; I also needed sound assets for purposes like ambient noise, sound effects, and character dialogue. The first two types I had a good bit of luck finding randomly through research. YouTube was a great source of finding royalty free sounds that I could use for my game. These included several sound assets such as sword clashes, goblin screams, birds chirping, and footsteps, which were all easy enough to convert to WAV files and drop directly into the Unreal Engine. All of these assets are accredited in Appendix B. For character dialogue, I would need to record voice actors, and I was fortunate enough to have friends that were willing to volunteer their time and voices to my project. To record the audio, I used my Logitech G930 headset, and to edit and mix the audio I used Audacity, a free open source digital audio editor and recording software. The program was simple and user friendly such that, as long as you know what you want to do, you can easily accomplish it. Using Audacity, I edited the clips to remove excess audio from recordings and added effects such as pitch change and fades to make audio sound more natural.

Chapter 6

Conclusion

Virtual reality is a newly forming medium that has a strong pioneering presence in the video game industry. Video games are inherently unique because of their ability to provide an interactive storytelling experience. VR video games are currently in an infantile and primitive state, being based merely on spectacle, and offer little narrative experience, similar to the birth of motion pictures. They will eventually evolve to include strong narrative experiences like their flat screen predecessors, but the current techniques for storytelling in video games will not transition to a VR environment without adaptation.

As I have exemplified through the development of *Sword Quest II* on Unreal Engine, creating a narrative video game in VR has strong limitations which must be addressed in some fashion. To overcome the limitation of motion restriction from motion sensing technology, I chose to forego use of Leap Motion technology in favor of traditional keyboard and mouse input. To be consistent with an immersive experience, I chose to present the game from a first-person perspective. The game is presented in real-time, with the only lapse in time being level transition; this could have been overcome by connecting the levels physically with an undetectable load cycle. Many traditional UI features were discarded as they were not feasible in a VR setting, and those that were not discarded were added to the world as 3D widgets, although

all could not be narratively relevant. To include subtitles, I presented them as text within the world, breaking immersion, but it was a tradeoff I was willing to make.

This product is by no means a perfect example of what a narrative VR game should be. Rather it is an experimentation to highlight the inconsistencies between flat screen gaming and VR gaming. Larger teams with more creative talent are likely to develop better solutions, but the principles and concepts still apply. The large pervasive theme is to discard the auxiliary screen widgets in favor of diegetic, narrative-relevant UI. A stronger use of conveying knowledge through story relevant and diegetic channels offers a more immersive and credible experience. The challenge is that eventually functions will not always have a diegetic solution available to them. Perhaps then, the solution should be to scale back usage and reliance on UI for a more minimalist experience. Again, it is not the goal of this thesis, to provide solutions, but to facilitate discussion of issues and experiment with possibilities.

6.1 Future Expansion

Game development is never truly finished, it merely reaches a significant version number based on public deadline. Such was the case with this thesis, and, therefore, there is still much work that could be done to further the integrity of the game. Foremost, the UI elements need to be refined for a VR setting; while experimentation was done for discussion purposes, to truly transition the remaining UI elements to a diegetic state would require a recreation of the UI base Blueprints from the ground up, conserving the design and functionality already implemented. Aside from that major goal, there are smaller changes that could enhance the state of the game. The addition of more fully dialogued quests, using the current quest infrastructure, would enhance the quality of the narrative, which is the primary focus of the game. Fleshing out the

nature of the protagonist through the actions he must take would add to the overall storytelling experience. Some fun ideas for improvement of gameplay might be: ranged combat styles, more resource/fabrication skills, an inventory system, and a follower system. But it must be known that these game augmentations should not take precedence over presenting the story, else the narrative will suffer – a pitfall that afflicts most games in industry.

6.2 Speculation on the Future of Gaming

The experiences of writing this thesis and viewing the current shortcomings of VR narrative storytelling have drastically affected the thesis writer's outlook on the future of gaming. At the beginning of the thesis, the marketing trend of the rise in VR sales of Oculus and HTC Vive, as well as Samsung VR and PlayStation VR, made it seem that VR would be the next progression in gaming. On the contrary, it seems that VR gaming will be held stagnant until the issue of movement restriction can be overcome. It is not enough to simply enlarge the room calibration until it fills the size of a warehouse; the VR experience needs a break from the physical world completely. Perhaps this breakaway can be attained once technology has a capability to communicate directly with neural impulses, intercepting the feeling and motion in the natural world and applying them instead to the virtual one. Such a technology would be similar to that of whole brain emulation or mind uploading, and, while such technology is not currently in practice, it is possible that it may emerge sooner than expected.

Appendices

Appendix A

Asset Attribution List

Asset Name	Author	Link
Basking Shark (Animated)	Patrick Petersen	https://forums.unrealengine.com/showthread.php?61541-Community-WIP-Fish-Schooling-AI-and-Assets-Package-(Open-Source)
Foliage	fighter5347	https://forums.unrealengine.com/showthread.php?59812-FREE-Foliage-Starter-Kit
Water Shader	DotCam	https://forums.unrealengine.com/showthread.php?42092-WIP-Gerstner-Based-Ocean-Shader
Footsteps Sound	SoundEffectsFactory	https://www.youtube.com/watch?v=QrwufQDsWJY
Sword Slash Sound	SoundEffectsArchive	https://www.youtube.com/watch?v=X3liPsg21Cg
Goblin Scream Sound	Sound Effects Public Domain	https://www.youtube.com/watch?v=Y8bqgyR52rc
Goblin Cackle Sound	Sound Effects Public Domain	https://www.youtube.com/watch?v=vDTquxpaHDI
Quest Complete Sound	Sound Effects Public Domain	https://www.youtube.com/watch?v=rSTITV8-iuo
Sword Clash Sound	32cheeseman32	https://www.freesound.org/people/32cheeseman32/sounds/180819/
Goblin Death Sound	spookymodem	https://www.freesound.org/people/spookymodem/sounds/249813/

Bibliography

[1] Steve Denning and Laura Dietz. The Science of Storytelling. *Forbes Leadership*. Mar 2012.

[2] Kamal Sinclair. The Art of Storytelling in Gaming. *Sundance Institute*. Dec 2014.

[3] Thomas Grip. 5 Core Elements of Interactive Storytelling. *Gamasutra*. Aug 2013.

[4] Sophie Charara. The Problems Facing VR Game Designers and How to Fix Them. *Wearable*. Mar 2015.

[5] Head-mounted Displays (HMDs). *Virtual Reality Society*.

[6] Review: Oculus Rift DK2. *FlatPanelsHD*. Aug 2014.

[7] Will the Oculus Rift Make You Sick? *Doc-Ok*. Aug 2012.