

# The Call of Karen

A Major Qualifying Project  
Submitted to the faculty of  
WORCESTER POLYTECHNIC INSTITUTE  
In partial fulfillment of the requirements for the  
Degree of Bachelor of Science in  
Interactive Media and Game Development  
And for the  
Degree of Bachelor of Arts in  
Interactive Media and Game Development (Technical Art Concentration)

**Authors:**

Kate Olguin (IMGD BS)  
Thomas Tawadros (IMGD BA)  
Diana Kумыkova (IMGD/CS)  
Mikel Matticoli (IMGD/CS)

**Advisors:**

Farley Chery (IMGD)  
Dr. Gillian Smith (IMGD/CS)

**14 May 2020**

# Abstract

*The Call of Karen* is a PC simulation comedy game about a 1950s suburban housewife whose home is invaded by Cthulhu. The player controls the titular housewife, Karen, who is trapped in an unhappy marriage taking care of an ungrateful child, and must keep up appearances in the face of a home gone mad. The goal of this MQP was to create a fun, funny, and full-fledged game of high enough quality to be released on a major distribution platform such as Steam or the Epic Games Store. We also wanted to create a compelling narrative that deconstructed the nostalgia associated with the 1950s by telling a story from the often neglected and demeaned perspective of a 1950s suburban housewife.

This report describes the production process of *The Call of Karen*, including details of the evolution and completion of the game's design, art, gameplay, playtesting, and showcasing procedures. The game at the time of this report is functionally complete, and playtesting and showcasing results indicate that we did achieve our goal.

# Acknowledgments

We'd like to thank our advisors, Professors Farley Chery and Gillian Smith. Their experience, input, and feedback was invaluable to us during this process. We'd also like to thank all of our playtesters who gave us their time and useful feedback, and the people outside the team who lent their voices to the game, Fiona Doyle and Zeke Feldman. Finally, we'd like to thank reference librarian Laura Robinson, who helped us out at the start of the project.

# Table of Contents

<b>Abstract</b>	<b>2</b>
<b>Acknowledgments</b>	<b>2</b>
<b>1 Introduction</b>	<b>7</b>
<b>2 Background</b>	<b>9</b>
2.1 Representations of the 50s in Media	9
2.1.1 Suburbia: A Man's World?	9
2.1.2 Influence of J. G. Ballard	10
2.1.3 Your Friendly Neighborhood Cthulhu	12
2.1.4 Anxiety in the Americas	12
2.1.5 Genre Titles	13
2.2 Representations of Women in the 1950s	17
2.3 Representations of Cthulhu in Media	22
2.4 Humor in Games	25
<b>3 Design</b>	<b>28</b>
3.1 Experience Goal	28
3.2 Comparables	28
3.3 Story	30
3.3.1 Story Summary	30
3.3.2 Characters	31
3.3.3 Karen	31
3.3.4 Husband	32
3.3.5 Child	33
3.3.6 Radio	33
3.3.7 Susan Jones	34
3.4 Mechanics	34
3.4.1 The Idea of Horror	35
3.5 Tasks	35
3.5.1 Smoothie/Breakfast	36
3.5.2 Lunch	37
3.5.3 Cleaning Up	37

3.5.4 Vacuuming	38
3.5.5 Cooking Meatloaf	39
3.5.6 Setting the Table/Cleaning the Table	41
3.6 Flavor Tasks	42
3.6.1 Ordering Silver Bullets	42
3.6.2 Receiving and Using an Ancient Book	43
3.6.3 Modifying Vacuum with Bullets	44
3.6.4 Final Vacuum Modification	44
3.6.5 Sealing Away Cthulhu	44
3.6.6 Revving the Cthulhu Vacuum	45
3.7 Days	45
3.8 Progression and Escalation	47
3.8.1 Escalations by Task	49
3.8.2 Breakfast	49
3.8.3 Cleaning Up	50
3.8.4 Vacuuming	51
3.8.5 Dinner Dash	51
3.9 Other Methods of Conveying Cthulhu	51
3.10 Greyboxing & Paper Prototyping	52
3.11 House layout	53
<b>4 Technical Implementation</b>	<b>55</b>
4.1 Key Software/Tools	55
4.2 Why We Chose PC Over VR	55
4.3 Engine: Why We Chose Unreal	56
4.4 Version Control	57
4.4.1 Why we chose Git and not Perforce	57
4.4.2 Workflow	58
4.5 Game Structure	59
4.5.1 Actors, Components and Triggers - Oh my	59
4.5.2 High-Level Documentation	60
4.5.3 TaskManager	61
4.5.4 C++ Integration and BP_GameManager	62
4.5.5 Data Storage: Data Tables and Dictionaries	64

4.6 Gameplay: How was the game made?	65
4.6.1 Day Night Cycle and Task Loading	65
4.6.2 Escalations and AI	65
4.6.3 Movement System	67
4.6.4 User Interface	67
4.7 Implementation Difficulties	70
4.7.1 Blueprints vs C++	70
4.7.2 Git Merge Data Loss	70
4.7.3 Semi-Random Data Overwrites and Deletion	71
<b>5 Art</b>	<b>72</b>
5.1 Inspiration and Concept	72
5.2 Colors	77
5.3 Asset Creation	78
5.3.1 Modeling	78
5.3.2 Texturing	79
5.4 Technical Art	81
5.4.1 Shaders	81
5.4.2 Rigging	84
5.4.3 Particle Effects	85
5.4.4 Cosmetic Blueprints	85
5.4.5 Lighting	87
5.5 User Interface	90
<b>6 Sound Design</b>	<b>92</b>
6.1 Voiceover	92
6.2 Music	93
6.3 Sound Effects	94
6.4 Implementation	95
<b>7 Playtesting and Showcasing</b>	<b>96</b>
7.1 Institutional Review Board Approval	96
7.2 Playtesting Protocol	97
7.3 Alphafest	97
7.4 Playtesting Session 1	103
7.5 The MassDiGI Game Challenge	107

7.6 Playtesting Session 2	109
7.7 PAX East	114
7.8 COVID-19 and Playtesting	114
<b>8 Conclusion and Recommendations</b>	<b>116</b>
8.1 Scope	116
8.2 Team Dynamics	116
8.3 Team Structure	117
8.4 Project Management	117
8.5 Future Work	118
8.6 What Went Wrong	118
8.7 What Went Right	118
<b>9 References</b>	<b>120</b>
<b>10 Appendices</b>	<b>126</b>
Appendix A: Final Game Progression List	126
Appendix B: IRB Exemption Example Survey	128
Appendix C: Alphafest Survey	129
Appendix D: Playtesting Survey 1	132
Appendix E: Playtesting Survey 2	135
Appendix F: MassDiGI Game Challenge Slides	138

# 1 Introduction

This Major Qualifying Project (MQP) was completed by four Worcester Polytechnic Institute (IMGD) students. The project's artists were Kate Olguin, majoring in Interactive Media and Game Development (IMGD), and Thomas Tawadros, majoring in Interactive Media and Game Development with a concentration in Technical Art (IMGD BA). Kate Olguin also served as the project's producer. The project's programmers were Diana Kumykova and Mikel Matticoli, both majoring in Computer Science and Interactive Media and Game Development (CS/IMGD).

*The Call of Karen* is a PC simulation comedy game developed in Unreal Engine 4 during the 2019-2020 school year at Worcester Polytechnic Institute. The game is about a 1950s suburban housewife whose home is invaded by Cthulhu. The player controls the titular housewife, Karen, who is trapped in an unhappy marriage taking care of an ungrateful child, and must keep up appearances in the face of a home gone mad. The player completes tasks stereotypically associated with 1950s suburban housewife life. These tasks quickly spiral out of control, resulting in gameplay like *Cooking Simulator*, but with more eldritch horror.

Our main goal with *The Call of Karen* was to make a full-fledged game that would inspire humor and keep players consistently engaged throughout its playtime. We also wanted this game to be of a high enough quality to be released on a major distribution platform like Steam or the Epic Games Store. Worth noting is that going through the actual process and steps of release is outside of the scope of this MQP (though it is something we plan on doing). However, for this project, we wanted to create a product that was of high enough quality where we would be able to release it if we wanted to.

As students, it's often hard to find the time to get a big project out the door. Additionally, with the way the term system works at WPI, we don't get too many opportunities to go through the entirety of the game development process under academic supervision. Professors and outside industry experts alike have told us that there is great value in bringing a project from start to finish, so we set out to make a game that would allow us to go through as much of the game development process as possible.

Another key goal with *The Call of Karen* was to tell a compelling story that deconstructed the nostalgia associated with the 1950s through the eyes of a period accurate suburban housewife. The 1950s is often portrayed through rose-tinted glasses in media (Dwyer, 2015), glasses that conveniently ignore the fact that this era was a terrible time to be a woman. Additionally, suburban housewives and their work is consistently demeaned and subject to significant negative stereotyping (Neuhaus, 2011). We thought that there was value in exploring this neglected perspective in an accurate way to provide social commentary on the patriarchal society of the 1950s.

In chapter 2, we detail our research findings that allowed us to accurately and thoughtfully portray and deconstruct the 1950s. Chapter 3 discusses the overall design of the game, including the influence our findings in chapter 2 had on our narrative and game structure. While chapters 2 and 3 discuss the *why* of our game, chapters 4, 5, and 6 discuss the *how*, exploring the technical, artistic, and sound design choices made throughout our development process. Chapter 7 details our results and experiences with playtesting and showcasing our game, and Chapter 8 concludes our paper with an analysis of some of the aspects of development that led us to success that weren't covered in previous chapters, as well as what

went wrong and what went right.

By keeping a tight hold on scope and maintaining consistent, close communication, *The Call of Karen* enjoyed a fairly smooth production process. We believe we were able to go through the majority of the game development process. Our observations and data from playtesting and showcasing detailed in chapter 7 indicate that we met our experience goal and our game is of high enough quality for release. As a result, we'd say this MQP was a success.



## 2 Background

*The Call of Karen* is one part 1950s housewife simulator, one part Lovecraftian horror experience. With the game being set in a historical time period, we used a variety of sources to better understand and recreate the world of the 1950s. Of particular interest was the life and struggles of women from the era, as well as how they are represented today. Intersecting this historical setting is H. P. Lovecraft's Cthulhu Mythos (Loucks, 2012). The work of Lovecraft in general, and the Cthulhu Mythos in particular, have become immortalized in popular culture. In recent years one can find everything from novel series to video games themed on Lovecraft's work (Ghodrati, 2013). Research into this area guided creative decisions around task escalation and enemy design. Finally, we researched humor in video games, how it functions, and how best to use it. *The Call of Karen* straddles a line between self-aware comedy and cosmic horror, so getting the balance right was key to creating an enjoyable experience.

### 2.1 Representations of the 50s in Media

The "suburbia" of 1950s America has become immortalized in popular culture. When our team first began to brainstorm what the 1950s means today, we came up with images of well-to-do households furnished in gaudy colors, housewives in polkadot skirts, and the static sound of a radio announcer rattling off in the background. The word "Hoover" entered the American lexicon as a reference to the eminent vacuum brand (Blitz, 2016). The Red Scare, and the Space Race weighed heavy on the minds of Americans nationwide (History.com Editors, 2020). Much of our current idea of 50s suburbia comes from a history of representation in media including books, television, and film (Huq, 2013). Looking back at these representations gave our team some guidance on how to go about building a game that takes place during that time.

#### 2.1.1 Suburbia: A Man's World?

Defining the term suburbia is not entirely straightforward. What quickly comes to mind, involved with the images above, are suburban neighborhoods: row upon row of neat, cookie cutter houses lined with white picket fences. This "physical" setting is the most concrete manifestation of suburbia. Location, however, is only a piece of the whole picture. Huq (2013) considers suburbia a "sociocultural category," in this case, of the American experience. The social, economic and political background of the day are all important in painting a complete picture. While it was not typically the subject, 1950s suburbia provided the background for, and subsequently became defined by, popular works of fiction including books, tv shows, and movies. The earliest works come from the 50s themselves. Preeminent among these is Sloan Wilson's novel *The Man in the Grey Flannel Suit* (1955).

Both the book and subsequent movie of the same name were popular hits. The novel hit a “cultural nerve” (George 2016). Through the character of Tom Rath, it explored the conflict between conformity and individualism in America’s working man. It put the idea of the “straight suburban mindset” into public consciousness, but more generally, it solidified a theme of



Poster for *The Man in the Gray Flannel Suit* (1956)

dissatisfaction present in working-class suburban life. This story focuses on the male perspective of suburban family life. However, Tom’s wife Betsy gives an impression of the female perspective. Dissatisfaction with the current state of affairs and her husband’s lack of ambition illustrate that everything is not smooth sailing in a superficially picturesque marriage. This aspect of the nuclear family was something we sought to recreate when portraying the time period from the perspective of a housewife.

In *The Call of Karen*, we wanted to accentuate the dissatisfaction of the suburban status quo as seen from the female perspective. Throughout the game, the player experiences Karen’s frustration with her family. But the dismissal with which her husband and child treat her, while real, never leaves the house. In this way, it is implied, the appearance of an ideal family is kept afloat, and Karen herself bears the brunt of keeping up this appearance. The catchphrase of the game illustrates this sense of obligation: “When horrors come to town, your property value will *not* go down.”

### 2.1.2 Influence of J. G. Ballard

Despite, or perhaps due to, its ordinary nature, suburbia is often the setting for unlikely events. With the threat of Soviet attack weighing on the collective American conscience, the suburbs were often painted as a place to escape to in the event of nuclear armageddon. The

cold war “engendered a ‘bunker’ mentality – seen literally in the 2009 film of Christopher Isherwood’s 1964 set *A Single Man* where the main character George’s neighbours are building bomb shelters” (Huq, 2013).

English novelist J. G. Ballard is renowned for his depictions of an apocalyptic post-war landscape. Beginning in the 50s, Ballard wrote stories prominently featuring “20th-century middle-class people devolving into savagery,” dystopian landscapes, and dehumanized sex and technology (Encyclopedia Britannica 2011). Such settings and topic contrasted heavily with the straight-laced ethics of 1950s suburbia, but contribute heavily to subsequent public imagination of the era. He foresaw a media-saturated society, a population oblivious to the struggles of the real world. Again, we see a “worst-case scenario” attitude toward the future. On the surface, this strongly juxtaposes the social conservatism of the day, which mandated an unflinching belief in the unstained superiority and excellence of the Western Bloc. But as time progressed, the



Poster for *It's a Wonderful Life* (1946)

psychological weight of mutually assured destruction led to a progressively bleaker outlook, both in the US and around the world. The prospect of a world torn apart by human conflict seemed, and likely was, closer than ever. A yet uninvented medium, video games, would go on to capitalize on this era of fear.

Our game plays on the Ballardian theme of apocalyptic occurrences in the proverbial backyard. The suburbs, neighborhoods born of a cookie cutter mentality, had an already low tolerance for inexplicable events, let alone an eldritch nightmare taking dominion. The juxtaposition of 1950s suburbia and Lovecraftian nightmare is amusing due to the contrast between the two topics. However, this conceit is only believable because of the sense that the tranquil suburb is exactly the place where something like this would happen.

### 2.1.3 Your Friendly Neighborhood Cthulhu

Proceeding books, film and television representations of the 1950s color our perception of what they were like. In 1946, *It's a Wonderful Life* immortalized on the silver screen the wholesome, "All-American" values that would later come to dominate the genre of "feel-good" suburban film. Ten years later the aforementioned adaptation of *The Man in the Gray Flannel Suit* showcased a more dramatic picture of a similar world. Both films portray the male perspective. Ambition and the need to succeed in work contrast with a responsibility and commitment to the family. Ultimately, upstanding morals come out on top. In contrast, Huq points out that "retro-representations of the 1950s/1960s are quite unlike the actual films from the era which tended to portray suburban perfection." Starring Toby Maguire, *Pleasantville* (1998), tells the story of two siblings that are transported into a sitcom from the 50s. An idyllic midwest town, Pleasantville epitomizes post-war suburbia. Old-fashioned American values, centered on the nuclear family, take center stage. The film turns black-and-white, but as the main characters challenge the moral framework of the town, they slowly bring color to this world. Ultimately, the film portrays the 50s as a simpler, but more restricted time, when individual self-expression was not as valued as today.

Starting in 2007, the *Mad Men* tv series explored the seamy underbelly of corporate life in the 1950s and 60s. It depicts misogyny, greed, and unrelenting power struggles that were certainly a part of life in the 50s, but a part rarely explored in movies or television from the time. Huq points out a theme in suburbia on television of 'them' and 'us'. The tv showed spacious, comfortable suburban homes, while viewers were more likely stuck in a more cramped, less idyllic reality. Again, retrospective media paints a different picture.

Superficially, *The Call of Karen* is a kind of *Pleasantville*. At a glance, the game is colorful, bright, and lighthearted. The game does not take itself too seriously, and thus the player experience's Karen's daily routine at a distance. It does not hit close enough "to home" that they feel the game is describing physical reality. However, the psychological and emotional reality of Karen's life, like those of the citizens of *Pleasantville*, is one of monotony and stagnation. Only when Toby Maguire (literally) steps into the picture does anything significantly change in the town of *Pleasantville*, and much the same can be said about our game's rendition of Cthulhu. In a way, the Great Old One was the catalyst that Karen needed to break through the prison of her stay-at-home life. Likewise, at the end of the game, she wields her hallowed vacuum cleaner, revving it in response to her husband's unwitting selfishness.

### 2.1.4 Anxiety in the Americas

Perhaps a reason that games have used the 50s as a setting is the perceived prosperity, and vulnerability of the time. The U.S. experienced an economic boom after the second World War. According to Moffatt (2020), the period from 1920 to 1960 saw the nation's gross national product rise from \$200 million to over \$500 million. The end of the war effort meant industry could finally relieve pent-up consumer demand. American markets and goods penetrated foreign countries. The "baby boom" was steadily increasing the number of consumers with disposable income. The rest of the world was struggling to recover from the aftermath of the war. Americans, largely untouched by Axis firepower were, for all intents and purposes, on top of the world. Progress and prosperity became national ideals. This aura of economic growth

fueled a particular brand of American exceptionalism. This became increasingly obvious during the Cold War.

But in 1957, much to their chagrin, Americans got news that Russia had become the first nation to put an artificial satellite into space. “Just when [they] were feeling self-confident and optimistic about the future, along came the crude, kerosene-powered Sputnik launch...America was agog and unnerved” (Dickson 2007). With exceptionalism came anxiety. The U.S. saddled itself with the responsibility of beating Russia in the space race. At the same time, mutually assured destruction via atomic warfare circulated in the public imagination. From across the globe, the world’s two largest superpowers were aiming at each other with hair triggers. A nuclear apocalypse did not seem like such an unlikely future.

In *The Call of Karen*, the radio announcer acts as a conduit for the anxiety of the outside world. Karen, and thus the player, never leave the house. Likewise they must rely on the radio for news of current events. With his talk of freak storms, communist sympathizers, and the stars being right, the radio announcer epitomizes the anxiety and hesitation about the outside world that was felt in the 50s. The game emphasizes this further, only letting the player out of the house when Cthulhu is at the doorstep.

### 2.1.5 Genre Titles

Four decades after the launch of Sputnik, one game developer would ask the question: what would the world look like if the 50s *did* end in nuclear apocalypse? Developed and published by Interplay Entertainment, *Fallout: A Post Nuclear Role Playing Game* released in 1997. It introduced players to a vision of an America ravaged by nuclear war. According to Hall (2018), the *Fallout* universe diverges from our own after the 1945 surrender of Japan to the Allied Powers. Technology progresses, and the world experiences a number of nuclear conflicts that eventually ravage the planet. But the game is quick to remind us that some things, like war, never change. Social and aesthetic development seem to have stopped mid-century. The game



*In-game advertisement for Vault-Tec in Fallout 4 (2015)*

takes place in the year 2161, but more resembles a twisted, nightmarish version of the 1950s. In the Fallout timeline, the USSR declined and communist China replaced it as America's rival super power. The U.S. remains in a state of cold war for over a century. Technological development, both civil and military, centers around the exploitation of atomic power. Visually, the game is "distinctly retro-futuristic." (Hall 2018). The Fallout world takes inspiration from World's Fair imagery from the 50s and 60s. In this universe, the technology of the future looks just like people in the 50s thought it would. Old fashioned diners, car chassis, and jukeboxes straight out of the 50s populate the environment. But the graphics of the original *Fallout* are limited. Most of the game is played from a zoomed-out, isometric perspective. Closeup character screens show off more of the aesthetic, but the graphic fidelity is low by today's standards.



*Screenshots from the original Fallout. Left: Isometric perspective; Right: Character interaction screen*



*Promotional image for Fallout 4 (2015)*

The original *Fallout* met critical acclaim, and since then *Fallout* has come to be a household name in the gaming community. Later games in the *Fallout* series would benefit from developments in graphics rendering technology. Starting with *Fallout 3*, developer Bethesda Softworks allowed the player to experience the game from a first-person perspective. More recently, *Fallout 4* (2015) and *Fallout 76* (2018) take advantage of modern physically-based rendering (PBR) to bring the world to life. Although the *Fallout* games are set in the near to far future, they represent the social, technological, and aesthetic sensibilities of post-war Americana.



*Promotional image for Fallout 76 (2018)*

The *Fallout* games are likely the most well known video games to use 1950s America so heavily as both set and inspiration. They certainly are not the only ones. 1950s iconography and tropes have inspired a variety of games spanning genres. Among these is the game *Destroy all Humans!* (2005), which explores an apocalypse of a different nature. Developed by Pandemic Studios and published by THQ, *Destroy all Humans!* put the fate of humanity in the hands of 2005's console gamers. Players control Crypto, a flying-saucer piloting alien bent on extracting human brains for the survival of his species. The game takes place in locations across 1950s America, from suburban *Santa Modesta* to *Capitol City*, parodies of Santa Monica and Washington D.C. respectively.

The game is self-aware and filled with dark comedy, and references to 50s pop culture abound. The first army detachment sent to intercept you can be heard chanting:

*"I don't know what I've been told, Joe MCarthy's good as gold."*



*Town of Santa Modesta in Destroy all Humans! (2005)*



*Capitol City in Destroy all Humans! (2005)*

Throughout the game, non-player characters (NPCs) theorize that Crypto is part of a communist plot to undermine the country. The ability to mind-read nearly anyone also provides an entertaining look into Pandemic's satirized version of the 50s.

The popularity of *Destroy all Humans!* Spawned a sequel, and a remake was released over the time this paper was written. The previous games showcase the myriad ways that the 1950s have been captured and represented in games over the last two decades. *The Call of*





*Mind-reading in Destroy all Humans! (2005) reveals humorous undertone*

*Karen* draws elements from all of them, weaving together a self-aware comedic narrative with high graphical fidelity. Our game stands on the shoulders of giants, but brings a fresh perspective that has been missing for quite some time.

## 2.2 Representations of Women in the 1950s

The titles mentioned above were (and continue to be) popular hits, but smaller games also show the widespread influence of 50s suburbia. Particularly relevant to *The Call of Karen* is the Kickstarter game *Aberford* (2015) from Sketchy Panda Games. Set in the titular small town, the game would feature 4 housewives as they



*Aberford (2015) Kickstarter promotional image*



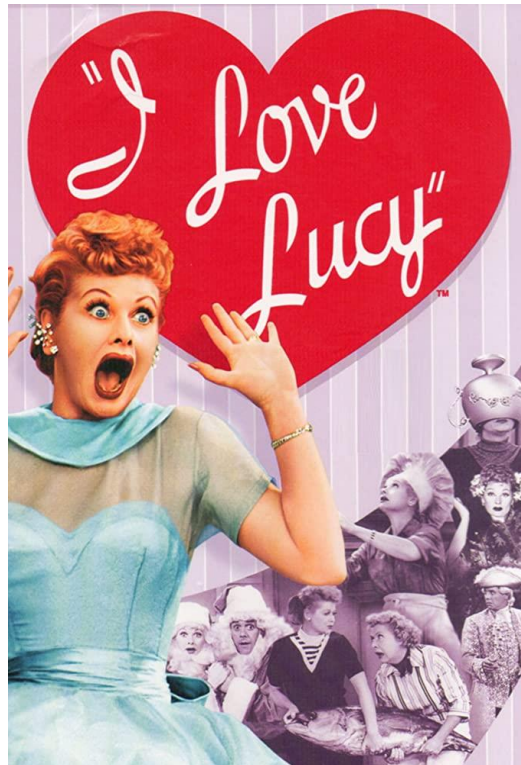
*Aberford (2015) blends 50s housewife personas with retrospective internet memes like the one above*

battled back hordes of zombies, formerly the men of their community. The game takes cues from the popular “50s housewife” persona seen in advertisements from the era. These reinvented housewives, however, have a crucial difference: their capacity for violence. Cooking and cleaning are replaced by bludgeoning and beating. It was the video game manifestation of a trend seen on Pinterest and other social media sites. Images of housewives from advertisements like the ones above were edited and captioned, in the style of ironic memes. Most of this seems to be for the sake of comedy, but much of what makes these images funny is the contrast between the reader’s internal persona of the 50s Housewife, and the one suggested by the image. The prevailing sense one gets from the original ads of the era is one of perfection, of order, of a certain conservative mindset. A 50s housewife is always well dressed, well mannered, and knows how to cook, clean, and tend to a household. She is the symbol of well-to-do suburban life, iconic in her polka dot house-dress, baking sheet in one hand, oven mitt glued to the other. Most importantly, she is the foundation of a stable household, putting the wellbeing of her family before everything else. But the idea of an ideal housewife is just that: an ideal.

In reality, this pristine outer persona is a kind of propaganda, a hollow image of what upper middle-class Americans valued and wanted at the time. In-line with the social conservatism of the day, it eschews messy, realistic portrayals of women for an easily digestible fantasy. Who needs subtlety when nuclear weapons and communists threaten to destroy your very way of life? Memes like the one above cut rather violently across the 50s housewife persona. What if things were actually not as hunky-dory as they look? What if the weight of social conformity was driving women of the era insane within the prison of their own homes? They frame the outer veneer of perfection as completely superficial. It’s fundamental function is to hide instability and reinforce a status quo. It creates a character out of a flawed human being.

In the age of social media, we see idealized personas all the time. Instagram, Facebook, and other social media sites have made it easier than ever for bodybuilders, fitness models, and the like to portray a particular image of who they are. What’s different is that we are now beginning to realize the society-wide detriments of this practice. Indeed, research linking

internet use to declines in mental health have been surfacing since the late 90s (Pantic 2014). Social media use, especially among teenage girls, has been linked to significantly higher rates of depression. Another crucial difference is the perceived enforcer of the status-quo. At a glance, it's tempting to say that back in the 50s, social-conformity was enforced by interpersonal relationships, whereas today it is enforced by the media. In reality the distinction is not so black and white. During the 50s, television started to broadcast what the ideal family should look like. Shows like *Father Knows Best* and *I Love Lucy*, began to influence the American identity. Clearly, media contributed to a sense of shared values, influencing us much like media does today. So, on some level, media influence has been part of the social fabric since at least as early as the 50s. But apart from the iconic look, what did people from the era think about the role of women in society?



*Poster for I Love Lucy (1946)*

The 50s themselves were seen as a time when women were predestined to become homemakers. Having just come out the end of the second World War, women who had been previously involved in the manufacturing work for the military-industrial complex began to be displaced by returning GIs. Industry had evolved since the beginning of the war, but not enough that the returning workforce was completely alienated. So, women who had answered the call of Rosie the Riveter left the factories and mills. And with the baby-boom in full swing at the end of the 40s, American women were more than ever becoming homebound.

Women were also viewed as the stable backbone on which countries including post-war Britain would rebuild. "This was due in part to the significant contribution women had made to the war effort and their on-going commitment to postwar reconstruction" (Beaumont, 2016). America, being more isolated from the destruction seen in other parts of the world, likewise

viewed women as the anchor of the nuclear family in a time when the world was just beginning to recover from the largest war it had ever seen. This high regard, however, did not come without tradeoffs. Women were portrayed as wives and mothers first. “In contrast to the dynamic possibilities of women's new role in postwar society, the prevailing view of women at this time, as illustrated in popular women's magazines, was that the vast majority aspired only to marriage and motherhood” (Beaumont, 2016). Whereas women had already proven they could be valuable members of the workforce during the war effort, they were nonetheless relegated to the role of mothers and wives. Beaumont makes explicit that “Women during the 1950s were not just housewives and mothers but were also workers, campaigners, consumers, spinsters, widows, lovers, divorcees, prostitutes and citizens.” The full array of women’s experiences and aspirations during this period, however, have been largely lost to history, or at least to public imagination. What persists, however, are the colorful, iconic images found in magazines of the ideal 1950s housewife.

Many magazines and advertisements of the time sought to market home appliances. It was a time when technology for the sake of convenience was a celebration of success. American exceptionalism was entering home life in the same way it does today: consumer goods. This phenomenon alone inspired a few aspects of *The Call of Karen*, including the use of a “Hurston” vacuum, and mail ordering silver bullets through a magazine catalog.



*Our in-game catalog was based on the popular mail-order catalogs of the day.*

The 1950s was the golden era of mail-order. The Sears-Roebuck catalog, started in the late 1880s, saw explosive growth throughout the coming decades. By the 1950s, Sears had exceeded \$1 billion in sales, and “had opened more than 700 stores in the United States” (Pruitt, 2019). Sears had become a retail titan, expanding to neighboring Mexico and Canada.

Their economy of scale allowed Sears to sell everything from socks to entire homes. In the middle were appliances: dishwashers, stoves, vacuums, and the like. And who better to market such goods to than the American housewife.



1950s advertisement featuring housewife persona

1950s America was imbued with the aura of technological progress unhindered by the notion of externality. New and innovative appliances were created to help people, including the ordinary housewife, in their day-to-day duties. The perception was that such technology would make life easier and more free, and to some extent it was true. Anyone who has ever washed clothes by hand can attest to that. But an invisible cost of all this innovation was that it commodified the labor of the women who used such appliances. Appliance advertisements like those pictured above deepened the already strong gender role assigned to women during the 50s. With the Sears and other catalogs now immensely popular, they then spread this housewife persona far and wide through the use of advertising.

The commodification of the homemaker also served to isolate them. Discouraged from seeking employment, housewives spent their days slaving away at the same repetitive tasks in service of what we can only hope was a loving family. But apart from the few ideal cases, works like *Grey Flannel Suit* (and general human experience) show that families rarely function perfectly. The breadwinner, the “working-man” of the 1950s, was the paragon of the household. Working husbands providing income meant that housework was deemed secondary, the needs of the housewife secondary to those of the husband. The difference, however, is that while the working man pursued his ambitions and competed for more desired opportunities, the housewife was relegated to the home, with no outlet for her ambitions, entirely at the service of her family. From a modern perspective, with women now integrated into the global workforce, it is not that difficult to see why some women in the 50s would feel trapped by their circumstances. With the constant pressure of monotonous servitude, who knows what kind of psychological stress they could have been under. As may be clear by now, this was the primary perspective we wanted to explore in *The Call of Karen*. Behind its playful and light-hearted exterior, the game is pinned to the psychological struggle of the 1950s housewife. The problem, in our case, was portraying this struggle. Luckily, there is sufficient precedent in this area, stemming from the long celebrated author of the Cthulhu myths.

## 2.3 Representations of Cthulhu in Media

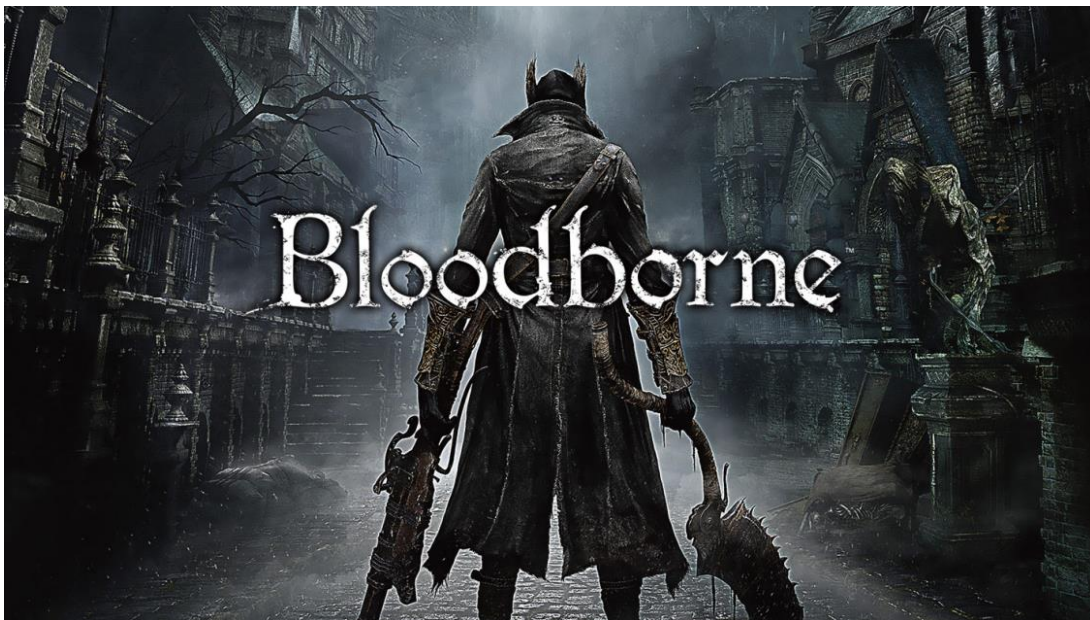
H. P. Lovecraft was born in 1890 in Providence, Rhode Island. Though he began publishing stories at an early age, he was little known during his lifetime (Somers, 2020). Only in the time since his death has Lovecraft's work in the realm of horror fiction risen to prominence. The terms "cosmic horror," "weird horror," and most recently "Lovecraftian Horror," all stem from the emergent ubiquity of Lovecraft's work. In contrast, the phrase "Cthulhu Mythos," the term describing the entire pantheon of fictitious stories, settings, and characters started by Lovecraft, was first used by one of his contemporaries in correspondence (Ghodrati, 2013). Of particular interest is that Lovecraft's work has travelled far outside the circles of literature in which it was born, penetrating into pop culture. Much like the writers that have since emulated Lovecraft's style and subject matter, creatives in all areas of media have been inspired by, and directly use the horrific universe built by Lovecraft in their own works. Eminent writers including Neil Gaiman, Alan Moore, F. Paul Wilson, and the prolific Stephen King have Lovecraft to thank. In King's words: "[Lovecraft] opened the way for us. He looms over all of us. I can't think of any important horror fiction that doesn't owe a lot to him" (Ghodrati, 2013). Director Guillermo Del Toro and surrealist artists H. R. Geiger and Jean Giraud (Mebius) also credit Lovecraft as inspiration. The supplemental literature, and more recently cinema, have had a tremendous effect on the general perception of the Cthulhu Mythos. Most recently, tabletop and video games have begun to incorporate elements of the Cthulhu mythos. Some, as we'll discuss, employ narratives constructed entirely within the Cthulhu Mythos.

Of particular importance to this paper, and to the design of *The Call of Karen* as a game, is the way in which Lovecraft told his stories. "Cosmicism" is the term given to the operational manual by which Lovecraft constructed his Eldritch universe. Cosmicism is "the idea and philosophy that mankind is absolutely insignificant and irrelevant...that life is genuinely inconceivable to the human mind, and the universe is fundamentally indifferent or hostile toward mankind" (Ghodrati, 2013). Glimpsing this vast incomprehensibility tends to drive characters insane, and it's often this glimpse that ends or forebodes the end of any hope of a positive conclusion for a protagonist. From the moment William Dyer hears a report of the titular range in *At the Mountains of Madness* (1931), the story is imbued with an eerie sense of hopelessness. And from the moment Thurston describes the small bas-relief in *The Call of Cthulhu* (1928), he carries the weight of knowledge too heavy for human minds (HorrorBabble, 2017, 2018). Lovecraftian horror stories rarely have unambiguous or happy endings.

Ghodrati captures the contemporary appeal of Lovecraft: "[His] 'gods,' names, incidents and settings were all created as if for our age; for the age of scientific and technological wonders and for the age when those weird paragraphs of cosmic horror and the Old Ones could be put into concepts for video games, arts and especially digital arts, modern horror writing and special effects in cinema" (2013). According to Ghodrati, Lovecraft first began to enter cinema after the second World War. The "proliferation of atomic weapons", had caused the public to question the "sanity of the scientific world" (Ghodrati, 2013). The official Lovecraft internet archive provides 3 categories for cinema involving Lovecraft: those films based directly on the original stories, Lovecraft inspired films that reference the original content, and films that have a "Lovecraftian feel to them," that do not explicitly reference Lovecraft (Loucks, 2020). Among this third category are Ridley Scott's *Alien* (1979) (heavily inspired by the art direction of H.R.

Geiger), and John Carpenter's *The Thing* (1982). The movies in the *Alien* franchise, most recently *Prometheus* in 2012, have made heavy use of the concept of "ancient astronauts" that inhabited the earth before the rise of humanity. This concept itself was heavily influenced by stories like *At the Mountains of Madness*, where characters stumble upon ruins of what seems to be an ancient, space-faring civilization.

As previously mentioned, interactive media inspired by or even set in a Lovecraftian universe have become increasingly popular in recent years. In the latter category video games like *Call of Cthulhu* (2005) and *The Sinking City* (2019) place players directly within the Lovecraftian universe, with references to locations like Arkhan, and Innsmouth that were re-occurring in the original literature. Likely the biggest video game of former category is the recent *Bloodborne* (2015). Releasing for both consoles and PC, *Bloodborne*, has a plot and backstory distinct from the Cthulhu Mythos, but draws heavily from the sense of cosmic horror established by Lovecraft. In particular, the "insight" system is a mechanic with *Bloodborne* that, once the player attains enough insight, allows the player to perceive ghastly monstrosities throughout the world that were previously unseen.



*Bloodborne (2015) promotional image*



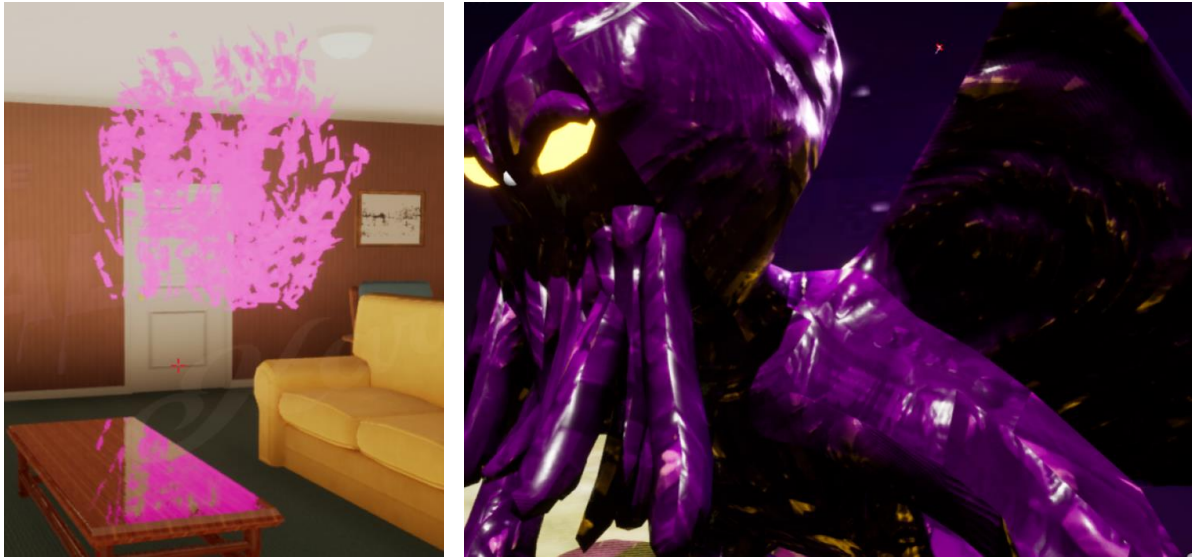
*Increased insight allows eldritch horrors to be seen in Bloodborne (2015)*

This progressive ability to see into an alternate reality, or to see something otherwise hidden is heavily utilized in the critically acclaimed mobile game series *The Room* (2012). Every game in *The Room* series is a 3D puzzle solving game steeped in Lovecraftian and cosmic horror. So far, every game in the series has included a “looking-glass” as a core mechanic, which allows the player to see glowing text, symbolism, and extra-dimensional objects that would otherwise be invisible. The most recent mobile release, *The Room: Old Sins*, features ancient aliens themes and puzzles, connecting it to the same web of inspiration as *At the Mountains of Madness* and *Prometheus*.

For *The Call of Karen*, we were not aiming to make a hopeless story. The focus was to be how Karen could overcome her circumstances, and rise to the challenge of her peculiar circumstances, without the help of her uncaring family. However, the encroaching presence of Cthulhu was not simply set dressing. We wanted to portray a psychological struggle in a way that wouldn't be completely foreign to players. The ubiquity of Lovecraftian fiction meant that (because players knew the presence of Cthulhu doesn't typically bode well) we could use references to the Cthulhu Mythos to progress the story. It also allowed us to easily and starkly



juxtapose the two sides of the conflict: “1950s housewife simulator meets eldritch horror” became an effective and enticing hook.



*Floating text and undulating textures in The Call of Karen*

The game itself incorporates direct references to the Mythos, as well as stylistic decisions that put it in the same category as the “Lovecraftian” films mentioned previously. The radio, in addition to progressing the narrative, provides many of these references. The player’s first hint of cosmic horror upon starting the game comes in the form of a radio jingle:

*“There’s a new kind of evil about...don’t let the octopus into your home.”*

From there the radio narration only gets more suspect, mentioning a “freak storm” over the Atlantic, and eventually quoting Lovecraft directly: “The Stars are Right, your number one guide to The Stars Being Right.” Abnormal visual effects, including floating text, camera effects including chromatic aberration, and grotesque, undulating textures add to the sense of otherworldly horror.

## 2.4 Humor in Games

From the beginning, *The Call of Karen* was designed to be as much about humor as it was about eldritch horror. Video games, as entertainment, frequently incorporate humor, and even some of the darkest, grittiest games take moments, like any good story, to add comic relief. *The Call of Karen* is far from gritty; the decision to make a game that didn’t take itself too seriously occurred early in the design process. But making a game where the humor actually resonates, and doesn’t fall flat, is easier said than done.

In *A Review of Humor for Computer Games: Play, Laugh and More* (2009), Dormann and Biddle divide theories about why humor exists into three categories: superiority, relief and incongruity. Superiority theory states that humor occurs at the expense of others. Relief theory emphasizes that humor can occur with the release of nervous tension. Finally, incongruity

theory focuses on how humor tends to occur when people find humor in unexpected or surprising situations. The order these are presented in also reflects the chronology of the respective theories, with superiority theory dating back to the mid 1800s (Dormann & Biddle, 2009). Thus, incongruity theory is the most modern, and most well regarded of the theories about the underpinnings of humor.

Kallio and Masoodian note that “while humor exists in some shape or form in many video games, unlike in film and literature, comedy is not considered an established genre of its own in video games” (2019). Comedy does not seem to be a cut-and-dry segment of the video game market as much as a by-product of the play process: “Eastman points put that humor is a form of play in itself, and that ‘No definition of humor...will ever stand up, which is not based upon the distinction between playful and serious’ ” (Kallio & Masoodian, 2019). In this way the humor in a game can be “emergent.” For example, in multiplayer games, getting hit by a vehicle and having your character ragdoll a dozen yards. Military and combat-focused games like the *Battlefield*, *Call of Duty*, and *Halo* (2001-2020) series all produce comedic gameplay experiences in this way. It relies on players using open-ended game mechanics and systems to produce humorous, laughter inducing moments. But what if humor is intentionally built directly into the game? According to Kallio and Masoodian, “Ludonarrative” comedy involves “active” interaction with an intentionally humorous gameplay/story-driven experience. They identify 3 primary methods, and a handful of secondary methods for “incorporating comedy into [the] ludonarrative of video games” (2019). Primary methods include visual style, the sources of comedy, and conflicts in comedy. Secondary comic “enhancers” are “useful for captivating the players’ curiosity, but can also prepare them for the full experience of humor (Kallio & Masoodian, 2019). These include humorous worlds, environmental storytelling, challenging normality, comic relief, and superiority humor.

*The Call of Karen* utilizes many of these ludonarrative strategies. Primary methods include comedy through source and conflict. An unexpected source of comedy is the radio announcer, who leads players through the game in a humorous way. Some of the game



*Mysteriously floating books are one of the first signs of danger*

mechanics themselves are also sources of humor. Floating household items, upgrading your vacuum with silver bullets, and chasing down a possessed meatloaf fit into this category.

Humor is also generated from the central conflict of the story; Karen, an average 1950s housewife, is at odds with the likes of an eldritch god, a Great Old One descended from the stars. *The Call of Karen* also includes many secondary elements for generating comedy. Dirt and slime spawning on the ceiling and walls reinforces a humorous world and environmental storytelling. The game also challenges normality, by establishing a relatively believable setting, and then breaking rules as Cthulhu encroaches further and further into Karen's suburban abode.

## 3 Design

In this chapter, we describe the design of *The Call of Karen*, including the experience goal in chapter 3.1, narrative design in chapter 3.3, mechanics in chapter 3.4, and overall game structure in chapters 3.5 - 3.9. We also discuss our comparables and how we drew from them in chapter 3.2, our greyboxing and paper prototyping process in chapter 3.10, and our level design in 3.11.

### 3.1 Experience Goal

Our main experience goal with *The Call of Karen* stemmed from wanting the player to feel as though their surroundings were spiraling out of control. This sort of feeling is usually used to inspire frustration or fear, but we wanted to use this feeling to inspire humor instead. *A Review of Humor for Computer Games: Play, Laugh, and More* details two of the major theories of where humor comes from: relief theory and incongruity theory (Dormann & Biddle, 2009). Relief theory suggests that we are likely to laugh when we're somewhat nervous, in order to break the tension. Incongruity theory posits that we laugh at things that are unexpected, surprising, and have a certain degree of nonsense to them. The high concept of *The Call of Karen* takes advantage of both of these humor theories. Seeing Cthulhu slowly ramp up its dominion over Karen's house inspires some tension, which allows us to tap into relief theory, and the conceit of Cthulhu invading a 1950s suburban housewife's home inspires humor with incongruity theory.

While the above goal was largely what our efforts were focused on, we had other goals as well. In addition to humor, we hoped to inspire amusement. We wanted players to be consistently engaged with and enjoying the game. We also wanted the player to feel pride in their work. We wanted to incorporate an element of satisfaction in the tasks the player was completing, so as to entice them to play further and to make sure they noticed when Cthulhu's presence started to affect these tasks.

We defined these goals early on in the game development process, the summer before our MQP officially began. We believed it was imperative to make sure we were all on the same page about what kind of game we were making as soon as possible. This experience goal significantly shaped our design process, and remained largely the same throughout production. The only minor changes we made were based more on the humor theories we utilized than on the actual goal itself. While we kept relief theory in mind, as production continued, we found ourselves leaning more and more towards the use of incongruity theory to inspire the humor we wanted.

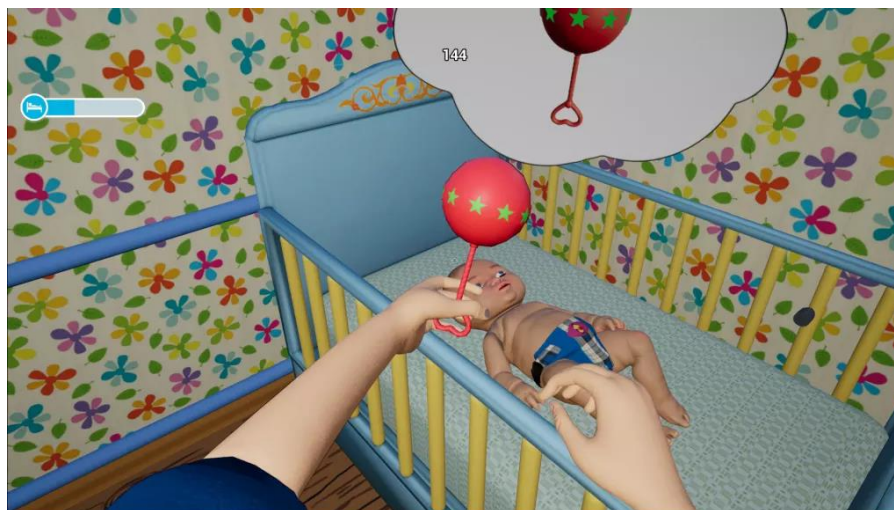
### 3.2 Comparables

Throughout the development of *The Call of Karen*, we drew inspiration from several games to inform how our game would play. Our main comparables were *Job Simulator*, *Mother Simulator*, and *Cooking Simulator* (Owlchemy Labs, 2016; PlayWay, 2019; Steppe Hare Studio, 2018). All of these games allow the player to interact with different objects in a physics-based environment, and are structured around task based gameplay, both of which we wanted to

replicate in *The Call of Karen*. *Cooking Simulator* was our closest comparable, since it was close in scope to what we had the capability of accomplishing and also has no fail state. We also drew from *Job Simulator*'s step-by-step instructions on completing tasks as well as the overall goofy tone of the gameplay, and were inspired by the hectic nature of *Mother Simulator*, though we decided not to emulate its intentionally unwieldy controls.



*Job Simulator* (2016)



*Mother Simulator* (2018)



*Cooking Simulator (2019)*

## 3.3 Story

While a large portion of our design process was focused on gameplay and mechanics, we had to dedicate plenty of time to setting and story as well. While we hoped to inspire humor with *The Call of Karen's* conceit and gameplay progression, it became evident early on that we would need a cohesive story in order to tap into the game's fullest potential. We knew that we had to keep our setting and story obvious and readable—the more that was immediately apparent to the player, the less we had to go out of our way to convey with added systems, and the more manageable our scope would remain.

### 3.3.1 Story Summary

The titular Karen of *The Call of Karen* seems to have everything a 1950s housewife could want: an affluent husband, a gorgeous house, and a bright young child. However, despite her seemingly picturesque lifestyle, she is deeply unsatisfied. Her husband is largely absent and dismissive of her, working late hours and rarely engaging with her during the hours that he is at home. He doesn't lift a finger to help with any housework, instead choosing to sequester himself in their room to write his novel. Karen's son, Francis, displays all the ungrateful mannerisms typical of 10-year-olds, and spends most of his time with his friend Timmy, whose passive aggressive mother constantly gets under Karen's skin. This dissatisfaction with her family and the expectation that she maintain the household in a perfect state of cleanliness wears on her. An unhappy house wife in her house life, Karen often falls asleep at the kitchen table, wine glass in hand.

During a seemingly normal week, unsettling news begins to air on the radio. There's a freak storm over the Atlantic, and the radio host seems to have become a bit unhinged, occasionally speaking in tongues in a garbled voice. Karen's house starts to behave oddly as well. Food flies out of the fridge once she opens it, and slime and floating furniture pop in and out of existence before she can really get a good look at them. As the week progresses,

stranger and stranger occurrences happen in Karen's house, and she is left trying desperately to clean it all up before anyone notices.

After a week of solving eldritch horror-related problems, Karen decides that she's had enough. She modifies her trusty vacuum with as much demon-fighting paraphernalia as possible and challenges Cthulhu face to face. She seals it in her vacuum, and the horrors finally cease.

### 3.3.2 Characters

One challenge we faced with characters is that, in order to convincingly portray a typical 1950s suburban housewife, we needed Karen to have a husband and at least one child. However, fully voicing, modeling, and animating additional characters is extremely time consuming, and creeps dangerously towards the uncanny valley if done poorly. After some deliberation, we decided to make it so that Karen's husband and children are never seen in-game, only heard through voice lines shouted across the house. We believed that this added to the humor of the game and was true to the often isolating lives of 1950s housewives (Lamb, 2011, p.29). This implementation of the husband and child also emphasize the fact that Karen is unappreciated; she spends all day cleaning, and they don't even bother to speak to her face to face. Additionally, we thought having the kid and husband run circles around Karen without ever seeing her was a funny gag.

We have a few other characters in addition to the husband and child, to give the impression of a world outside of Karen's house. Additionally, we wanted at least one other character to be affected by or acknowledge Cthulhu's presence, since we wanted to lean away from the idea that everything happening was all in Karen's head. We also needed a vehicle for more humor and commentary on the 1950s, so we incorporated a radio with a host that is also affected by the encroaching eldritch threat.

### 3.3.3 Karen

When creating the character of Karen, we wanted to make sure that we portrayed a 1950s housewife recognizably, but also didn't buy into any of the harmful stereotypes surrounding 1950s housewives—at least, not without seriously examining them first. 1950s housewives are generally represented as happy, subservient women, despite the prejudices and expectations set upon them by their families and society. We wanted to avoid the harmful misogyny present in these tropes, and create a more realistic, rounded character. Karen is unhappy in her marriage, and is well aware of the societal pressures and expectations forced on her. She doesn't like it, but she sticks with her husband and keeps maintaining the house in order to keep up appearances. The idea of maintaining normalcy and keeping up appearances is integral to the culture of the 1950s, so we wanted this idea to be important to Karen as well. As a result, Karen doesn't want to fight Cthulhu because it's an unspeakable evil that needs to be stopped, but because it's affecting her ability to maintain normalcy.

During the beginning of production, Karen was largely voiceless, and her personality was only expressed through one or two lines. As production continued, we realized that it was imperative that Karen have more dialogue and be a fully-fledged character, so people would relate to her more and so we could really clarify that she was not happy with her lot in life. Karen

talks to her family politely, but when she's alone in the house she becomes sarcastic and beleaguered, revealing her true feelings.

Worth noting about Karen is that she has a background in fighting monsters, which is hinted at through a few lines in the game. This aspect of Karen's background is generally skated over for laughs, but is worth noting—the living room closet in the house is dedicated to monster fighting paraphernalia, including bottles of holy water, wooden stakes, garlic, crucifixes, salt, and knives.



*Karen's monster fighting closet. Not pictured are the crucifixes on the wall, just out of range of the camera.*

The monster fighting closet ties into another key aspect of Karen's personality. While she is a victim of the misogyny and expectations of her time, she is also incredibly proactive. She is aware of the horrors encroaching on her house, and takes concrete steps (like ordering silver bullets and modifying her vacuum to shoot them) to keep everything in order.

### 3.3.4 Husband

With Karen's husband, we didn't mind as much if we bought into the stereotypes surrounding 1950s men, because generally those stereotypes were not as harmful or pervasive as the ones surrounding women. As a result, Karen's husband is the typical 1950s husband. Most depictions of 1950s husbands gloss over their negative attributes, so we made sure to emphasize them. He's a working man, and as a result is out of the house all day. He offers little to no help with general day to day care around the house, and doesn't interact much with Karen or his child. He's dismissive of Karen and the work she does, and through Karen's dialogue it's implied that he, like many men of his era, is a misogynist. While he is certainly a bad husband, we avoided making him overtly physically or emotionally abusive or manipulative. This game tackles a lot of parts of 1950s housewife life, but we aren't equipped and weren't prepared to properly portray a legitimate abusive marriage. Also worth noting is that an unhappy marriage can be played for laughs, while an abusive marriage should not be. Karen's husband is named John, a stereotypically generic caucasian male name to fit his stereotypical personality. Due to his general absence, he is only heard throughout the game when leaving for work or giving



excuses for not coming to dinner. He is oblivious, and does not notice the eldritch horror infecting the house, or his wife's unhappiness.

### 3.3.5 Child

Similar to the husband, we didn't mind if we bought into stereotypes surrounding children. Francis is Karen's only child, named for Francis Wayland Thurston, the narrator in the original *The Call of Cthulhu*. Francis is generally ungrateful, but not to the point of caricature—he's just your average entitled and somewhat rude child. Francis is about 10 years old, in middle school, and generally out and about. We didn't want Francis to be a surly teen, but we also didn't want him to be so young that he was extremely reliant on Karen, or required too much supervision to leave the house. We wanted Francis to have items that he would leave strewn around the house for Karen to clean up, so we settled on trading cards, a traditional pastime for kids his age in the 1950s. Francis, like Karen's husband, doesn't notice Cthulhu invading the house. It was important to us that we didn't put Francis in any serious danger—any game that enters the realm of seriously hurting children strays too close to the kind of horror we wanted to avoid entirely. We also used Francis as a vehicle for the rivalry between Karen and one of her neighbors, Susan, by having Francis hang out with Susan's son. While Francis is certainly a source of stress in Karen's life, we intentionally focused on the husband as the bigger problem in the house. Children are stressful and uncivilized just by nature of being children, while Karen's husband is old enough to know better.

### 3.3.6 Radio

We knew early in production that we wanted a radio in the game to provide more insight into the outside world. It was important to us to have a window into life outside of Karen's house, to give the impression of a greater, bigger world. We listened to several old radio broadcasts to make sure we portrayed the radio persona right. We learned fairly quickly that 1950s radio recordings were few and far between, and that the kind of radio announcers associated with the 1950s were present throughout the 1960s. As a result, we largely used 1960s radio as our reference, including but not limited to the *Ron Lundy Show*, Lucille Ball comedy sketches, and *WMCA "Goodguy" Radio* (Ball, 1950; *KOMA*, 1964; Lundy, 1962; Murrow, 1950; O'Brien, 1965). 1950s radio announcers were generally very upbeat white men who did commentary, interviews, and jokes in between sponsorships and songs playing, so incorporated these aspects into the personality of our radio host. Our fictional radio station, PMCR radio (which stands for Ph'nglui Mglw'nafh Cthulhu R'lyeh, part of a chant from *The Call of Cthulhu*), is hosted by Charles, named for the protagonist of one of Lovecraft's other novels, *The Case of Charles Dexter Ward* (1941).

Radio hosts and their jokes were products of their time, so we decided to utilize the radio for commentary on the 1950s. Charles reinforces McCarthyism, misogyny, and intense materialism on his show, as many of his real-life counterparts did, exaggerated and played up for satire. Charles also makes comments with uncanny accuracy on what Karen is doing throughout the day for some extra commentary and humor. We decided that Charles would slowly get corrupted by Cthulhu, to reinforce the idea that Cthulhu's presence and impact is not limited to Karen's house and mind. Charles maintains his happy-go-lucky and showboating

affect even as his voice becomes garbled and he begins speaking in tongues. He seems to suffer minor nervous breakdowns throughout the week, though once Karen seals away Cthulhu, he returns to normal.

### 3.3.7 Susan Jones

A classic trope of many narratives involving suburban housewives is rivalry with other suburban housewives. We found that we could utilize this trope while avoiding the misogyny that defines many other pieces of media depicting suburban housewives, and decided that we could incorporate some humor here without demeaning or devaluing Karen or her work. Susan Jones is the local PTA president, and the mother of one of Francis's friends. She is catty and makes backhanded compliments towards Karen fairly often, and dialogue from Karen indicates that the disdain is mutual. Worth noting is that, while Karen is displeased with her lot in housewife life, Susan fully buys into it, at least externally, going so far as to make statements like "women shouldn't drive." Susan's personality is inspired by the sad fact that women raised in sexist environments (especially in the 1950s) have a significant amount of internalized misogyny, and sometimes buy into the idea that they are less than men and only good for housework (Charles, Guryen, & Pan, 2018; Lamb, 2011, p.57).

## 3.4 Mechanics

We kept scope in the forefront of our minds as we designed our mechanics, since we had a limited amount of time and number of programmers and weren't looking to create complex systems. We decided to focus on logical and/or spatial difficulty when deciding on our mechanics, with the intent to keep gameplay straightforward and fairly easy. We avoided fine-motor difficulty, since we didn't think it would mesh with the experience goal of the game, and we weren't sure how well fine motor difficulty would tie into the experience of being a suburban housewife. After all, how often do you need twitch reflexes when cleaning up the house?

We settled on two major gameplay systems: picking up/putting down objects and interacting with them. Picking up functions similarly to picking up items in Bethesda Softwork's *Skyrim* (2011). Upon the press of a button (in *Skyrim* E, but in *The Call of Karen* left click) the item is held out in front of you. The item is locked to the center of your viewport, and when you move, it moves. You put down items with the same button click, and if they are not on the floor, they fall to the floor. You cannot move items through solid objects as well. Items can be picked up and put down in certain positions to trigger certain events or complete certain tasks. For example, putting down a plate on the dining room table is part of the "setting the table" task. Interacting with objects is accomplished with right click. Large objects with a highlight around them will open and close, while holding an object and then pressing right click will cause that object to perform its specific function. For example, cabinets open and close upon right clicking, but the vacuum, which is held in your hand, turns on and sucks up grime with the same button. Some objects change others based on interaction—for example, a raw meatloaf becomes cooked when placed in the oven.

We kept the mechanics on the simpler side to establish a solid, easy to understand groundwork with the player. The high concept of the game already requires significant

suspension of disbelief, so we want to keep the general mechanics very readable so as to not completely alienate the player. This simplicity also allows us to convey a sense of normalcy to the player easily and effectively, which makes breaking that normalcy all the more noticeable.

### 3.4.1 The Idea of Horror

As made clear by our experience goal, we did not intend to inspire horror with this game. However, we can't completely ignore the influences the horror genre had on our design process. After all, Lovecraft's creations are all usually described as "horrors". Additionally, we have to use certain horror elements to communicate to the player that Cthulhu is invading Karen's home. So, how do we invoke the idea of horror, while staying away from actual horrific elements themselves? The art section of the paper details the visual side of this conundrum, but here we're talking about mechanics.

One of the main elements of many horror games is lack of agency. In classic horror staples like *Outlast* (2013-2018) and *Amnesia: The Dark Descent* (2010), when there are monsters coming, you can't fight back against them; your only choice is to run (Frictional Games, 2010; Red Barrels, 2013). With *The Call of Karen*, we were aware of this use of agency, and designed around it. From a narrative standpoint, Karen lacks a lot of agency in her home life. She's stuck doing the same or similar tasks every day, and until the end of the game, she is unable to directly combat Cthulhu, instead dealing with byproducts of his presence. However, when it comes to mechanics, Karen has a ton of agency, able to interact with most objects and accomplish tasks at her own pace. Giving Karen agency in her mechanics but limiting her agency in the narrative allowed us to strike the balance between horror influence and actual horror that we wanted.

## 3.5 Tasks

A key part of *The Call of Karen* is the tasks Karen completes through the day. These tasks are essentially chores, typical activities that a suburban housewife would do throughout the day. Tasks are composed of granular subtasks, and completion of these subtasks leads to completion of the overarching task. For example, the first overarching task of every day is making breakfast. Making breakfast is broken down to the following subtasks: cook two eggs, two strips of bacon, put them on a plate, and place a fork on the plate to eat them. These subtasks are meant to be fairly self explanatory, and use real-world logic to guide the player. Cooking eggs requires the player to get a pan, place it on the stove, and then place an egg in the pan. Designing our tasks around real-world logic lessened the amount of tutorialization necessary—players were already familiar with the general principles involved with the gameplay. All other tutorialization was accomplished via short, onscreen prompts.

We knew early on that we wanted tasks, not minigames—minigames are too hard to fine-tune, and tasks were what our primary comparables used. We wanted each task to be fairly short, and reminiscent of what a suburban housewife would do so that we could effectively utilize real-world logic. We also wanted tasks that could be feasibly modified to indicate the presence of Cthulhu.

We ideated a wide variety of potential tasks that a typical 1950s suburban housewife would complete during the course of her day, then pared them down. We tried to keep the number of tasks as low as possible, keeping in mind what we could make fun with the mechanics we had and what best represented a typical suburban housewife's day. We initially ended up with seven tasks in total: Make a smoothie, make lunch for your son, clean up, vacuum, cook meatloaf, set the table, and clean the table. As we prototyped and iterated, we learned that some tasks were not fun, not necessary, or didn't effectively evoke the 50s, and thus were changed. We also later added "flavor" tasks, one-off tasks that add some variety and some extra worldbuilding.

### 3.5.1 Smoothie/Breakfast

Making a smoothie was a task that took place in the morning of every in-game day. It involved placing a certain number of combinations of fruits and vegetables into a blender, turning on the blender, then drinking the resulting smoothie. We wanted it to be the main, routine-setting task, the only thing Karen really does for herself throughout the entire day, and the first thing the player does in the game. Making a smoothie was one of the earliest tasks we had in mind for the game, a relic from before we had decided that the game would take place in the 1950s. Once we did make this decision, we considered changing the smoothie task, as it didn't seem period accurate for the 1950s (though research indicated that it was). We kept it in due to the consensus that, while it didn't evoke the 1950s, it was a task typically associated with suburban housewives.

That wasn't the end of our smoothie-related discussion. After playtesting at our first event, Alphafest, we noticed several players struggling to put the fruit in the blender—the transparency of the blender combined with the fact that the fruit didn't cast shadows made for a significant depth perception struggle. Additionally, our feedback from both Alphafest and the later MassDiGI Game Challenge indicated that we were right in our first concerns; the blender and act of making a smoothie felt out of place in an otherwise obviously 1950s aesthetic. As such we decided to change the task. However, we were left with a slight conundrum. While the smoothie task had problems, we didn't want to completely redo or eliminate it. Eliminating the smoothie would cause significant script rewrites, and starting from scratch would mean completely redoing the framework of the task and potentially greyboxing it again.

After some discussion, we changed the task from making a smoothie to cooking breakfast. This switch maintained the same level of suburban housewife-iness, while keeping with what people expect from the time period. Additionally, we were able to reuse a lot of the same task framework. Cooking breakfast involves getting a pan out of the cupboard and placing it on the stovetop, then getting eggs and bacon out of the fridge and placing them in the pan. The player must wait a few seconds for them to be cooked, then place them on a plate. Once plated, they must add salt and pepper to their food, then place a fork on the plate to eat it. The breakfast change was helpful to us in several ways. The use of the pan eliminated the depth perception problem we had with the blender, and cooking eggs and bacon was generally easier to grasp than finding all of the different fruits and vegetables.



*Cooking Breakfast.*

### 3.5.2 Lunch

Making lunch was another task decided early on, stemming from the idea that we wanted more chores to acknowledge Karen's motherhood. The idea of lunch was to take foodstuffs from the fridge and around the kitchen and place it into Francis's lunchbox, which would disappear from the table once he went off to school.

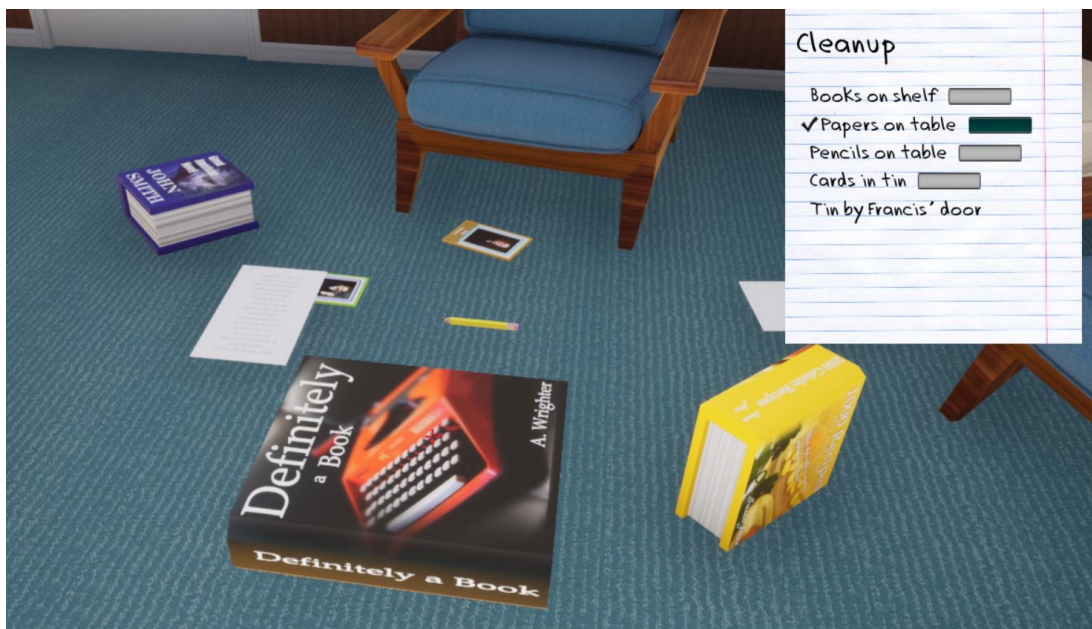
The main struggle we faced with lunch was implementation. Lunch didn't make it into the game in time for Alphafest, and after Alphafest we focused more on fixing the issues with the gameplay that was already implemented. After a while, we made the decision to leave lunch out. We had concerns that adding lunch late would result in a host of new problems that we would then have to balance, similar to the first implementation of the smoothie. Additionally, we found that lunch wasn't integral to the experience. The cleaning up task involved picking up Francis's belongings, so we weren't bereft of tasks that indicated his presence in the house. No voice lines directly referenced the act of making lunch, so we wouldn't cause significant rewrites by leaving it out. Additionally, the actual task itself, picking up and putting different types of food into a receptacle, was quite similar to the smoothie task. Since lunch occurred immediately afterwards, it didn't seem especially necessary.

### 3.5.3 Cleaning Up

Cleaning up was an integral task due to the fact that it is both a key part of the 1950s housewife experience and that it effectively communicates the fact that Karen's family doesn't help her with any housework. Cleaning up involves picking up papers, pencils, books, and trading cards strewn around the living room, objects belonging to John and Francis. Books must

be placed on the living room shelf, papers and pencils must be placed on the coffee table, and trading cards must be placed in a trading card tin and left by Francis's door.

Cleaning was a task that changed very little throughout the development process. It did receive some tweaking after playtesting, as we discovered that we had done too good of a job leaving the room in a state of disarray and the task took far too long. This issue was fixed rather easily by lowering the number of objects that needed to be put in their proper place. The only main issue we faced was that, for a significant part of development, we had no indication of which door was Francis's. That problem was solved by adding a hand-lettered sign to Francis's door. Overall, cleaning was easy to grasp and conveyed Karen's plight effectively, making it one of our most invaluable tasks.

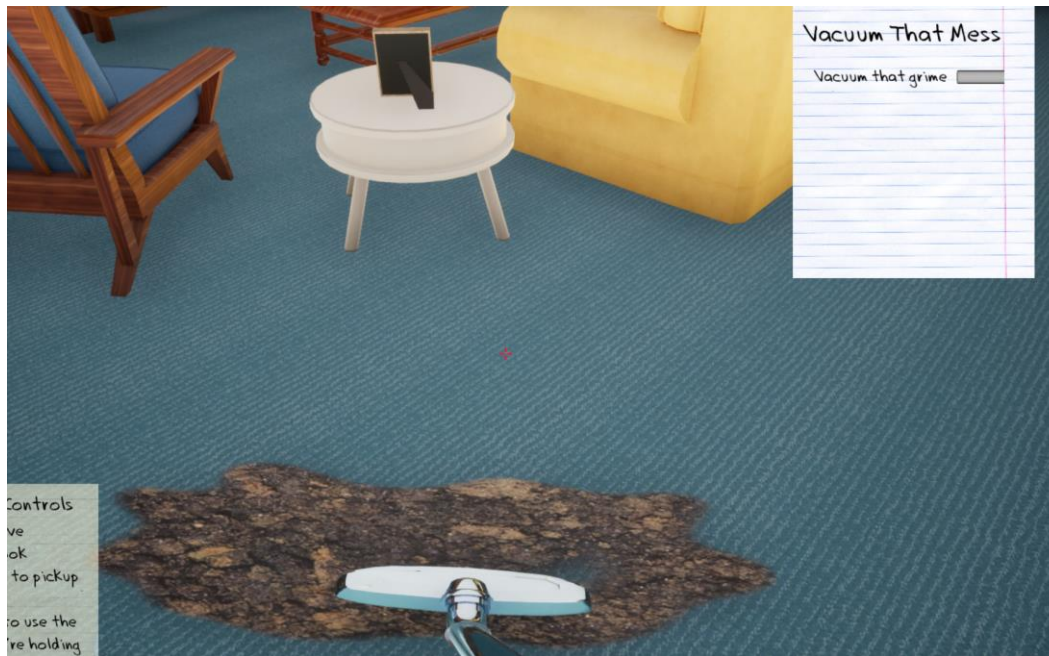


*Cleaning up.*

### 3.5.4 Vacuuming

Vacuuming was designed hand in hand with cleaning up as a typical suburban housewife task that we wanted to include. Vacuuming involves picking up the vacuum object, holding it close to a patch of grime, and holding right click until the grime fades away. Vacuuming didn't go through any major changes during the course of production, though we made plenty of minor tweaks. Players struggled to figure out how to use the vacuum due to the fact that grime patches faded out fairly slowly. Players would be using the vacuum correctly but not see immediate results, and as such would think it was broken or that they were using it wrong. We remedied this problem by speeding up the rate at which grime patches faded out. We also found that players would often stop vacuuming the moment the patch of grime became invisible. This behaviour was a problem due to the way the grime faded out, as the grime was invisible for a very small period of time *before* it was counted as completely vacuumed up. Thus, players would move on to a different stain, and have no way to figure out where the old stain was upon realizing their error. We fixed this issue by altering the way the grime fades out. Once

the grime had almost faded to the point of invisibility, we dialed down the opacity rapidly to fade it out the rest of the way and count the patch as completed. Vacuuming, similar to cleaning, was extremely useful to us, and served the added purpose of giving us a weapon for Karen to fight Cthulhu with.



*Vacuuming up grime.*

### 3.5.5 Cooking Meatloaf

It was important to us that Karen cook for her family, but we worried that complex cooking systems or mechanics would dangerously balloon our scope. We considered several ideas, and had to balance between what was interesting, period accurate, and recognizable. We hovered around the idea of having Karen make aspics for a while, gelatin-based savory dishes that were all the rage in the 1950s.



*A typical 1950s aspic.*

However, while we did enjoy the idea of aspics, we decided that they weren't quite recognizable or iconic enough as regular 1950s fare. Given that this is a game about doing housework in the face of a home gone mad, we thought it would be more valuable to start with a cooking task that was rooted in normalcy, then take it into the realm of madness, and starting with an aspic would already be significantly out of the ordinary for most people. After kicking around several ideas, we landed on cooking meatloaf.

Meatloaf was ideal for us because the act is quick and doesn't require any complex cooking systems, and could be performed in conjunction with setting the dinner table. Making meatloaf in *The Call of Karen* involves picking up raw meatloaf, placing it in the oven, waiting a few seconds for it to cook, bringing it to the dinner table, and then dropping a knife on it to slice it.

Meatloaf was another task that underwent very few changes during the course of production. The main struggle that we faced with meatloaf was technical, as it was sometimes difficult to get the meatloaf to sit in the oven to cook it, and once the meatloaf was done, it was hard to extricate it from the oven. However, the overall structure of the meatloaf task always remained the same.





*Cooking meatloaf.*

### 3.5.6 Setting the Table/Cleaning the Table

Setting the table involves placing plates, forks, and cups on the table to prepare for dinner, and upon being placed anywhere on the table, the dinnerware will immediately snap to its proper position. Setting the table and cleaning the table were designed to go hand in hand with cooking meatloaf. The idea was initially that Karen would set the table, then cook the meatloaf. After dinner was prepared, Karen would be drawn out of the dining room for one reason or another, and in that moment her family would eat dinner in a mad scramble, leaving the table in disarray before returning to their respective rooms. Karen would then return to the dining room and place the dishes in the dishwasher.

After a while, we decided to cut the cleaning the table aspect of the task. Firstly, we couldn't quite figure out a good system to get Karen out of the dining room and obstruct her view of it for long enough to start the dinner sequence without locking the player's movement and camera, which we didn't want to do. Additionally, similar to lunch, cleaning the table didn't make it into the build in time for Alphafest or playtesting. We decided that it was repetitive and didn't add enough to the game to be worth the trouble, so we left it out.

Worth noting is that setting the table and cooking meatloaf were functionally combined into one overarching task, the "daily dinner dash". This combination meant both tasks could be completed during the same period of time; for example, the meatloaf could be placed in the oven, and the plates could be put on the table before the cooked meatloaf was taken out. This combination made sense to us considering that both tasks were fairly short and revolved around preparing for dinner.



Setting the table.

## 3.6 Flavor Tasks

Flavor tasks were added to the game after several industry experts at the MassDiGI Game Challenge suggested that we add unique tasks that directly addressed Karen trying to combat Cthulhu. We kept these tasks short and simple, as we were fairly deep into development and didn't want to add anything that would need intensive playtesting or balancing.

### 3.6.1 Ordering Silver Bullets

Ordering silver bullets begins when Karen receives an edition of the fictional *Being* magazine (a play on *Life* magazine, a popular read in the 1950s) in the mail. The magazine offers free silver bullets if the receiver's address is filled in and mailed back to the distributor. The player must place a pencil on the magazine to fill out the information, then place the magazine in an envelope, then take the envelope to the mailbox. A few days later, the silver bullets arrive.

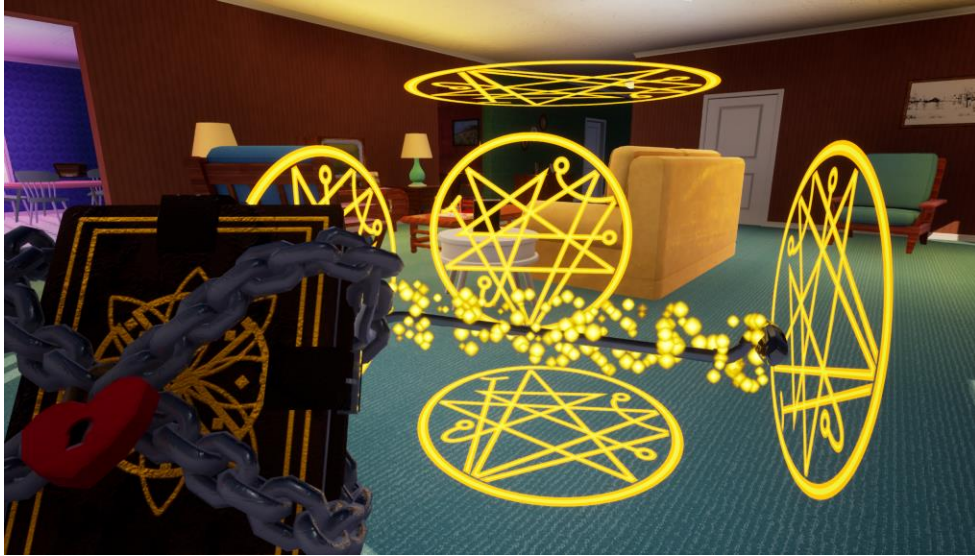


*Being magazine and the advertised silver bullets.*

Silver bullets were added specifically because of their connection to monster hunting. While silver is not necessarily a weakness of Cthulhu's, in many mythologies silver is used to fight against or ward off evil supernatural beings. As such, we decided to include silver bullets as a method of fighting against Cthulhu. We were more concerned with portraying recognizable demon-fighting iconography than being accurate to the Cthulhu mythos, and assumed that most people wouldn't have enough knowledge of *The Call of Cthulhu* to note that fighting Cthulhu in this way isn't accurate.

### 3.6.2 Receiving and Using an Ancient Book

Karen receives another package in the mail the day after ordering her silver bullets. The package is an ancient book, sent over to her by her catty neighbor Susan. Attached is a passive-aggressive note from Susan noting the general disarray of Karen's home and informing her that the ancient book will help. Picking up the book and right clicking the vacuum with it while the book is being held will cause the vacuum to become enchanted, and able to vacuum up the eldritch slime covering Karen's walls and ceiling.



*The vacuum is enchanted, with ancient book in hand.*

After the vacuum is upgraded, Karen must now (begrudgingly) write Susan a nice thank you letter. Writing the thank you letter involves picking up a pencil and placing it on paper, then putting the paper in an envelope, then putting said envelope in the mail slot. While this is the last we see of the ancient book, this is far from the last we see of upgrading the vacuum.

### 3.6.3 Modifying Vacuum with Bullets

Once Karen's bullets arrive in the mail, her vacuum must be modified to be able to use them. This modification is accomplished in the same way that enchanting is, holding the bullets while right clicking the vacuum.

### 3.6.4 Final Vacuum Modification

After several days of eldritch horrors invading her home, Karen decides that she's going to combat Cthulhu directly in a final showdown. Before doing this, she has to upgrade her vacuum one last time. This task involves picking up the vacuum, putting it in the previously mentioned monster-fighting closet, and closing the door and waiting for a few seconds. Upon opening the door again, the vacuum will have several new Cthulhu-fighting modifications from the objects within the closet.

### 3.6.5 Sealing Away Cthulhu

Sealing away Cthulhu, and the ending of our game, was something we discussed at length. We did want to end the game on a positive note for Karen, as we felt that we would lose a lot of the humor if she remained miserable. We also didn't want to destroy or explicitly show Cthulhu, as that goes against the spirit of the Cthulhu mythos; a core part of the idea of Cthulhu is that it is unknowable and cannot be killed. We considered a twist ending, revealing at the end that Karen's husband and child were eldritch demons, and that's why the house was spiraling out of control. We ended up discarding this idea because we felt that it undermined the

exploration of Karen's unhappiness as a 1950s suburban housewife. If Karen's family was actually eldritch demons, than it would be easy to chalk up their behaviour and her unhappiness to the fact that they're evil entities. The misogyny of the 1950s is easy to sweep under the rug when the main misogynist in question is an evil demon.

We definitely had a conundrum on our hands. Since Karen is fighting against Cthulhu all week, it would make sense that she'd want to stop him so everything would go back to normal. The problem is, normal wasn't working for her. Even if Cthulhu is stopped, her marriage remains unhappy and her child remains ungrateful. We kicked around several ideas that might give Karen a satisfying ending, one of them even being that she would marry Cthulhu—we thought there was something to the thought that an eldritch horror would be a better husband than a typical 1950s man.

Eventually, we landed on a happy compromise: Karen would seal Cthulhu away in her vacuum. That way we would never have to show Cthulhu or destroy it, and considering that in *The Call of Cthulhu* lore Cthulhu being unsealed from his deep slumber in the sunken city of R'lyeh is what would bring about the end times, it's logical to assume that to stop Cthulhu, it would have to be sealed away again.

For this task, Karen must pick up her newly upgraded vacuum and step outside, the first time she's been outside during the entire game. The house is surrounded by a thick fog, and the dim silhouette of Cthulhu is above her. The player must click to throw the vacuum at Cthulhu. The game cuts to black after she does so, and skips to the next day.

### 3.6.6 Revving the Cthulhu Vacuum

There is one final flavor task that occurs on the final day of the game, after Cthulhu has been sealed away. The player must find and pick up the vacuum Cthulhu is sealed in. Turning on the vacuum indicates that Cthulhu's power is sealed within and can be manipulated by the vacuum. This indication of the vacuum's new power is key to the ending of the story. Immediately after trying the vacuum, Karen's husband chides her for the house being less clean than usual during the week of eldritch horrors, upon which Karen revs the vacuum once more.

## 3.7 Days

While tasks are at the forefront of *The Call of Karen*, the game's structure is day-based as well. The game takes place over the course of an in-game week, with different tasks on Karen's agenda depending on what day it is. We utilized this day-based framework for several reasons. Firstly, we wanted to add some variation to our gameplay. We were concerned that doing the exact same tasks in the same order day after day would get boring and repetitive fast, and different tasks on different days would break up some of the monotony, while still allowing us to set up a general routine for Karen. Additionally, we were worried that doing the same tasks every day would break some of the illusion for the player. Considering that Cthulhu exists in *The Call of Karen's* universe, we're clearly not going for the picture of realism, but considering that our game's mechanics rely on real-world logic, it's easier for players to swallow that Cthulhu is real than that the floor needs to be vacuumed every day.

Establishing our use of a day-based system early in our development also gave us more concrete control of the scale and scope of the game. The length and pacing of the game could be adjusted by removing or changing days, or changing the number or type of tasks that take place during each day. Initially, we settled on completing two in-game weeks, with four or five tasks during each day. Our initial estimate was that tasks would take roughly two minutes to complete, so each day would take roughly eight minutes. Two in-game weeks would give us a little over an hour of gameplay, which we thought was feasible given our timeframe.

As development continued, we realized that our initial two week estimate was a bit optimistic at the pace we were moving at. Additionally, we found during playtesting that tasks took longer than planned to complete, roughly three to five minutes for each one. We also suffered from pacing issues—playtesters felt like the game didn’t introduce Cthulhu early enough. As such, we cut the number of in-game days we would complete in half, settling on one week. We also lowered the number of tasks each day to three or four, or lower depending on how difficult the tasks were. These changes were necessary to improve pacing and make sure that the days we had were polished and high-quality given the amount of time we had left to complete them.

Early on, we established a routine for tasks relating to days. Some tasks, like making breakfast, are completed every day. Other tasks depend on how recently another task was completed—you wouldn’t vacuum two days in a row. Tasks are divided into morning, afternoon, and night tasks, and there’s usually at least one of each category task in a day. We created a chore order map to keep track of what chore happened on each day, and in what order.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
smoothie	1	1	1	1	1	1	1
making lunch	2	2	2	2	2	2	
vacuum			3			3	3
cleaning*	3		3		4+		2
setting table	5	5	5	5	4 **		4
meatloaf	4	4	4	4	3 **		3
*different kinds at different points of the day							
"+": night cleaning							
**pizza night							

*Our initial weekday chore order map.*

Due to the changes mentioned earlier, we had to make significant changes to our chore order map to fix pacing and scope issues.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
breakfast		1	1	1	1	1	1
vacuum			2	2	3		
cleaning		2					2
dinner dash		3	3	4	4 **		
flavor task				3	2	2	1 & 2
**pizza night							

*Our final chore order map*

### 3.8 Progression and Escalation

An important aspect we wanted to harness in our game was the act of establishing a routine and the uneasiness that comes from breaking it. As a result, tasks were designed with “escalations” in mind—changes to the base tasks that would convey Cthulhu’s presence and the exponential descent into madness so common in Lovecraftian fiction. These escalations could be as subtle as randomizing placement of cutlery in an odd way, or as complex as chasing an evil meatloaf through the house to consecrate it with holy water. The escalations would not affect the goal of the overarching tasks, instead modifying the subtasks that the overarching tasks were composed of. The goal of making breakfast would remain the same, but would have the added step of plucking eggs and bacon out of the air to cook them while gravity isn’t cooperating.

Early in production we created a progression map that illustrated what tasks would be performed on what days, and what level of escalation it would be at. We initially decided that the first two days would be for establishing base game mechanics and ideas, then use the days after to slowly introduce some weirdness.

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
smoothie		1	1	1	1	1	1
making lunch		2	2	2	2	2	
vacuum			3			3	3
cleaning*		3		3	4+		2
setting table		5	5	5	4 **		4
meatloaf		4	4	4	3 **		3

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday	ZZZTURDAY
1	1	1	1	1	1	1	1
2	2	2	2	2	2		2
	3			3		3	3
3		3		4+		2	2
5	5	5	5	4 **		4	5
4	4	4	4	3 **		3	4

Escalation levels
very low
low
medium
high

*The first progression map and legend for our planned two weeks of gameplay.*

We initially divided our tasks into 4 stages of progression: Very low, low, medium, and high. Very low was to be so low that the player might not even notice it. For example, silverware being in a different place than usual when going to set the table. Low is a slight step above very low, relatively unobtrusive but a bit more obvious. For example, silverware and plates being in a different place than usual, then, after the player puts the silverware on the table, the plates move back to their original locations. Medium escalations start to enter the realm of making the player’s tasks a bit more difficult. Medium escalations involve objects that defy gravity, texture changes, and other oddities of that nature. High escalations make subtasks significantly different or more difficult. High escalations generally involve objects becoming sentient and or moving in abnormal ways.

As previously mentioned, after playtesting we realized we had a problem with our pace. Players didn’t notice or question the very low escalations at all, and the utter lack of Cthulhu in the early days led players to boredom fairly quickly. It turned out to be fairly easy to swap around the progression, so we were able to make quick changes based on our playtesting and get them in front of more eyeballs faster. As a result, we ended up changing the progression of the game significantly, and much more by ear than our initial meticulous planning. To indicate the presence of Cthulhu and add intrigue early on, we implemented a medium escalation— books floating in the living room, then falling to the floor—on the first day of the game. This addition gave us the immediate intrigue we were looking for, while still allowing us to maintain a relative sense of normalcy and routine.



After we cut the first week of content, we had to adjust our progression significantly once more. Due to our rapid changes, we didn't end up returning to our old progression map to make changes, instead creating a separate document to list what would happen each day. Also factoring into our decision to abandon our old progression map was the addition of flavor tasks, which didn't quite fit into our map's structure. Our new progression list can be found in our appendices. We also did away with very low escalations altogether. No playtesters seemed to notice them, and they made less sense to use now that the gameplay was condensed to one week.

### 3.8.1 Escalations by Task

As the escalations for some tasks are vastly different than the escalations for others, it is worth denoting here what escalations occur for each task.

### 3.8.2 Breakfast

For the low escalation on the breakfast task, food, pans, and plates fly out of the fridge and cupboard at the player. They float around unaffected by gravity for a few seconds before dropping to the floor, and the task progresses as normal. For the medium escalation on the breakfast task, food, pans, and plates are in zero gravity for the entirety of the task, necessitating that the player pluck them out of the air in order to complete the task. For the high escalation, food, pans, and plates are unaffected by gravity, but also "pulse" up and down every few seconds, making it significantly harder to find and utilize them.



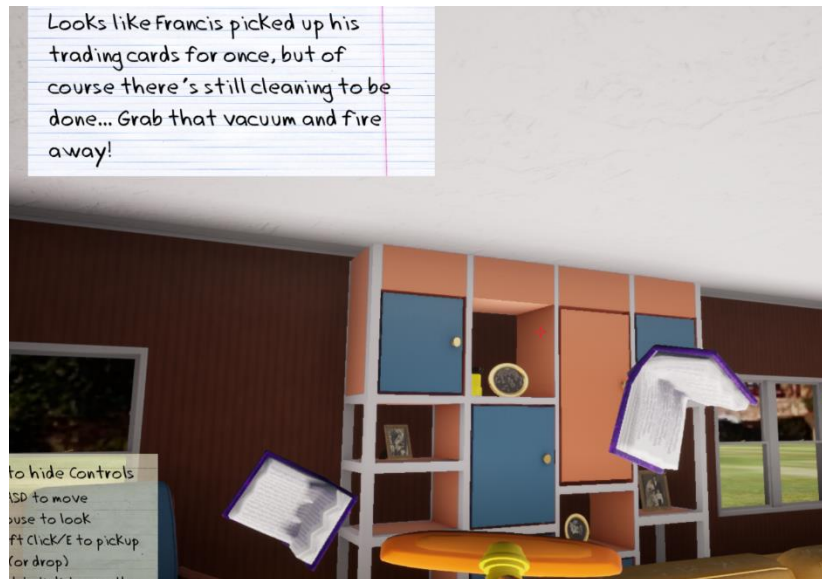
*A delicious floating breakfast.*

### 3.8.3 Cleaning Up

The low escalation of the cleaning task involved objects floating in the air, spinning rapidly, then dropping to the floor. The medium cleaning escalation was the same, with the notable difference that the objects never dropped to the floor. The high escalation utilized the modified vacuum that could shoot bullets. Books with their pages creased into fangs float in the living room, chomping in the air, and the vacuum cleaner must be used to shoot them down.



*Cleaning up in zero gravity.*



*Hungry hungry books.*

### 3.8.4 Vacuuming

The low escalation of the vacuum task involved patches of grime on the walls and ceiling instead of on the floor, and the medium escalation involved these patches of grime being transformed into patches of pulsing purple ooze. Notably, the vacuum does not have a high escalation. Considering that the vacuum is used for the high escalation of cleaning and the sealing away of Cthulhu, we didn't think it was especially necessary for the vacuum to be featured in its own task again.



*Who put slime all over my perfectly lovely ceiling?*

### 3.8.5 Dinner Dash

The dinner dash's first two escalations are exactly the same as cleaning up: floating, spinning objects that fall to the ground, then floating, spinning objects that remain unaffected by gravity. The dinner dash's high escalation involves the meatloaf turning to slime with eyes and skittering around the floor of the house. Karen must get a holy water bottle from the closet and spritz the meatloaf for it to turn back to normal, then cook the meatloaf as usual.

## 3.9 Other Methods of Conveying Cthulhu

Despite our adjustments of the progression and escalations there was still some concern that we weren't implementing Cthulhu fast or soon enough. These concerns resulted in the creation of a blueprint called the Yikes Machine, designed to be a quick and easy way to invoke Cthulhu without changing major gameplay mechanics. At a variable time averaging around 20 seconds, the Yikes Machine does something strange to the house. This strangeness lasts for about 10 seconds, then disappears. These oddities come in four forms:

- Swapping existing materials out for Cthulhu-esque materials.
- Spawning random, large objects in random places they shouldn't be. For example, an oven floating in the living room.

- Spawning pillars, spheres, and other objects composed of swirling text from the chant in *The Call of Cthulhu*.
- Adding post processing effects to the main camera like film grain, chromatic aberration, and intense bloom.

These oddities were received fairly well by playtesters, and had the added effect of invoking the Lovecraftian theme of seeing strange happenings and questioning your own sanity. Many players didn't notice the effects at first, and would often turn away before going back to get a second look to confirm what they had seen. Because the Yikes Machine effects don't last very long, oftentimes the effect would be gone before the player could get a good look at it, making them question what they had seen.



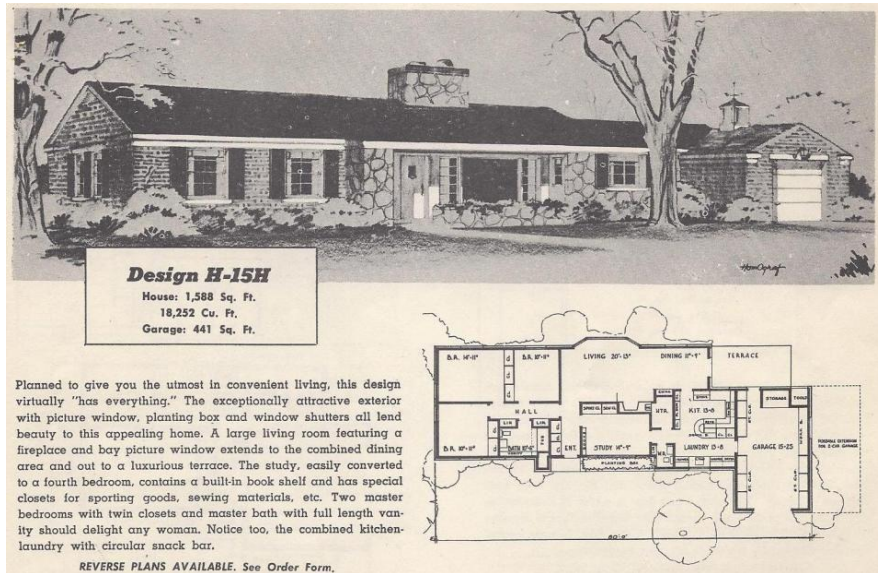
*It doesn't seem particularly ergonomic to put a chair there.*

### 3.10 Greyboxing & Paper Prototyping

Before we entered full production, we wanted to make sure that our ideas were sound. We “paper prototyped” our tasks, though with our simple control scheme, it was difficult to create an accurate 1-1 paper prototype. We used several markers and pieces of paper as stand-ins for our objects and moved around a room picking up items and putting them down as we planned for in the main game. We found this level of paper prototyping to be substantial enough to convince us that our gameplay mechanics would work, especially considering that we had some solid comparables to build off of. After paper prototyping, we began greyboxing each task and the layout of the house with simple shapes to get a feel for the core gameplay. After we were satisfied with our greyboxes, we added models and textures to make them complete tasks.

### 3.11 House layout

From the beginning, we knew that the game would take place entirely in one house. The art, design, and technical cost of creating and implementing multiple areas was outside of the scope of this project. As such, creating a quality layout for the house was important to us. This layout was something we iterated alongside our greyboxing. We wanted a layout that was both reminiscent of a classic 1950s house and designed for good gameplay. We referenced blueprints of 1950s houses during this design process, and kept the play spaces of our comparables in mind.



*Blueprints for a house layout from the 1950s.*

We realized we needed a “loop” between the dining room, kitchen, and living room. These three rooms are where most of the gameplay takes place, and thus we had to make sure it was easy to navigate between all of them. We also intended to “break” the line of sight between those three rooms as much as possible. Breaking the line of sight would allow us to manipulate elements of different rooms without the player noticing, which was key for switching between different tasks. Additionally, we decided against a second floor. Our research indicated many 1950s homes were single-story, and since all of the gameplay would take place on the first floor anyhow, it didn’t make sense to craft a new floor that would be entirely cosmetic and serve no functional purpose. We cycled through several layouts, and changed them as we went through the greyboxing process, eventually landing on the layout below.



*A top-down view of Karen's house.*

## 4 Technical Implementation

Over the course of this MQP our main goals for programming were to keep things simple to implement, easy to understand, and easy to build upon. We also hoped to make the process of integrating code and art updates as clean and painless as possible. In order to achieve these goals, we took steps to start planning the code structure for main gameplay elements such as the day/night cycle, task loading, and actor modifications fairly early, but always made sure to keep an open mind about refactoring old code when necessary. In regards to combining code with the artists' work, we tried to merge branches on Github, our source control of choice, approximately every two weeks to make sure everyone's branches were updated with the most recent art and codes changes, as well as to limit merge conflicts further down the line. However, as with most projects, we did hit some bumps along the way - mainly, issues finding Unreal resources, difficulties with integrating C++, and not quite keeping up with that "merge branches every two weeks" plan as much as we'd hoped.

### 4.1 Key Software/Tools

A variety of tools were used to enable and facilitate the development of Call of Karen. On the technical side, these fit into three major categories:

- *Engine:* Unreal Engine 4 + Unreal Editor
- *Version Control:* Git + Unreal Diff Tools
- *Documentation:* Google Drive + GitHub

### 4.2 Why We Chose PC Over VR

Pre-production for *The Call Of Karen* began in early June of 2019 and continued well into the summer, and much of that time was spent deciding the basis for this game: what was it about, what genre is it, what story do we want to convey, and **what platform will it be on?** This last question turned into a contentious debate that stretched several weeks as our team vacillated between developing the game for Windows PC or VR.

At first we intended on making the game in VR due to several reasons: the programmers had experience developing in VR, one of them had a vested interest in the area, the market was less saturated, and we thought it would be easier for the player to empathize with Karen's plight if they had to physically do the chores through VR mechanics. VR templates in game engines typically include code for interacting with objects pre-built, so that was one less task for our programmers to tackle, and there was more opportunity for playing around with sinister and/or interesting 3D sound if the player was in a surround-sound headset. Finally, we wanted to make use of WPI's extensive VR resources - the school owns a few different kind of VR headsets as well as VR backpack sets that would allow players to roam around a playspace unhindered by wires and/or lighthouses, which we thought we could take advantage of to maybe create a game that capitalizes on the larger play area and higher graphic quality of the new headsets.

Ultimately, however, we decided on PC as the primary platform - though we have not ruled out converting *The Call Of Karen* to VR eventually if the development team wishes to. The key reasons why we settled on PC are as follows: PC games are easier to setup and playtest,

they allow for more creativity in terms of art (our opinion), they are easier to access for the general market, they (usually) don't require specialized equipment and, most importantly, *The Call Of Karen* didn't **need** to be in VR except for the gimmick as far as we could tell.

Early on we decided that we wanted to get in as much playtesting and feedback as we could on this game, and to facilitate that we needed an easy setup and cleanup, as well as easy access to necessary equipment. For PC games, this amounts to having a working laptop handy and charged; for VR games, this includes wiring a VR headset to a VR capable computer or laptop, potentially setting up lighthouses, and then dealing inevitably with a variety of connection issues. All of our team members have either attended or heard of playtesting sessions that took a lot longer than needed because of VR issues, and we wanted to avoid that.

When it comes to art, our team generally agreed that VR excelled when its art style was more realistic, and we didn't necessarily want a realistic style at the time. We didn't want to limit our options in case we did decide to go a more stylistic/cartoonish route.

As to the market concerns - most good VR headsets, those that don't cause as much motion sickness and have nice graphics, run in the price range of \$200-\$1000 generally speaking. This is a huge money sink, especially for those who maybe only want to play a few VR games or only just ours, so we felt it wouldn't be reasonable to expect people to go out and buy a headset just to buy our game as well. If we decided to only focus on people that already had a headset - well, that is a fraction of the consumer market, and therefore a fraction of the number of buyers for our game as the number would be if we targeted PC gamers.

Finally, *The Call Of Karen* didn't need VR - by this we mean, there was no mechanic or story element that required the game to be created in VR and VR only. While yes, we loved the idea of doing cool things with surround sound and really immersing the player into Karen's lifestyle, all of that is also doable in a first person game with careful attention to the audio; which is subsequently what we did.

### 4.3 Engine: Why We Chose Unreal

Within the first month of development we already knew we wanted to do a 3D game, with an emphasis on excellent lighting and a polished art style. It was for these reasons that we leaned towards using Unreal Engine 4 from the beginning, as it has a very well developed art-pipeline, an easier to use and modify lighting system, and higher fidelity graphics rendering than comparable free-use engines such as Unity, in general. In addition, developing in Unreal gave us a greater chance to do well in any Epic Game scholarships such as the Epic MegaGrants program or in the Epic Game store, if we did decide to publish the game there.

Another benefit of Unreal was that its Blueprint system was relatively easy to pick up, and the object-oriented structure of the blueprint object was very familiar to both programmers. While official documentation on some aspects of blueprinting can be sparse, there is a wealth of information online in the form of community tutorials, Youtube videos, and basic Unreal documentation that allowed all team members, most of whom had little to no Unreal experience, to pick up the necessary skills to start developing the game within a week or two.



## 4.4 Version Control

### 4.4.1 Why we chose Git and not Perforce

The team knew immediately that some form of source control was necessary for this project in order to allow for all four of the team members to work on the game concurrently. In addition, this source control had to be easily accessible to both programmers and artists at any skill level, so as to minimize the learning curve and speed up production. As both programmers had extensive prior experience working with Git, in conjunction with GitHub's added collaboration tools, that was the source control initially leaned towards and used for the first 4 months of the project. Git's many different GUIs also sweetened the deal for us, because they allowed for each team member to interact with Git in whichever manner they found most comfortable and efficient. For the most part on our team, programmers stuck to using git command line, while artists used a mix of Git Kraken, GitHub Desktop, SourceTree, and GitHub Web. However, using Git's default diff tools with Unreal posed some troubling issues, such as major difficulties in merging blueprints, especially level blueprints, across the different member branches.

Perforce Helix Core (Perforce) was introduced as a potential alternative to Git, as we were advised by various classmates as well as our team advisors that Perforce was the industry standard for working with Unreal Engine on large teams ("Perforce Helix Core", 2019). Perforce is used internally at Epic Games, and at several other large game studios (Neverender, 2014). Unlike Git+GitHub, it is a commercial piece of software that is licensed for free to academic institutions. The Helix Core server is used in conjunction with P4D, a standalone front-end that runs on the desktop (not on the web). To give it thorough consideration, we reached out to Perforce about their educational licensing and obtained a trial license to set up.

The setup of Helix Core immediately proved complicated, as it required a manual on-site server-side installation of the software, for which Perforce provides minimal support to non-enterprise customers. Despite difficulties with the server, we were able to install and test out P4D (the user-facing frontend) and the merge tools packaged with it. After looking into the viability of using Perforce and the effectiveness of its merge tool, the team ultimately decided to stay with Git for the main source control. While Perforce did offer specially made merging tools, they were designed for diffing text files, and were therefore not Unreal Blueprint compatible. The minimal diff tools available for Unreal's binary file formats were the same for Perforce as for Git. We found the overall setup, management, and user interface for the Perforce repository to be much clunkier and less intuitive as well, and switching over would have provided a dramatic increase in work tangential to development of the game with no clear benefit.

Through the process of researching Perforce, we discovered a solution to our Blueprint merging issue in the form of the Unreal blueprint diff tool. Unreal's built in diff tools allowed us to compare the current version of a blueprint with previous versions from the same git branch, as well as versions on other branches, to resolve any merge conflicts. In previous versions of the Unreal Editor, these diff tools were integrated with Perforce and SVN, but not git, which was likely what led to the misconception that Perforce provides diff support for Unreal. As of UE4.22, git support is in beta but fully functional, and integrated well with our established workflow. This tool has been essential in reducing lost and/or duplicate code due to merges.

## 4.4.2 Workflow

Our approach to having multiple people collaborating on an Unreal project through Git was to: always have a working copy of the game on master, have everyone make changes on branches, and merge as often as was feasible. Master was locked to ensure that no one person could push to it by accident and overwrite our production copy of the code - to merge changes into master a team member had to create a merge request which was then approved by at least two other members, typically the programmers. Generally speaking each team member would maintain 1-2 personal branches from master, with each member communicating what they planned to change. It was extremely important that everyone on the team knew what the others were up to in order to avoid multiple people making the same edits, or even more fatal, multiple people making edits on the same file. In an effort to mitigate the dangers of overlapping edits, each member usually created their own sublevel in Unreal to put in their changes if they were significant enough, i.e. not small bug fixes or material changes. Using sublevels allowed each team member to make changes to the map in scene without affecting the main Game.umap binary file, or any other sublevel binary file, both of which are not able to resolve merge conflicts. When it was time for someone's work to be merged into master, one or more members would handle moving the temporary sublevel changes into the main Game.umap, and the temporary sublevel would then be deleted.

When it comes to merging overall, we aimed for a complete merge every one to two weeks - we rarely, however, met that goal, until the final few weeks before our beta build was completed. At the beginning merges were less frequent due to work going slower, but as time wore on it became clear that combining the work of four people at once was much more daunting than anticipated. Unreal's reluctance to work well with Github only exacerbated the issue, with virtually any file worked on by two people needing to be manually merged in order to avoid work being lost - we understand now, why Perforce has the policy of only allowing a file to be checked out by one person at a time. The frankly frustrating amount of work that went into each merge caused a push back of the merge process which resulted several mega-merges which lasted multiple days. Thankfully though, as we approached spring and became more comfortable with each others' development time and processes these mega-merges became fewer in number. In addition, we discovered the Unreal Diff tool, which allowed us to more effectively solve merge conflicts on blueprints - however, this was only within the last few months of development, and **only** effective at solving clear-cut merge conflicts on blueprints. If there was a merge conflict on any of the map files, the system config, or any other binary files, it was often a case of copying over as much as we could remember and hope we discovered introduced bugs before they went into production. The technical details of these issues are discussed in more detail in the Implementation Difficulties section.

The software we used for source control was all variations on Git - the programmers stuck exclusively to the git cli, more or less, while the artists started off using Git Kraken, and switched to Git Desktop and SourceTree, which they found to be more comfortable to work with Unreal. Overall, our workflow decisions were largely informed by Unreal's inability to integrate properly with these tools.

## 4.5 Game Structure

### 4.5.1 Actors, Components, and Triggers - Oh My...

All actors in this game which have any form of interaction were based off of two actor classes defined early on in development: `ReactiveStaticActor` and `ReactiveSkeletalActor`. The main difference between these classes is that `ReactiveStaticActor` contains an inherited static mesh component, while `ReactiveSkeletalActor` contains an inherited skeletal mesh component - this actor type was created specifically for objects which had animations, such as doors needing to open/close and the monster books, as static mesh components do not support animations. `ReactiveStaticActor` also contains inherited `Escalation` and `RotatingMovement` components for ease of applying escalations to task objects on later days.

Both `ReactiveStaticActor` and `ReactiveSkeletalActor` defined custom component events which would pass actor events such as `EventOnHit`, `EventOnOverlap`, etc. from the actor to its interactable components. In addition to this, custom events were also created for `LeftMouseButton` and `RightMouseButton` to pass down these event calls to components which would apply mouse functionality to certain objects.

As for components, the programmers decided early on in development to create a base interactable component from which all other components would inherit functionality. The only exceptions to this were the `Escalation` component, which does not have any interactivity functions, and the `CinematicComp` component, which handled screen cinematics such as fade ins and fade outs, and was attached solely to the `TaskManager` actor.

The first component created was `ActorEventHandlerAbstractComponent`, which defined an abstract function to catch the custom component event `ComponentEventHit` defined in `ReactiveSkeletalActor` and `ReactiveStaticActor` for children components to define on their own. Directly parented to this is the `Interactable` component, which the rest of the interactables inherit. Inside `Interactable` the custom event `EventInteract` was created, which defines how to treat object interaction depending on the type of actor (static or skeletal) and the other components the actor has. It also ensures that all objects which have a component of type `Interactable` have a few important characteristics such as simulating physics and physics body collisions in order for these actors to act realistically in game.

`Pickup` and `Tool` are the two components defined from `Interactable`, with `Pickup` inheriting directly from `Interactable`, and `Tool` inheriting directly from `Pickup`. The `Pickup` component, as it indicates, defines functionality for objects which the player is able to pickup - in addition, in the `FirstPersonCharacter` function `EventPickup` the object the player is attempting to pickup is always checked to make sure it has this component present. Typically the behaviors defined in `Pickup` relate to pickup objects which don't have any other defining features, such as cards, pencils, the cardtin, etc., and only affects how they act on `EventComponentHit`.

`Tool` defines behaviors for objects which carry out an action on right click - "Right click to use tool", as the game informs. There are nine tools defined in-game, with one or two that have duplicate functionality but utilize different assets - for example, "Salt" and "Pepper" are separate tools, but use the same function on right click with different particle materials. For the most part, the tools in game complete fairly simple actions - play a sound, create/delete an actor, call another quick function, change the material of its owner and/or another actor in-scene. Often the



- GameManager fires OnTaskChanged
    - Old task levels unloaded, new task levels loaded
      - Unload task/lighting sublevels for old task if applicable
      - Load task/lighting sublevels for new task
        - If no next task call QueueNextTask for next day
  - GameManager fires OnTaskLoaded
    - GameManager receives and sets up UI
    - TaskManager receives and sets up gameplay for current task
    - RadioManager receives and starts up audio for current task
  - *Player is free to move around/interact with gameplay Actors*  
 Gameplay Actors call GM\_SetValue
    - GameManager changes value in value store
    - GameManager calls OnValueChanged
    - GameManager calls CheckTaskComplete  
If task completion criteria met, fires OnTaskComplete
      - TaskManager Receives, sets up post-task UI/Task if applicable
      - RadioManager Receives, starts up end of task audio
        - If last day, break main game loop
        - Upon completion, call QueueNextTask for next task (*loop*)
- Open level Credits

### 4.5.3 TaskManager

While the entirety of our gameplay logic was implemented using blueprints, in part for ease of use by the artists, and in part to speed up development and compile times, some systems proved more reasonable to build using C++. One of the core mechanisms of the task-based gameplay was evaluating whether or not a task or subtasks had been completed. The parameters for this completion generally involved the state of different gameplay objects (e.g. whether the card tin is in front of Francis's door), higher-level game state (how many books have been placed on the shelf), and user-triggered events (hitting a plate with a fork). Checking completion for a task requires either storing all of this state data in one place, or having references to each individual object. We opted for the former, as it was easier for several different Actors or Components within the scene to dynamically get references to a particular object than to attach references of each of those objects to an instance of a single Actor class.

TaskManager initially housed all of this data in a pure blueprint format. Game state was stored in the form of Blueprint variables (mostly booleans and integers), as was the names and orders of tasks each day, task escalation levels, and day-to-day progression (arrays/maps of strings and asset references). The benefit of this format was that it was quick to prototype chunks of gameplay in order, and everything was laid out sequentially, making it easier to debug than a more abstract implementation.

As the amount of days/tasks and similar mechanics increased however, this resulted in large amounts of duplicate blueprint code and a lot of rewriting or refactoring those chunks of gameplay in each individual instance. This also meant that every blueprint class that had code affecting task completion had to search for and store a reference to the single instance of TaskManager, and update a corresponding variable manually defined in the TaskManager

blueprint. Additionally, events to check task completion for each task had to be defined individually in TaskManager per task, and in some cases with separate chains per game day/escalation. This led to major blockers and version control headaches, as the binary and non-diffable blueprint file that comprised TaskManager had unmergeable edits any time one of our programmers wanted to add core gameplay.

#### 4.5.4 C++ Integration and BP\_GameManager

Our initial TaskManager implementation helped to clearly identify the key benefits and drawbacks of using the blueprint system within our game:

- Sequential gameplay was quick and easy to wire and troubleshoot in blueprint
- Less sequential/more abstract mechanisms such as level and UI loading became tedious to reimplement repeatedly

To address the latter issue, we created the GameManager using Unreal’s C++ API. This enabled a more abstract representation of our game state in the form of an in-memory key-value store. Each key was a unique string representing a game state variable that corresponded with an integer value. This is a simplified form of the kind of state management used in many modern web application frameworks such as React (“React State”, n.d.). Implementing custom getters and setters for these state variables allowed us to do a couple of convenient things, such as defaulting values to zero (removing any need for initializing or checking the presence of a value), and firing off events when a game value was updated.

With all state updates happening through a single function, this also facilitated checking task completion - all of the keys and target values associated with completion of a task could be stored in a Datatable, and checked only on a TaskUpdate instead of on tick or by manually triggering a custom defined event in TaskManager. State value updates, task completion, and level loading completion were all exposed via Event Dispatchers that any blueprint could bind a custom event to immediately on creation (“Event Dispatchers”, n.d.).

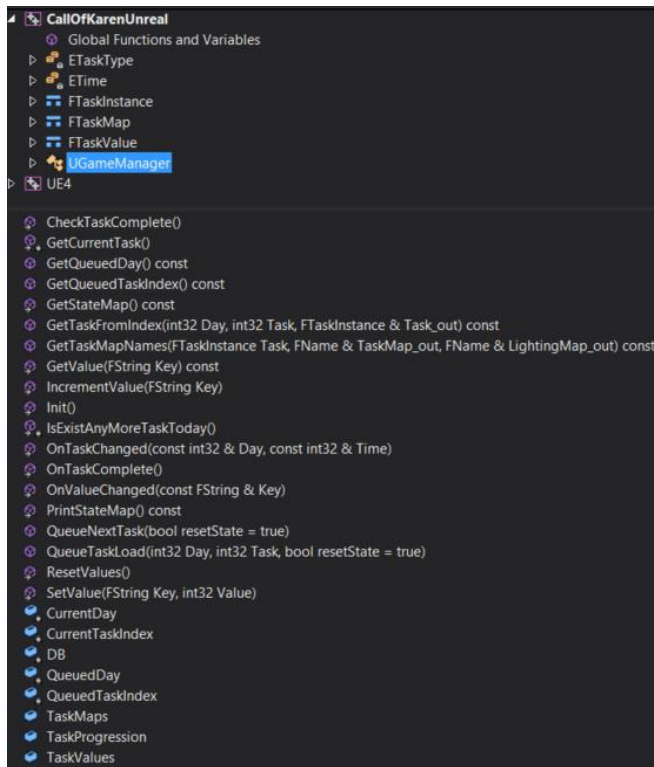
A key aspect of GameManager was its inheritance from the UGameInstance engine class (“UGameInstance”, n.d.). The GameInstance subclass designated in the project settings as the game’s primary GameInstance is instantiated automatically, and accessible statically from any blueprint class, including animation blueprints. This meant we no longer needed to search for and store references to the object holding our game state, and could simply obtain the engine-level reference to it on the fly to trigger/bind to events and update state.



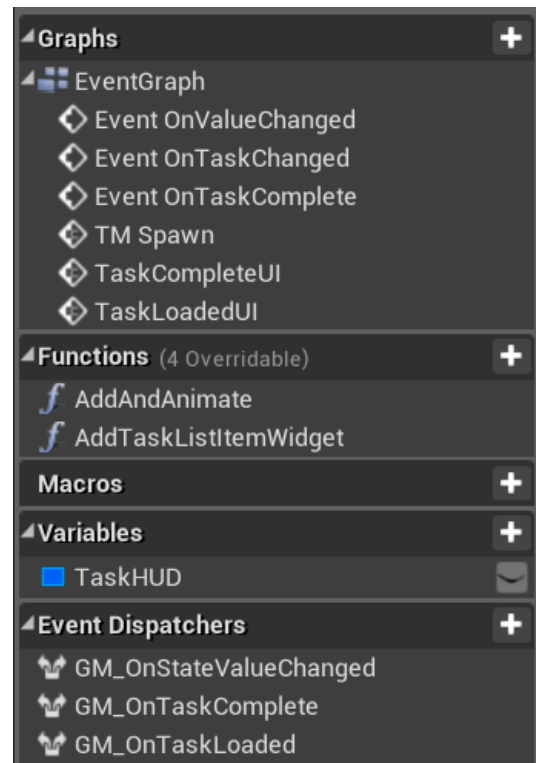
As established in our analysis of TaskManager, sequential logic was easier to integrate in Blueprint. While level loading fell into this category, certain aspects such as querying datatables and defining events were cleaner and easier to do in Blueprint. We addressed this through the creation of BP\_GameManager, a subclass of GameManager that defined any sequential logic (such as level loading) with helper functions for more abstract operations defined in GameManager. Features that made more sense to implement in Blueprint were declared as abstract functions in the C++ layer and overridden in the Blueprint layer. The

blueprint layer additionally provided an abstraction on the self-contained C++ functions that didn't need to be called elsewhere, and resulted in a much simpler interface for state management and level loading.

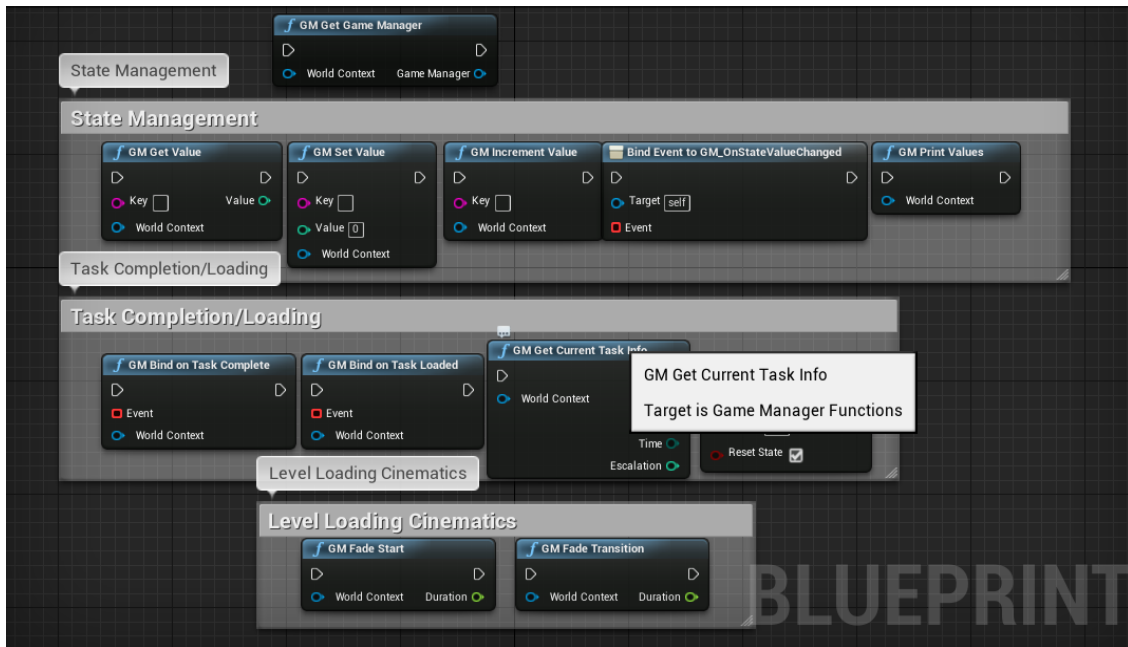
The result was a simple and clean Blueprint interface to use within the rest of the gameplay code for state management and task completion/level loading. Using a static function library, we made all of our key functions and event bindings accessible globally using a single node. Having the ability to update state without declaring a blueprint class variable in TaskManager, getting a reference to TaskManager in the corresponding blueprint, and checking that variable before getting/setting it saved us a lot of time and cut out unnecessary spaghetti.



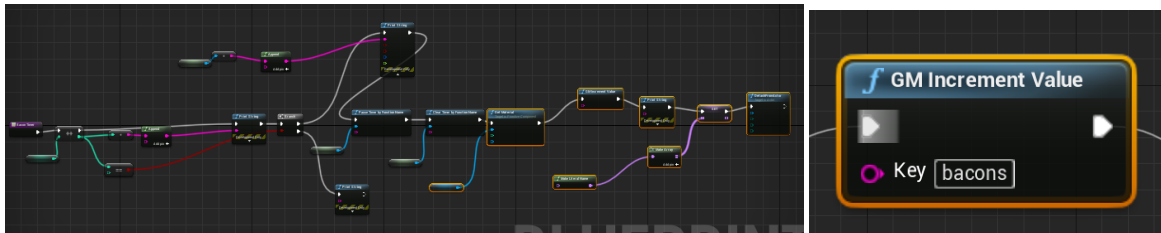
*GameManager (C++) Definition*



*BP\_GameManager (Blueprint) Definition*



*Final GameManager Interface (These nodes can be used in any blueprint)*



*Handling Egg Cooking Without vs With GameManager State*

#### 4.5.5 Data Storage: Data Tables and Dictionaries

All major data was stored inside a series of .csv files, some of which were backed up as Google Excel sheets on our shared team MQP drive. These .csvs were then imported into Unreal as Data Tables for use in engine. The type of data stored can be split into audio, and task management.

Audio tables included the MusicTable, which consisted of all the music clips used in game, DaySoundAudio, which was a table of dialogue from all characters keyed to their approximate order of use in game, and the SoundTable, which was where all the general sound effects were stored. An overarching DialogueSequence table kept track of which audio files, mainly dialogue interspersed with some sound effects, were to be played at what point during the game one after the other to ensure smooth transitions between audio clips.

Task management tables included TaskMaps, TaskProgression, TaskValues, and WidgetStrings. TaskMaps organized the sublevels used for each task in game and keyed them to their task names for ease of loading them on the fly, while TaskProgression kept track of the



timeline of the game - what tasks were played which day, in what order, what their escalation (difficulty) level was, and what lighting state was required to be loaded for a certain task. TaskValues was where all the task completion variables, along with their target values, were stored and updated in game - for example, in order to complete the vacuuming task the player must vacuum up all the spots of grime to reach the target number of dirt cleaned, so a "dirt" variable in the TaskValues table was updated on each spot removed to evaluate if the player had completed the task. On each update to a task variable the SetValue function called an internal CheckTaskComplete function to ascertain whether or not all variables associated with the current task had reached their target values. If so, an event TaskComplete would be fired from GameManager and received in either the TaskManager or RadioManager blueprints to be acted on (played task end audio, remove objects, load next task, etc). Widget Strings contained the keyed values for different task UI text, and was used by the base UI template to create dynamic UI in-game.

## 4.6 Gameplay: How was the game made?

### 4.6.1 Day Night Cycle and Task Loading

The gameplay loop of *The Call of Karen* consists of a repeating day-night cycle with a series of tasks/mini-games occurring on each day of increasing difficulties. Getting this loading of tasks and their associated escalation levels to work was our first priority, and the initial method for driving the day-night cycle was completed within a few weeks of development. This method relied on storing the data for each day, its list of tasks, and the escalation level for each task in a series of dictionaries (key value pairs) and ordered lists, which were passed into a function in TaskManager to be acted upon. Each task had a dedicated sublevel for ease of switching tasks and for ease of having multiple people work on the project at the same time, so the day loading system took advantage of this and set up each task by loading its sublevel and any UI that went along with it. This day loading system, while clunky, was effective for the initial prototyping phase.

During the latter half of development it became clear that certain things initially built in blueprint would be cleaner and more efficient/maintainable if built in C++ instead - one of these things was the day-night cycle framework. As mentioned previously in the section on C++ Integration and BP\_GameManager, one of our programmers took it upon themselves to do a partial rewrite of the entire cycle system, moving data such as the tasks and their level information into data tables for easier and neater retrieval, and setting up an event system which could be passed into blueprints to be acted on.

### 4.6.2 Escalations and AI

One of our main aims when it came to programming this game was to keep the implementation fairly simple and easy to build off of. Towards this end we planned the escalation system so that when it came time to add higher levels of escalations, and therefore higher levels of difficulties, all it would take was simply adding in a call to the next level function, i.e. call Level2 to apply level 2 escalations. Almost all of the escalation code was encapsulated

into a custom Unreal Component called Escalation, which was made an inherited component for all objects which used our standard actor types ReactiveStaticActor or ReactiveSkeletalActor. Each Escalation Component has an associated escalationLevel which would be set on task loaded in order to tell the component which level function to call and apply changes to the owner actor's behavior. Originally the reasoning behind having an individualized escalation level for each actor was to avoid accidentally applying escalation effects to other ReactiveActors in scene which didn't need them - for example, while we wanted all the food, plates and pans in the breakfast task to start floating and jerking around on escalation level 2, we didn't want the same thing to happen to the fridge or the spatulas in drawers, both of which were ReactiveActors and therefore also had Escalation Components of their own.

We ended up with three escalation functions - levels 1, 2, and 3 - along with three other general use escalation functions: normalize, randomizeLocation, and EXID. RandomizeLocation and normalize are fairly self-explanatory. The former takes in the current task and randomizes the location of all the objects associated within a given space in the house. The latter removes any and all effects on objects in a task that have had an escalation applied to them; this was used often when there was a need for floating objects to drop ominously on triggers, or to make it easier to complete certain escalated tasks such as putting the meatloaf in the oven when there is an antigravity escalation applied to it. EXID was a custom function to create the escalation for breakfast on day 6, which caused the gravity to reverse every 1-2 seconds to create a sort of wacky "possessed breakfast food" feel for the final day before the Cthulhu showdown. The functions Level1, Level2, and Level3 used a combination of these generalized functions, as well as custom features such as applying impulses and/or rotation vectors to objects to get more interesting movement patterns.

There were some unique cases where the escalation was not applied by the Escalation Component, but rather created dynamically on level load - this was typically because the escalation was so different from the baseline task, and/or utilized so many different objects, that it didn't seem worthwhile to try and fit the functionality into the component itself. In addition, often these tasks required the conversion of objects already in the level, such as the meatloaf and books, from ReactiveStaticActors to ReactiveSkeletalActors or Pawns so that animations and/or AI navigation could be applied to them. These instances were the day 4 dinner task, which utilized Unreal's AI framework to create a meatloaf that runs around, and day 6 cleaning, which spawns floating, chomping monster books which must be shot down to complete the task.

The AI meatloaf used for day 4 dinner was created using an Unreal Pawn as the base actor class, and uses a combination of a NavMesh Volume in the main map and pseudo-random movement within a defined radius to move about the level. For this task we didn't need particularly complex AI, as we never aimed to make tasks so difficult as to be frustrating to the player, or require a lot of dexterity, so the tricky-but-doable challenge of catching a twitchy little meatloaf rat that the AI created worked perfectly. This meatloaf character was always present in the dinner sublevel, but only made active on day 4.

Day 6 cleaning required the complete deletion of the normal objects being cleaned - pencils, papers, normal books, and cards - and the spawning of new monster books, which then had level 2 (float and no drop) escalation applied to them. As mentioned previously, in order for an actor to have an animation sequence attached it needs to have a skeletal mesh, so all of

these books needed to be spawned in as `ReactiveSkeletalActors` with the monster book animation sequence attached and playing.

### 4.6.3 Movement System

The movement system utilizes the Unreal first person shooter template's system, with some tweaks. In order to make the movement feel more grounded and less like a classic shooter game, the jump height was decreased while the player mass was increased. In addition, the default walking speed was increased so that walking around the environment wouldn't be a chore for the player. To accompany this, two additional movement options were added to further aid players in navigating the playspace: a crouch capability, and a run capability.

How to deal with object collisions while moving was a large part of tweaking the movement system, as often times in *The Call of Karen* the player finds themselves walking around a house strewn with various obstacles and items on the floor and next to walls. Ultimately, we decided on having any small object on the ground ignore collisions with the player character, while still generating overlap and hit events in case the object needed to respond to colliding with the player in any way. This method of dealing with collisions allows the player to navigate the house without "tripping" over any objects on the ground or getting stuck on relatively large objects such as books, resulting in a smoother navigation experience. Very large static items, like kitchen counters and living room furniture, still collide with the player character and block any attempts to overlap in order to maintain realism.

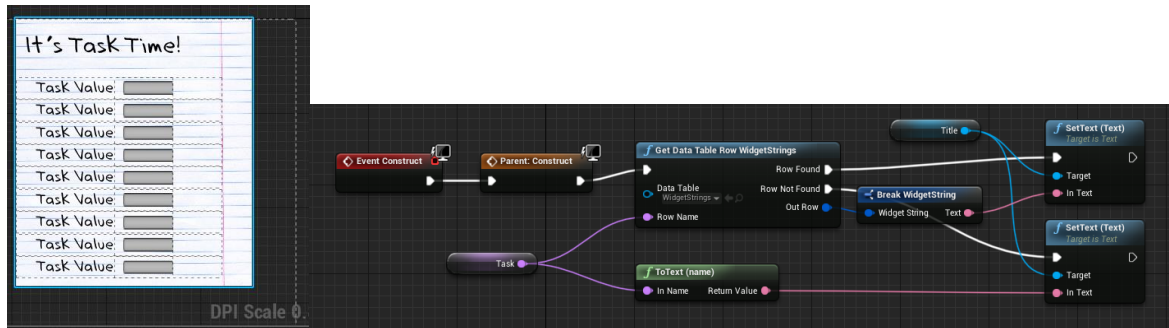
Initially, an aspect of the movement system was the ability to "step onto" any object in the game, including kitchen counters, tables and chairs. We had believed that leaving that ability in would give the player more freedom of movement around the play space and allow them to reach into perhaps more difficult spaces in the house, like above the fridge or in the very center of a large table. However, after receiving feedback from Alphafest in November in which several people complained about "jumping onto the counter" as if it were a bug, we ultimately decided to take out jumping onto objects and reconfigured some game objects' positions so that they would still be reachable by the player within the map.

### 4.6.4 User Interface

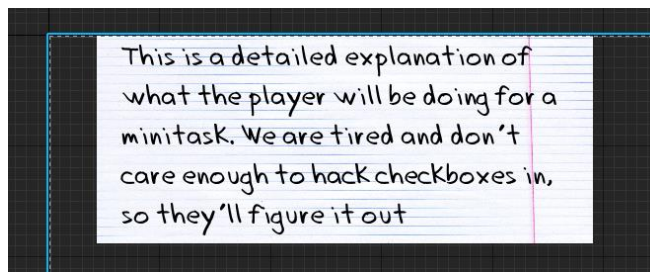
With our heavily task-based gameplay, effectively communicating how to complete those tasks to the player was critical. While our early prototypes had on-screen checklists, these were individually hand-built UI widgets for each task, which proved to be unscalable, and difficult to orchestrate in Blueprint. Our solution was to utilize a few data-driven Widgets, with UI strings stored in data tables so that we could focus on the gameplay and content of the UI and not have to worry about building out new UI upon completion of each new gameplay element.

Unreal allows UI widgets to inherit properties and structure like with any other Blueprint class, which allowed us to build a few generic, data-driven UI elements that could be populated from our existing `TaskValues` datatable on task load, and update themselves from the game state. Our base widget, `FadeInWidget`, contained no structural components but animations for fading/sliding on and off screen. `TaskWidget`, a child of `FadeInWidget`, contained a title element and a list of `TaskListItemWidgets`. Since our task loading procedure already loaded/parsed the data for the current task and the `TaskValues` associated with completion, all we needed to do

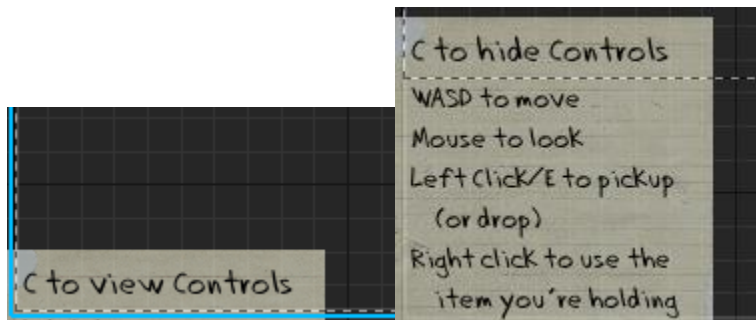
was pass this data into the widget constructors and stick them on the screen. TaskValues in the form of egg\_count were correlated with UI strings like “Cook 2 eggs” in the WidgetStrings datatable.



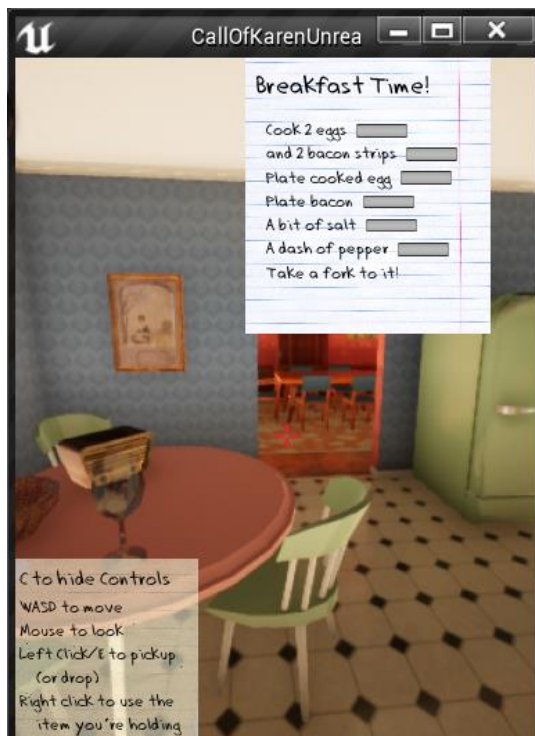
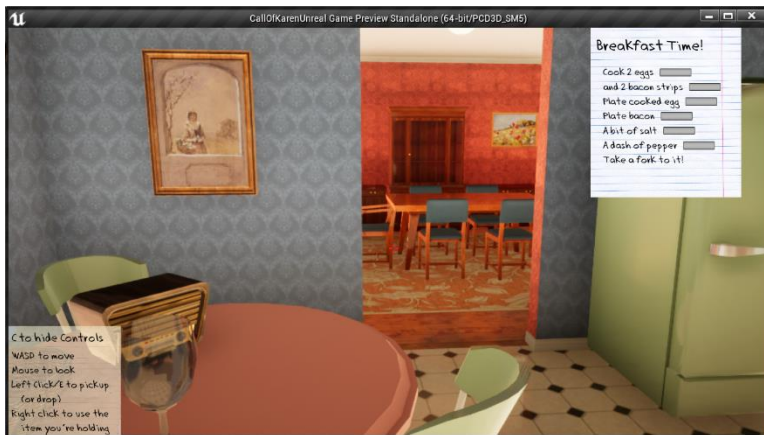
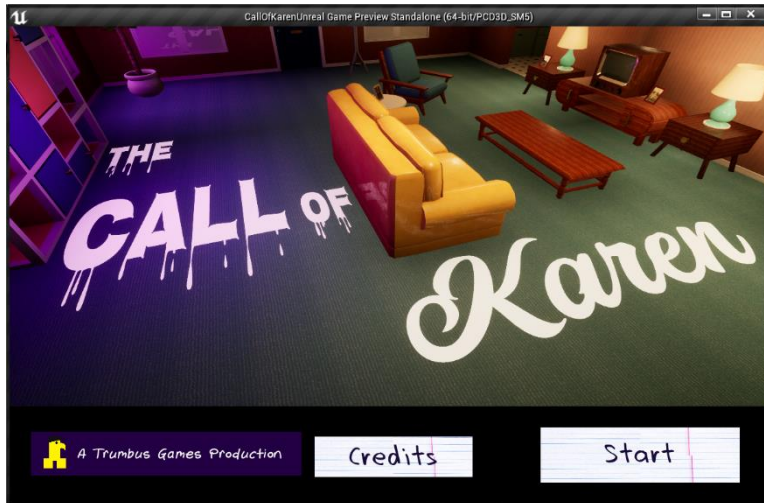
For user instructions that did not conform to this format, we created a simple popup widget inspired by toast notifications (Brocka 2016). This allowed us to provide added story and hints to the player during mini-tasks such as ordering silver bullets and writing a sarcastic thank you note to Susan, without having to add the overhead of making those items full-fledged tasks with TaskValues, dedicated levels, etc (or modifying the task and level loading systems to accommodate them).



In a similar vein, we added a persistent controls popup at the bottom left corner of the screen so the player could easily reference it if they forgot any of the controls. This popup appears at the start of the game and can be hidden/shown by pressing a single key (with a lingering but non-distracting prompt remaining in the HUD while it is hidden).



Unlike our prototype UI, all of these UI elements as well as active menu/credits UI elements were anchored to different corners of the screen, so that they scale to dramatically different screen sizes and aspect ratios effectively.



## 4.7 Implementation Difficulties

### 4.7.1 Blueprints vs C++

When we first began development in Unreal the blueprinting system was very easy to get used to and get working fast – we sometimes had prototypes for tasks up in a matter of hours. However, as we continued to work and wished to create more complex systems, such as the loading systems for days and task sequences at different difficulties, it became harder to implement things quickly and efficiently. Simple tasks that could be accomplished by writing 3 lines of C++ code took up enormous blocks of “spaghetti” blueprint space, and were basically unintelligible to the artists - even between the programmers, sometimes, it was difficult for one to read the other’s blueprint code. Blueprint functions become sprawling, messy masses very quickly, which led to much confusion, duplicate code, and bugs that should have been easy to fix, but weren’t. In addition to the incomprehensibility, blueprints also execute slower than C++ when it comes to looping functions, map loading functions, and functions with heavy mathematical equations. This slowdown became extremely apparent as more objects appeared in scene that needed to be looped over in code for modification, and posed a serious risk of slowing down our game to the detriment of the overall experience and quality.

### 4.7.2 Git Merge Data Loss

Imagine banging two different kinds of rocks together until they become one rock - except you have no hands and can’t see the rocks. Eventually you just give up and go find a different rock. That is how version control integration works for Blueprints in Unreal. Our programmers lost as much as tens of hours of work at a time over merge issues, and though they could write a standalone paper on it, have opted for a brief summary instead.

The root cause of this problem is the way Unreal handles asset storage. Assets (\*.uasset), maps (\*.umap), and most notably, blueprints, are stored in a binary format. This means separate changes to the same file can not be compared or merged by version control tools such as git. While Unreal provides a Blueprint merge tool that identifies conflicting changes, it does not allow you to merge them (even if they are functionally able to be merged cleanly), nor does it display this information in a useful way. The user can view sections of blueprint in the current local branch, the branch being merged in, and the most recent common ancestor in the version tree - but these graphs are read-only and include unintentional changes like accidentally moving and disconnecting/reconnecting nodes, resulting in the edit history being pretty much unreadable. File changes are also not refreshed without closing/reopening the diff tool, resulting in additional lost work unless the affordances of the tool are completely ignored and the “merge” and “save” buttons never used. To top it off, the merge tool frequently crashed the editor or even the computer when left open for extended periods of time (presumably some sort of memory leak).

Maps and assets, unlike Blueprints, could not be merged in any capacity, and if modified on two separate branches, one set of changes always needs to be entirely scrapped and redone. We were able to mitigate some of this by employing strategic workflows, such as having artists make asset changes by copying actors into sublevels, and manually merging those back

in. While this reduced the amount of work that needed to be redone from scratch, the process of manually comparing actor properties and overwriting/copying data between individual actors was extremely tedious. Some of our larger art merges took 6-8+ hours of programmers/artists going through several weeks worth of work together, and in a couple of cases multiple days when combined with the obscure data corruption bugs described in the next section.

For assets, there was no way to resolve merge conflicts, and it was time consuming to wait for each others' responses before editing a particular asset file to prevent conflicts. In many cases where conflicts were introduced unknowingly large amounts of work were lost and sometimes unnoticed for weeks, resulting in repeated "didn't we fix that 3 weeks ago?" moments (particularly when modifying collision boxes and animation properties, which are all stored in the same mesh asset file by Unreal). In some cases changes just flat out got lost/disappeared in merges, and had to be redone several times over the course of subsequent merges. At times, it almost felt as if our meddling Cthulhu gameplay mechanics were bleeding into our development workflows.

### 4.7.3 Semi-Random Data Overwrites and Deletion

An unexpected and unwelcome feature of Unreal that the programmers experienced often was data being overwritten and/or deleted randomly in the Unreal editor. Typically the victims were actor components and tags. Sometimes while working in the editor, either between builds or between reopenings of Unreal, actors would lose characteristics without explanation. This issue has been reported by other Unreal users in various contexts, though the cause remains unclear and there were no documented solutions that worked for us (Roberteker, 2018). Often this meant that upwards of two to three dozen objects would lose the ability to be picked up, or react to hit and/or overlap events correctly, and would in general cause unexplainable bugs until we caught on to what was happening. The issue could be solved by resetting the changes on whatever sublevel map the error occurred on. However, this usually wasn't a viable option because of essential changes that needed to be kept on that level. The result is that the programmers spent many hours redoing settings and replacing missing code several times over to fix issues that shouldn't have even occurred in the first place.

We couldn't find much information as to the cause of the data deletion - our best guess is that it's a side-effect of either Unreal's backup map data system, or an obscure memory-related bug in the implementation of the editor. Either way, it is not an exaggeration to say that between the two programmers a significant part of the level changes in this project was redone two to four times. In addition to the number of times blueprint data was lost to difficult merges, somewhere around 25-50% of the code was rewritten more than once, and in some cases several times.

## 5 Art

In this chapter, we cover all of the artistic choices that went into shaping the look of *The Call of Karen*. Chapter 5.1 details other works that inspired the look of our game, chapter 5.2 discusses our use of color, chapter 5.3 describes our asset creation process, chapter 5.4 discusses our use of and process for creating technical art, and chapter 5.5 discusses our user interface. Our user interface is only discussed at a cosmetic level here; for more information on user interface from a technical standpoint see section 4.6.4.

### 5.1 Inspiration and Concept

Early in the art design/visual development process, we wanted to be able to point to a select set of games as reference for our art style. We agreed our game should look pseudo-realistic; high quality assets paired with saturated colors and simplified textures were our initial ideas. Our first comparables we used were *What Remains of Edith Finch* and *Firewatch* (Campo Santo, 2016; Giant Sparrow & Annapurna Interactive, 2017) .



*What Remains of Edith Finch* (2017)





*Firewatch (2016)*

The former was used as a guide for overall level layout, since most of *Edith Finch* takes place in cramped interior environments, and it also established the quality level we wanted to have for our assets. *Firewatch* was primarily used for lighting reference. One compromise we made was that we didn't want our game to be choked with clutter, as many areas of *Edith Finch* are. We wanted players to understand what was and what was not interactable, and we felt that excessive amounts of clutter would work against this principle. Additionally, reduced clutter meant reduced art load, allowing us to focus on polish, lighting, and making sure the assets we did have were high quality. Additional style references included *Bioshock Infinite*, and *The Flame in the Flood* (Irrational Games & 2K Games, 2013; The Molasses Flood, 2016).



*Bioshock Infinite (2013)*



*The Flame in the Flood* (2016)

*Trover Saves the Universe* has the distinction of being one of the few comedy-centric games we used as a comparable. In general, comedy games seem to incorporate more colorful palettes (Squanch Games, 2019).



*Trover Saves the Universe* (2019)

Some feedback we received during the MassDiGI Game Challenge involved presenting *What Remains of Edith Finch* as a comparable. We were encouraged to swap it for a different game, principally because the eminence and critical acclaim of *Edith Finch* overshadowed its utility as a comparable; we agreed that comparing *The Call of Karen* to *Edith Finch* established unreasonably high expectations for our game. Thereafter, we decided to instead use *I Expect You to Die* as our main style and layout comparable (Schell Games, 2016).



*I Expect You to Die (2016)*

This well-known VR game is well regarded, and most importantly, smaller. Its similarities to *Edith Finch*, as well as the reduced amount of clutter in the game, made it a very apt choice as a comparable.

Additional comparables were also established for how we wanted to implement the presence of Cthulhu and corresponding milieu of eldritch disturbances. *The Sinking City* and *Darkborne* were our earliest comparables in this area (Bertz, 2019; Frogwares, 2019). Our initial thoughts on incorporating these elements involved the creation of pentagrams and symbols to cover walls, much like this example from *The Sinking City*.



*The Sinking City (2019)*

Fully committing to a 1950s style presented us with some new and interesting challenges when it came to art. Not only did we have to create a convincing house, but we had to make it something that truly evoked the 50s. We were initially tempted to utilize our own knowledge of the 1950s and furniture around our own homes as primary references, but were able to quickly find 1950s references that were more reliable and period accurate.

This need to evoke the 50s presented some interesting challenges, as sometimes props and layouts that were period accurate just didn't "feel" like the 50s. In order to create something that truly evoked the feeling we were going for, we had to account for everyone's nostalgia goggles. For example, many appliances that we included (the blender, electric stove, and dishwasher) were met with skepticism because, even though they were around in the 50s, many people didn't think they were, and thus they seemed historically inaccurate, despite not actually being so. Few people took too much issue with this, however, and some of these problems solved themselves. While utilizing the dishwasher was present in our initial design plans, we never ended up actually using it, so it seems like another cabinet and nobody really pays it any mind. The blender task was changed to breakfast (for a number of reasons, including period-accurate skepticism), so that problem took care of itself. As for the stove, not too many people batted an eye at that, and the way that the stove was textured meant that it matched the rest of the kitchen quite well anyhow.

Accounting for nostalgia was one part of the concepting process, but just as important was making sure we had period accurate references. We relied fairly heavily on the internet at first, but after a fruitful trip to the library, one of our artists was able to procure a book from the 1950s about how to be a good interior decorator. The *Good Housekeeping Book of Home Decoration* (Brandt, 1957) became an invaluable art resource. The book was chock-full of useful references and pictures from the 1950s. It had information on how every room in a house should be decorated, including many useful reference pictures. When creating props and furniture for our environment, this was the reference we used the most.



*On the left a typical 1950s kitchen, and on the right The Call of Karen.*

## 5.2 Colors

We aimed to keep our use of color deliberate and readable. Each room in Karen's house is a different color, keeping in line with sensibilities of the time. We also made sure not to texture any regular objects or walls purple, a color we left exclusively for Cthulhu. Our use of purple in this way was inspired by the use of saturated pink and blue in *The Legend of Zelda: Breath of the Wild* (Nintendo, 2017). The previously mentioned colors are almost exclusively used to denote danger posed by Ganon and his lackeys. They stand out in comparison to the other colors in the landscape and immediately alert the player that something is amiss.



*The Legend of Zelda: Breath of the Wild* (2017). Note the bright colors of the guardians.

We wanted to accomplish the same effect with the color purple. Cthulhu's iconography is historically portrayed as green, however, we found green to be fairly sickly and not unearthly enough. We found purple to be a choice that fit our mood much better, as it evokes danger, but not as much as red, and sticks out amidst the rest of the house. We used purple in both lighting and texturing in situations where we wanted to convey Cthulhu was present.



*Purple lighting during the day Karen confronts Cthulhu.*

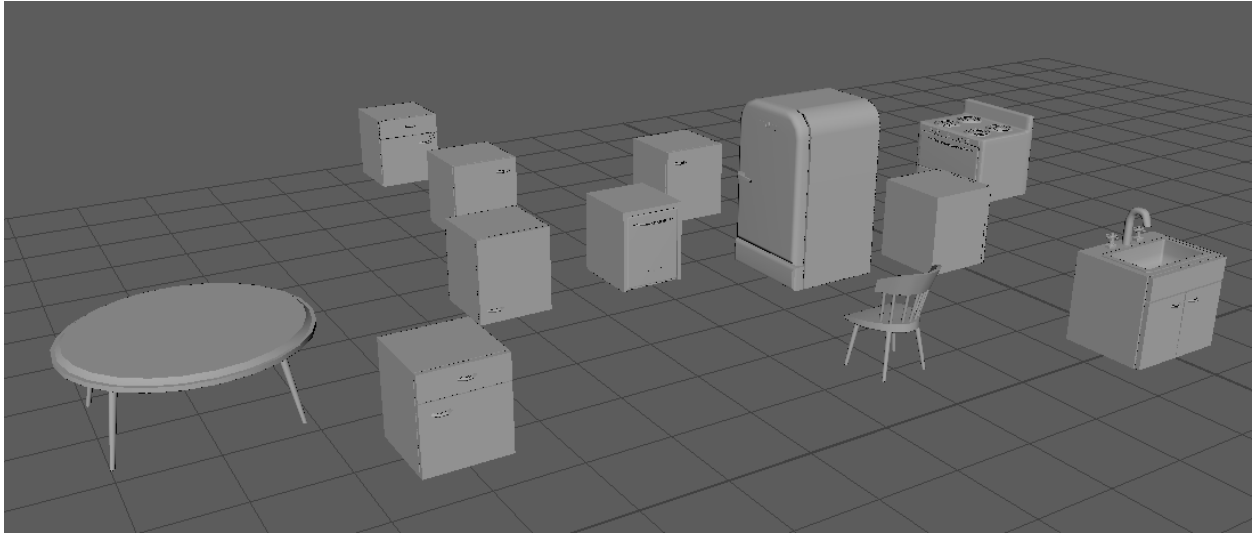
## 5.3 Asset Creation

### 5.3.1 Modeling

We created our 3D models with a combination of Autodesk Maya and 3DS Max, focusing most heavily on the former. We struggled to some degree with proper scaling—to be implemented correctly in-game, every model must be scaled correctly upon exporting, not afterwards. We had a hard time figuring out what was going wrong, and by the time we did, it was time to demo for our first event. To make everything look right for the build, we rescaled everything in-engine, then updated and reimported the assets after the event.

Asset creation was divided between our two artists. We kept and updated a thorough asset list so as to not step on each others' toes and maximize efficiency. We changed and updated our asset list as necessary when the game changed. As previously mentioned, we utilized plenty of reference when modeling, and while we tried to get everything done right on the first try, we did end up having to go back and make some edits.

Modeling was done in waves based on priority. By the time modeling began, we had a greyboxed version of our house layout with blocked out furniture included, so we prioritized necessary, interactable assets first. For example, fruit for the smoothie, the blender, the oven, plates, forks, knives, vacuum, et cetera. A lot of the prop modeling was done in the same scene, so as to make sure that everything was scaled consistently with each other. Once the necessary, interactable assets were complete, we moved on to necessary furniture, set pieces that would obviously be notable as missing if left as default cubes or cylinders for too long, like the couch and the kitchen table. Once these big setpieces were created, we moved on to set dressing: objects that would add some extra flavor to the environment but were not quite necessary like bowls, decor, and picture frames.



*All of the kitchen furniture in the same place.*

Our aim was to finish modeling as soon as possible so we could move on to shader creation and texturing and stay on schedule. We accomplished this goal to a large degree, but we did end up having to go back and adjust models later based on gameplay or how they looked in the engine. For example, fairly late in production we had to adjust the front door model, because we added a flavor task that required Karen to receive mail, and the original model lacked a mail slot.

All of our models were hard-surface and didn't necessitate the use of Zbrush. While we know that Zbrush is a powerful and excellent tool, it didn't make much sense to go out of our way to work in Zbrush when we had little to no organic models to make.

Worth mentioning was that we had to be diligent with our material assigning and naming practices while in these modeling packages. Leaving materials as their default names upon exporting assets causes the material to be imported with that default name into Unreal, resulting in many duplicate materials with unhelpful names like "lambert1". We struggled with this problem at the beginning of the project, but after some time and a significant amount of renaming and sorting, we were able to consistently name materials correctly.

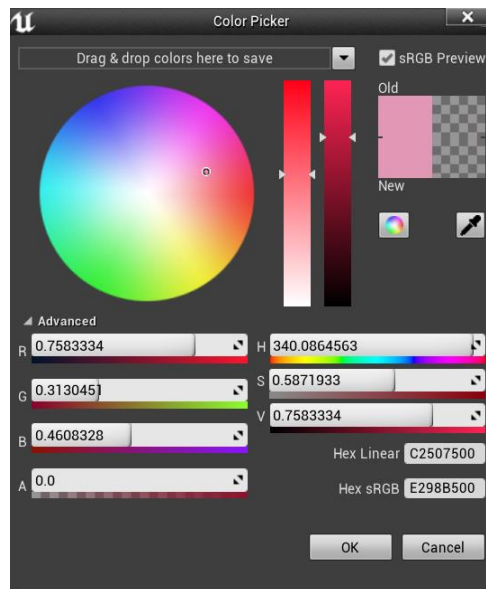
### 5.3.2 Texturing

Texturing was done using a combination of both Substance Painter and Adobe Photoshop. Substance painter was used primarily because, due to the large amount of props we had, we prioritized fast UV unwrapping over perfectly optimized UVs for most props. Substance Painter would allow us to texture with these quick UVs with significantly more ease than Photoshop.

While most props were textured with Substance Painter, Photoshop was useful for props that required a lot of images and text, an aspect of texturing that Substance Painter struggles with a bit more. For example, the couch, california closet, and kitchen implements were created with Substance Painter, but all of the books and trading cards were very text-heavy, and thus required Photoshop. Substance Painter makes it harder to edit previously input text, and is a poor choice for difficult to create, modular, easily swappable, precise textures as needed for the

trading cards. In many of these cases however, models were taken into Substance Painter to create maps other than the base color map, since it's easier to visualize maps like metallic and roughness in Substance. Only in a few cases (the books and envelopes) were all texture maps created in Photoshop.

There were many instances where we utilized basic functions in Unreal's shader graph to block out colors on textures before actually texturing them. For example, during a large portion of development, the colors of walls were determined via the VectorParameter module in Unreal's shader graph. This module allowed us to very quickly and easily see what colors would match together and look good in-engine. Once the color of the walls was determined, we used grayscale wall textures and combined them with the VectorParameter module to give the walls more character, and tweaked the VectorParameter module to adjust for the value change.



*Unreal's VectorParameter module.*

Texturing was split between our artists, utilizing the same organization system as modeling to keep from overlapping work as much as possible. We worked in a somewhat staggered way, and one of our artists was new to Substance Painter and had to learn it before they could begin. We rarely shared Substance files between each other, as they were quite large, and we had good enough communication where we were able to clarify any changes that needed to be made so that the artist with the original Substance file was able to change it themselves.

We did have a strange issue when importing our textures into Unreal, which persisted into our first showcasing event. When hooking up our textures with the appropriate materials in Unreal, the textures appeared dark and far too shiny. We were so baffled that slightly before our first showcasing event, we used our knowledge of Unreal's shader graph to artificially change the materials to be their correct roughness. Once the event was over, we discovered our problem and were able to fix it fairly easily—it was a matter of changing a setting on the textures themselves, then changing a parameter in the shader graph to reflect the texture change. When



using the Unreal Engine temple, Substance Painter, exports the Occlusion, Roughness, and Metallic maps of materials packed together into one image file. However, packing them this way means that the sRGB checkbox in the texture must be unchecked in Unreal before the material will properly show the material.

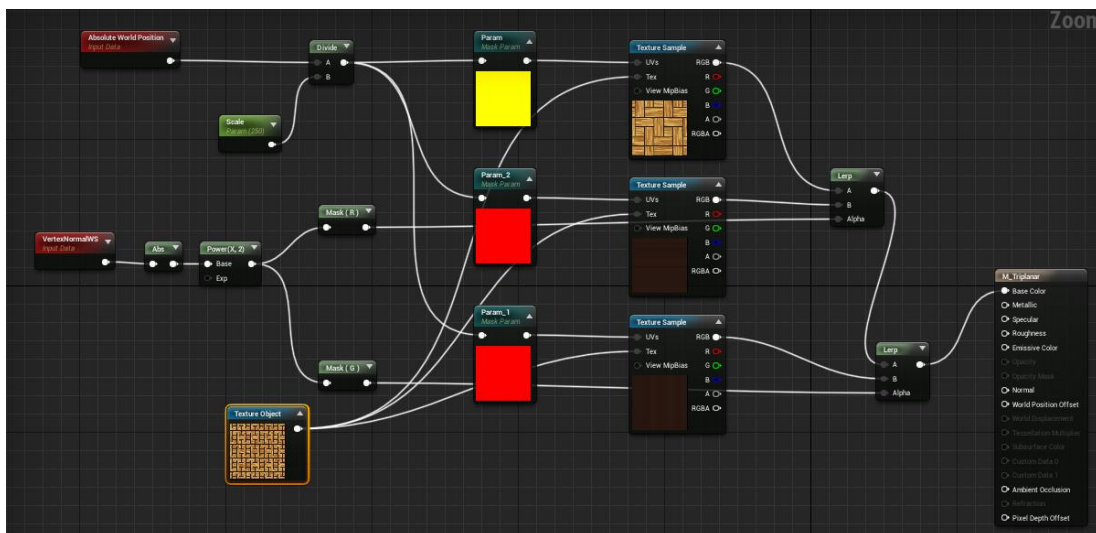
Also worth noting is that, for this project, the artists were the ones hooking up all of the textures into Unreal. Hooking up the textures gave the artists more familiarity with the shader graph (necessary if we were to create the shaders needed to convey Cthulhu) and also gave them more control over how the assets actually looked in-engine, which was invaluable to accomplishing our artistic vision.

## 5.4 Technical Art

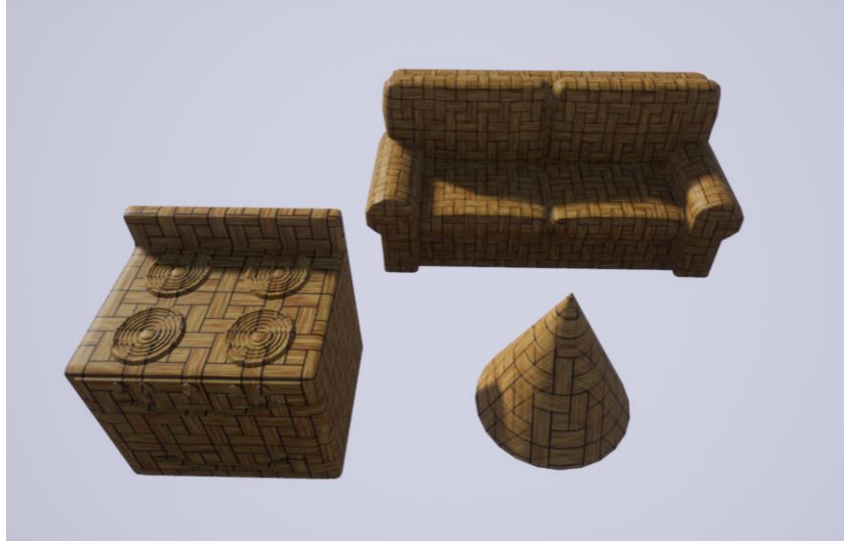
Both of the artists on the team possessed knowledge of technical art, and this knowledge was extremely useful to us as the project continued. This knowledge of technical art and the willingness to roll up our sleeves and learn different areas of technical art gave us new and interesting options to make our game look polished and evoke Cthulhu in fun and interesting ways.

### 5.4.1 Shaders

Part of communicating Cthulhu's presence involved liberal use of the shader graph. One of the first shaders we created was a shader that allowed us to do tri-planar mapping inside Unreal. Effectively, this shader would allow us to put any tileable texture on any object and have the texture applied correctly with no need to account for UVs. As part of our vision was swapping out textures on random objects, this was an ideal situation—this way, we would be able to swap out textures on any object we liked, and wouldn't have to be stuck with a limited amount based on what objects we used the correct UV maps for. This would also save on memory space, since we could apply one texture and shader to multiple objects with the correct results.

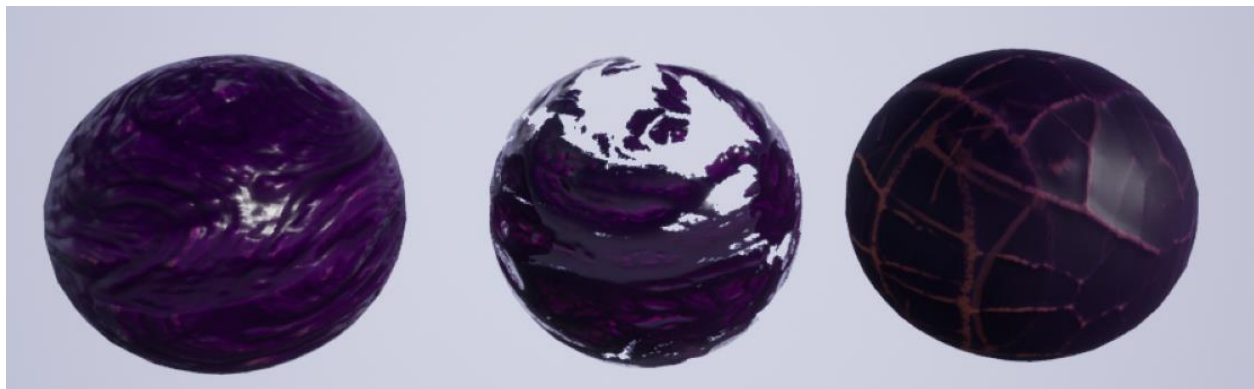


*The tri-planar mapping shader graph.*



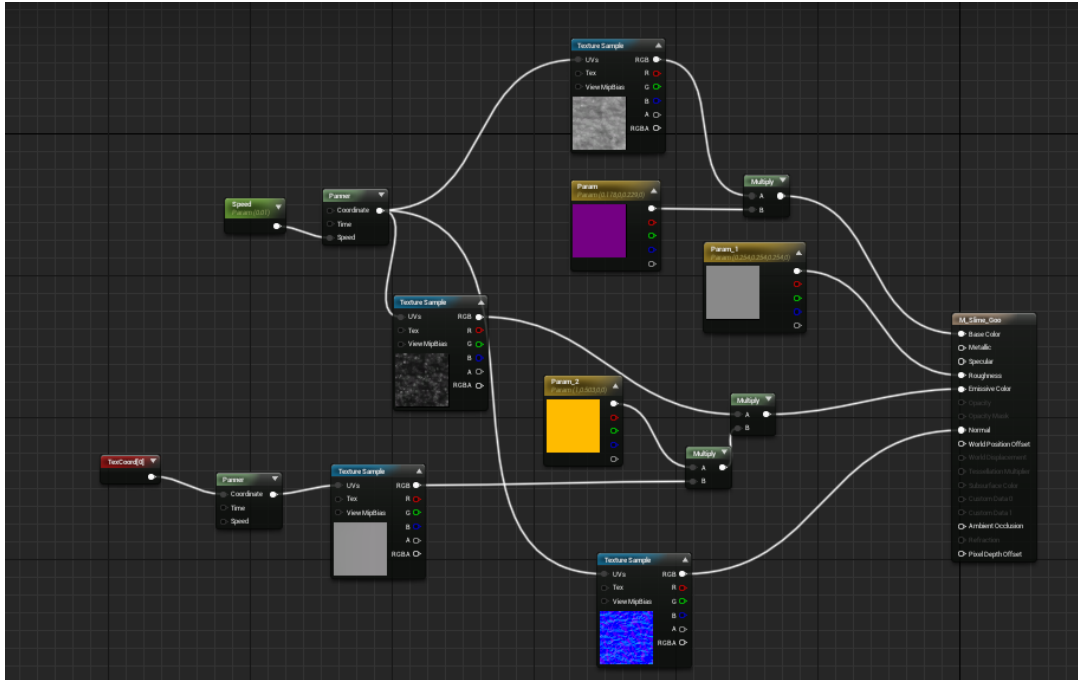
*The tri-planar mapping in action.*

Now that we had tri-planar mapping working, we needed eerie textures to use it with. We tried to stick with classic horror/Cthulhu iconography—mostly veins and slime—but we didn’t want to evoke any major feelings of fear or disgust in the player. The use of purple, as mentioned in chapter 5.2, helped us achieve this goal. The vein texture would be a lot more unsettling if it was skin colored, but the purple “skin” and orange veins, a color combination rarely found in nature, really helped us distance ourselves from any true horror while still evoking the idea of it.



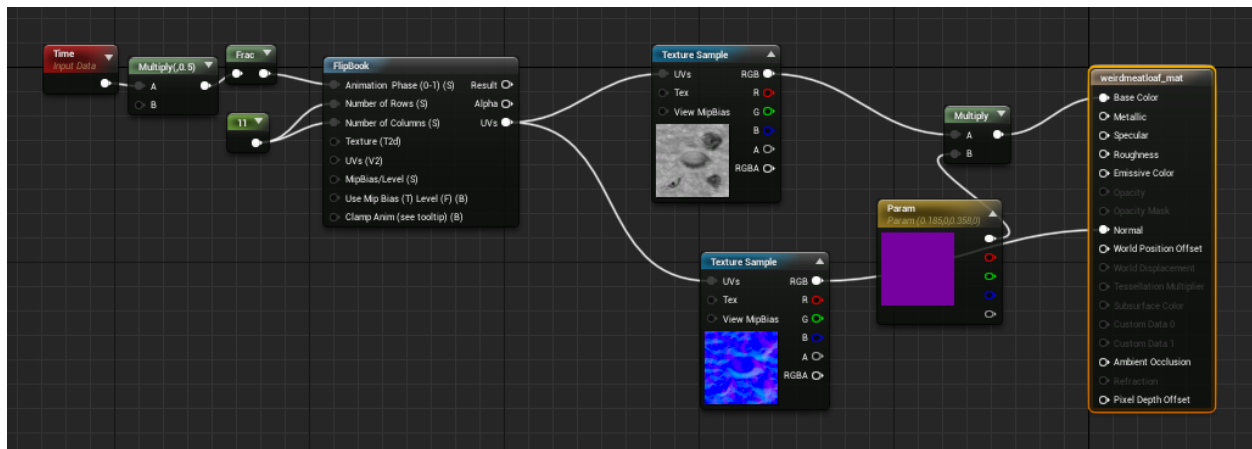
*Our Cthulhu textures. In-game, they also pulse and flicker.*

Most of our Cthulhu-esque shaders were animated in one way or another. Many didn’t use spritesheet animation, but instead made use of the panner module. The panner module allowed us to give the slime textures enough movement to feel just slightly unsettling, and meant that we didn’t have to import any new animations when they weren’t necessary. We also were able to use the panner module zoomed in on a grayscale texture multiplied with the opacity and emissive map to indicate that spots or veins were flickering and glowing, which added a nice dynamic element to our shaders.

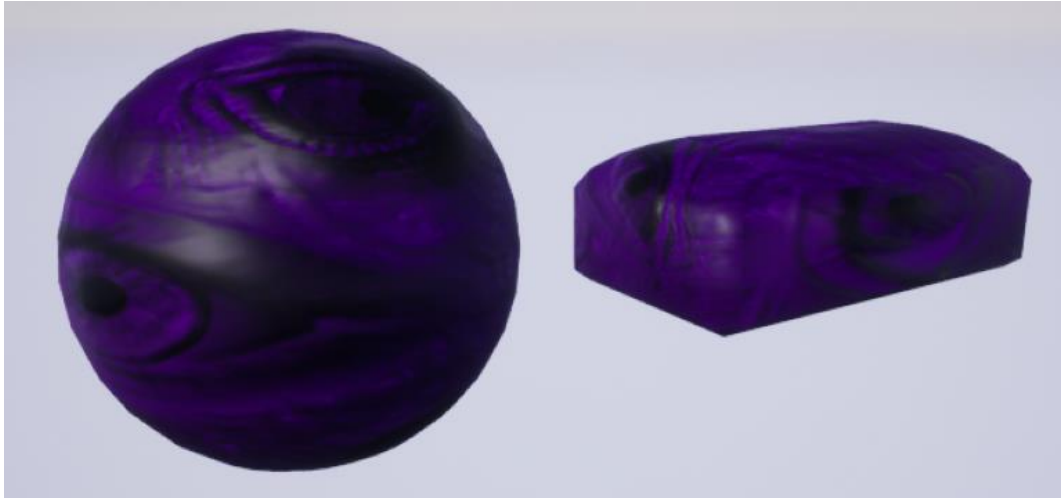


The shader graph for one of our slime textures.

While most of our textures were animated with the panner, we did utilize some spritesheets for our meatloaf texture. One of the dinner dash escalations involves the meatloaf being caught and consecrated, so we wanted the look of the meatloaf's texture to be more intense than the shaders that crop up around the house on occasion. As such, we decided to give the meatloaf several eyes that twitch around and blink, since creatures with multiple eyes are common in Lovecraftian iconography. These eyes were animated in After Effects and rendered as a gif, then converted into a spritesheet in Photoshop, and imported into Unreal. The flipbook node was then used to cycle through the series of images to animate them. It turned out fairly well—the only problem was that, with this flipbook node, we were unable to use the tri-planar mapping shader on the meatloaf, so some of the UV seams are fairly visible, and not all of the eyes are clear.



The eldritch meatloaf shader graph.

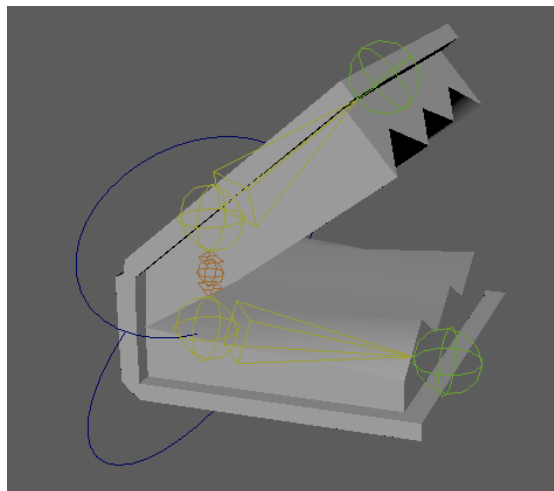


*The meatloaf shader on and off the meatloaf.*

### 5.4.2 Rigging

As we didn't have any animated characters, our rigging was kept to a minimum. We did struggle at first with figuring out how to get cabinets, the refrigerator, and the oven to open and close, and considered using a very basic rig for them. However, we ended up using animated mesh hierarchies instead. Even this method of opening and closing doors and cabinets ended up being a struggle, as skeletal mesh collisions were finicky and hard to work with so the programmers had to go in and essentially hard-code the opening and shutting of doors.

We did end up with one rigged asset. On day 6, some books in the living room float around and chomp at the air. In order to get these books to chomp, we used a simple 5-bone rig to get the effect we needed. We had to make significant adjustments to skin weights, since our books were fairly low-poly and any major mesh deformities would be glaringly obvious. We were able to achieve the effect we wanted fairly quickly, and also created and exported a simple biting animation for the books.



*Our rigged chomping book.*

### 5.4.3 Particle Effects

We didn't end up using too many particle effects, but there were some notable ones. Firstly, we had the salt and pepper particle effects—these effects didn't require much in the way of complex visuals, and as such were created by the programmers, not the artists. We also had a particle effect for when the vacuum gets enchanted, making it glow as it rises into the air. Finally, and probably most notably, we had a particle effect when Karen goes outside during the final section of the game, meant to look like a rising, swirling storm.

This last particle effect had two uses: to instill a sense of unease, and to obscure Cthulhu. One of the important things about Lovecraftian horror is that the horror comes from the unknowability and incomprehensibility of its subjects. As such, we felt that it would undermine Cthulhu's presence if we were to clearly portray it. We still felt the need for a final showdown with Cthulhu, but we needed to keep its form ambiguous. As such, we made a very rudimentary Cthulhu model complete with glowing eyes and an animated slime texture, gave it a backlight to help show a silhouette, and shrouded it behind a particle effect (seen below). We had to do a significant amount of iterating on this particle effect to get it to show just the right amount of Cthulhu, but eventually we reached a happy medium.



*Cthulhu's silhouette.*

### 5.4.4 Cosmetic Blueprints

While we were largely able to keep our artistic endeavors in Unreal confined within shaders and particle effects, there were a few cosmetic elements that the artists had to delve into blueprints to accomplish. Blueprints were used for our swirling text, postprocessing effects, protection circle, and upgrading the vacuum blueprint.

This swirling text was an idea that we initially wanted to incorporate into the cleaning up escalation, an idea which later got cut. The concept was to have floating, swirling text appear above an ancient book, and after a few seconds both the text and book would disappear. When designing the text material, we knew that we didn't want it to be a flat png. We wanted to be able to adjust the text's size and contents within Unreal. As far as we could figure out, creating this kind of adjustable text is impossible to do with the shader graph alone. As such, we made a

blueprint that took text input and applied it to a specified material. This swirling text blueprint and material combo may have been cut from cleaning up, but it was invaluable to us when it came to the Yikes Machine. The Yikes Machine allowed us to spawn unusual objects and then delete them shortly after, and we quickly discovered that large columns of swirling text worked quite well as one of these unusual objects. We applied the text blueprint and material to a sphere, cylinder, and default Unreal bush, which gave us the strange look we were going for. The text actually on the material is the chant associated with the Great Old One in *The Call of Cthulhu*:

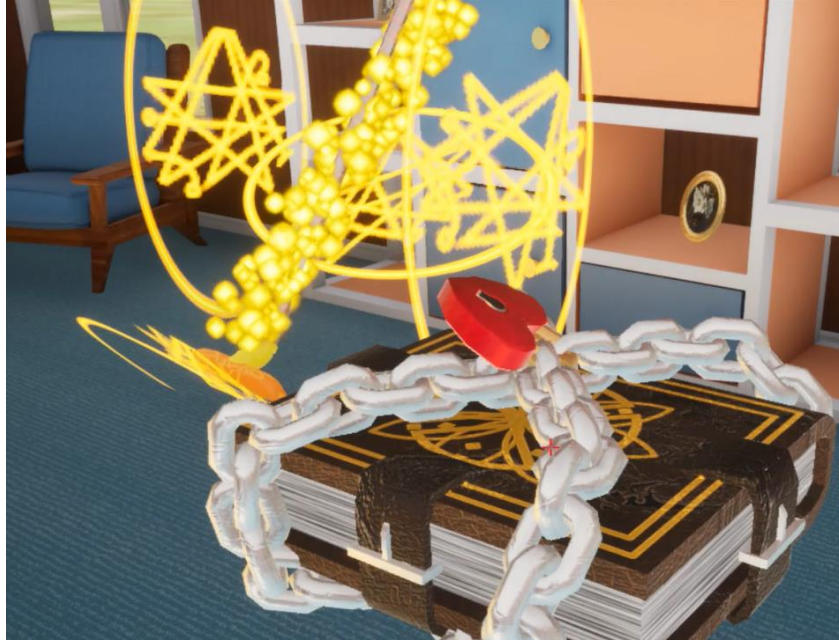
*“Ph'nglui mglw'nafh Cthulhu R'lyeh wgah'nagl fhtagn.”*



*Swirling Cthulhu text.*

We also created another cosmetic blueprint that worked in conjunction with the Yikes Machine. This blueprint allowed us to predefine different sets of postprocessing effects and apply them to the camera whenever we wanted. Paired with the Yikes Machine, this blueprint let us fill the screen with static or use chromatic aberration whenever we wanted to invoke a sense of uneasiness.

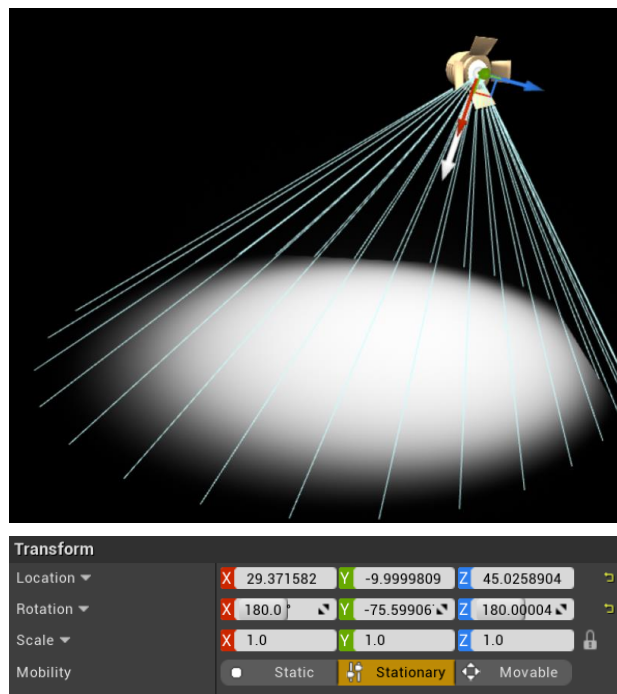
Another cosmetic blueprint, the protection circles, was similar to the swirling text in that we wanted to accomplish that would be much harder do in the shader graph. We needed it to expand and fade into existence upon creation, and we wanted the glyph on the inside of the protection circle to float up and down within the outer circle while the blueprint was active. Once we got the protection circle working, we incorporated it into the vacuum upgrade blueprint. The vacuum upgrade blueprint utilized particle effects and the protection blueprint to have the vacuum rise into the air as protection circles surrounded it, before falling to the ground as a different, upgraded vacuum.



*Enchanting the vacuum.*

#### 5.4.5 Lighting

A major change that occurred over the course of the development of *The Call of Karen* was the switch from static to stationary lighting. Unreal Engine includes 3 basic, mutually exclusive states for all lighting assets: static, stationary, and dynamic. As the name suggests, static lights are completely unchangeable at runtime, and therefore have the lowest



*A stationary spotlight in Unreal Engine*

computational cost, as well as the highest quality. Stationary lights are technically dynamic; some of their properties—color, intensity, etc.—can be changed at runtime, but they cannot be moved. Dynamic lights are completely changeable at runtime, and as you might guess, are the most computationally expensive. Though modern games are now embracing fully (or mostly) dynamic lighting—*Fortnite* (2017-2020) in particular—unrestrained use of dynamic lighting can quickly result in massive performance problems. Unreal Engine offers stationary lights as a compromise between fully static, and fully dynamic lights. Stationary lights allow for baked lighting (that is, lighting and shadowing that is baked into a 3D asset before runtime) in combination with dynamic shadow maps. Initially, it was thought that a mostly static lighting setup would achieve our desired quality level. Indeed, the game remained statically lit for most of its development. However, this prevented the game from having dynamic (moving) shadows. With light pouring in through the windows during the morning and afternoon, we wanted movable objects, of which there are plenty, to cast moveable shadows. Besides adding a layer of quality, it would assist with depth perception when handling interactable objects.

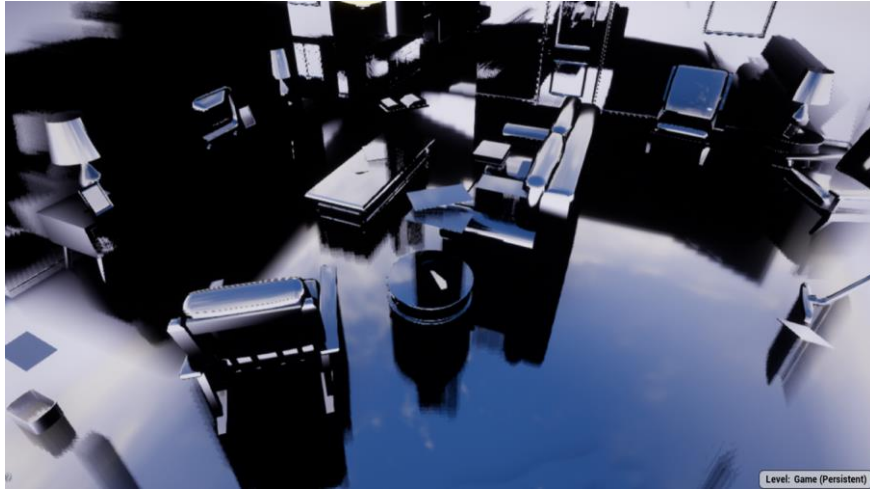


*Preliminary baked lighting with static objects*

The problem was that, even with the lighting set to stationary, there were still no dynamic shadows being cast. We believed this was due to the fact that the house, specifically the floor, was still a static asset. Like lights, mesh objects can also be set to static, stationary, or dynamic, allowing different levels of modification at runtime. We believed that only stationary and dynamic meshes could receive dynamic shadows, and our testing seemed to reflect that. In retrospect, there were likely other problems with the lighting that interfered with having proper dynamic shadows, as static objects *do* in fact receive dynamic shadows. Nevertheless, we proceeded to change as much of the house to stationary objects as possible. Dynamic shadows were now working, and the lighting was looking better than ever. But there was one major problem: anything in the house that was reflective was reflecting the outside sky; it was as if there was nothing blocking the light from the distant horizon hitting the glass on the kitchen table, or the



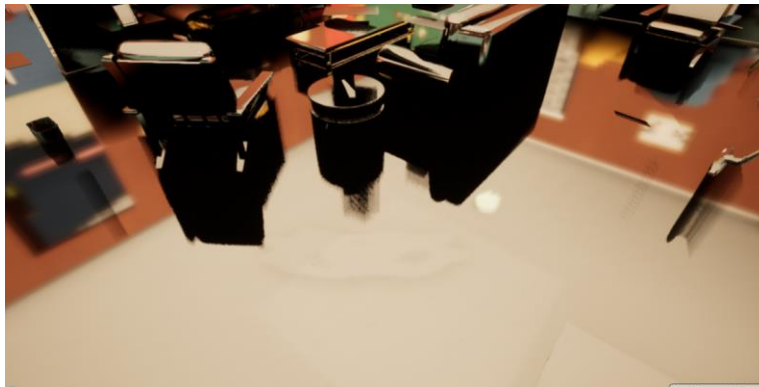
couch in the living room.



*Isolating reflections shows the interior reflects the sky*

This, unfortunately, was a consequence of converting the entire house into stationary meshes. Stationary meshes, unlike static ones, do not show up in reflection captures. Briefly, reflection captures act as a 360 degree camera, capturing everything they see, and then projecting this image onto reflective surfaces. Without reflection captures, materials (especially metals) look dark and matte, and glass is reflectionless. However, having the entire house be stationary meant that it was invisible to our reflection captures, making every reflective surface within the house directly reflect the sky. To solve this issue, a separate reflection environment sublevel was created, in which we placed static copies of all the objects in the house that had been made stationary. The new static objects then contributed correct reflections to everything within the house.

Understanding our previous misconception about static objects, this process of making the entire house stationary and then copying everything to make static reflections could have likely been avoided. But hindsight is 20/20, and the result we achieved was high quality lighting, with relatively low computational cost.



*Corrected reflections*



*Final morning lighting*

## 5.5 User Interface

While we recognize the importance of good user interface, beautifying the UI was our lowest priority. However, near the end of production, we took a little bit of time to upgrade the UI from the base given by Unreal. Karen's tasks are shown on a piece of paper, in a font resembling handwriting. We wanted this UI to be simple and reminiscent of a typical to-do list. Additionally, our menu screen utilizes the title render that we use for advertising, which we thought was a good fit.



*Task list UI.*



*The Call of Karen title render.*

## 6 Sound Design

This chapter details the creation and implementation of *The Call of Karen's* game audio. Chapter 6.1 covers the recording and processing of voiceover, chapter 6.2 discusses the creation and finding of music assets, chapter 6.3 discusses the creation of sound effects, and chapter 6.4 discusses the implementation of audio into the game.

### 6.1 Voiceover

As mentioned in chapter 3.3, a lot of *The Call of Karen's* narrative is conveyed via voiceover. As a result, voiceover was our first audio priority. Once our script was written and approved by our advisors, we decided to record placeholder audio so we could see how players responded to it. We were fairly close to Alphafest and didn't have much time to reach out to other people to perform, so our placeholder lines were recorded almost entirely internally. We thought it might be distracting that one teammate voiced almost every character in the game, so we used audio processing effects to try and mask this fact as much as possible. The only external recording done for this placeholder audio was for a one-off character, Jeff, who had two lines fairly late in the game. Jeff had accidentally been left out of the internal recording sessions, and since we needed his lines quickly, one of our teammates' roommates voiced him.

After Alphafest and our first playtesting session, we left the voiceover largely unchanged. We knew that it needed to be edited and re-recorded, but we knew that we would likely have to make even more edits in the near future. We thought it was a better use of our time to work with what we already had until we had sufficient time to iron out those edits, especially since we were making large changes to different parts of the game fairly quickly.

With the deadline for our build fast approaching and the script freshly rewritten for (hopefully) the last time, we knew we needed to re-record in mid D term. Unfortunately, we had a significant obstacle to overcome. Due to the COVID-19 outbreak, D term was completed entirely off-campus. We were all stuck at home, without immediate access to each other, other people that might be able to help with our recordings, or any nice recording equipment. As such, we had to figure out how to work with what we had, and what to do to get new recordings. Fortunately, we had previously decided that we liked the radio recordings well enough the way they were, and they had escaped rewrites largely unscathed. Since the radio's lines comprise the majority of the script, this took a lot of work off our plate. For Francis, while we would have preferred to have someone new, we couldn't quite think of anyone who would work that we could easily get ahold of. Considering that Francis has very few lines, we decided to leave him as-is as well. The husband's voice lines were also left as-is. Francis and the husband did require a few rewrites, but since they had been recorded internally, we didn't have to worry about wrangling outside voice actors for them.

The main conundrum that we faced was Karen. Karen was the most obvious placeholder voice, and the script had been rewritten to give her significantly more lines. We needed a new Karen to bring it all together, and with a stronger performance. Luckily, a friend of one of our

team members knew a fellow student, Fiona Doyle (who consented to have her name in this paper), who we thought would be a good fit. Scheduling during a global pandemic proved to be a bit of an obstacle, but eventually we were able to record. Karen's new voice was much more fitting than the old one, and really tied together the cast of characters.

One of our biggest obstacles during this time was the fact that all recording needed to be done with whatever microphone voice actors had on hand. Microphone quality was generally bad and extremely variable, and significant audio processing had to be done to try and get the audio to sound the way we wanted it to. The audio we ended up with obviously wasn't as crisp and polished as it would have been had we been using professional grade microphones, but it was definitely acceptable. Also, luckily, there were some rerecords that were likely to go unnoticed. Radio rerecords were largely undetectable, because radio lines were sent through a high pass filter with static playing in the background to emulate a classic 1950s radio sound. These components masked the different mic quality of the rerecords fairly well. Since all of Karen's lines were recorded by our new voice actress, they were all recorded using the same mic in the same recording environment, which meant that they sounded uniform. The husband's new lines were the trickiest ones—the mic quality was quite low, but with some postprocessing effects and an acknowledgment that the husband was shouting from across the house, the lines turned out alright.

An important aspect of the voiceover is the presence of Cthulhu in the radio. As the week continues, Charles, the radio host, speaks gibberish, chants in tongues, and shows other signs of descending into madness. We utilized audio effects to effectively convey this breakdown. We did combinations of reversing the audio, pitching it down, and doubling the radio announcer's voice to give it an eerie, demonic quality, while maintaining the cadence and tone of the announcer.

## 6.2 Music

We decided that it made the most sense for the in-game music to be coming from the radio—after all, if we had a diegetic reason for music to be playing, why not use it? While we did have a team member capable of music composition, we decided fairly quickly that it wasn't worth it for them to compose the game's entire score, considering how much music we needed and how much other work was left to be done. After it was pointed out to us that a lot of music from the early 1920s had entered the public domain, we decided to utilize this early 1920s music in *The Call of Karen*, assuming that many people would not be able to tell the difference between one type of old music and another. We were generally correct in that assumption, but unfortunately, some problems arose with this selection.

Firstly, people who were familiar with the 50s recognized almost immediately (and were disappointed by) the fact that we weren't using period-accurate music. Considering that people interested with and familiar with the 50s are among the people most likely to be interested in our game, the fact that we were alienating them the moment the radio turned on wasn't promising. Our second problem was significantly bigger: when double-checking the copyright laws of the recordings we were using, we realized we had a misunderstanding. A lot of music from the early 1920s had entered the public domain, but it was the sheet music, not the recordings, that were

allowed to be used. Tragically, this meant that all of our music we had been using had to be scrapped.

We took this search for new music as an opportunity to shift to something more period-accurate. Eventually, we found 12 songs that were closer to what was actually playing on a 1950s radio, transitioning from 1920s soft jazz to 1950s classic rock and more upbeat blues music. While we preferred music released under the Creative Commons 0 license (which allowed modification and required no attribution), we discovered that it's quite difficult to find this type of music, so we ended up with tunes that required attribution. However, giving some composers credits in our game was a small price to pay for the amount of time and effort getting the music from online saved us.

The only piece of custom music that we did compose was the jingle that the game opens with. The lyrics are as follows:

*“Listeners and patriots and lovers watch out!  
There is a new kind of evil about.  
Both family men and old maids on their own,  
Don’t let the octopus into your home!”*

We thought the jingle would be a nice tongue-in-cheek way to set the overall tone of the game and set up the radio aspect early on. We aimed to be period accurate with the instruments and the style of jingle, based on our previous radio research. The vocals were recorded by one of our artists and layered to give the impression of a chorus, similar to the all-female choruses present in many advertisements at the time. As for the “octopus”, there was a significant amount of capitalist propaganda in the 1950s that portrayed the rising communist threat as an octopus, so we thought that would be a fun way to reference both the atmosphere of the time and Cthulhu. And, for those who don't know about the communist-octopus reference, the radio announcer spells it out for them in the next line of dialogue.

## 6.3 Sound Effects

Sound effects were largely created using Creative Commons 0 sounds from freesound.org. For the majority of this MQP, we were fairly light on sound effects. We focused mostly on sounds that the game would feel odd without—sounds for the vacuum cleaner turning on or getting enchanted, sounds for the doorbell ringing when Karen gets a package, et cetera. The main other sound effects we incorporated were Cthulhu related. There are a few times in the game where Karen addresses Cthulhu directly, and Cthulhu responds in kind with a growl or shriek. We wanted these Cthulhu sounds to be eerie but also still be able to convey some kind of intelligent emotion, so these sounds were recordings from one of our artists which were then put through a significant amount of audio effects. Recording these Cthulhu sounds ourselves also gave us more control over the kind of mood we wanted for them.

As for soundscapes, we only had one soundscape, for the final day. That soundscape is mostly the sound of distant thunder, with some quiet, occasional chanting to set the demonic tone. The chanting isn't super easy to make out, but is as follows:

*“In perfect suburbs with darkened skies  
The ancient ones will once more rise  
Disrupting homes is his new brand  
Cthulhu spreads across the land.”*

This chant both adds a bit to the humor and sets the mood.

## 6.4 Implementation

Due to our lack of a dedicated audio person, and also in keeping with our manageable scope goal, we decided to aim for fairly simple audio implementation. All the implementation was handled by one of the programmers when they were in between larger programming edits, and utilized 3D sound features built into the Unreal Engine.

Priority went first to putting in the radio dialogue for all days, as it was the main element of storytelling in the game. Similarly, after the final script rewrite, Karen, her family’s, and Cthulhu’s lines (or roars) were high priority and next to be put in to complete the overall narrative. Last to go in were environmental sound effects such as vacuuming, upgrading/enchanting the vacuum, and minor task sounds such as circling a magazine ad. All sounds used in the game were implemented with a simple `PlaySoundAtLocation` function in Unreal except for Cthulhu’s groans, which we aimed to have more of an “omnipresent” sound - Cthulhu is literally an ancient cosmic entity/God, so it made more sense to make it impossible to pin down where its responses were coming from. The radio and voice lines each used custom Sound Attenuation settings to replicate real world 3D sound, with special focus on having the radio sound realistically quieter/louder as the player moved around the house, but still always intelligible. It was extremely important that the player could always hear the radio and all dialogue, so the volume levels were exaggerated a little to accommodate moving around the environment.

All audio files were kept in a series of spreadsheets, both on the team shared Google Drive as well as on the Github repository, and were converted to Unreal Data Tables for use in engine. Music, sound effects, and dialogue were kept in their own data tables, with an overarching `DialogueSequence` table that kept track of multiple lines of dialogue between Karen, her family, and Cthulhu for each task and each day. All of the audio sequences passed through a generalized function within our audio blueprint `RadioManager` called `PlayAudioSequence`, which played each audio clip in order and from the correct location in the house based on a passed-in variable. This way we avoided duplicate code while also spawning sounds from multiples points in the house through one method. Other one-off sound effects and miscellaneous lines were typically played when the player passed over a trigger, like passing through the door connecting the kitchen and living room, or on a tool use, like vacuuming or using a spray bottle.

## 7 Playtesting and Showcasing

This chapter details our playtesting sessions and showcasing events. Chapter 7.1 discusses the process we went through to receive Institutional Review Board approval, and chapter 7.2 discusses the playtesting protocol we followed during playtesting sessions. Chapter 7.3 discusses our first event and playtesting opportunity, Alphafest, and analyzes the results we received from this event. Chapter 7.4 discusses our first formal playtesting session and breaks down the results, and chapter 7.5 details our preparations for, time at, and information gathered from participating in a local game pitching competition, the MassDiGI Game Challenge. Chapter 7.6 discusses our second playtesting session, and chapter 7.7 discusses our time and findings from showcasing at PAX East 2020. Finally, chapter 7.8 discusses the impact that the outbreak of COVID-19 had on our ability to effectively playtest.

### 7.1 Institutional Review Board Approval

Before we could start any playtesting or showcasing, we had to get approval from the Institutional Review Board (IRB), an organization that prevents unethical treatment of participants in data collection or studies. The IRB was created with psychological and biomedical experiments in mind, but despite the fact that we didn't quite fit into that category, we were still required to get approval before we could begin any data collection.

Following the approval process was difficult. The IRB's website was inscrutable to us, and we struggled to figure out how to start. Every process or guideline we tried to follow only raised more questions. The form asked us what kind of study we were, and we had no idea what to classify ourselves as. Some IRB studies require the administrators to go through online ethics training and pass an assessment. Would we need to do the same? We also discovered that there are some studies that pose a low enough risk to participants that they don't require IRB approval, and go through a different process altogether to get exemption. Were we one of those? Additionally, it seemed as though studies that are not exempt from the IRB require those who participate (in our case, anyone who played) to fill out and/or sign a form before playtesting. This discovery was quite disappointing. Bringing along a stack of forms and making each player sign them would definitely hamper our efforts to get playtesting feedback from people at events, where it's already hard enough to get people interested in your game without the addition of paperwork. Full of questions and finding very few answers, we decided to reach out to some IRB administrators at WPI.

After meeting with IRB Secretary Ruth McKeough in person and explaining our study to her, she was kind enough to walk us through the process. It turned out that, thankfully, we were one of the studies that was exempt from the IRB. We had to go through a process to formally receive exemption, but once we received it we would not need our players to sign any waivers or fill out any forms before playing. All we needed to do was give each playtester a general overview of the game before they began, and assure them that they could leave at any time. Once we had IRB exemption, we were free to begin our playtesting and showcasing.



## 7.2 Playtesting Protocol

An important part of gathering the most valuable feedback possible from playtesting sessions is conducting these sessions in the correct way. During playtesting sessions, administrators must take notes on both what the player is doing in-game and their expressions and body language as they do it. Another key element of playtesting protocol is that the administrators should not talk to the playtester unless the tester is stuck or asks a direct question. Playtesting is meant to provide information on how the game would be received by the general public if it was released immediately, and the general public won't have the developers guiding them through every step of the game or answering every player's questions. Most questions should be ignored, in favor of the playtester figuring out the answers on their own. Direct questions should only be answered and instruction should only be given if the playtester is incapable of progressing without intervention. And if that point has been reached, you'll already have learned what you need to know about that part of the game.

Another important aspect of playtesting is encouraging playtesters to voice their thoughts aloud at the beginning of each session. Even with a reminder at the beginning of a session, playtesters don't always voice their thoughts, since plenty of people hesitate to give feedback. However, it's important to give this reminder anyways, since valuable information can be learned from playtesters speaking as they play the game.

Also worth noting is that, as required by IRB, we opened every playtesting session with a brief spiel explaining what the game is about, and assuring that playtesters can leave at any time. We also informed them that we would be taking notes on what they were doing, but would not record their names, so any specific things they said or did could not be traced back to them in our paper or notes.

Our playtesting protocol did make the sessions feel a bit awkward at first. After all, it's a bit odd to have someone hovering silently over your shoulder, as you play through something they made. It was especially awkward on occasion when the testers would ask questions and we had to remain silent. However, once we and the testers got used to it, the sessions became significantly less awkward. And the data gained from our sessions run this way was invaluable to improving our game.

After each playtesting session, we had our playtesters fill out a short survey. These surveys are available in the appendices, and used neutral language to avoid skewing the results. Most of our questions were formatted using Likert scales, though we did have a few open ended questions. Two of these questions gave us data that was sometimes hard to parse but was useful when we could do so: specifically, "Describe your experience with one sentence" and "Name 3 emotions that you felt while playing this game". These questions were designed to help us figure out if we were hitting our experience goal.

## 7.3 Alphafest

Our first showcasing and playtesting opportunity came in the form of Alphafest. AlphaFest is an annual showcasing event held by the IMGD program in late B term, an event which all IMGD MQPs are required to showcase at. Alphafest is a significant milestone for IMGD MQPs, as it's usually the first time the IMGD student body and other IMGD professors get

to see what state other projects are in. It's also usually the first time project groups can get feedback on their project by people who know nothing about it. Knowing this, we were determined to give a strong showing, and get some solid data in the process.

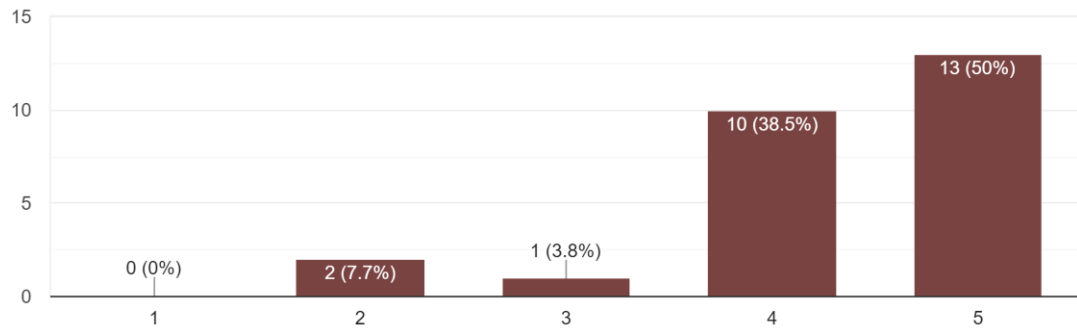
As this was our first deadline for an actual build, and we had been working in a fairly insular environment, we were anticipating many bugs and generally mixed feedback from Alphafest attendees. Still, we pushed ourselves to get the best possible product we could in front of everyone, and set deadlines and frequent check-ins to make sure that we were on track. By the time Alphafest arrived, we had the first day or so of the game ready to show off. We had implemented some audio lines and had the smoothie, tidying up, and dinner dash tasks completed.

Something we weren't quite expecting was just how popular Alphafest and *The Call of Karen* were. There were several factors that contributed to this popularity. Firstly, the IMGD program had recently instituted a new policy on playtesting, which required students to participate in two playtesting sessions/events per IMGD class they were enrolled in. As Alphafest was a playtesting event, it was packed with people wanting to knock out the requirement. Additionally, it was held in the Foisie Innovation Studio and offered free food, attracting many students working in the building who were looking for a break or a snack. This event also drew several people outside IMGD to attend, people either looking to see what WPI's resident game developers were up to or who wanted a sneak peek on what a friend, roommate, or significant other's MQP looked like. Due to these factors, the event was packed. *The Call of Karen* had also strategically picked out a spot by the door to get the attention of attendees who were coming in or leaving, which definitely added to our popularity. We set up two playtesting stations, and consistently had a line of two to three people waiting to play for the rest of the night. We took turns managing the stations and taking diligent notes following our playtesting protocol, but there ended up being so many playtesters that we set up an extra station for administering the post-game survey so we could reset the game faster and keep people cycling through.

There's nothing quite like playtesting for ferreting out new and unforeseen bugs, and Alphafest was no exception. We got a lot of brand new information to parse from both our notes and the 26 people who filled out our survey. Worth noting before we break down the following graphs is that, while this format doesn't display what the values 1 through 5 on the Likert scale mean, the actual survey does display these values (see appendix C). For all questions, 1 is the least desirable outcome, while 5 is the most desirable.

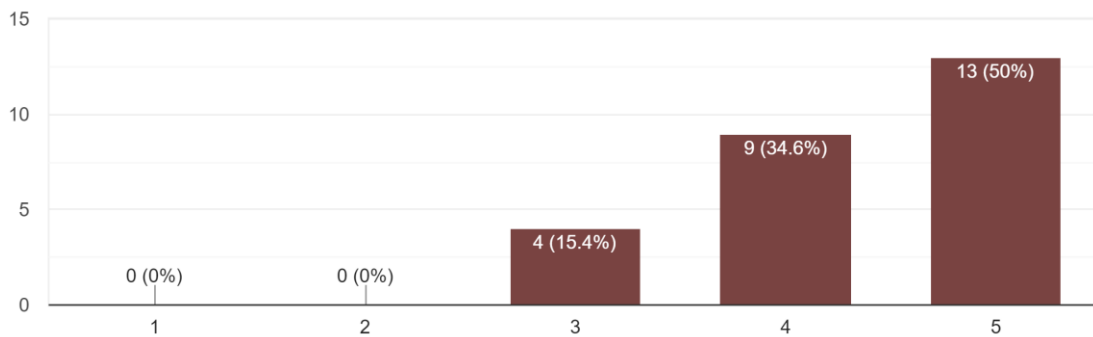
### How understandable were the controls?

26 responses



### Did you understand what you were supposed to do?

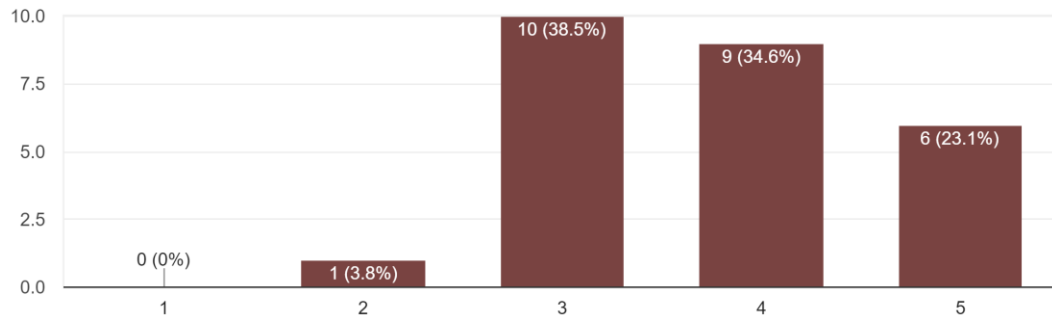
26 responses



Players seemed to generally understand the controls and what they were supposed to do in the game, but we had to take this data with a grain of salt. Most players didn't notice the controls popup, and to keep things moving so we could get as many playtesters as possible, we gave a rundown of the controls to every player at the beginning of their session. The same goes for the tasks, though we definitely didn't explain those quite as much. We were still exploring how to tutorialize and implement the UI explaining these aspects of the game, so not having solid data for this first event didn't bother us all that much.

How difficult was it to navigate through the play space?

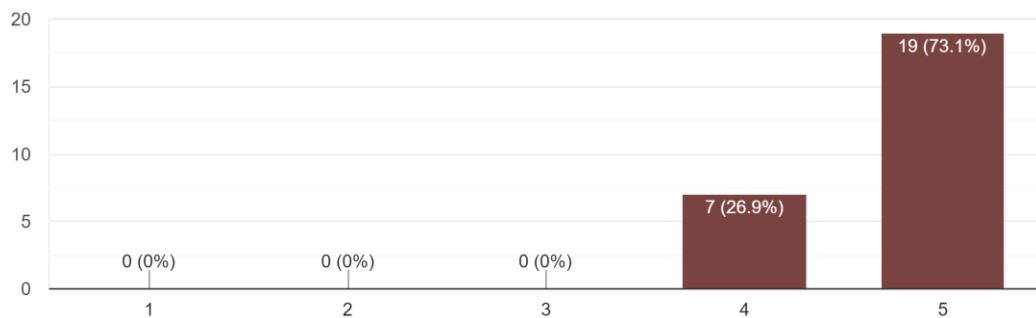
26 responses



The results of the survey indicated that, while the playspace was navigable, there was room for improvement. Despite all of our iterating on the house layout, it seemed we still had some tweaks to make. However, given the observations that we made of the playtesters themselves, we had a hypothesis on what was tripping players up. The player camera's field of view was pulled fairly far out in this build, which caused some disorientation. After Alphafest, we applied some minor tweaks to the walls and rooms in the house, and shrank the field of view, hoping that this would fix the issue.

How did you feel the art style worked with the game?

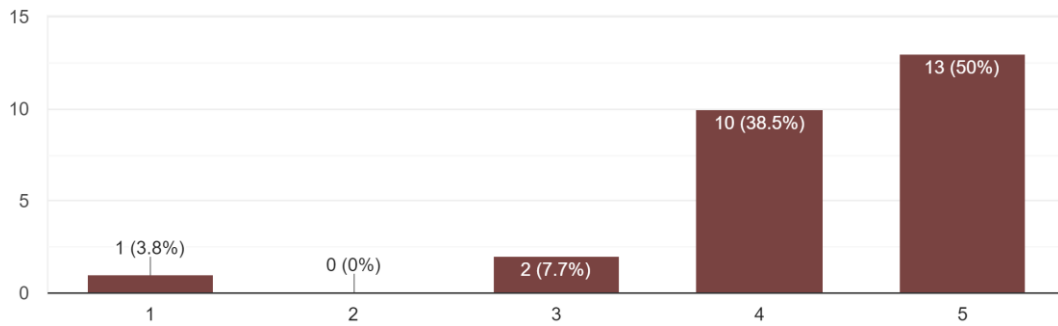
26 responses



We had a few art issues going into Alphafest, but were happy to see that players overwhelmingly liked the art style. This response, alongside the fact that we wouldn't be changing the art style at all for the rest of production, led us to omit this question in future forms. Looking back, it may have been valuable to keep it anyways, but hindsight is 20/20.

### How did you feel about the lighting?

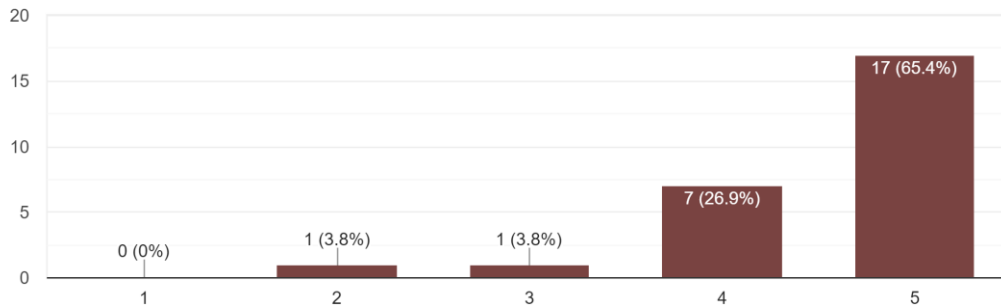
26 responses



Lighting was also received generally positively, but we felt it still needed some work. Lighting was something that we continued to iterate throughout production, but it was nice to hear that we at least had a decent base to work off of.

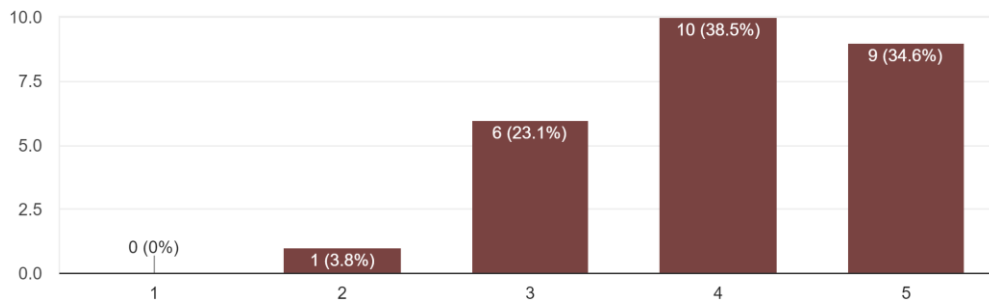
### How did you feel about the radio?

26 responses



### How did you feel about the writing?

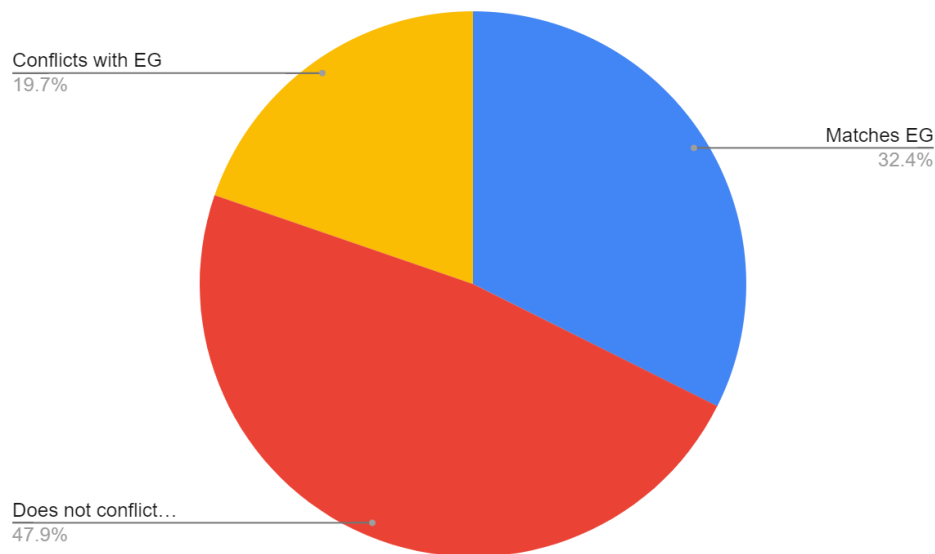
26 responses



The radio was received positively, and the writing received mixed to positive reception. This data also didn't influence our next iteration of the game much, because a lot of the writing and radio hadn't been implemented at the time of Alphafest. We had about two lines of dialogue (lines that were on the longer side, but still only two lines), and no music aside from the opening jingle. We attributed the mixed response of the writing to these factors—after all, it's hard to judge the overall quality of a game's writing based on a very small amount of it.

Worth noting is that, despite humor being a key part of our experience goal, we didn't have any questions directly addressing whether the game was funny or not. We struggled to figure out a way to ask this question in a way that would give us actionable feedback without priming the player. Instead of asking directly about humor in the survey, we observed the playtesters to see if they were smiling or laughing, and relied on the question "What are three emotions you felt while playing this game?", which was meant to help us determine if we were reaching our overall experience goal. After Alphafest, we classified the emotions we got from the survey into three categories: Matches our experience goal, doesn't conflict with our experience goal, and conflicts with our experience goal. Ideally, most of the responses would match or at least not conflict with our experience goal.

Parsing these emotions was sometimes difficult. The one that gave us the most pause for thought was "confusion", because it was hard to tell where the confusion was coming from. Was it coming from a lack of understanding of the game systems or mechanics? If so, then we were doing something wrong. Was it coming from not understanding why Cthulhu was invading the house? Then that was alright. In the end, we chalked confusion up to not conflicting with our experience goal, since it was so hard to quantify.



These initial results showed some promise, but we still clearly had a lot to work on. The majority of the emotions given (47.9%) did not conflict with our experience goal, and a significant amount matched our experience goal (32.4%), but we had a sizable amount conflicted with it (19.7%). The majority of the words that conflicted with our experience goal were words like "frustrated" and "annoyed". Given that our game was fairly buggy at the time

and some tasks, like the smoothie, were difficult to complete, we attributed the majority of these emotions to those issues, and doubled down our efforts to fix them.

Overall, we considered Alphafest to be a great success. While we did have a lot to improve on, we also gained plenty of useful information, and several actionable things to do to improve the game. Additionally, the notes on our playtesters themselves indicated that we were heading in the right direction. Many seemed engaged and were laughing and smiling as they played.

## 7.4 Playtesting Session 1

Our first formal playtesting session was in C term. We sent out a mass email to IMGD majors and those taking IMGD classes asking for playtesters. The playtesting requirement had made people ravenous for playtesting opportunities, and every time slot was filled in a little over two minutes.

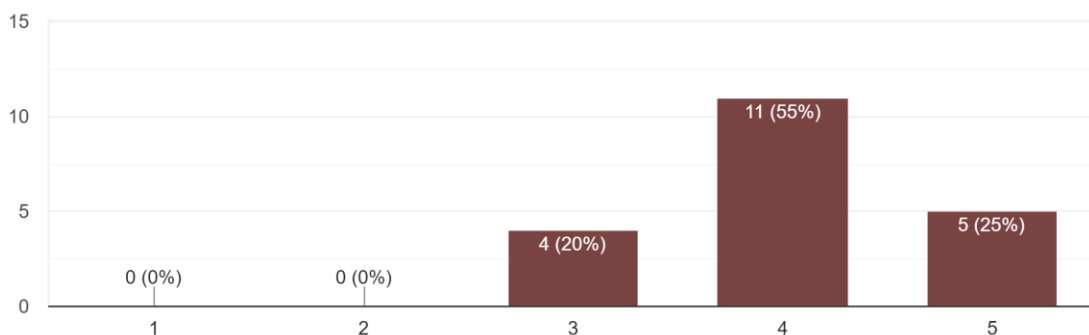
For this and every other formal playtesting session, we had two participants at a time come in and play the game, with at least one team member observing them both. They would play until they either completed the content we had, or until roughly 25 minutes had passed. Afterwards, we would ask them to fill out our survey.

Since Alphafest, we'd made several adjustments to *The Call of Karen* based on the feedback we received. Significantly more of the voice lines had been implemented, the smoothie task had been adjusted to hopefully be more readable, and many bugs had been fixed. Additionally, days 1-3 were largely complete (though not bug-free).

For this first round of playtesting, we had 20 playtesters. Most of our survey questions were the same as the ones from Alphafest, though as previously mentioned, we removed our art style question.

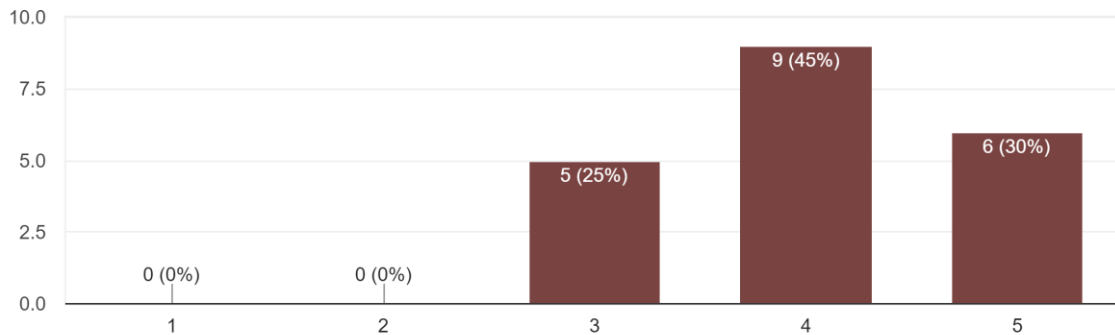
How understandable were the controls?

20 responses



Did you understand what you were supposed to do?

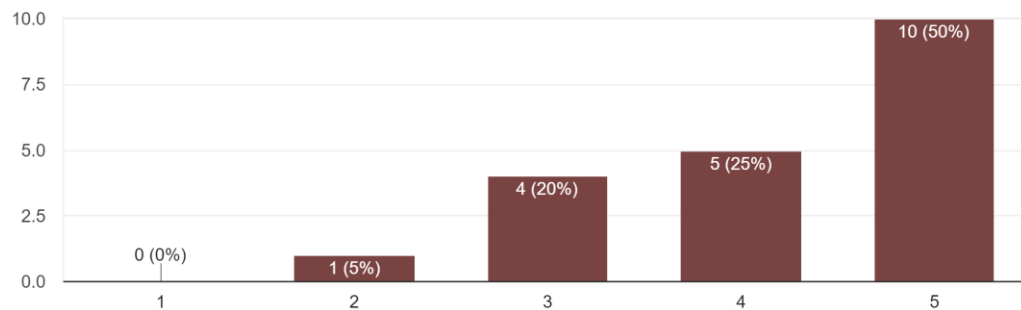
20 responses



These mixed to positive results from this round of testing confirmed our suspicions that our positive results from Alphafest were likely due to the fact that we explained our game to the people playing. We did not offer any such explanation this time, which we believe is what resulted in this reception. Not terrible, but we still had a lot to improve on. The main gameplay concept that players struggled to grasp was right clicking while holding an object to use it, so we decided to try and fix that sticking point in later builds.

How difficult was it to navigate through the play space?

20 responses

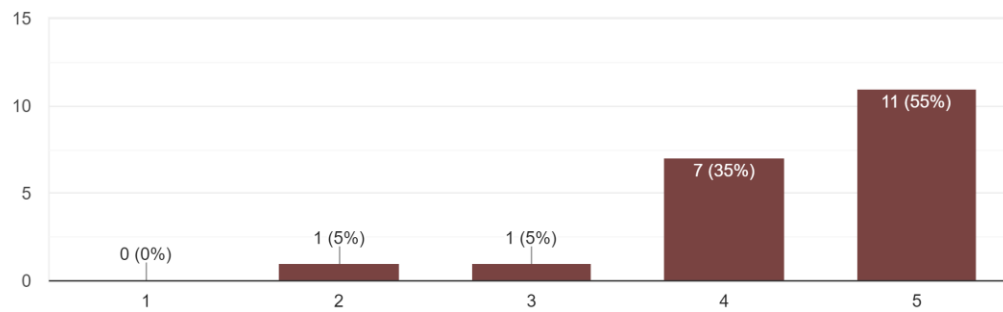


Our FOV and layout tweaks seemed to have worked, since players found it significantly easier to navigate through the play space during this round of testing. While the results seemed to indicate that there was more we could do to improve the playspace, we decided to leave it alone for the rest of development. We were satisfied with these results, and with the way that we had modeled the house and its walls, it just didn't make sense to continue to make tiny changes until it was absolutely perfect. There were other, more pressing matters we wanted to focus on.



### How did you feel about the lighting?

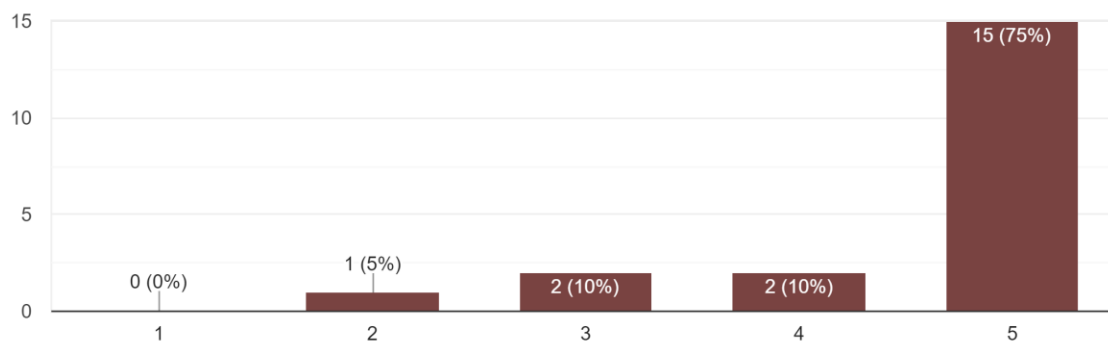
20 responses



The lighting had undergone some changes since Alphafest, but not to a significant degree. As such, the feedback regarding the lighting remained largely the same. We still had plans for more lighting changes, but these changes wouldn't be finished until D term, so we anticipated the overall lighting feedback to remain the same for the playtesting sessions following this one.

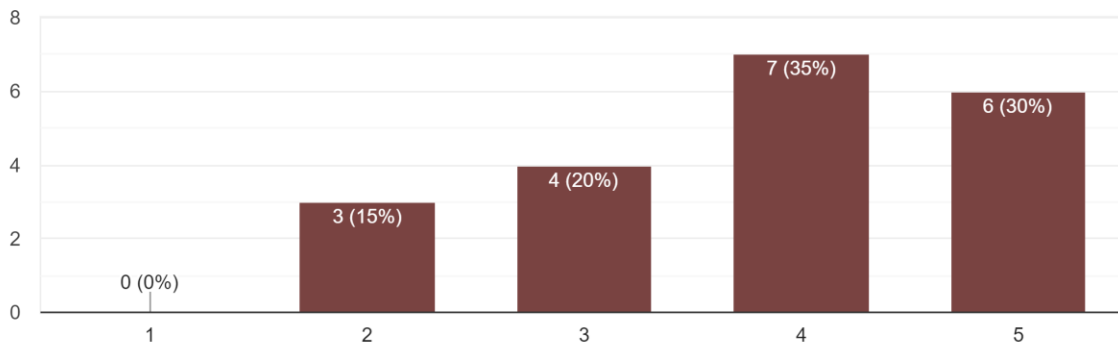
### How did you feel about the radio?

20 responses

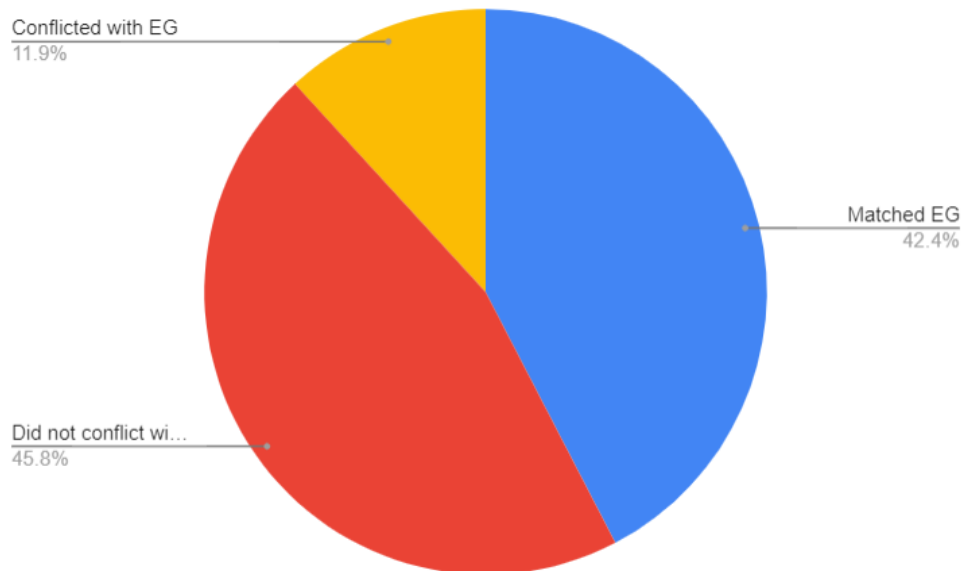


How did you feel about the writing?

20 responses



As previously mentioned, since Alphafest more voice lines were implemented in the game and the radio now played music in the background during the tasks. The radio was received overwhelmingly positively, while the writing received mixed to positive reception. The mixed reception on the writing gave us some pause for thought. What could we do to make the writing more appealing to everyone? Additionally, while some people didn't seem to enjoy the writing, we did observe several people laughing and smiling at the jokes, and those that did enjoy it liked it enough to point it out specifically in our survey question, "What's one sentence you would use to describe our playthrough?" While our script did undergo a few changes during D term, we were unfortunately unable to playtest these changes.



Our first round of playtesting also revealed that we were doing much better matching our experience goal. The percentage of responses that conflicted with our experience goal was down to 11.9% from our previous 19.7%, and the number of responses that matched our experience goal increased by 10%. Similar to the results we saw before, some of the conflicting

responses were along the lines of “frustrated”. These responses were significantly fewer than before, and considering that since Alphafest our number of progress-impeding bugs was significantly down (though not zero), we believe we correctly assessed that the frustration we saw from the Alphafest playtesting was a result of these bugs.

However, there was a new contender in the category of emotions conflicting with our experience goal: “bored” and “monotony”. Emotions like these comprised over half of those that conflicted with our experience goal, emotions that we did not see as much from Alphafest. Additionally, observations of playtesters indicated that we weren’t doing enough to convey Cthulhu early on in the game. Payers barely noticed the tiny sprinklings of madness we scattered about (randomizing object placement or tossing objects all over the ground), and were fairly bored with the slow build-up to Cthulhu. We knew that we needed to remedy our pacing problem, but unfortunately didn’t have time to do so before our next event, the MassDiGI Game Challenge. However, we saw the majority of these playtesting results as proof that we were improving our game and moving in the right direction, even if there was still more that needed to be done.

## 7.5 The MassDiGI Game Challenge

The MassDiGI Game Challenge is an annual pitching competition hosted by the Massachusetts Digital Games Institute, or MassDiGI. Indie and collegiate game developers prepare and give a 7 minute long pitch to a wide variety of industry professional judges. The competition takes place over the course of two days, with a miniature showcase and the first few waves of judging occurring on the first day, and final presentations and the final wave of judging on the second day.

Our producer was the only one familiar with the event, having previously competed in the game challenge. Thus, we all agreed that participating would be a valuable opportunity for us to get even more valuable feedback on our game. We spent the week leading up to the Game Challenge polishing our game and rehearsing our pitch. We also did some things to make us and our setup seem a bit more professional—we got Trumbus (our team name and logo) pins, decided on a color scheme to wear (black with a purple accent), and got props to put on our booth to fit our 1950s housewife brand. We were nervous, but excited.

Worth noting is that the MassDiGI Game Challenge is less for playtesting and more for getting concrete and direct feedback from industry experts. Also worth noting is that the MassDiGI Game Challenge has a significant focus on marketing strategies and marketability. As such, we had to dedicate a significant chunk of our presentation to our research and marketing strategy for our game. Another point to note was that, if our team did make it to the second day, we would be one member short—our producer had a prior commitment that she was beholden to during the time of the game challenge.



*Our setup for the first day of the MassDiGI Game Challenge.*

During our first day, we presented our pitch to three waves of two to three judges. Our practice paid off, and our pitches went smoothly and without incident. We adjusted our pitches slightly each round based on feedback we received from the judges. However, we didn't have to adjust all that much. Feedback from the judges was generally very favorable, and praised us for our high concept, humor, and art. The criticism generally came in two categories: with our marketing plan and with our implementation of Cthulhu.

While our marketing plan was definitely something we wanted to keep in mind during production, we weren't all that concerned with making it absolutely perfect. Going through the actual marketing and steps of release was outside the scope of this MQP, and we didn't have any dedicated marketing people on our team. However, the implementation of Cthulhu was definitely an area that we wanted feedback on. Judges confirmed what we had learned from our previous playtesting feedback; Cthulhu needed to show up earlier in the game, and its presence needed to be known in other ways besides antigravity. They also suggested adding tasks that directly addressed Karen's battle against Cthulhu, which we hadn't considered previously. This feedback directly influenced our game, spurring us to create the Yikes Machine and flavor tasks.

The MassDiGI Game Challenge was also helpful in that it allowed us to directly connect with other game developers in the greater Boston area. The 2020 MassDiGI Game Challenge attracted teams from Becker College, Northeastern University, and independent game developers all around the greater Boston area, and we played each other's games and offered feedback and ideas. We found meeting and talking to other developers in the greater Boston area to be a valuable experience, and we're happy we were able to do so.

During the second day of the game challenge, all of the finalists for each category of games (neatly organized into pastry-based categories, we were cronuts) presented one final time in front of an all new panel of judges. Despite the absence of our lovely producer/artist Kate, the presentation went remarkably well, with generally favorable feedback from this new

panel. They focused mainly, again, on the marketing plan, and suggested we flesh that out more as expected. In addition to this they paid special interest to the 1950s setting, and advised us to make sure it was obvious what time period our game was set in, as well as making the house feel lived-in. As a result, our artists spent much of the later development period adding personality and more period-accurate details around the house to great effect. Finally, the second-day judges also brought up the need to focus on bringing the presence of Cthulhu to the forefront. We finished as runner-ups in our category and were extremely pleased with the result.



*Finalists and runners up in our category! (Kate present in spirit)*

## 7.6 Playtesting Session 2

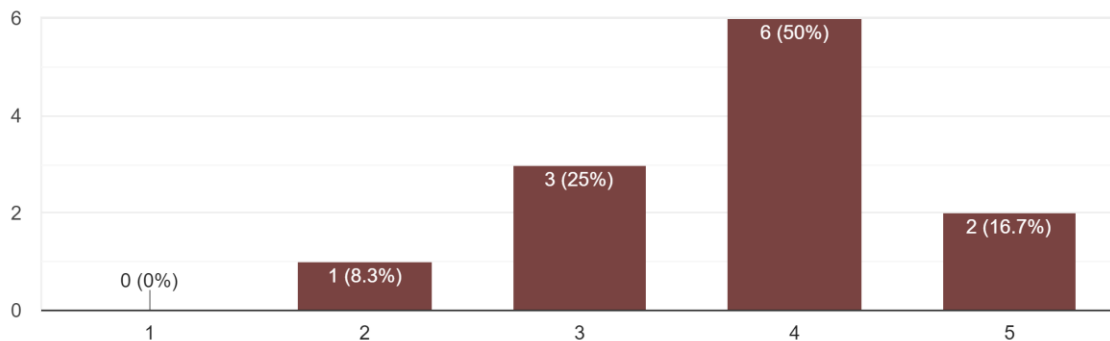
After our success at the MassDiGI Game Challenge, we knew we had to get back to the grindstone and start implementing our changes based on the feedback we got there. The first thing we tackled was our pacing problem. We needed to spark intrigue earlier on to keep players consistently engaged. We also needed to implement some more Cthulhu into the house, a problem brought to our attention by both playtesting and our advisors. As such, we made a few sweeping changes. We adjusted the progression of the game and added a task escalation on day one: floating objects in the living room during the cleaning up task. We also added the Yikes Machine to give us some extra eldritch flavor. To gauge how effective these methods of conveying Cthulhu were, we added another question to the survey: “How much did you feel the presence of Cthulhu?” Also worth noting is that this round of playtesting was when we decided to remove the smoothie task and replace it with cooking breakfast.

This playtesting session was very important to us, since it was to be the last playtesting session before we would showcase at PAX East the following week. We wanted to put our absolute best foot forward for showcasing at an event as renowned and popular as PAX East, and as such needed desperately to know about any critical bugs or issues. Due to our shifting schedules, we offered less playtesting slots than before, but still believed we had enough playtesters to get useful data. Similar to last time, all of our slots filled up extremely quickly, though we had several no-shows the day of.

For this round of playtesting, we had 12 responses. We had the first three days of the game completed and largely bug free at this point. Aside from the added Cthulhu question, the survey remained the same as it had been previously.

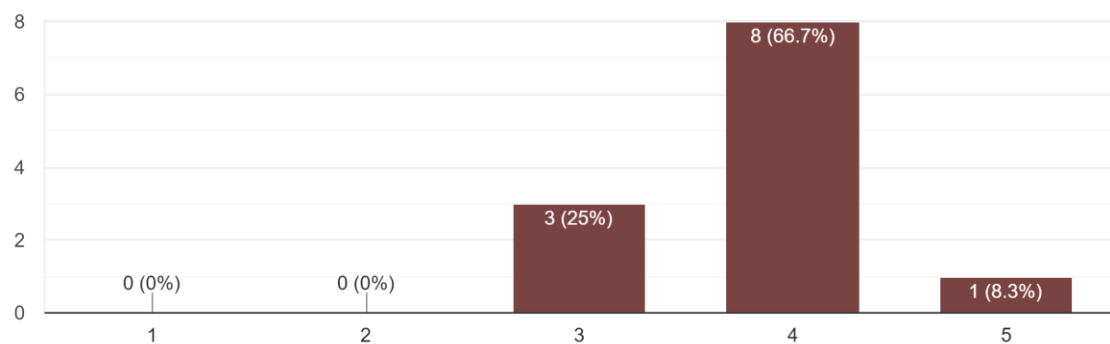
How understandable were the controls?

12 responses



Did you understand what you were supposed to do?

12 responses

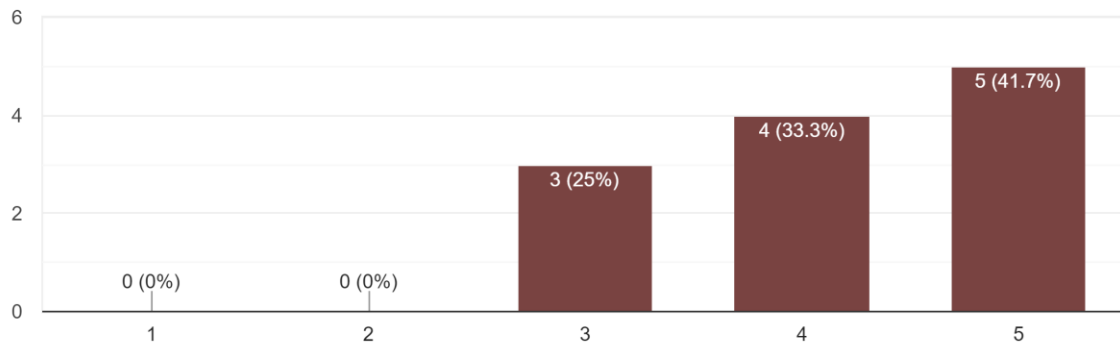


The results for these questions were generally the same as before, and considering that we hadn't made major changes to the systems affecting these responses, we didn't really anticipate any big changes. However, this general understanding of the controls and objectives satisfied us that the build would work for PAX East. While players were still occasionally confused about right clicking while holding an item to use it, they generally understood the

overall mechanics and what they were supposed to be doing. Additionally, during our observations of the playtesters, they didn't get stuck to the point of being unable to progress, and didn't ask nearly as many questions about the controls as they had in the past.

How difficult was it to navigate through the play space?

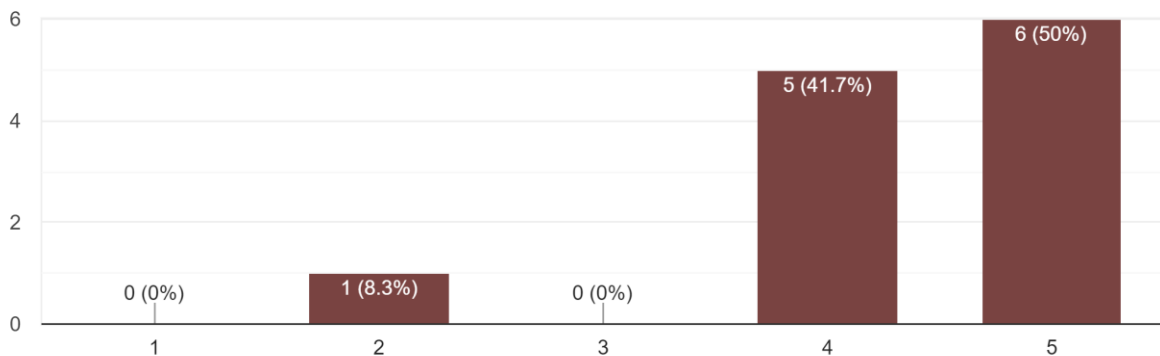
12 responses



Considering that the playspace hadn't been changed from the previous playtesting session, we expected roughly the same results that we had received before, and that was what we got. Similar to the controls-based feedback, our observations indicated that players were generally fine navigating the playspace, so we were content with our results. There was only one issue worth noting: during the cleaning up task, the player is supposed to put cards by Francis's door. However, Francis's door wasn't labeled, so it was impossible to know which door to put the cards by. This was remedied before PAX by adding a sign to Francis's door, which seemed to solve the issue.

How did you feel about the lighting?

12 responses

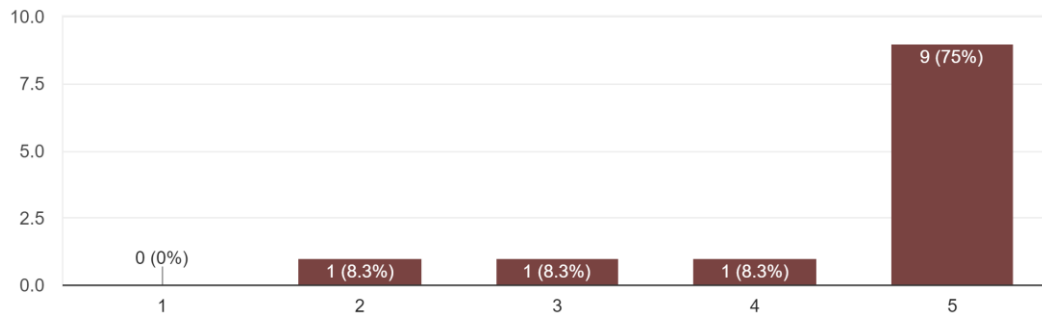


Lighting had been tweaked, and was overwhelmingly well received (with one outlier). We were pleased with these results, though we did end up continuing to refine and improve the

lighting as production continued into D term. However, these results did assure us that our lighting was good enough for PAX East.

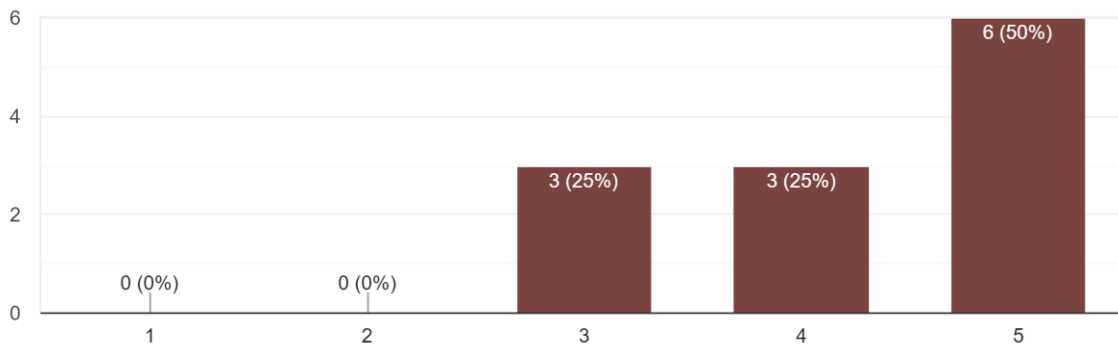
How did you feel about the radio?

12 responses



How did you feel about the writing?

12 responses

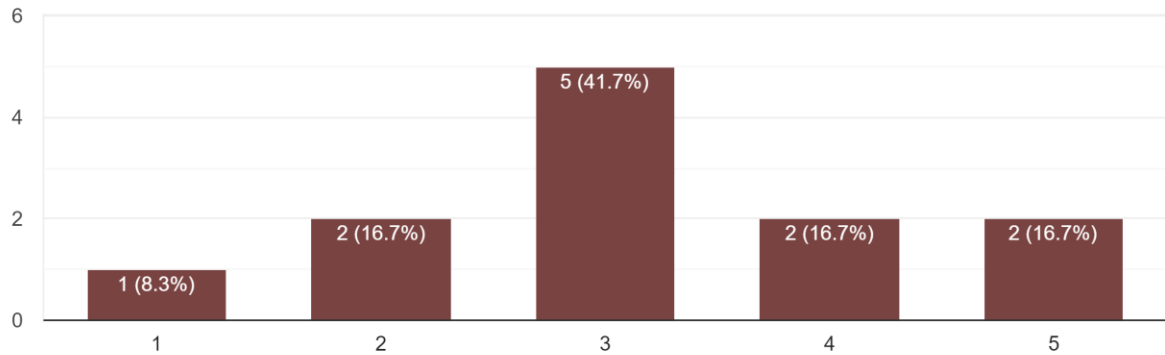


The radio and writing were also well received. Nothing had changed with the writing, so we were curious why our data had shifted to see writing in a more consistently favorable light. Our only hypothesis is that perhaps, with the Cthulhu aspect of the game revealed earlier, all the cheeky jokes about everything being fine landed a lot better. Whatever the explanation, we were happy to see this improvement.

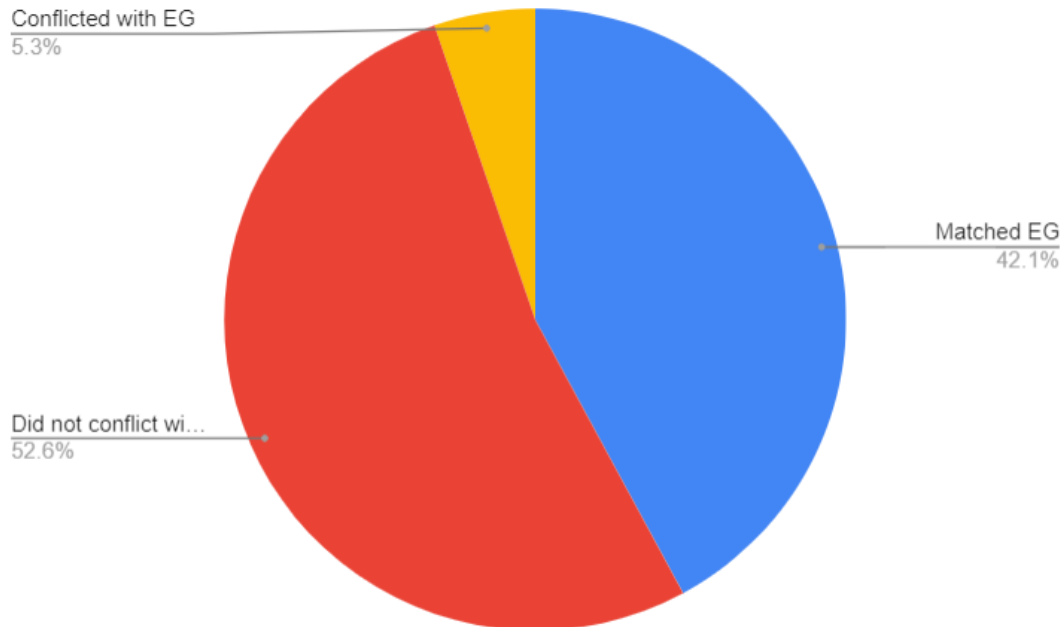


How much did you feel the presence of Cthulhu?

12 responses



The presence of Cthulhu question ended up being a bit of a mixed bag. However, the responses we received were, to some degree, the responses we were aiming for. These responses were results from the first few days of gameplay, where Cthulhu isn't very present or active. As such, the fact that Cthulhu's presence was somewhat middling tracks with what the player would have experienced during these first few days.



We definitely seemed to be improving when it came to meeting our experience goal. While the number of emotions that matched our experience goal remained the same (42.1%), the number that conflicted had decreased significantly (5.3%). We were satisfied with these results, and felt like they told us we were close to or almost matching our experience goal.

Overall, these playtesting results gave us a lot to think about, but once again reinforced that we were heading in the right direction. We were doing better matching our experience goal, our pacing had improved, and we would have a good product to show at PAX East.

## 7.7 PAX East

PAX East, the biggest gaming convention on the eastern seaboard, was our next and final (at least within the scope of this MQP) stop for showcasing and playtesting. PAX East is a showcasing event for small and large games alike to be viewed by the general gaming public, an event that typically boasts tens of thousands of attendees. PAX East this year was haunted by the spectre of COVID-19, but that didn't seem to affect its high attendance numbers.

WPI has a booth at PAX East every year that showcases student games, run by an IMGD IQP team that accepts game submissions of all kinds from the IMGD program at large and selects four games to be displayed at the booth. *The Call of Karen* submitted and was fortunate enough to be selected. Our last playtesting session was the previous week, and we scrambled to make bug fixes and design tweaks based on our playtesting feedback. We went into PAX with the first three days of *The Call of Karen* largely feature complete.

PAX East was easily the most difficult event to get quantifiable feedback from. First of all, due to the event being four days long and the WPI booth being supervised in shifts from different games, we were not able to see every single attendee playthrough of *The Call of Karen*, as we had been able to with previous events. Also, due to the fact that we were there to help showcase all of the games at the booth, not just our own, we were not able to closely observe players and take notes the same way we had with events prior. We also did not administer the playtesting form at PAX East. We didn't mind this lack of data, since we all knew that PAX East was not an event for playtesting. PAX East was strictly for showcasing the product you have and for garnering interest from the general public.

However, we did still manage to get some information from PAX East. Several attendees asked us where they could purchase the game or find it online, which we found notable. The vast majority of games available at PAX East are those that have been or are going to be commercially released by professional game developers, so the fact that people felt like our game was high enough quality to be available commercially told us we were succeeding at our goal of creating a game of high enough quality for commercial release.

## 7.8 COVID-19 and Playtesting

It would be a disservice to this paper to talk about playtesting without discussing how difficult it became during the final term of this project. Halfway through D term break, it was announced that due to the increased spread of COVID-19 and the risk of infection posed by large gatherings, D term would be completed entirely remotely off campus. We were lucky that we all had the hardware necessary to continue the MQP without access to the computer labs on campus, but it certainly did have an effect on our ability to playtest.

We briefly discussed the possibility of playtesting remotely, but ended up scrapping the idea. Playtesting remotely severely limited the amount of data that we would receive, and the process of sending builds out to and managing testers remotely wasn't worth it to us to get this

limited data. As a result, we do not have any playtesting data from D term. While this lack of data is unfortunate, we still feel that we had enough data to continue improving the game and draw accurate conclusions on whether or not we succeeded with this MQP based on our feedback from our last playtesting session and PAX East.

## 8 Conclusion and Recommendations

This chapter details our conclusion and recommendations for future MQP teams, and analyzes some elements of our developments that were not mentioned in previous chapters. Chapter 8.1 details our scope for this project, chapter 8.2 discusses our team dynamics, chapter 8.3 discusses our team structure, and chapter 8.4 discusses our project management. Chapter 8.5 describes our plans for future work on *The Call of Karen*, and chapters 8.6 and 8.7 discuss what went wrong and what went right respectively.

### 8.1 Scope

On a high level, our scope was defined in parallel to our goal: to create a complete game of high enough quality for release. Many of our design decisions were informed by our emphasis on limiting scope. On the art side, this included not modeling characters, keeping the house small, and leaving out areas not essential to gameplay such as the bedrooms and bathroom. This allowed the artists to focus on building a dense and detail-rich gameplay environment that feels authentic and alive. On the technical side, the gameplay was built off of simple, reusable mechanics: picking up/placing objects, and occasionally activating secondary behaviors for held objects (running the vacuum, pouring salt). Our programmers also opted to utilize the blueprint system instead of diving into C++ right away in order to quickly prototype and graybox gameplay elements. This also gave us room to adjust our scope progressively, as we could easily add in/cut out tasks without adversely affecting the overall gameplay experience.

The largest outside factor limiting our scope was our decision to use Unreal Engine. While the underlying engine itself is very powerful, and the skills we picked up through using it were very useful, the editor itself is very slow and unstable, and introduced several nightmarish technical constraints over the course of the project. Sections of the documentation are also very lacking, and we were further disadvantaged by the Unreal Wiki and all its content being spontaneously taken offline during D term.

At the time of writing, Unreal is still a relevant tool in industry, and a good option for realtime applications and cinematic rendering. It makes it easier to attain high-quality aesthetics, and provides some powerful VFX tools for artists. From a technical perspective however, for the scope of an MQP aiming to create a complete game, we would strongly recommend picking an engine that integrates well with version control systems and provides a non-visual scripting environment that does not require recompiling the engine.

### 8.2 Team Dynamics

Our team dynamics worked out astoundingly well. Our team members did not all know each other especially well going into the project, but we got along well early on and were all excited about taking on the task of making a game from start to finish, as well as about the game's concept once it was decided. More importantly with everyone working on a student schedule, all our team members regularly communicated their progress and constraints. We did so in a largely informal manner - aside from timelining major milestones, we kept up-to-date todo lists, Google Docs, and kept each other updated via a Facebook Messenger group chat. In

general, we found the most critical aspects of our team dynamics to be clear communication, matching motivation/commitment, and a willingness to re-evaluate/re-scope aspects of the game and regularly adjust our timeline based on our individual progress/constraints.

## 8.3 Team Structure

As mentioned in chapter 1, this Major Qualifying Project (MQP) was completed by four Worcester Polytechnic Institute (IMGD) students. The project's artists were seniors Kate Olguin, majoring in Interactive Media and Game Development (IMGD), and Thomas Tawadros, majoring in Interactive Media and Game Development with a concentration in Technical Art (IMGD BA). Kate Olguin also served as the project's producer. The project's programmers were Diana Kumykova and Mikel Matticoli, both juniors majoring in Computer Science and Interactive Media and Game Development (CS/IMGD). The small size of our team meant that teammates often had to step into several different roles. Both artists were also effectively technical and VFX artists, responsible for lighting, pipeline management, rigging, shader creation, and occasional blueprinting. We lacked dedicated audio personnel, but one of our artists and programmers were able to share the audio load. Nobody was officially denoted as a designer, but each teammate contributed significantly to the design of the game. Additionally, while our producer did hold us to deadlines and assigned tasks, the team was autonomous to the degree where close management was unnecessary.

The team's makeup, size, and the multidisciplinary nature of its members proved to be an excellent balance for our project. All of the teammates had a steady stream of work to complete throughout the course of development, and there were no points when team members simply had nothing to do. While we didn't have dedicated audio or design personnel, we are inclined to believe that the addition of dedicated audio or design personnel may not have helped us significantly. We were able to create and implement a significant amount of high quality audio without needing an official sound designer, with the one exception being that we don't have original music. Additionally, while a dedicated designer may have been helpful in the early stages of the game, but as production continued we believe they may have run out of work to do had they not possessed any other skills. As it was, the division of labor worked out well for us.

## 8.4 Project Management

As touched upon in the previous section, one of our artists served as the producer for this MQP. As the producer, she organized and directed team meetings, assigned tasks, and held the team to important deadlines. The team proved to be easy to manage, as members had a high level of autonomy and could be trusted to complete tasks quickly with a high level of quality. Early on in the project it was made clear that we needed clear and frequent communication, and as a result we were able to avoid any major miscommunications throughout the course of development. Additionally, we had no interpersonal conflicts throughout the course of the project, so no conflict management was necessary.

Worth noting is the fact that the producer also served as an artist on the team. Similar to our concerns about a dedicated designer, while a dedicated producer would have certainly had a more constant flow of work throughout development, we feel as though a producer on a team

this size must also work in another discipline so as to be useful when there is less managing to do. Additionally, our producer possessed knowledge in many areas of game development and professional game development experience through internships, and was able to use this knowledge to communicate effectively with team members and draw from industry standard production practices. It is our producer's belief that lessons learned from previous game development experience and knowledge of these practices was a large part of the reason production continued smoothly. Of course, the team members themselves also cannot be undersold—everyone worked hard and wanted to see the game come to fruition, and were model teammates.

## 8.5 Future Work

The goal of this MQP was to create a fun, funny, full-fledged game that was of high enough quality for release. We believe that we have achieved this goal, but while our minimum viable product is complete, there is still more we want to do to polish the game for release. There's more audio that we want to implement, a few textures to tweak, and several other odds and ends that we want to wrap up before sending this game out into the world. We plan on doing a bit of marketing as we polish the game, then releasing it onto an online distribution platform. Our target platform is Steam, but we want to put more consideration into the Epic Games Store or Itch.io before nailing anything down too quickly. We plan on releasing some time in the early summer, and will continue work on it until then, though likely at a slower pace compared to the school year.

## 8.6 What Went Wrong

Our biggest setbacks were obscure technical issues introduced by Unreal, resulting in lots of lost/repeated hours of work, and personal time constraints. Despite the benefits of the tools we used, using industry standard version control tools in combination with Unreal engine proved an unexpectedly massive headache. While we had a really great team comprised of competent and talented individuals, our team members were also constantly busy if not overworked for much of the duration of the project. With multiple fluctuating schedules coming into play on top of the technical issues, it was difficult to forecast the time cost of different phases of the project, and we inevitably ended up running into D term and had to cut out some things last minute.

## 8.7 What Went Right

After a year of design and development, we achieved all our major goals. We created a completed game, with a fun, funny and cohesive play experience. We created an immersive environment and an engaging story, with all of the art, code, and dialogue designed and built from scratch, that will be ready to release with a few more weeks of relaxed polishing. We were able to accomplish this in spite of aggressive technical difficulties, and maintained a positive and enjoyable team environment despite the stress of balancing these difficulties with a tight timeline. We also learned a whole lot along the way about the tools, process, and nuances of

creating a production-quality game from start to finish, and got the chance to show our game at PAX East, MassDiGi Game Challenge (where we won an award for our pitch), WPI's Virtual Research Showcase, Showfest, and CS/IMGD Project Presentations.

## 9 References

- 1950s housewife Pinterest meme. (2012). In *TeamJimmyJoe.com*.  
<https://teamjimmyjoe.com/2012/12/1950s-housewife-meme/>
- 20th Century Fox. (1956). The Man in the Gray Flannel Suit Movie Poster. In *Internet Movie Database*.  
<https://www.imdb.com/title/tt0049474/mediaviewer/rm2867336448>
- Ball, L. (1950, January 30). *My Favorite Husband*. Stardust Records.  
<https://www.youtube.com/watch?v=V-MTi0v3wI8>
- Beaumont, C. (2016). What Do Women Want? Housewives' Associations, Activism and Changing Representations of Women in the 1950s. *Women's History Review*, 26(1), 147–162. Taylor & Francis Online.  
<https://doi.org/10.1080/09612025.2015.1123029>
- Bertz, M. (2019, April 2). *Project Wight Reemerges As Darkborn*. Game Informer.  
<https://www.gameinformer.com/preview/2019/04/02/project-wight-reemerges-as-darkborn>
- Bethesda Softworks. (1997a). Original Fallout character interaction screen. In *Internet Movie Database*.  
<https://www.imdb.com/title/tt0134648/mediaviewer/rm706179072>
- Bethesda Softworks. (1997b). Original Fallout isometric player perspective. In *Internet Movie Database*.  
[https://store.steampowered.com/app/38400/Fallout\\_A\\_Post\\_Nuclear\\_Role\\_Playing\\_Game/](https://store.steampowered.com/app/38400/Fallout_A_Post_Nuclear_Role_Playing_Game/)
- Bethesda Softworks. (2015a). Fallout 4 screenshot with robot. In *Steam store*.  
[https://store.steampowered.com/app/377160/Fallout\\_4/](https://store.steampowered.com/app/377160/Fallout_4/)
- Bethesda Softworks. (2015b). Prepare for the Future game poster. In *Internet Movie Database*.  
<https://www.imdb.com/title/tt4741306/mediaviewer/rm1744991232>
- Bethesda Softworks. (2020). Fallout 76 Wastelanders monster screenshot. In *Steam store*. [https://store.steampowered.com/app/1151340/Fallout\\_76/](https://store.steampowered.com/app/1151340/Fallout_76/)
- Bioshock Infinite*. (2013, March 26). [Game]. Irrational Games, 2K Games.
- Blakemore, E. (2016, May 11). *How Jell-O Wobbled Its Way to Pop Culture Greatness*. JSTOR Daily. <https://daily.jstor.org/jello-in-pop-culture/>
- Brandt, M. (1957). *The Good Housekeeping Book of Home Decoration*. The



Hearst Corporation.

Brocka, B. (2016, April 16). *terminology - What is a "toast notification"?* User Experience Stack Exchange; Stack Exchange Inc.

<https://ux.stackexchange.com/questions/11998/what-is-a-toast-notification>

CBS. (1951). I Love Lucy promotional poster. In *Internet Movie Database*.

<https://www.imdb.com/title/tt0043208/mediaviewer/rm2306015744>

Charles, K. K., Guryan, J., & Pan, J. (2018). The Effects of Sexism on American Women: The Role of Norms vs. Discrimination. In *nber.org*.

<https://www.nber.org/appendix/w24904/effects-sexism-american.pdf>

ClickAmericana. (2019). 1950s housewife at the oven. In *ClickAmericana*.

<https://clickamericana.com/topics/home-garden/how-to-be-a-perfect-fifties-housewife-in-the-kitchen>

*Cooking Simulator*. (2019, June 6). [Game]. PlayWay.

Dickson, P. (2007, November 6). *Sputnik's Impact on America*. Pbs.Org.

<https://www.pbs.org/wgbh/nova/article/sputnik-impact-on-america/>

Dormann, C., & Biddle, R. (2009). A Review of Humor for Computer Games:

Play, Laugh and More. *Simulation & Gaming*, 40(6), 802–824.

<https://doi.org/10.1177/104687810934139>

Dwyer, M. (2015). *Back to the Fifties: Nostalgia, Hollywood Film, & Popular Music of the Seventies & Eighties*. Oxford University Press

*Event Dispatchers*. (n.d.). *Docs.Unrealengine.Com; Epic Games*. Retrieved March 2020, from

<https://docs.unrealengine.com/en-US/Engine/Blueprints/UserGuide/EventDispatcher/index.html>

*Firewatch*. (2016, February 9). [Game]. Campo Santo.

*Freesound - Freesound*. (2012). Freesound.Org. <https://freesound.org/>

George, A. (2016), Gray Flannel Suit or Red Strait Jacket? Anticommunism and

the Organization Man in Postwar Fiction and Film. *J Pop Cult*, 49: 1320-1340.

doi:10.1111/jpcu.12494

Ghodrati, N. (2013). *The Creation, Evolution and Aftermath of Lovecraftian*

*Horror* [Master'sThesis].

<https://www.duo.uio.no/bitstream/handle/10852/37090/NeginxGhodratix-xMasterxxThesisxxMayx2013x.pdf?sequence=1&isAllowed=y>

Hall, C. (2018, May 30). *The Fallout timeline*. Polygon; Polygon.

<https://www.polygon.com/features/2015/11/9/9646378/fallout-timeline-4-3-2>

- Health, C. for D. and R. (2019). IDE Institutional Review Boards (IRB). *FDA*.  
<https://www.fda.gov/medical-devices/investigational-device-exemption-ide/ide-institutional-review-boards-irb>
- HorrorBabble. (2017). "The Call of Cthulhu" by H. P. Lovecraft / Cthulhu Mythos (4/14) [YouTube Video]. In *YouTube*.  
<https://www.youtube.com/watch?v=2xB4yHCgly8>
- HorrorBabble. (2018). "At the Mountains of Madness" by H. P. Lovecraft / Cthulhu Mythos (8/14) [YouTube Video]. In *YouTube*.  
<https://www.youtube.com/watch?v=XTk0iROhSOc>
- Huq, R. (2013). *Making Sense of Suburbia through Popular Culture*. London: Bloomsbury Academic. Retrieved March 20, 2020, from  
<http://dx.doi.org/10.5040/9781472544759>
- I Expect You to Die*. (2016, December 13). [Game]. Schell Games.
- Job Simulator*. (2016, April 5). [Game]. Owlchemy Labs.
- Kallio, O., & Masoodian, M. (2019). Featuring comedy through ludonarrative elements of video games. *Entertainment Computing*, 31, 100304.  
<https://doi.org/10.1016/j.entcom.2019.100304>
- KOMA - Oklahoma's Greatest Hits*. (1964, January 5). KOMA.  
<https://www.youtube.com/watch?v=vQHPEFw0zHI>
- Lamb, V. M. (2011). The 1950's and the 1960's and the American Woman : the transition from the "housewife" to the feminist. In *HAL*. <https://dumas.ccsd.cnrs.fr/dumas-00680821/document>
- Lin, L. yi, Sidani, J. E., Shensa, A., Radovic, A., Miller, E., Colditz, J. B., Hoffman, B. L., Giles, L. M., & Primack, B. A. (2016). ASSOCIATION BETWEEN SOCIAL MEDIA USE AND DEPRESSION AMONG U.S. YOUNG ADULTS. *Depression and Anxiety*, 33(4), 323–331. Wiley Online Library.  
<https://doi.org/10.1002/da.22466>
- Loucks, D. K. (2020, January 20). *Lovecraftian Movies and Television Shows*. [Www.Hplovecraft.Com. http://www.hplovecraft.com/popcult/moviestv/](http://www.hplovecraft.com/popcult/moviestv/)
- Lundy, R. (1962, February 28). *Ron Lundy Show*. WIL.  
<https://www.youtube.com/watch?v=4WHGOBj1YNk>
- Moffatt, M. (2020, January 27). *What Caused the Post-War Economic Housing Boom After WWII?* ThoughtCo; ThoughtCo. <https://www.thoughtco.com/the-post-war-us-economy-1945-to-1960-1148153>

- Mother Simulator*. (2018, March 20). [Game]. Steppe Hare Studio.
- Murrow, E. (1950, December 12). *Hear It Now*. CBS.  
<https://www.youtube.com/watch?v=kH0tlgqvuw0&feature=youtu.be>
- Neuhaus, J. (2011). *Housework and Housewives in American Advertising: Married to the Mop*. Palgrave MacMillan.  
[https://books.google.com/books/about/Housework\\_and\\_Housewives\\_in\\_American\\_Adv.html?id=Um89YgEACAAJ&source=kp\\_book\\_description](https://books.google.com/books/about/Housework_and_Housewives_in_American_Adv.html?id=Um89YgEACAAJ&source=kp_book_description)
- Neverender. (2014, April 9). *Perforce (and other version control) best practices*. Unreal Engine Forums. <https://forums.unrealengine.com/development-discussion/c-gameplay-programming/2678-perforce-and-other-version-control-best-practices>
- O'Brien, J. (1965, February 24). *WMCA "Goodguy" Radio*. WMCA.  
<https://www.youtube.com/watch?v=UQnSBkzga6Y>
- Omidasalar, A. (2018). Posthumanism and Un-Endings: How Ligotti Deranges Lovecraft's Cosmic Horror. *The Journal of Popular Culture*, 51(3), 716–734. Wiley Online Library. <https://doi.org/10.1111/jpcu.12681>
- Outlast*. (2013, September 4). [Game]. Red Barrels.
- Pantic, I. (2014). Online Social Networking and Mental Health. *Cyberpsychology, Behavior, and Social Networking*, 17(10), 652–657. PMC.  
<https://doi.org/10.1089/cyber.2014.0070>
- Perforce Helix Core Version Control | Perforce*. (2019). *Www.Perforce.Com*.  
<https://www.perforce.com/products/helix-core>
- Poplawski, Stephen J. 1885 - 1956. (2017, August 8). Wisconsin Historical Society.  
<https://www.wisconsinhistory.org/Records/Article/CS11926>
- Pruitt, S. (2019, March 13). *An Ode to the Massive Sears Catalog, Which Even Delivered Houses by Mail*. HISTORY. <https://www.history.com/news/sears-catalog-houses-hubcaps>
- React State*. (n.d.). *Www.W3schools.Com; W3Schools*. Retrieved March 2020, from  
[https://www.w3schools.com/react/react\\_state.asp](https://www.w3schools.com/react/react_state.asp)
- Retro House Plans and Home Plans for Retro Style Home Designs*. (n.d.).  
 Www.Familyhomeplans.Com. Retrieved May 3, 2020, from  
[https://www.familyhomeplans.com/search\\_results.cfm?housestyle=28](https://www.familyhomeplans.com/search_results.cfm?housestyle=28)
- Roberteker. (2018, November 29). *Overnight, half of my objects disappeared*. Unreal Engine Forums. <https://forums.unrealengine.com/unreal-engine/unreal-studio/1557134-overnight-half-of-my-objects-disappeared>

- RKO Radio Pictures. (1946). It's a Wonderful Life Movie Poster. In *Internet Movie Database*. <https://www.imdb.com/title/tt0038650/mediaviewer/rm3862243328>
- Sears Roebuck & Co. (1960). 1960 Sears Catalog Cover. In *Mid-Century Living*. <http://midcenturyliving.blogspot.com/2011/04/sears-1960-catalog.html>
- Silver Has Mystic Powers*. (n.d.). TV Tropes. Retrieved May 3, 2020, from <https://tvtropes.org/pmwiki/pmwiki.php/Main/SilverHasMysticPowers>
- Sketchy Panda Games. (2015a). Aberford Peggy says... promotional image. In *CNET*. <https://www.cnet.com/news/aberford-1950s-housewives-saving-the-world-from-zombies/>
- Sketchy Panda Games. (2015b). Aberford promotional image. In *Kickstarter*. [https://www.kickstarter.com/projects/910316779/aberford-a-game-of-zombies-and-50s-housewives?ref=nav\\_search&result=project&term=aberford](https://www.kickstarter.com/projects/910316779/aberford-a-game-of-zombies-and-50s-housewives?ref=nav_search&result=project&term=aberford)
- Skyrim*. (2011, November 11). [Game]. Bethesda Game Studios.
- Somers, J. (2020, March 27). *Biography of H. P. Lovecraft, American Writer, Father of Modern Horror*. ThoughtCo. <https://www.thoughtco.com/biography-of-h-p-lovecraft-american-writer-4800728>
- Sony Computer Entertainment. (2014). Bloodborne promotional image. In *GotGame.com*. <https://gotgame.com/2014/09/16/apply-bloodborne-alpha-test/>
- The Editors of Encyclopaedia Britannica. (2019, November 11). *J.G. Ballard | British author*. Encyclopedia Britannica; Encyclopædia Britannica, inc. <https://www.britannica.com/biography/J-G-Ballard>
- The Flame in the Flood*. (2016, February 24). [Game]. The Molasses Flood.
- The Legend of Zelda: Breath of the Wild*. (2017, March 3). [Game]. Nintendo.
- The Sinking City*. (2019, June 25). [Game]. Frogwares.
- THQ Inc. (2005a). Capitol City promotional image. In *MobyGames*. <https://www.mobygames.com/game/destroy-all-humans/promo/promoImageld,148166/>
- THQ Inc. (2005b). Santa Modesta promotional image. In *MobyGames*. <https://www.mobygames.com/game/destroy-all-humans/promo/promoImageld,148160/>
- THQ Inc. (2011). Destroy All Humans! mind-reading screenshots. In *YouTube.com*. <https://www.youtube.com/watch?v=VqngJZxmoVE>
- Trover Saves the Universe*. (2019, May 31). [Game]. Squanch Games.

*UGameInstance*. (n.d.). *Docs.Unrealengine.Com; Epic Games*. Retrieved March 2020, from [https://docs.unrealengine.com/en-](https://docs.unrealengine.com/en-US/API/Runtime/Engine/Engine/UGameInstance/index.html)

[US/API/Runtime/Engine/Engine/UGameInstance/index.html](https://docs.unrealengine.com/en-US/API/Runtime/Engine/Engine/UGameInstance/index.html)

*What Remains of Edith Finch*. (2017, April 25). [Game]. Giant Sparrow, Annapurna Interactive.

Whirlpool Corporation. (1950). My Washdays are Holidays Now! advertisement. In *Pinterest.com*.

<https://i.pinimg.com/originals/45/fb/6e/45fb6ef2dae71be7ef478d65fe6d2e8b.jpg>

# 10 Appendices

## Appendix A: Final Game Progression List

### Day 1

- Make breakfast
- Cleaning (weird)
  - Floaty
- Set table

### Day 2

- Breakfast (float & drop)
  - Food flies out of the fridge at you
- Vacuum (Regular)
- Set table (slightly weird)
  - Float and then drop

### Day 3

- Breakfast (weird)
  - Floaty
    - No drop
- Vacuum (weird)
  - Slime & on walls/ceilings
- Order silver bullets
  - Take magazine, right click with envelope, put in mail slot
    - We need a mail slot I guess
- Set table (weird)
  - Floaty

### Day 4

- Make breakfast (same as day 2)
- Get ancient book in the mail from Susan
- Modify vacuum
  - Modify slime getting vacuumed up
- Vacuum (weird, slime on walls and ceilings snakes around the room)
  - Amoeba distortion
- Write a sarcastic thank you note to Susan for the ancient book
  - Right click paper with pencil, put in envelope, put in mail slot
- Open the closet
  - Closet is full of demon fighting things, grab the holy water from the top shelf
- Set table
  - Meatloaf is weird, Consecrate meatloaf with holy water

### Day 5

- Make breakfast (weird again, floaty no drop)
- Modify vacuum

- Right click vacuum with silver bullets, it's got silver bullets now
- Day end, no dinner

#### Day 6

- Make breakfast (force pulses every 1-2 seconds) up and down
- Clean up
  - Shit floating ominously in the air, go up to it with vacuum and blast it out of the air, pick it up
  - Only 4 or 5 books

#### Day 7

- Throw the vacuum in the demon fighting closet
  - The vacuum is COOL
- Step outside, fog, use vacuum on Cthulhu
- Fade to black or whatever
- Next day, husband sasses you, rev up your vacuum

## Appendix B: IRB Exemption Example Survey

### The Call of Karen Survey

Your feedback helps us improve our game!

**1. Did you understand what you were supposed to do?**

*Mark only one oval.*

	1	2	3	4	5	
Not at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Definitely

**2. Did you find the game humorous?**

*Mark only one oval.*

	1	2	3	4	5	
Not at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Definitely

**3. What did you think of the art?**

*Mark only one oval.*

		1	2	3	4	5	
It did not fit the game	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	It fit the game very well

**4. What did you think of the controls?**

*Mark only one oval.*

		1	2	3	4	5	
Not intuitive at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Intuitive

**5. Please list 3 emotions that the game made you feel.**

---

**6. What would be one sentence you would use to describe your experience?**

---

---

---

---

---



7. Any additional comments for us?

---

---

---

---

---

**Thanks!**

---

## Appendix C: Alphafest Survey

### Call of Karen Feedback!

Help us improve our game please, thanks!

\* Required

1. How understandable were the controls? \*

Mark only one oval.

	1	2	3	4	5	
I did not know what was happening	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	They were intuitive

2. Did you understand what you were supposed to do? \*

Mark only one oval.

	1	2	3	4	5	
I had no idea what was happening	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I completely understood

3. How difficult was it to navigate through the play space? \*

Mark only one oval.

	1	2	3	4	5	
Very difficult	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Easy

4. How did you feel the art style worked with the game? \*

Mark only one oval.

1      2      3      4      5

---

It didn't fit at all      It fit very well

5. How did you feel about the lighting? \*

Mark only one oval.

1      2      3      4      5

---

I didn't like it      I liked it

6. How did you feel about the radio? \*

Mark only one oval.

1      2      3      4      5

---

I didn't like it at all      I liked it

7. How did you feel about the writing? \*

Mark only one oval.

1      2      3      4      5

---

I didn't like it at all      I liked it

8. What are 3 emotions you felt while playing this game? \*

---

9. What's one sentence you would use to describe your playthrough? \*

---

---

---

---

---

10. If you found any bugs, please describe them here.

---

---

---

---

---

11. Any final comments or thoughts?

---

---

---

---

---

12. Our next question asks for your email. What do you want yours to be for? Check all that apply.

*Check all that apply.*

- Confirm that I playtested for this game so I can get credit
- Infrequent updates on how the Call of Karen is doing as we get closer to release
- I'm not putting in my email

13. Please enter your email. (If you aren't doing this for playtesting or updates, you can just leave this blank)

---

Thanks for your help! We really appreciate it.

---

## Appendix D: Playtesting Survey 1

### Call of Karen Feedback!

Help us improve our game please, thanks!

\* Required

1. How understandable were the controls? \*

*Mark only one oval.*

	1	2	3	4	5	
I did not know what was happening	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	They were intuitive

2. Did you understand what you were supposed to do? \*

*Mark only one oval.*

	1	2	3	4	5	
I had no idea what was happening	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I completely understood

3. How difficult was it to navigate through the play space? \*

*Mark only one oval.*

	1	2	3	4	5	
Very difficult	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Easy

4. How did you feel about the lighting? \*

Mark only one oval.

	1	2	3	4	5	
I didn't like it	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I liked it

5. How did you feel about the radio? \*

Mark only one oval.

	1	2	3	4	5	
I didn't like it at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I liked it

6. How did you feel about the writing? \*

Mark only one oval.

	1	2	3	4	5	
I didn't like it at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I liked it

7. What are 3 emotions you felt while playing this game? \*

\_\_\_\_\_

8. What's one sentence you would use to describe your playthrough? \*

---

---

---

---

---

9. If you found any bugs, please describe them here.

---

---

---

---

---

10. Our next question asks for your email. What do you want yours to be for? Check all that apply.

*Check all that apply.*

- Confirm that I playtested for this game so I can get credit
- Infrequent updates on how the Call of Karen is doing as we get closer to release
- I'm not putting in my email

11. Please enter your email. (If you aren't doing this for playtesting credit, you can just leave this blank)

---

12. Any final comments or thoughts?

---

---

---

---

---

Thanks for your help! We really appreciate it.

---

## Appendix E: Playtesting Survey 2

### Call of Karen Feedback!

Help us improve our game please, thanks!

**\* Required**

1. How understandable were the controls? \*

*Mark only one oval.*

	1	2	3	4	5	
I did not know what was happening	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	They were intuitive

2. Did you understand what you were supposed to do? \*

*Mark only one oval.*

	1	2	3	4	5	
I had no idea what was happening	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I completely understood

3. How difficult was it to navigate through the play space? \*

*Mark only one oval.*

	1	2	3	4	5	
Very difficult	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Easy

4. How did you feel about the lighting? \*

Mark only one oval.

	1	2	3	4	5	
I didn't like it	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I liked it

5. How did you feel about the radio? \*

Mark only one oval.

	1	2	3	4	5	
I didn't like it at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I liked it

6. How did you feel about the writing? \*

Mark only one oval.

	1	2	3	4	5	
I didn't like it at all	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I liked it

7. How much did you feel the presence of Cthulhu?

Mark only one oval.

	1	2	3	4	5	
There was Cthulhu?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	I felt it a significant amount.

8. What are 3 emotions you felt while playing this game? \*

\_\_\_\_\_



9. What's one sentence you would use to describe your playthrough? \*

---

---

---

---

---

10. If you found any bugs, please describe them here.

---

---

---

---

---

11. Our next question asks for your email. What do you want yours to be for? Check all that apply.

*Check all that apply.*

- Confirm that I playtested for this game so I can get credit
- Infrequent updates on how the Call of Karen is doing as we get closer to release
- I'm not putting in my email

12. Please enter your email. (If you aren't doing this for playtesting credit, you can just leave this blank)

---

13. Any final comments or thoughts?

---

---

---

---

---

Thanks for your help! We really appreciate it.

Appendix F: MassDiGI Game Challenge Slides





## TRAILER

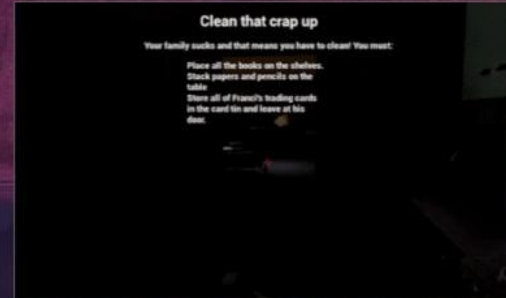


# GAMEPLAY

DAY 1



- Task-based
- Varies by day
- More days → More Strange
- Cooking Sim/Mother Sim but with a meddling Cthulhu

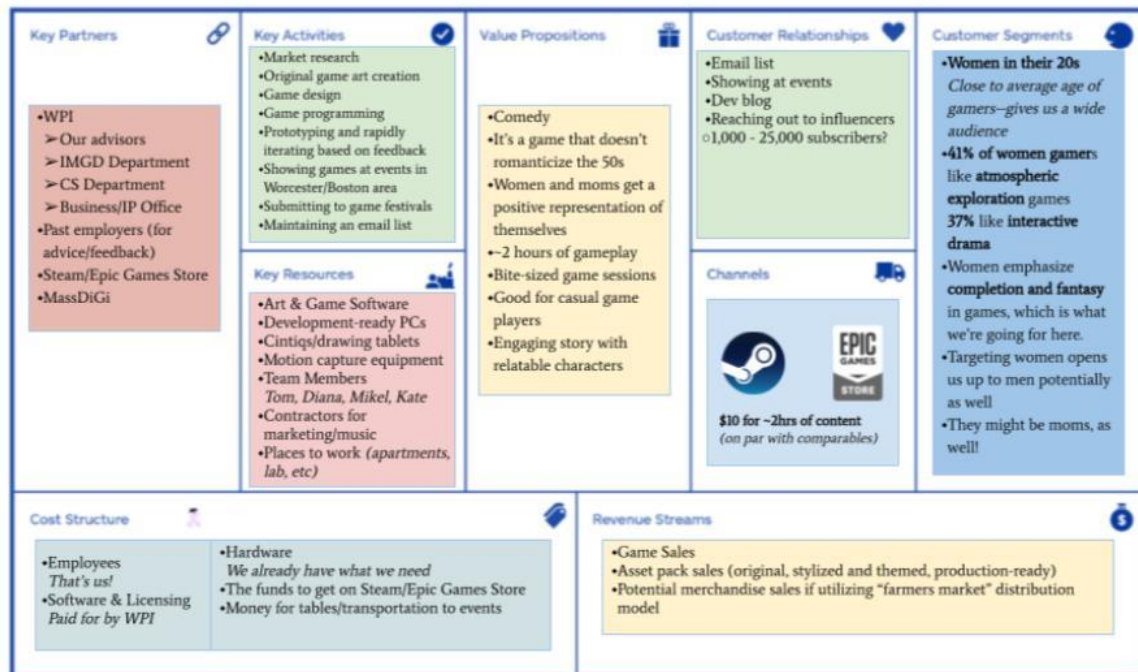


DAY 3

# ART STYLE

THE CALL OF KAREN






**Customer Relationships**

- Email list
- Showing at events
- Dev blog
- Reaching out to influencers
- 1,000 - 25,000 subscribers?

**Channels**



**\$10 for ~2hrs of content**  
*(on par with comparables)*

**Customer Segments**

- Women in their 20s**  
*Close to average age of gamers—gives us a wide audience*
- 41% of women gamers** like **atmospheric exploration** games
- 37% like interactive drama**
- Women emphasize **completion and fantasy** in games, which is what we're going for here.
- Targeting women opens us up to men potentially as well
- They might be moms, as well!

# MARKETING

- Email List
- Events
- Influencers



## COMPLETED

- ◇ Key Gameplay & Mechanics
- ◇ Core Artwork
- ◇ Story
- ◇ Email list started

## IN PROGRESS

- ◇ Shaders
- ◇ Playtesting & adjustments based on that
- ◇ UI

## TO BE DONE

- ◇ More playtesting and adjustments
- ◇ Updating audio
- ◇ Showing at events
- ◇ Visual polish
- ◇ Email list
- ◇ Reach out to influencers

SEPTEMBER 2019

RELEASE TIMELINE

MAY 2020



