SPI-Glass

(Society for Paranormal Investigation) - Augmented Reality Mobile Game

A Major Qualifying Project Report
submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the

Degree of Bachelor of Science
in Computer Science

Degree of Bachelor of Science
in Interactive Media & Game Development

Degree of Bachelor of Arts
in Interactive Media & Game Development

Dr. Gillian Smith (IMGD/CS)
Ben Schneider (IMGD)

Table of Contents

Attribution

Abstract	Mason Moore
Introduction	Mason Moore, Keenan Porter, Jaliah Hippolyte, Jimmy Martella, Joshua Cohen,
Game Design	Joshua Cohen, Mason Moore, Keenan Porter, Jaliah Hippolyte, Jimmy Martella
Tech (Design)	Keenan Porter
Narrative (Design)	Thomas Jolicoeur, Mason Moore
Art (Design)	Jaliah Hippolyte, Mason Moore
Audio (Design)	Jimmy Martella
Tech (Implementation)	Keenan Porter, Joshua Cohen
Narrative (Implementation)	Jaliah Hippolyte, Thomas Jolicoeur, Mason Moore
Art (Implementation)	Jaliah Hippolyte, Mason Moore
Audio (Implementation)	Jimmy Martella
Post Mortem	Mason Moore, Keenan Porter

Abstract

SPI-Glass is a multi-year project with the goal of designing an Augmented Reality game for mobile devices. This report documents the production of SPI-Glass during the 2025-2026 academic year. SPI, or the Society for Paranormal Investigation is a research body and institution that protects the public from encounters with the supernatural. Players take the role of a new recruit, utilizing their phone camera to find and disperse ghosts in an immersive first person perspective.

Our Team

SPI-Glass, was developed by five undergraduate students at Worcester Polytechnic Institute. The team consists of Producer and Technical Artist Mason Moore, 2D Artist and Technical Artist Jaliah Hippolyte, Lead Programmer Keenan Porter, Designer and Programmer Joshua Cohen, and Audio Designer Jimmy Martella. Additionally, three graduate students provided support throughout development, Programmer Thomas Branchaud, Narrative Designer Thomas Jolicoeur, and Designer Minyun Pan.

Acknowledgements

We would like to give special thanks to advisors Professor Ben Schnedier and Professor Gillian Smith for their guidance and feedback throughout the project, narrative advisor Katherine Crighton for their advice, research, comments, and conceptual support during development, and computer science advisor Professor Matt Ahrens for his technical counsel.

We would also like to thank Professor Josiah Gutherie and the students from the IMGD 4099 class for their independent contributions through the length of the 4099 course.

Special thanks to Sophie Maguire, Daniel Hudon, Charlie Doud, Eric Maher, and Renée Michaud from IMGD 4099, who contributed assets to the development of our project.

1 - Introduction

SPI Glass is an augmented reality (AR) ghost investigation and hunting game, where players join a secret society of paranormal investigators tasked with researching the supernatural,

protecting the public from spiritual hauntings, and helping lost ghosts find peace. Using tools to navigate a hidden world of spiritual energy through their device, the player will research the secrets of the supernatural world and dispel the ghosts which lurk around every corner.

While traveling, players will be notified of spiritual hauntings in the world around them, which they can choose to investigate. Arriving at these hauntings the player must search the area in AR, uncovering markings, tracks, and other hints to the nature of the hauntings and the kind of ghost causing it. Using what they learned, players force the ghost to show itself, where they can then combat and disperse it.

Following each encounter, discoveries are automatically recorded in their journal, contributing to ongoing research of the spiritual world. Over the course of the game they will discover different ghost classifications, behaviors, and elements of spiritual energy, which will help them become more efficient at dealing with hauntings. When a ghost escapes your grasp or you fail to dispel them properly, they may return, stronger than before. As these ghosts become more notorious, nearby players are more likely to encounter them as well, with each player observing the shifting hierarchy of the spiritual world. The most powerful and elusive of these ghosts create widespread disturbances. Locating and dispersing the most infamous ghosts takes a cooperative effort from the local SPI community, each player can gather clues to the overarching anomaly while investigating nearby haunts.

Creative Vision and Pillars

Our vision for SPI Glass is a game where players join a fantastical society of paranormal investigators, empowered by a strong arsenal of tools for revealing, capturing, and cataloguing ghosts around them. This project will focus on a future-friendly framework that is scalable and accommodates multiplayer features.

- ➢ **Explore a Hidden Reality**
    - Attractive, empowering, and inviting, as opposed to frightening or intimidating.
    - Encourages players to see their device as more than a phone (SPI Glass as more than a game-), but as a tool for sensing and interacting with otherworldly phenomena just beyond normal perception.
    - Simulates ghost-hunting in an accessible and fun format.
    - Horror and similar themes will be common, but the player should **never** feel powerless or threatened. The only threat the player should ever encounter should be “mission failure” (ghost escapes, etc).
- ➢ **Social Collaboration with a Special Community**
    - Creates the feeling of an exclusive club where members are privy to privileged knowledge (without truly excluding anyone with cell reception).
    - Puts players in the position of a researcher working with the organisation to help ghosts move on.

    - Tasks players to work with other researchers to collect and share more info about different types of ghosts and encounters.
- ➢ **Procedurally Built and Adaptive**
    - Players build their own narrative through their actions and experiences.
    - Able to be played in any location.

2 - Design

Game Design

The player starts by viewing their position on the world map, given by location data. At points on the map nearby, the player will see leads on spiritual disturbances, selecting these leads will allow them to start an investigation of the area.

The player is given the following information from leads:

- ➢ The general location of the lead.
- ➢ The notoriety of the ghost on location.
- ➢ Some indication as to the nature of the disturbance.

Investigation

The player is then transitioned to an AR scene using their device's camera. In this scene, the player has immediate access to a selection of tools for discovering the type of ghost they are facing, locating the ghost in the space, and dispersing the ghost.

Each ghost is defined by three unique traits that are assigned procedurally that the player can discover.

Traits:

1. **Name**
2. **Classification** *(Spirit, Wraith, or Demon)*
3. **Element** *(Fire, Water, Earth, or Air)*

The investigations will always take place in AR. The player is looking for behavior patterns that indicate the ghost's classification, sigils that exist in the playspace that can combine to reveal the ghost's name, and elemental markings left by the ghost. The player can also find answers by communicating with the ghost through the *Ouija W.R.I.T.E.R.*, or analyzing their environment using the *SPI-Glass*.

Summoning

The player must summon the ghost by writing its name through sigils, or finding the position it's hiding and dowsing it in its elemental weakness. This phase may be used multiple times depending on the ghost.

Dispersal

The player's ability to see and interact directly with the ghost defines the Dispersal phase. Once the ghost has shown itself, the player has to work to suck up and break down the *Ectoplasm* (spiritual energy) around the ghost's core. The danger level represents both the player's progress towards exposing the ghost's core / dispersing it, and the ghost's progress towards powering up and forcing the player to prematurely end the dispersal.

At some intervals, the ghost will guard itself and become immune to losing danger level. The player must use the *P.R.I.E.S.T* to break its defenses, making the ghost vulnerable again allowing the player to decrease the ghost's danger level.

Each ghost classification also has a unique ability that it will trigger to defend itself from a guard break, the player must use tools to counteract the ghost's actions and break their guard.

The Investigation ends when either:

- ➢ The danger level is fully depleted, representing the player successfully dispersing it.
- ➢ The danger level is filled, representing the ghost increasing in power and forcing the player to flee and reassess the ghost later.

When the encounter ends the results of the encounter are displayed and the player returns to the map where they can take on more ghost encounters.

Tools

Tools are a fundamental system in how the player interacts with the game environment. Each has a unique feature that allows for players to find creative solutions to ghost dispersal. No two tools should feel the exact same. Each tool should have a specific purpose and a specific style, making them distinct from one another as well as encouraging the player to develop a unique approach to ghost encounters.

Tools can be classified in different ways, here two possible structures:

Phase Specific

Investigation	
CHALK	*Allows the player to draw symbols of significance, summoning surges of elemental spirit energy, as well as summoning Ghosts after finding the Sigil representing its name.*
E.M.F. 985	*The Energy Magnifier & Finder 985, follows the general rules of a normal EMF reader. This means the player should take out this device and direct the device around the space. The device then gives haptic and audio feedback of the ghosts direction and location. When near the ghost the E.M.F. will automatically shut off so it doesn't explode.*
Ouija W.R.I.T.E.R. 616	*The Ouija Witness Retention Investigation Tool Energy Recycler 616 is a way for the player to more calmly gain information directly from the ghost than a regular Ouija board (which tends to anger the ghosts). This item is placed down, and the ghost will have a chance to interact with it, answering a selected question by the player.*
G.L.A.S.S. 417	*The Geometric Lighting Analyzation Series System 417 is a way for the player to visually inspect the investigation area for Ectoplasm and Elemental Effects. This can also reveal sigils left by restless ghosts that can also reveal their name. This tool is the intended primary method for investigation.*
Dispersal	
S.U.C.K 609	*The Super Universal Capture Kit 609 is a heavily modified Electromagnetic Vacuum. This device removes and disperse ectoplasm from the ghosts, reducing their volatility and eventually releasing the ghost from the mortal plane. This fires in a direct line and requires precise movements to track the ghosts*
NaCl 117	*The Natural Aggregate Chambered Launcher 117 uses specially formulated salt to restrict the ghosts movements. By firing the device at a location in the room, it covers objects and spaces in salt, blocking the ghost from moving into the space. If a ghost is hit by a blast, it has its movement slowed. The device carries only three charges due to budget restraints.* ***Strong against Water element***
P.R.I.E.S.T 266	*The Precision Reducing Investigation Episodic Solvent Tranquilizer 266 is based on a design to fight fires in remote locations away from water sources. This model has been modified to fire a concentrated blast of holy water. This weakens a ghost's defenses allowing players to continue to disperse the ghost. It fires in an arch, forcing players to judge distance and angle to best affect the ghost.* ***Strong against Fire element***
S.B. 002	*The Sponge Box 002 is a device designed to trick evasive ghosts into slowing down long enough for the player to react and disperse the ghost. If a ghost inhabits it, it ejects the ghost and removes a portion of its ectoplasm. This is best placed in the investigation phase.*

Interactions Specific

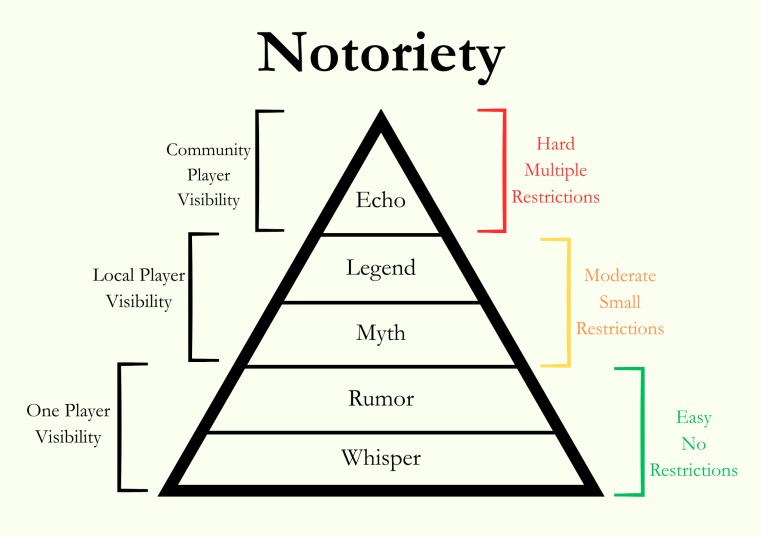
Placeable - *Players using these tools will be able to place them down in the space, this then will be interacted with by ghosts and have effects when they do.*
Ouija W.R.I.T.E.R. 616
S.B. 002
Ranged - *These tools have abilities that allow them to interact with objects far away from the player, or affect the ghost in some way from a distance. These tools will have projectiles.*
S.U.C.K 609
NaCl 117
P.R.I.E.S.T 266
G.L.A.S.S. 417
Passive - *These tools DO NOT directly interact with the world around them. They give particular information or enhance the player in some way that the player can then use to do other things.*
E.M.F. 985

Ghost Types

Each ghost classification also behaves differently and has a unique ability it can use which the player must combat accordingly.

- **Spirits**
  - Will sometimes multiply rapidly and swarm the player. The player has to disperse all the copies separately before they can continue to reduce danger level.
  - Copies will fly at you, raising the danger level.
  - Will float around attempting to distract the player from finding the actual Spirit.
- **Wraiths**
  - Will sometimes hide by possessing an object, requiring you to locate it again.
  - Flies around very quickly, requiring the player to slow it and restrict its movement using Salt.
- **Demons**
  - Will confront you directly and can face-tank a lot of vacuuming.
  - Will throw game objects around the scene at you, raising danger level.
  - Resistant to guard break requiring the player to target specific vulnerable points to break guard.

Notoriety



Another system that will influence this gameplay will be a system known as *Notoriety*. The Notoriety system comes in tiers of increasing difficulty. The base notoriety of any anomaly is whisper. Whisper ghosts can only appear to a single player as a lead. These are the simplest and most basic form of a ghost type and dispersed easily. The next Tier is Rumor. Rumor ghosts are slightly harder and may come with an additional modifier to make dispersing the ghost slightly harder. This tier is still only visible to a single player. The first tier that is visible to multiple players is the Myth tier. They have one or two modifiers that makes them a decent challenge to disperse. Multiple players may have a chance to disperse them, reaping good rewards. Legend Tier ghosts are quite difficult with multiple modifiers that force a *particular* method of dispersal. They are a group of agents to fight and disperse completely. The final tier of the Notoriety system is Past Echoes. Past Echoes are large community events where a multitude of ghosts haunt a region of players. This can include fighting multiple higher notoriety ghosts at once or fighting ghosts of varying types. These events should allow for multiple skill level players to try and tackle and aid in dispersing the haunt.

Importantly, SPI-Glass is not a horror game; When interacting with the ghost, the player should feel empowered, and have multiple options for things they can do to take control of the game state. How the tools feel is incredibly important to how players experience that control, so we aimed to emphasize that with their design. Every tool is operable with only a single finger on the screen, and the margins of error (aiming, pattern recognition, etc) are wide to reduce frustration and accommodate for the limited control that a mobile AR app provides. The UI is minimal and only provides the most important information, so as to not clutter up the screen and leave more room for the player to see into the AR space. Our intent was for every action with the tools to have prominent audio cues to let the player know if what they were trying to do was successful, and visual cues should predominantly exist in the AR world space rather than solely through UI. Most tools are intended to be used multiple times in a single full game loop, and our

team made an effort to reduce times when a tool can only be used once through any given loop. In particular, we wanted each tool to have at least one role in both phases of investigating hauntings (Investigation and Dispersal), to give the tools depth and versatility.

We designed the ghosts' behavior to encourage engagement with the unique nature of AR. The gameplay always has one event happening at any given time, and the player should not have to look around themselves wildly to find what they need to see. Props are generated in the game as well to better blend AR and reality, as well as to give certain ghosts the ability to interact with the space it exists in as well as the player. During the Investigation phase, the ghosts try to ward off the player and raise their Notoriety. The player must find the ghost while the ghost is trying to get to the player (with the exception of the Demon). During the Dispersal phase, the ghosts try to confuse and distract the player so that they can escape and further raise their notoriety.

Tech

The technical aspects of this development revolve around the use of Niantic's Lightship API. This allows us to bring our ghost hunting experience to anyone with a phone without requiring LIDAR. Other technical aspects include the procedurality of the game and its leads, ghost statistics, and object placement, the organization of features through scripting, and development of documentation practices that can be carried forward into the future.

GPS Maps

The main method of portraying a lead is to give the location of the lead on a player's map in the description of a lead. This will be handled by another API using GPS locations. These maps will be stylized and fit the theme of the journal, giving the player the location of the disturbances.

Previous Implementations

As this was the second year of development during this project, time was spent investigating what had been accomplished during last year's development. What we learned was that multiple API's and methods were tested for AR game development, and the leading API was Niantic's Lightship API.

Looking through the game code, the team found that many of the calls used by the previous year were deprecated as Lightship had been overhauled a few months prior to the start of this year's development.

Additionally, when looking at the game content created by last year's implementation, the team decided that it did not fit the vision of the game as a whole. This meant all of the technical development had to be started from scratch. This decision was unfortunate, but necessary in establishing a more versatile codebase for the future.

Naming Conventions Table

Constants	CONSTANT_CASE
Global Variables	camelCase
Serialized Game Objects	_Underscored
Serialized Variables	PascalCase
Debugging Variables	flatcase
Audio Variables	Title_Case
Function Names	PascalCase

Version Control

This project used Github as the primary method of version control. The general Practices of each team member followed the same steps. If something was being simply added or adjusted in a small way that wouldn't significantly affect gameplay or game functionality, it could be committed to main provided they were given permission by the repository manager. If any significant changes were made, a new branch would be created. Branch names would be descriptive and may use the commit tags. Merging branches was handled by both the Repository manager, and the branch owner.

Commits would follow a general format of a tag followed by the most significant action and a description. Tags would have the following categories:

Git Etiquette:

- [Tag]/[add, remove, modify]-[Message]
    - **Feature** = Means some new item, ability, or mechanic added/modified
    - **Texture** = some material was added or modified
    - **Model** = some model was added or modified
    - **UI** = some UI was added or modified
    - **Story** = Some text or story specific Item was added
    - **Audio** = Some audio was added/modified
    - **MISC** = unsure or not categorized was added or modified

Goals

The technical goal of this project was to create a foundational game for future development including, basic class structures, a basic game loop, API integrations, and clear documentation of how to proceed with development and in what fashion.

This present three essential pillars to Technical design:

1. Build Basic Classes
2. Complete Game Loop
3. Develop Documentation

Narrative

The organization SPI, finds itself at the center of paranormal activity in the modern day. Facing a world where hate, love, sorrow, grief, and beyond can hold someone back from passing on, and that festering feeling grows into something far more dangerous. With the advancements made since the 1920s SPI Glass is the only organization that can help these lingering wisps find peace and move on, and by proxy protect the unknowing people of the world around them. Through using paranormal detection tools, Rituals, and a Spirit Medium's connection to the ectoplasm that we leave behind. Agents of SPI Glass root out, confront, and disperse these ghosts right under our noses, before the ghosts grow and evolve from the feelings that we experience every day.

World Building Overview

The story of the SPI organization truly begins with the man they revere, the figure that gave their organization a foundation without ever meeting him. The man who discovered ectoplasm, Thaddius Godwin, the creator of the SPI glass, a device used to peer into and perceive ectoplasm, the emotional remnants of humans who have passed on.

Thaddius was a man of wealth and renown within the innovation community of the UK in the 1920s. Thaddius's loss of his brother from the end of WW1 led to a great push and desire to comprehend if we can truly communicate with the dead. While many spiritualists focused on vocal and communal aspects, the visual traces of ghosts had been something that really couldn't be proven in a field of science. Something that had been explored but never invested in beyond the surface, but for Thaddius an optometrist of his time, eyes, sight, and the way we perceive the world was a natural world of thinking for him.

With his skills, and his desire to speak to his brother again Thaddius began his great work that became the Godwin glass, a multilayered lens device that could peer upon different forms of ectoplasm, and with this, the discovery of real evidence of ghosts was made. Thaddius, thrilled in his discovery and the chance to meet those again he lost in the war, began to spread his knowledge to the world, and this great act of discovery would be his downfall. As the world learned more and believed the day he went to contact his brother would be the day he disappeared with all that would be left of his legacy and his history would be the tool that became the quintessential piece of SPI Glass today.

All of his research would culminate in a tragic event, Thaddius's final seance, where he went to commune with his fallen family at the battlefield where they were found. The nature of paranormal events vary in strength, size, and scale. Many are insignificant occurrences, like flickering lights or the moving of objects. That is not what happened here at Thaddius's final

seance, what would occur on his arrival was the largest paranormal event recorded to date. These ghosts and the ectoplasm they are made of are fueled by belief and attention. They want to be seen, so they feast upon that attention and recognition like emotions they are. This happened here at the remnants of this battlefield. So many people believed in his research on ghosts that paranormal activity hit a critical threshold in that area. As Thaddius communed the earth shook, stray mines were set off, the weather swirled and changed. The Ghosts that had been gaining strength from acknowledgment began to discorporate Thaddius, leaving nothing behind but the SPI Glass. Following his disappearance people began to treat his research as a hoax, the site of his demise returned to a safer state. No one truly knows what happened that day.

After some years, when Thaddius Godwin's fame had died out, and his legacy destroyed, seven spiritualists would encounter his work. These seven hosted seances regularly between each other, where they'd discuss all manner of parapsychology, leading them to find the remnants of Thaddius's and with his work the proposal that ghosts were in fact real and they truly could commune with their lost loved ones was genuinely attainable physically. This led them to seek out the site of his demise unknowingly, in search of clues or remnants of the man's work. As the group finally arrived at the site, they stumbled upon the gold mine that was Thaddius's sole glass and with the anomaly of a device in their palms, they learned the truth of ghosts and the secret they had to keep. For if the world knew of Ghosts then they would only grow in power and if they were to grow beyond the tipping point of common knowledge and acceptance. Then the world would be thrown into paranormal chaos born of the emotionally restless dead. So they keep to secrecy, fighting a noble but silent battle, giving ghosts release through dispersal and keeping the world safe and unaware of one of its darker truths.

Those 7 who seeked the site of Thaddius's last moments have no idea of the story of the man behind it but revere its creator as the genius who proved them right and gave them their great noble path. A hero who gave them the tools to keep the world safe. Most importantly to anyone outside their circle the creator's absence was hidden, and this inventor is very well a figure that was "real" a mode to convey their collective decision-making and a way to inspire strength and awe in the agents below them.

Those who came after, the parapsychologists found only the glass itself and nothing of the man who came before. This singular glass is the cornerstone of their work and without it the founding members of SPI would be unable to do their noble, silent work. For with this they learned that the belief of ghosts is what strengthens them and, more importantly, that they can disperse them from the earth to grant them peace and protect the world. The secrecy is a necessary step in that the paranormal threshold doesn't go over manageable levels. Spiralling out of control, creating full-blown paranormal attacks across the world that can't be controlled. This is why SPI focuses on secrecy and dispersal to protect the world on a hidden battlefront and help ghosts move on.

Art

The mood for SPI Glass is playful and campy. For Ghosts our goal is to take visual inspiration from horror tropes to feel somewhat unsettling while still primarily being inviting and non-threatening. Taking inspiration from the visual language of games like Don't Starve, Little Nightmares, Hades, and Pokemon, SPI Glass will emulate a horror theme through a few visual guides. Human characters and tools concerning the SPI organization use a steampunk twist on 1920s culture and technology.

General Guidelines and Core Pillars

- ➢ Graphical Style:
  - 2D UI and Character Sprites.
  - 3D Ghosts and Tools.
  - Stylized / Cartoony Ghosts.
  - Cell Shading in 3D and 2D.
- ➢ Main Camera
  - Top Down flat Map View.
  - AR Camera Perspective, with the phone held vertically.
    - Held Tools Appear at the bottom right.
- ➢ Time Period
  - Set in the early 2000s, but concerns topics and history from the 1920s and the turn of the century.
- ➢ Atmosphere and Mood
  - Playful.
  - Campy.
  - Slightly Creepy and Unsettling.
  - Mysterious and Otherworldly.
  - Antique.
  - Academic and Studious.
- ➢ General Color Palettes
  - High Contrast / Use of Shadows. Ghosts lit mostly from underneath, and with few color steps to evoke the vibe as the underlit flashlight during a ghost story.
  - Heavy use of brown, orange, and gold. These colors support the vintage and steampunk style, highlighting metallic bronze.

Folder Structure

This is a general guide for the folder structure for assets particularly relevant to artists. In the assets folder in both the google drive and the unity project, art assets are found in these folders.

2D - For any 2D assets.

UI - Any UI related assets

Concept Art - Any Concept Artwork

3D - For any 3D assets

SKM - Standing for Skeletal Mesh, any assets with bone hierarchies

SM - Standing for Static Mesh, any assets without bone hierarchies. Tool objects and test ghosts meshes are found here

AM - Animation controllers and clips.

Materials - Materials for 3D objects.

Textures - 2D Texture map files.

VFX - Contains visual effects graphs.

Shaders - Contains shader graphs and subgraphs for the custom modular lighting system.

Naming Conventions

3D Assets	
prefix_filename_suffix_0(iteration)	
Prefixes	
SK_	Skeletal Mesh
SM_	Static Mesh
T_	Texture
M_	Material
PS_	Particle System
AN_	Animation Asset
Suffixes	
_D	Diffusion
_R	Roughness
_M	Metallic
_AO	Ambient Occlusion
_P	Packed

_H	Height / Displacement

2D Assets	
prefix_filename_0(iteration)	
Prefixes	
C_	Concept Art
SP_	2D Sprite Asset
UI_	UI Assets

Inspiration and References

Throughout the process of creating our tools, characters, and ghosts, our baseline structure was having solid material to refer back to throughout the ideating process. During PQP we put together moodboards and pinterest boards for different aspects of the design- ranging from overall style to specific details for the ghosts and characters. Throughout work on the project, these boards were upkept and are intended to stay up and available for further review.

Our goal was to take our inspirations from existing games and convert them to the ‘spooky’ and exaggerated approach we wanted to utilize to ensure the ghosts came off non-threatening. Additionally, we used these as a way to develop an understanding of the 2D art style.

Inspiration Pinterest Board

Inspiration Pinterest Board

Ghosts

Design Philosophy

With our ghost designs our intent is to evoke ‘spooky’ rather than scary. This means that while some design elements should aim to be unsettling, the overall impression of the Ghost shouldn’t rely on intimidating aspects.

Ghosts' visual language takes a lot of inspiration from the Pokemon franchise, specifically by distancing Ghosts from being too human, allowing them to take on more animalistic features. The intention is to create beings that, on initial observation, feel more like mythical creatures than the souls of dead humans. Utilize this visual language to strengthen the campy and playful side of the design.

One way to balance this with creepier, horror themes is to play with expectations for human anatomy, and drift into the uncanny valley. Removing features of the face like eyes or the

mouth, or transforming limbs into something unfamiliar can help bring an unease to the design. Exaggerating anatomical proportions can also support unease, choose features to stretch into unnaturally lanky sizes or shrink.

Ghosts combine some semi-realistic elements with cartoonish ones, with the dichotomy creating unease in the design, while enabling cuter and less intimidating aspects to shine through. Use mostly round shapes and simple silhouettes, with the intent to draw attention to the few more detailed and realistic features.

Inspired by the description of Mistwraiths in Mistborn, a primary design feature of all Ghosts types is to include bright bone structure in the interior of the Ghost's body that glows with a white light, adding visual interest to the mostly transparent and analogous colors of the Ghost's body.

Design Ghost types to have unique, recognizable silhouettes and gestures, so they can be easily distinguished from other Ghost types. During the design process, it's imperative that how the ghost moves or engages with the player is visualized as well. This helped significantly during the rigging and animation process in understanding how the rig needed to be handled and how the ghosts move.

Notes for Asset Development

2D Assets

- ➢ The 2D concept work for the assets should end up using flat colors, to better demonstrate the patterns for the ghosts.
- ➢ The 2D work for assets generally went from being just lineless thumbnails or sketches to having defined lineart which finalize the silhouette. This lineart pass of the ghosts will also provide a basis for the colored layers.
- ➢ The colors used are not the color of the ghost, but will be converted to greyscale during the texturing process. In the 2D concept art, these colors give an idea of what a ghost may look like with any element.

3D Assets

- ➢ Aim for under 30k quads for any Ghost meshes total polycount.
- ➢ Smooth the rounded shapes of the model and avoid noise on the model surface..
- ➢ To integrate well with transparency, the main body of the final mesh must be merged and have their vertices welded. Avoid using a compound of separate objects or subtools, as after textures are applied the interior geometry will be visible creating breaks in the fresnel highlights and becoming distracting.

Texturing

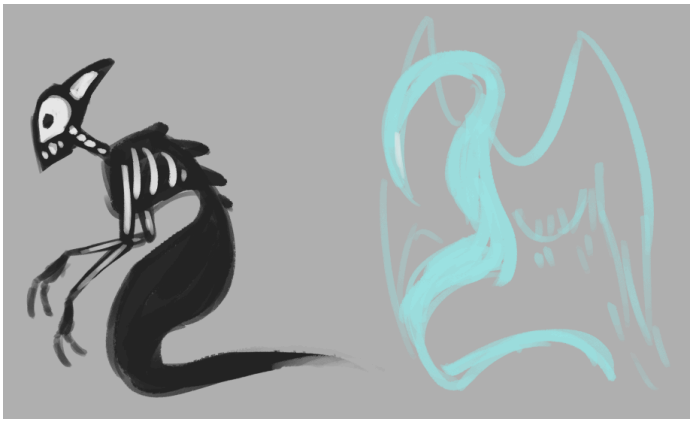
- ➢ Cell shading is done by Unity shaders, so using a basic specular pipeline will work well for creating the 2D texture maps.
- ➢ Coloring is also taken care of by the shader, It's built modularly allowing for each Ghost type to have multiple materials for each of the four elements. So the base color map for

Ghost types should be a greyscale map, where whiter parts will be less saturated and shine brighter, while darker parts become more saturated and darker.

- ➢ Do not use much if any metallic value on the ghosts textures, as the reflections complicate the ghosts visually and make them feel less wispy.
- ➢ Use a small to moderate amount of roughness for a subtle shine.

Ghosts Overall

Throughout early ideation processes, several sketches were put together to get an overall idea of the ghosts and what they may look like. This process was key in establishing a jumping off point for the ghosts as a whole. They would be created during meetings by Jaliah Hippolyte and Mason Moore to help define where the project's visual design would end up. Core elements of the design, such as the visible bones and overall 'dark' colorations and flowy bodies.



Spirits

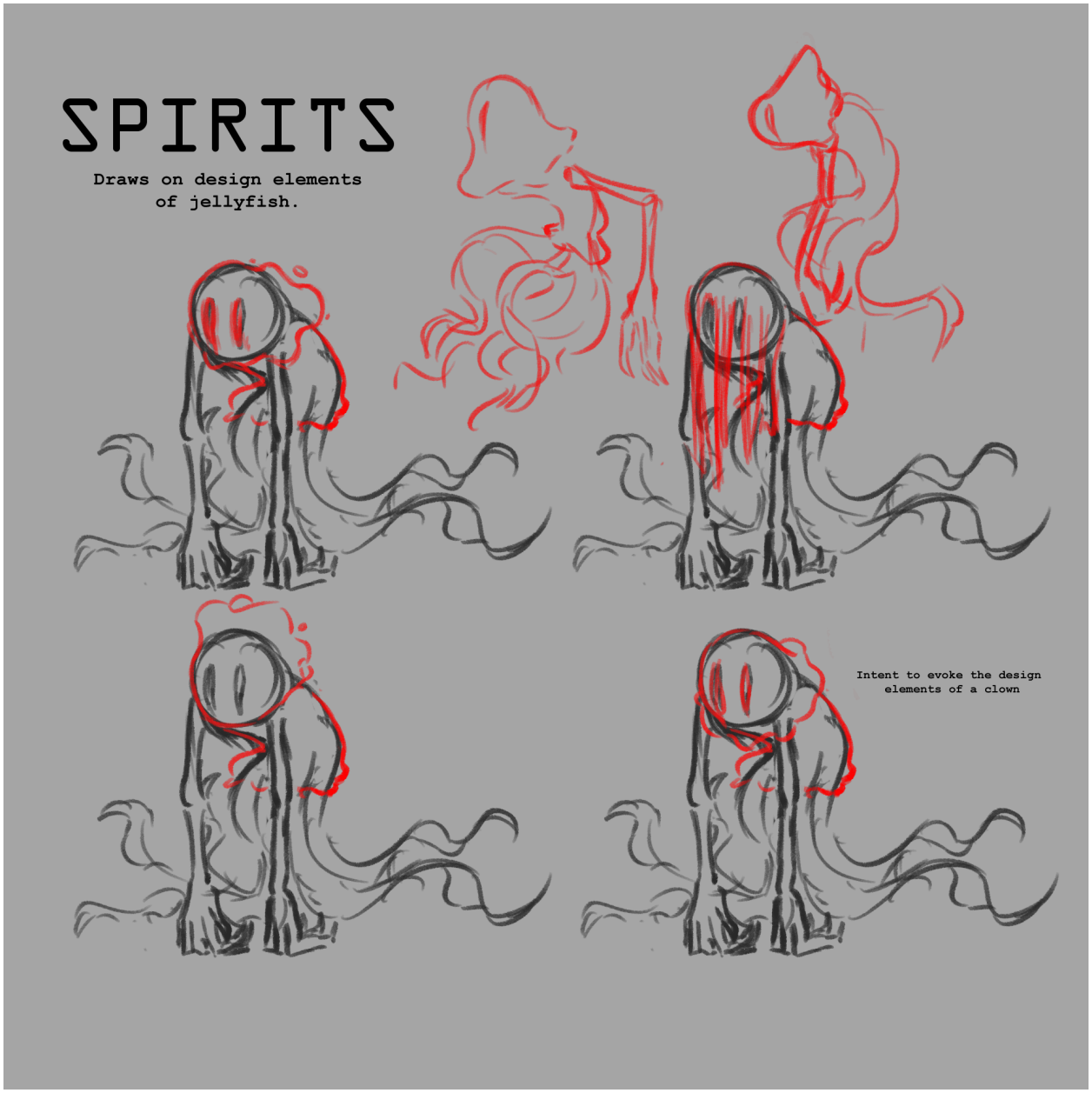
Spirits embody feebleness and fear, as such they're designed to seem like less of a threat, and more of a pitiable nuisance. The Spirits silhouette was inspired by Jellyfish, which is the source of the skirt-like shape around its torso resembling the margins of a Jellyfishes hood, as well as its long draping tentacles.

Spirits were given large eyes to seem innocent and vulnerable. We designed several ideas for draping hoods to cover its head as it's hunched over, to compliment the tentacles and further push the sorrowful energy. While the tentacles can be somewhat unsettling, the absence of a mouth, its thin elongated arms and fingers were all incorporated for additional uneasiness.

The design for the Spirit went through four major iterations.

First Concept Sketch by Mason Moore

Second Concept Sketch by Jaliah Hippolyte



Final Concept Art by Jaliah Hippolyte

Wraiths

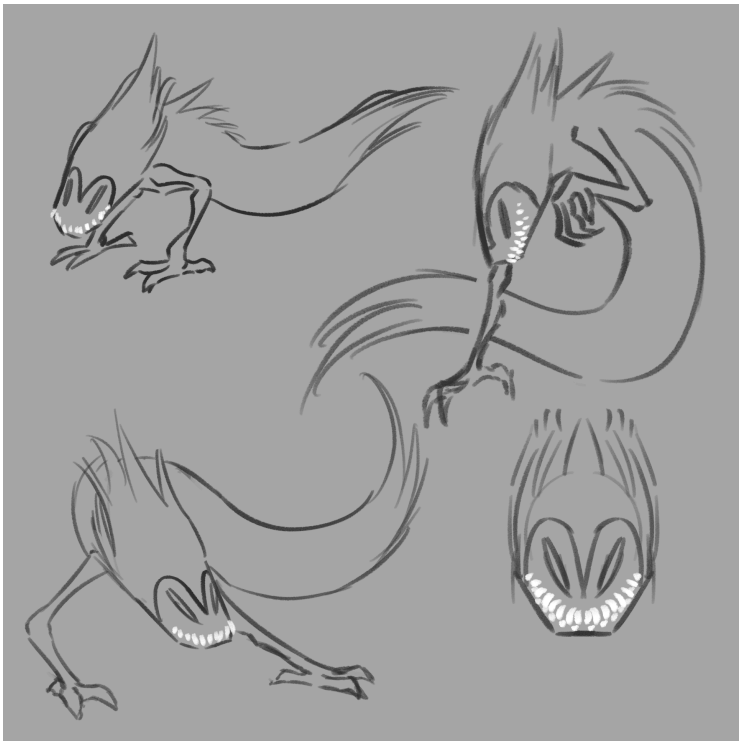
The Wraith is designed to highlight its agility and slipperiness. Originally inspired by salamanders, its long slicked back head shape and long body are perfect for quick snappy movements and evasive maneuvers into the safety of possessed objects. The Wraith's unnaturally broad smile and always visible teeth are intended to display its playful and mischievous nature.

The design for the Wraith went through two overall iterations, as well as a few varying iterations for its texture patterns.

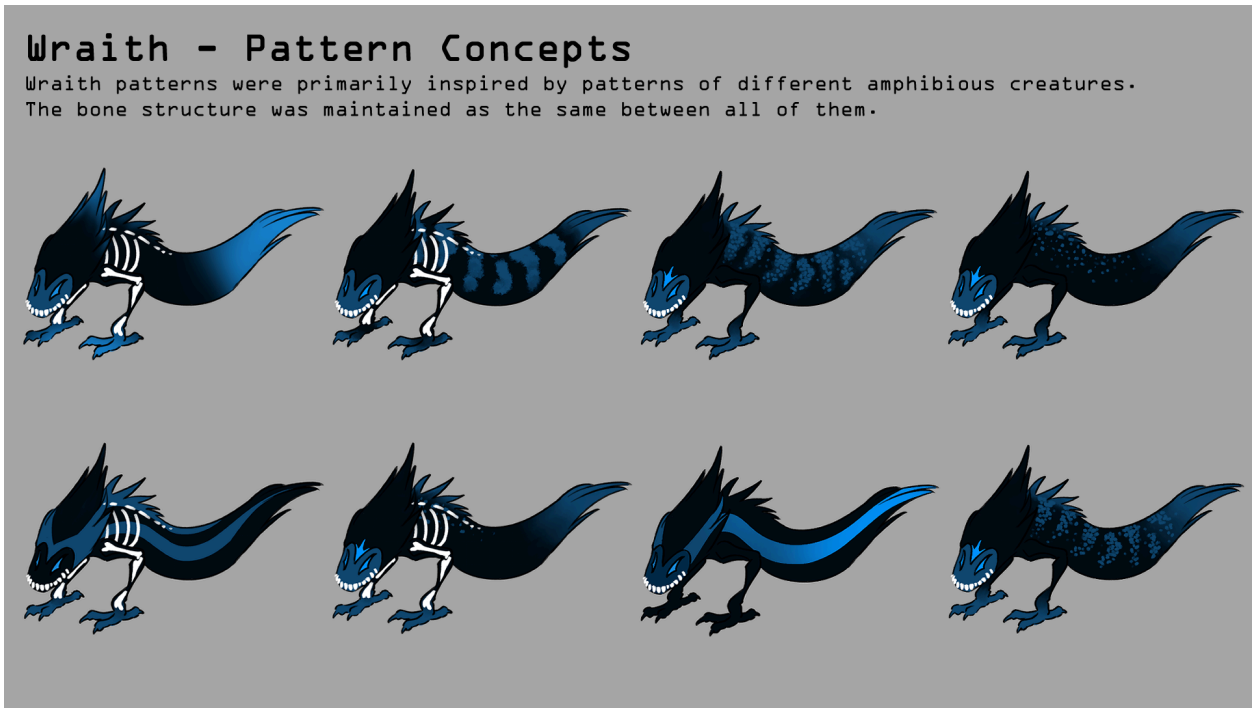
First Concept by Mason Moore

Initial Concept Sketches by Jaliah Hippolyte

Final Concept Sketches by Jaliah Hippolyte



Color Passes by Jaliah Hippolyte



Final Design by Jay Hippolyte

We also considered a winged wyvern like silhouette for the Wraith.This design was more bat-like, intended to invoke the visual similarities between some spirits and flowing curtains. While we really enjoyed the idea, it was decided that the more amphibious design fit the gameplay conception of the Wraith better.

Demons

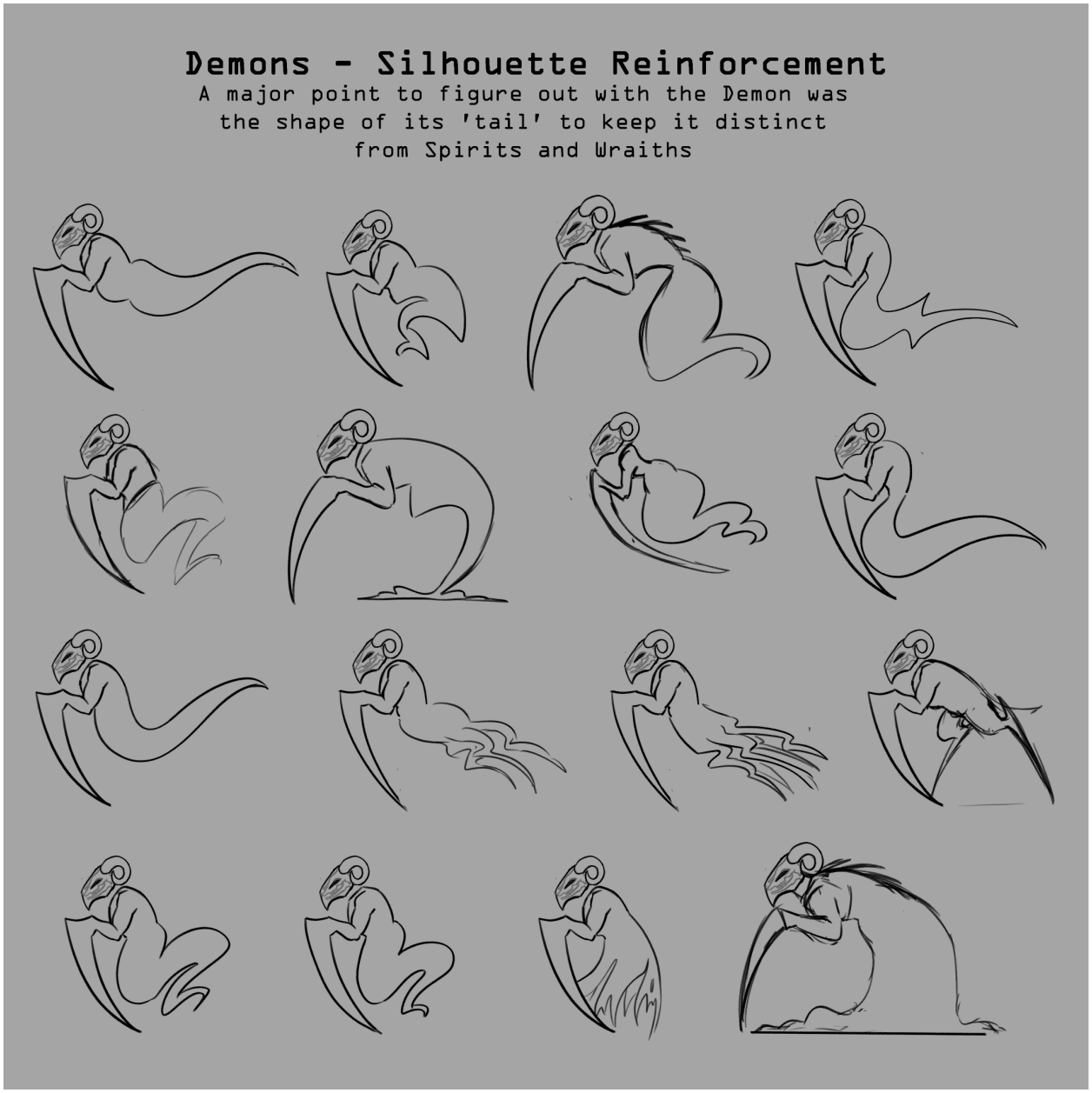
The Demon was designed to give the impression of distinct strength, intended to appear as a heavy hitter to most accurately portray its difficult to handle and aggressive behavior. Its inspiration derives from more ‘anticipated’ demon visuals: horn and goat skulls. Additionally, the scythe-like hands are used to convey death. Its body is bulkier to help give it a defined silhouette compared to its counterparts, with an appearance intended to feel more ‘put together’and powerful than the other ghosts.

The Demon went through a few sketch iterations before falling on its final design.

First Concept by Mason Moore

First Pass Sketches by Jaliah Hippolyte

Second Pass Tail Sketches by Jaliah Hippolyte



Final Design by Jaliah Hippolyte

Procedurality

To allow each Ghost to stand out, they should have a number of wearable props or anatomical adjustments that can be randomly assigned to them. These can be hats, scarfs, face wear, or other clothing items. The intention is that when they are arranged on a Ghost procedurally, each Ghost should have some unique visual additions, giving them character and memorability.

Tools

Design Philosophy

SPI's technological development split from the outside worlds at its founding. While their tech is advanced, they're mostly ongoing improvements on technology rooted in the original discoveries of Thaddius Godwin back in the 1920s. The primary goal of tool visual design is to showcase this alternate development with a vintage steampunk style.

Major inspiration was taken from weapons in the Splatoon franchise, for the way they adapt common objects like hoses, and paintbrushes into weapons. We used this philosophy, and took from iconic technology like radio, phonographs, and typewriters, adapting them into weaponry in a similar manner. Tool design makes major use of gears and clocks, which are placed generously around a lot of the tools. We also took icons from popular tropes in Ghost hunting and spirituality, like candles and drawn sigils.

SUCK

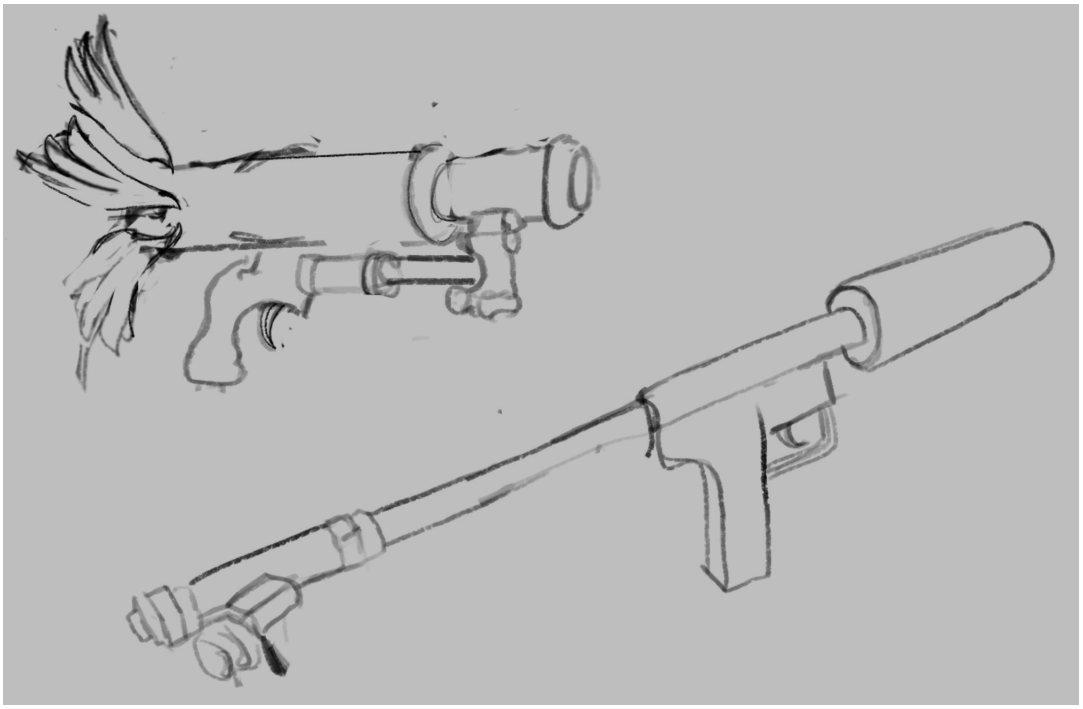
The SUCK was initially inspired by the Poltergust weapons found in the Luigi's Mansion series. However we wanted something that felt more like a weapon and would distance itself a bit from the function of a vacuum; where vacuums collect and capture Ghosts, the SUCK actively disperses Ghosts.

The SUCK design is composed of a couple references. The front section has a Phonograph shaped muzzle and elongated barrel, resembling a vacuum, except that the top of the barrel has a handle at the top. This front handle is held alongside a back handle near the holder's hip, resembling a minigun. The barrel leads into a chamber with fans, inspired by superchargers found in car engines. Ghost Ectoplasm is spun around and torn up in this chamber before being spat out from an opening at the back, taking after wood chippers or snow blowers.

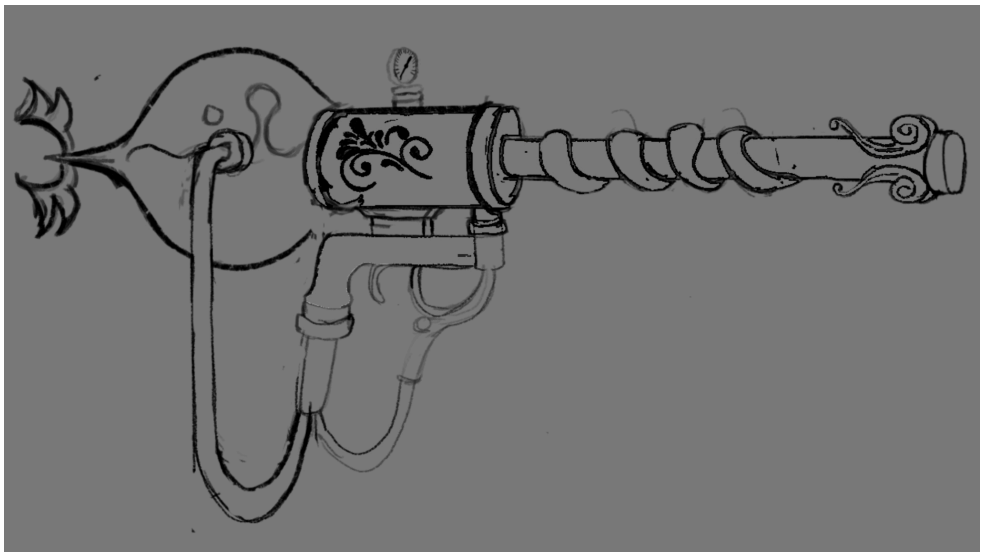
PRIEST

The PRIEST brought some unique design challenges. Firstly, we struggled to find a good silhouette. We originally considered a slimmer flamethrower or power washer shape, but eventually settled for a one handed water gun look. The PRIEST also wants to call on religious imagery without making overt allusions to specific religions and beliefs. To solve for this we used architectural imagery and ornamentation, which has relation to religious concepts but not typically categorized as such. The Muzzle of the SPI Glass is framed by two Ionic Columns, and the spherical container holding water tapers to a point resembling a domed chapel or mosque. The sight is ornamented by winged shapes, which are often used in religious imagery for revered creatures or angels. Some bronze piping, hoses, and clockwork was implemented for a steampunk flair.

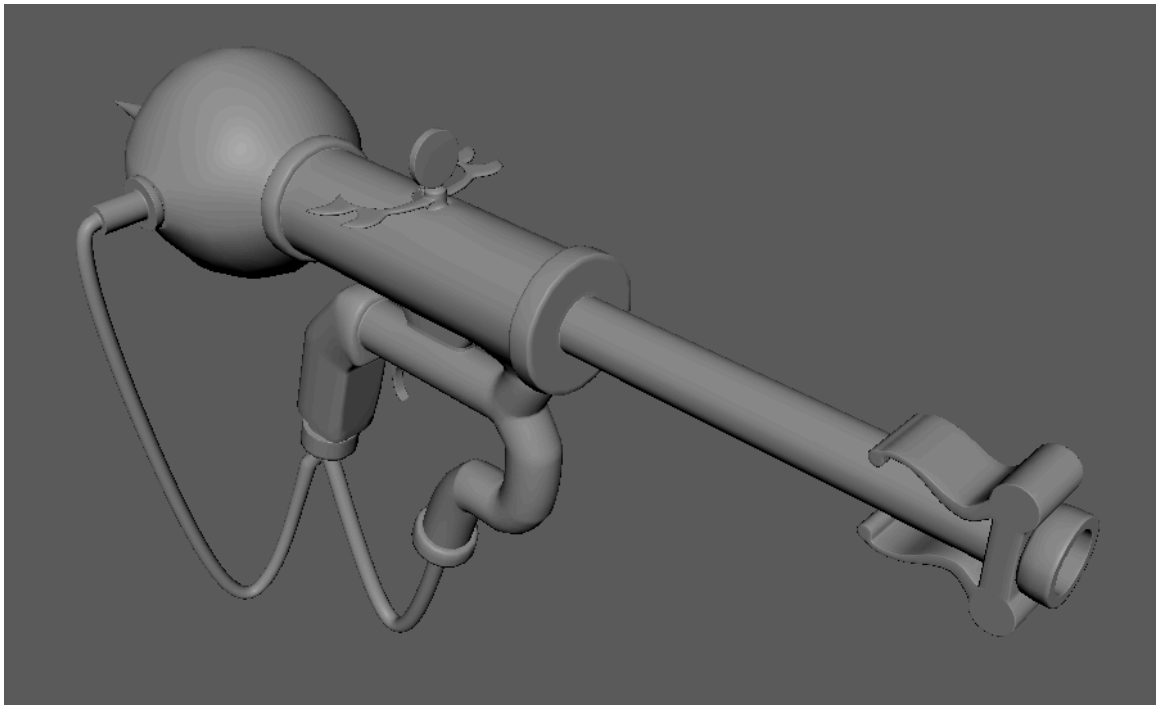
First Concept Sketch by Mason Moore

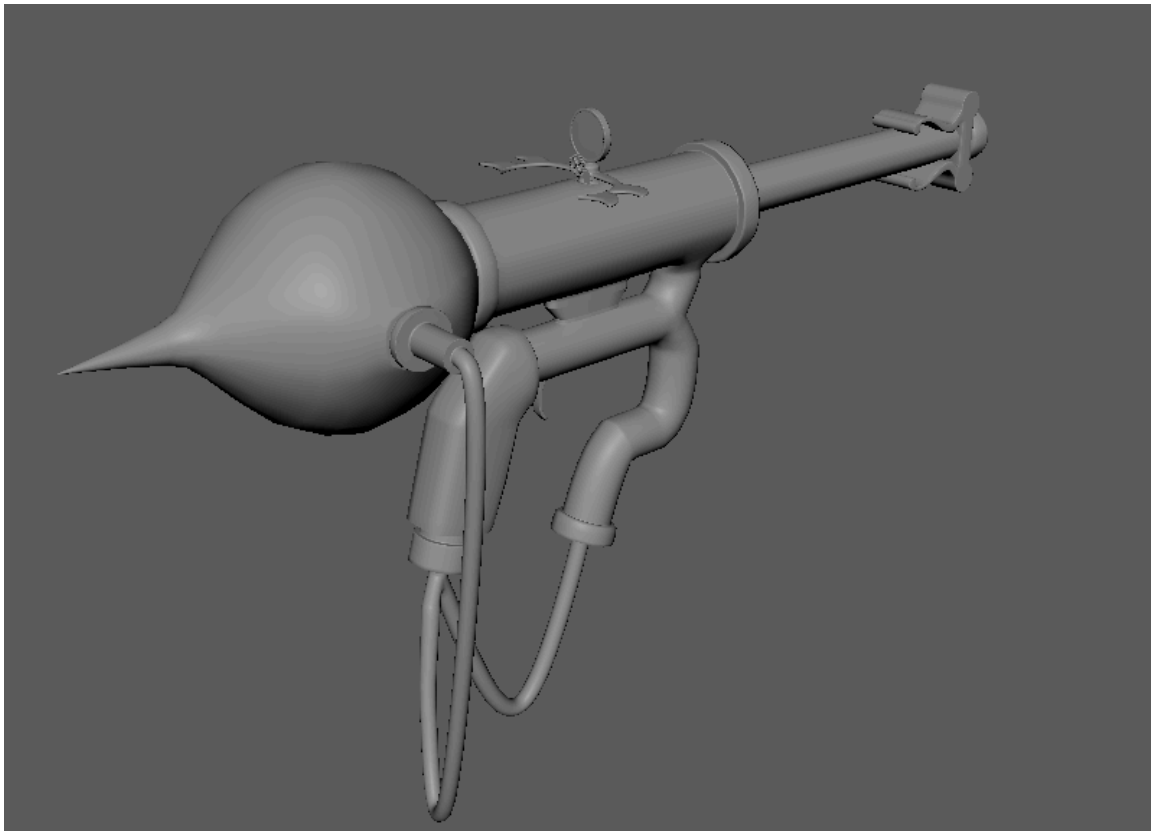


Second Pass Sketch By Mason Moore



3D Model by Mason Moore

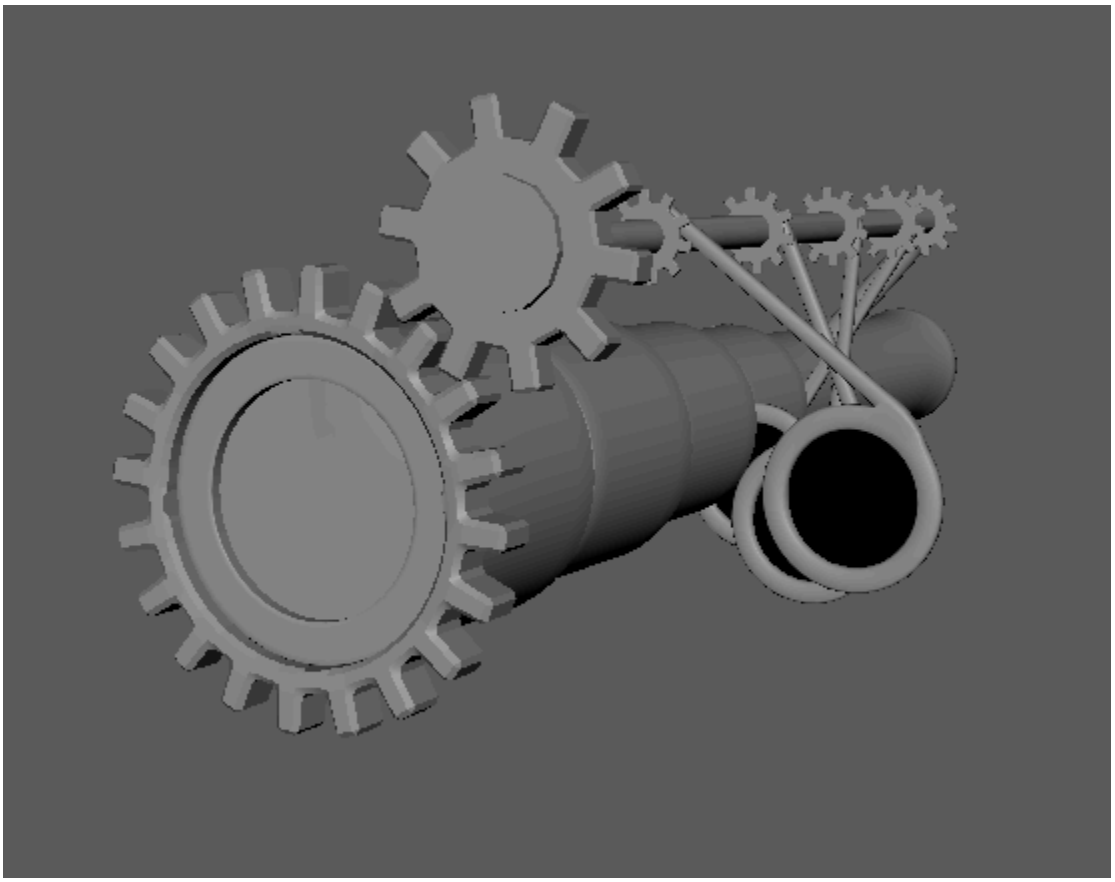




SPI Glass

The idea for the SPI Glass's design came from the multi lensed instruments used by optometrists to judge and measure eyesight. This concept was combined with a traditional spy glass or telescope, allowing different lenses to view different elements within ectoplasm. The SPI Glass design segments a traditional spy glass and attaches a lens swapping mechanism to it with several gears.

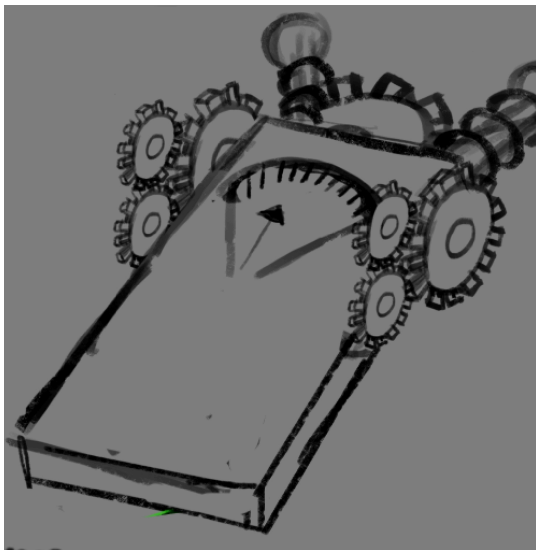
3D Model by Mason Moore



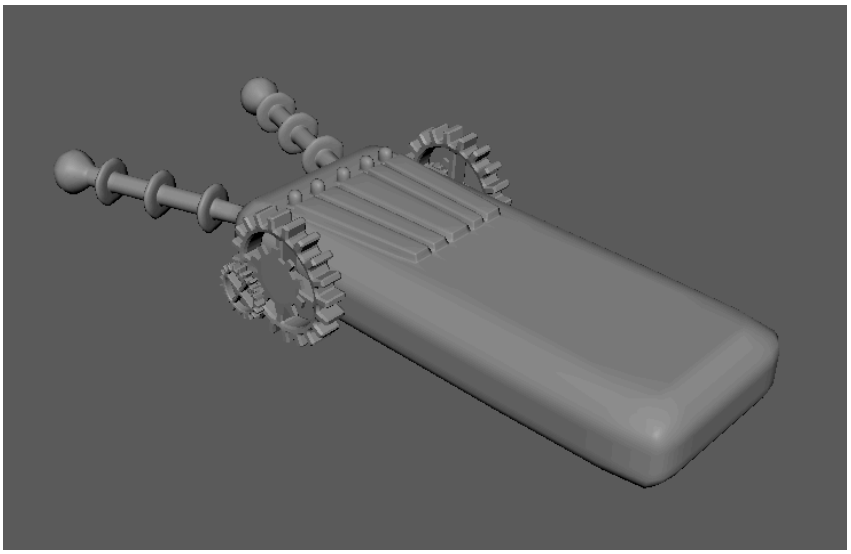
EMF Reader

The EMF Reader was designed to give a vintage and steampunk flair to a relatively simple design. EMF Readers are typically depicted as simple rectangular handheld devices, with colored sensors or indicators. Our final design takes heavily from the EMF depiction found in Phasmaphobia, with colored bars and lights used as the primary indicator for distance

First Sketch by Mason Moore



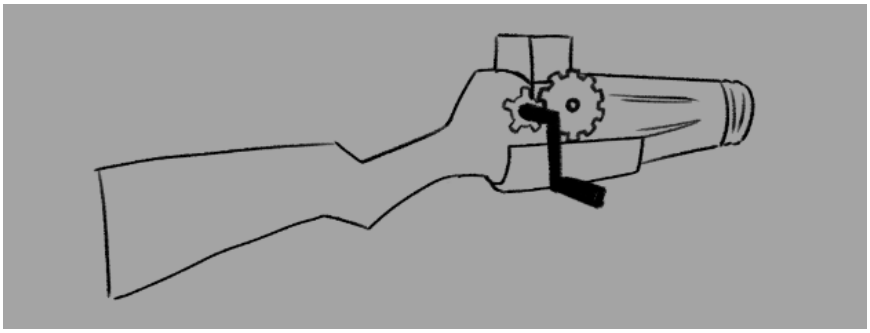
3D Model by Mason Moore



NaCL

The NaCL adapts the silhouette of a shotgun, adding a couple lighthearted quirks to give it character. The muzzle is replaced by the dotted holes of a salt shaker, justifying the intended projectile spread. The receiver is replaced by a grinder, where instead of loading cartridges, solid rock salt is placed to be grinded up and loaded into the chamber. Photobashing was utilized to make a baseline design concept, before drawing it out.

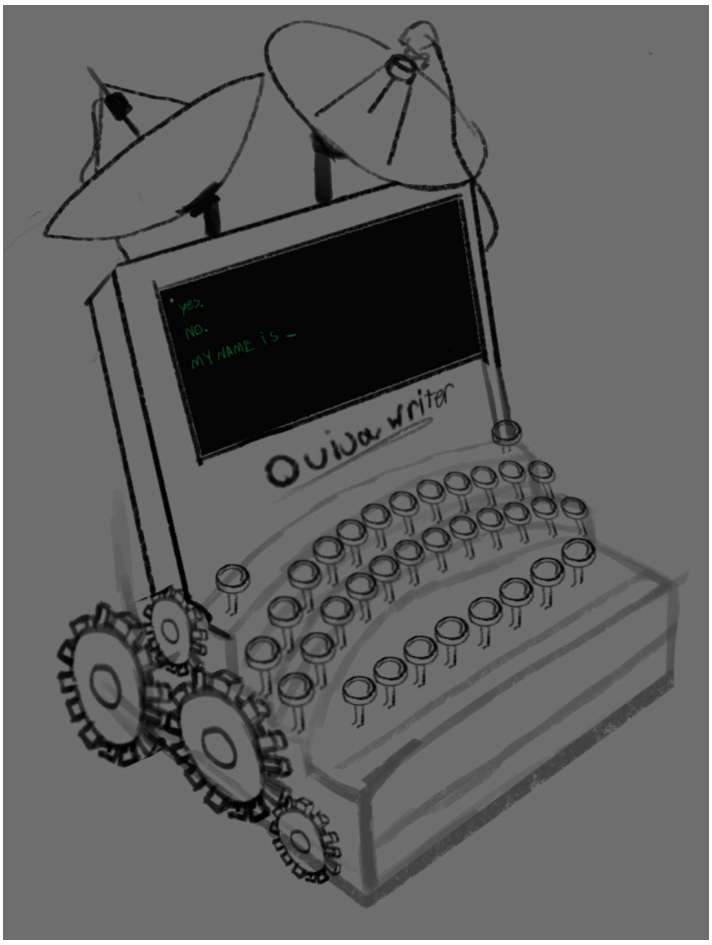
First Sketch by Jaliah Hippolyte



Ouija Writer

The Ouija Writer is essentially a Ghost translator, and combines the imagery of a Ouija board and a Typewriter. Despite using a modern LED screen to display questions and answers, the keyboard uses typewriter keys, arranged in the curved shape characters have on Ouija boards. The mounted satellite sensors record spiritual energy from Ghosts, allowing the device to autonomically type out the Ghosts answer.

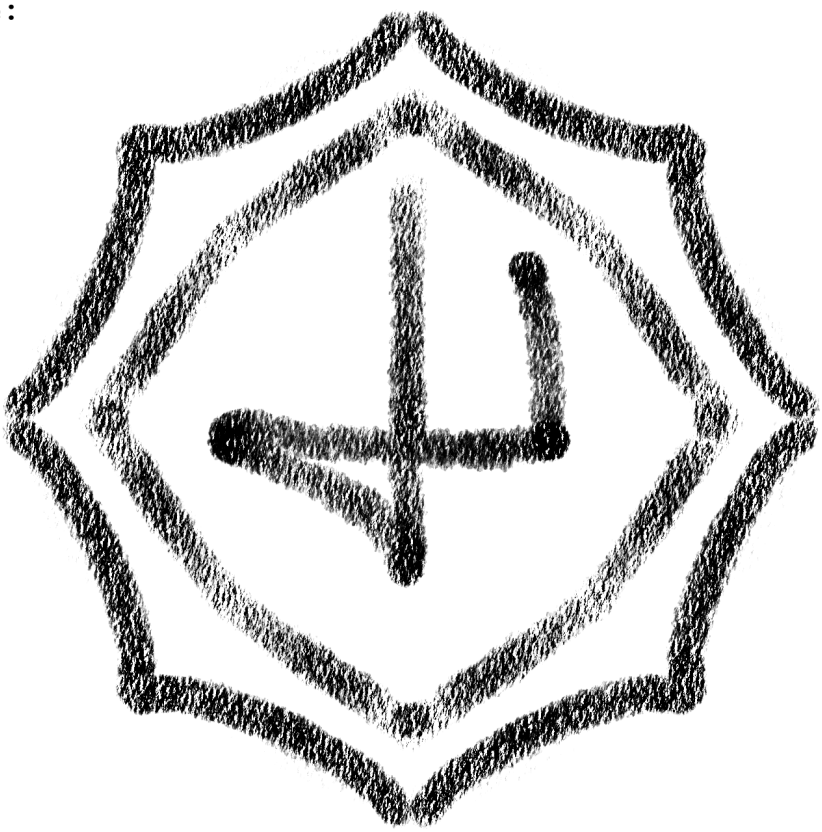
First Sketch by Mason Moore



SIGILS

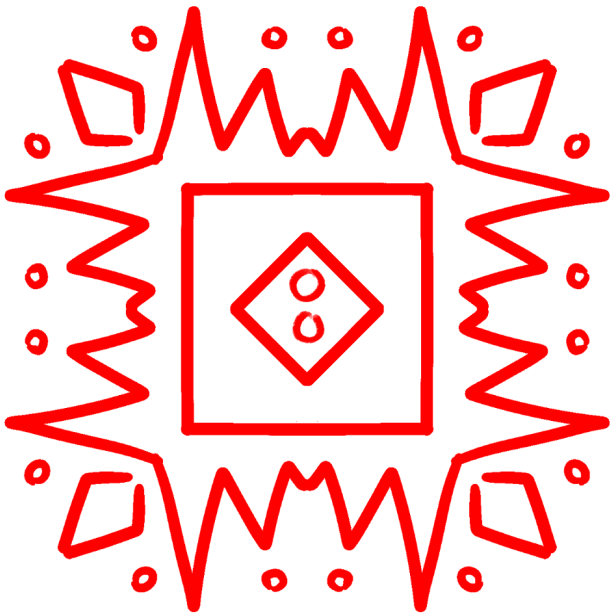
Sigil guides/visuals should look hand drawn overall, using some amount of texturing like pencil/chalk/crayons for the Assets.

Example:



This is an example of an 'incorrect' sigil.

- The outer rings are very complicated, with a lot of dots.
- The outer rings have straight lines
- There are more than 2 outer rings
- The inner symbol does not abide by the 8 point circle rule
- The inner symbol cannot be drawn with 1 continuous line
- The lines are "clean" rather than made with a texture similar to pencil/chalk/crayons



When designing sigils, you should also avoid making symbols from pre-existing, real world religious/occult connotations.

These symbols should not translate to a 'real world' symbol.

Humans

Design Philosophy

When designing humans for SPI, what they wear is the most important part of conveying their connection to the setting. Humans should be designed with clear intention for what they do for SPI in mind, which can be achieved through multiple different routes. Someone who works under SPI glass should look ‘dated’or ‘out of place’ in terms of their outfit, as it should fall under the umbrella of steampunk when looking at reference material. A major portion of these designs come down to jackets, straps, and technology. Tools designed for SPI work well as additional props to character designs that can add more connection between the character and the player.

Additionally, members of SPI Glass sport masks in their design, which should draw from the same concepts as their outfits. Adding technology, particularly gears or clocks, when it comes to parts that reasonably could move adds more points of interest to the visual design of the character in question.

While we only intend to use character portraits, designing the character from head to toe was also necessary for understanding what elements connected to where, even if they didn’t visually show up in the final assets.

Notes for Asset Development

2D Assets

- ➢ Assets should be lined with either flat or cell shaded, maintaining a semi-stylized approach.
- ➢ The baseline canvas should be 1280 x 1280, with 40 pixels at each edge being buffer space for the image. The final asset should be 1240 x 1240.
- ➢ Character Portraits include only the bust.
- ➢ Character should be semi-stylized, having mostly realistic proportions but exaggerated shape language when necessary

The Handler

When designing the handler, Jaliah made several different options to choose from in regards to body shape, facial shape, hair, clothing and masks. Her design was worked on in tandem with narrative decisions regarding her. As we came to conclusions on her behavior, some selections would be made. Her design was meant to invoke the idea of someone who was independent and capable, but very keen on working on her own. This led to decisions that would be further cemented as aspects of SPI, such as the masks, as a way to keep her visually interesting while also being emotionally ‘closed off’.

Handler Concepting Part 1 by Jaliah Hippolyte

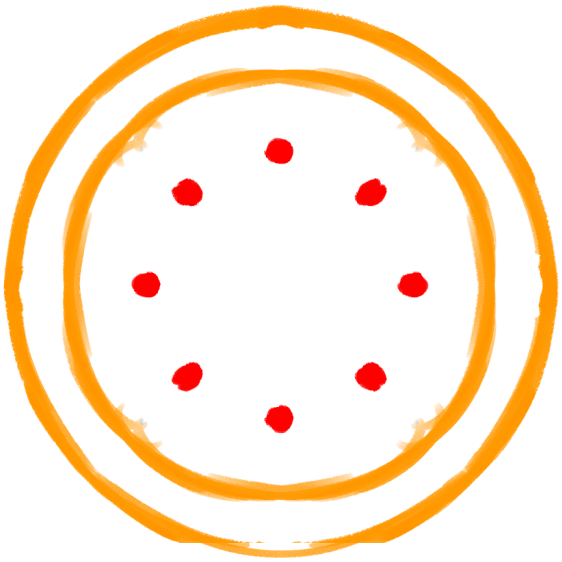
Handler Concepting Part 2 by Jaliah Hippolyte

Handler Concepting Part 3 by Jaliah Hippolyte

- **Sigils should consist of 1-2 outer 'rings' and one inner symbol.**
- **Outer 'rings' should not have any straight lines, but can have points.**
- **The inner symbol should be able to be drawn with straight and/or curved lines between 8 points in a circle.**
- **The inner symbol should only be lines between 3-5 points.**
- **A big design priority is to try and make a sigil you can draw with your finger with 2-3 continuous strokes**

- Outer 'Rings'

- Inner Symbol



UI

Design Philosophy

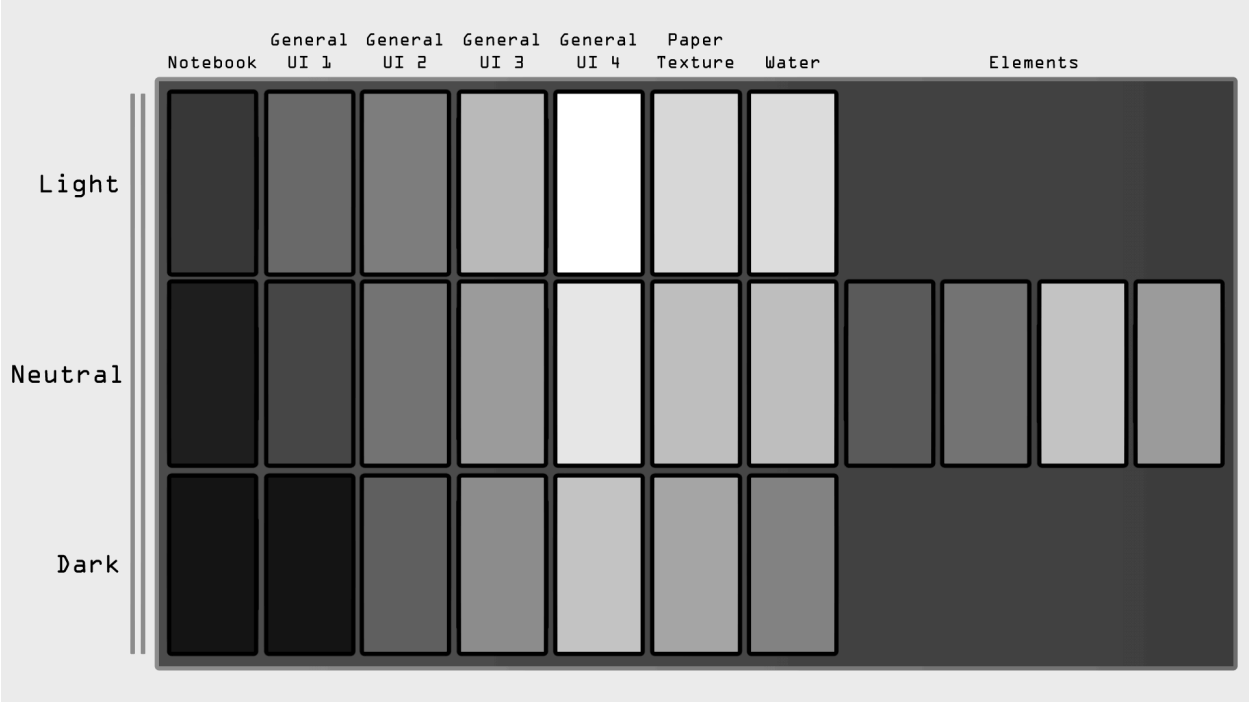
The primary goal of the UI elements were to be both simple and invoke the feeling of the ‘steampunk’ relation through the coloration. UI elements should avoid being modern in technological references because of this. For example, the notebook is utilized as the players menu and main way to access the game overall, rather than a phone. Similarly, the design of the tool wheel takes light inspiration from rotary dials.

While they do not follow a specific era in their stylization, they do gear towards a very ‘clean’ variation of steampunk stylization. Straight lines and rounded corners, but still an amount of sharpness in the shapes. Gears and axles should be placed in spaces that look like they would be moving.

Notes for Asset Development

Palette

- The palette is built on primarily very warm tones, focusing on bronze-goldish hues. The palette was built to have clear changes in value between each color, to better provide accessibility under as many variants of color blindness as viable.
- Primarily General UI 1-3 were utilized, 1 generally would be used for backgrounds or as ‘darkness’ when 2 was already in use. 2 would generally be used for darker values, and 3 for lighter values.



Icon Elements

- ➢ All UI Elements should, overall, gear towards being ‘lineless’ in style. Lineart tends to become muddied at smaller scales and easily lost.
- ➢ When creating the assets, scale should be kept in mind. With the various sizes, multiple scales of a UI element may be necessary. As such, the final assets should be stored as vector art in whatever art software is being used. Clip Studio Paint’s vector system was used to produce all of the current UI assets, but any program supporting vector artwork will work. Generally, they were made on a canvas at a larger size, and then set up in such a way that they could be scaled down as needed.
- ➢ The canvas, at minimum, should be 1024 x 1024, at maximum it should be 4096 x 4096
- ➢ Things should be kept simple, using one variation (Light/Neutral/Dark) of General UI 2 and 3.

Animation

Design Philosophy

Animating the ghosts are primarily only limited by the rigs themselves. Ghosts generally need to feel 'floaty' in their animations, and their animations as a whole should be exaggerated. If a movement is quick, it should be bigger to make sure the player can see it, as each animation is a key element of player information. If a movement is small, these still can be exaggerated to make it clear to the player what's happening. No movement should be small and fast, however, as both the player camera and the ghost will be frequently moving, making them easy to miss.

Notes for Asset Development

- ➢ All ghost animations were managed at 30 FPS
- ➢ All animation needs to be managed in 3ds Max as we use the CATrig system
- ➢ It's recommended that each animation either be saved as its own Max file or as a clip using the CAT system to easily re-access it when necessary.
- ➢ The Morph Target system needs to be used for any animation involving squash and stretch.

Spirit

The Spirit is meant to emulate jellyfish, with very flowy movements being ascribed to its tendrils. To achieve this, we created start and final poses for each tendril, and would then move each layer of tendrils frames to ensure the child bones lagged a few frames behind the parents by moving all of the existing child bones frames. From here, recreating the very 'flowy' look of the tendrils was relatively easy.

- ➢ Tendrils are animated in a separate layer from the rest of the model, the local 'Tendrils' layer, rather than the absolute 'Overall' layer. This makes it easier to move Tendrils animations between different animated clips for the sake of animation blending.
- ➢ Poses that fade back into the idle usually begin and/or end with the starting pose of the idle.
- ➢ The Spirit can be exported with 'animation only' selected as an option.

Wraith

The Wraith's animations are meant to fit a ghost who is quick and mischievous, and as such they are meant to keep a very flowy, almost snake-like motion to how it moves. When its body shifts back and forth, a major detail we have in mind is to keep the head facing forwards, with counter movement on another layer if necessary.

- ➢ When exporting the Wraith, 'animation only' must be unchecked to export with morph targets.

Provided is a link to an animation reel, showcasing how animation was implemented in-engine.

https://youtu.be/YeEUYLaApwY

Environment

Augmented Reality experiences create a special hurdle when it comes to designing environments. Ultimately any environment one designs has to coexist with incoming camera footage that will inevitably dominate the scene. For SPI Glass, the goal is to adapt to this world and not obstruct it.

Environmental Art consists of 3D furniture and props to be scattered about the scene, ambient particle effects, and post processing to set the tone for the AR environment. For Environmental props we found inspiration from turn of the century architecture, characterized by generous ornamentation.

The scene should be underlit to best emphasize the shadows on Ghosts and Objects, creating a ghost story by the campfire vibe. Since Tools are viewed from above though, attach lights to the parent object for tools so they can follow the tool position and keep them illuminated.

SPI Glass will be played in bright areas so that the camera can recognize the floor and other objects in the world. This poses a bit of a problem however, as the heavily lit scene will disrupt the spooky tone we're aiming for. We can adjust for this with some post processing magic, decreasing the overall brightness, adding highlight and shadow adjustments, and other effects like film grain and chromatic aberration to sell the player's transportation into the game.

Audio

The original primary audio goal of SPI-Glass is to create a sound identity that aligns with and enhances the spooky but slightly playful vibe of the game. And whilst the unexpected tech challenges proved limiting for what could be fully implemented, we do believe that we achieved this goal, particularly when you look at the total number of assets that have been created and audio design groundwork that has been laid out for future iterations.

Important Note: As audio cannot be displayed visually by image, at the end of each subsection we have included links to the corresponding demo reels.

Music

Given the 1920s inspiration of the project, we decided to use Jazz as the primary genre. There are a total of five original compositions: "Subtle Now," the main menu/title screen theme, "Us, The World," the world map and menu theme, "Harper," a theme for the handler character, "Detect, Dispel, Disperse," the encounter theme, and "Dé," the mission complete jingle. Additionally, the "Detect, Dispel, Disperse" and "Dé" both have multiple arrangements, four for the former (the investigation phase and one each for the three ghost types during the dispersal phase) and six for the latter (a victory and defeat theme each for the three ghosts, titled "DéSclose" and "DéFeat" respectively). The entire soundtrack, including alternate arrangements, can be found here.

Sound Effects

There are four primary groups of SFX: Ghosts, Tools, UI, and Ambiance. Ghosts have four types of short barks: Attacking, being damaged, being dispersed, and a general bark specific to each type, groaning for Spirits, laughing for Wraiths, and growling for Demons. Each ghost type also has a slightly different personality framework, with Spirits more raspy and feeble, Wraiths more giggly, mischievous, and childish, and Demons more growly and animalistic. Tools have been brought to life using a wide variety of Foley, some with fairly straightforward sources such as a sink tap for the holy water gun and a physical keyboard for the typewriter placeable, and others through more unconventional places such a microwave for the EMF RADAR and some uncooked pasta for the salt gun. Likewise, all UI SFX for the menu uses the sound of paper to match the journal aesthetic. For Ambiance, in a sort of middle ground between VA and Foley, Jimmy used their own throat to create a series of creaking sounds, as well as the static and call hang up sounds for the otherwise traditionally voice acted snippets of radio dialogue. All sound effects have been created from scratch using Foley and editing, and are showcased alongside the voice actor and/or Foley source in this demo reel.

Voice Acting and Directing

Since each type of ghost is intended to represent a variety rather than an individual, after Jimmy recorded two samples for each type, take advantage of the 4099 contractors (and Mason after one of them fell through). This meant not only getting more thorough experience with voice acting, but voice directing, a completely new experience for all parties (in the sense of both Jimmy actually doing the directing and the three volunteers being directed). The biggest learning curve in this process quickly became the need to give in the moment feedback; to truly provide direction in what they needed to do instead of just accepting their first takes. And in the end, it provided some valuable differentiation to the collection. A demo reel of all self-directed VA done by Jimmy can be found here in addition to one for all VA directed by Jimmy found here.

Sound Design

This is the area in which we faced the most struggle. On one hand, the vast majority of created assets did get fully implemented internally within Wwise. However, with the challenges faced by tech limiting the progression of the project, much of that implementation did not properly end up in the game, in some cases only in an incomplete state, and in others not at all. Nonetheless, though the full extent of the time and effort put into both asset creation and internal audio design could perhaps not be fully reflected in the final product, what did make it in does meaningfully enhance the experience, particularly the encounter music which is set up such that it dynamically adapts to both the current phase and the specific ghost in the encounter. A demo reel of some of the audio design in the final product can be found here.

Roadmap

Sprint	Assets	Tech
1	Development and Finalization of 2D Concept Art for Handler, Ghosts, Tools. Initial UI Work for Tools. 3D Model, Rig, Textures, and Basic animations for **Spirit**. Spirit should be implemented in-engine. Tools (SUCK 609, SPI Glass) have completed geometry. One completed environmental object. Narrative(Spirit Info, Overall world bible surrounding core concepts), Tutorial Outlined Tool and Ghost SFX Encounter Music	Immediate Level functionality UI working w/ buttons Levels swap between w/o crashing AR Functionality Ghosts Basic Features Implemented ● Slight Stat Variation (Proc. Gen) ● Pathing Base ● Tools Basic Tool features implemented Git Repo Created
2	2D Sprites for the Handler and completed UI Icons for tools. **Wraith** 3D Model, Rig, Textures, Animations.Tools (EMF, NaCL, P.R.I.E.S.T, Ouija W.R.I.T.E.R., (S.B. 002)) have completed geometry. Narrative, Tutorial, Dialogue Current SFX Implementation	UI has Menus Leads Exist Map added Tools ● Placeable tools are ready Ghosts ● Spirits Implemented ● Appears in AR level
3	**Demon** 3D Model, Rig, Textures, Animations. Finalized 2D Elements (Sigils, Journal, Menus) Narrative, Tutorial, Dialogue UI+ SFX Design/Implementation Finalise Music	UI has text and tracks progress Leads work Player Danger level Tools All tools complete Ghosts ● Wraith Implemented ● Objects can be inhabited ● win/loss condition
4	Tool animations, particle systems, shaders, overall refinement period. Notification Dialogue, Tutorial Sound Cleanup/Refinement	Stats tracked Leads are Randomly generated Map tracks player location and contract location Demon Implemented

	Overall Finalization	• Objects Populate space and maybe throwable • Proc. Gen Models Tools all implemented

3 - Implementation

Tech

Features:

Only a certain amount of the outstanding objectives were able to be met during this development period. Of the ones achieved, a majority fell under goals one and three.

For goal one, the basic classes for tools, ghosts, and elements were made and some additional classes that inherit from those classes. Manager Classes were also developed to organize what objects were active and when. These manager classes are used to either connect functionality across different scripts or connect UI to features.

Goal three was met and exceeded with all of the code that was developed during this MQP being commented and documented in this Tech Bible. Additionally, Code from the Lightship API and another external script were commented and documented as well. This will hopefully help future developers of this project understand what is being used and how it should be used for other features.

Establish Basic Classes:

Our primary goal of this project was to make base classes that could be easily expanded on and adapted to fit future needs without much or any changes being made to the foundational features of the game.

The first class developed was the base Ghost class. This class' objective was to handle all features that would be consistent across all ghost types. This included movement, danger level, scene changes, win/loss conditions, behavior states, etc.

Additionally, a goal was to handle all changes to variables to the basic class with setters and getters. The team decided to use setters and getters over allowing direct modification of the variables in this class because we could ensure that the values would never be something that could break the functionality of the foundational features of the game.

For example, we implemented a new destination function in the Ghost class, this function checked if the given destination was on the navigation mesh and corrected the position if it wasn't. This allows for future development to avoid needing to add unnecessary checks in the future and abstracts ghost handling to make the process of development simpler.

The Second major base class created was the tool class which followed similar design structure and configuration. This class did the required legwork for tools to connect to other systems in the game and be usable. The team included several functions across all base classes that could be overridden, allowing for further modification as needed. We established certain functions that extended Unity functions like *Start* or *Update* so that we could ensure certain actions were being performed.

Establish Game Loop:

Due to time constraints and unexpected road blocks, the technical team could not fully implement all of the features planned in our PQP or implement all assets created by other domains like art and audio. Because of this, the main game loop was prioritized and several features were cut such as: Demon Ghost class, Wraith Ghost class, Ouija Writer Tool class, and the NaCL Tool class.

The game loop developed by the designers for this project had two phases, different events happening across all of them. These phases were further divided up into **four** behavioral states with basic functions. These states were given a numerical value and a general purpose:

- **Investigation**
  - **Hidden (Value: 0) -** The hidden state is the primary state of the Investigation phase. This state is when the ghost is not attacking the player, allowing them to explore their surroundings and find clues or set up traps.
  - **Haunting (Value: 1) -** The haunting state is the secondary state, this is when the ghost performs some action that affects the player or reveals information about the ghost. The Spirit in this case spawns copies of itself to confuse the player.
- **Dispersal**
  - **Flight (Value: 2) -** The flight state is the primary state of the Dispersal Phase. This is when the ghost is panicking and the player is allowed to disperse parts of it, eventually wholly dispersing the ghost.
  - **Fight (Value: 3) -** The fight state is the secondary state. This is when the ghost will protect itself by either guarding or attacking the player.

These states also may have individual transitions that can be overridden to for animations or certain functions like hiding or revealing copies, hiding or revealing itself, setting movement speeds or initial destinations. These four states encapsulate all behavior stated in the team's game design.

Future development Goals:

This project had many stretch goals that were not able to be met during this development period. These are recommended as starting points for future development.

Features:

The features that are missing from the game that need to be implemented are the wraith behavior, interactable object behavior, demon behavior, procedural generation of ghosts, creating a tutorial for players, and complete tool implementation. These features are not necessarily required if development branches off from the ideas of this year's project, the features here are what could not be done.

- **Wraith Behavior:**
  - Following the team's Game Design Document. The wraith should be faster than the Spirit. It also is required to be able to inhabit objects dotted throughout the scene. This requires the Wraith to know where all objects are in the scene and to then also implement a state where it is considered "inside the object"
- **Interactable object behavior:**
  - Objects created in the scene need to be able to be hit by some tool to affect a Wraith. The objects additionally have the requirement to be able to be thrown by a ghost and increase the Danger Level of the ghost if it "hits" the player. This also destroys the object.
- **Demon Behavior:**
  - The demon is the most aggressive version of a Ghost. It doesn't run away or go back into hiding. Once it is revealed, it will rampage until the player is forced to leave or is dispersed. The Demon gains notoriety by throwing objects at the player and being generally intimidating. This means the Demon also needs to know where objects are in the scene and be able to grab them, making the objects projectiles.
- **Procedural Generation:**
  - The missions that players go on should vary in difficulty and type. To do this, the game must generate a mission with a particular Notoriety rank to dictate health and other factors, and decide what Ghost is to be used with what element. This information also has to be transferred into the AR scene and activated correctly.
- **Complete Tools:**
  - There are four of six tools implemented in the game currently. The two remaining tools are the NaCL, a device that restricts the movement of ghosts by firing shots of salt into space. This tool has limited ammo and should follow the functionality listed in the game design document. The other tool is the Ouija writer. This tool can attract a ghost and reveal important information about them like name, element, or type. This tool is important to the investigation phase and should pair well with the SPI-Glass. This tool is able to be placed in the scene and interacted with by both the player and the ghost. Demons will act aggressive towards it

- **Elemental Attack:**
  - An important component of the complete game loop is the elemental attack. It is intended to be how the player transitions from the Investigation phase to the Dispersal phase, and while we were unable to complete the implementation of the elemental attack into the game, the system is fully functional and works correctly. The player draws one of the four elemental symbols in the center of the screen while the G.L.A.S.S. is active, and a script called GestureRecognizer.cs compares the drawn symbol to the saved patterns in the file gestures.json to determine which element the player is intending to draw. If it finds a match, then it creates an Elemental Attack in front of the player. The original GestureRecognizer script comes from a public Github repository made by user Oponn-1, and the version implemented in the game has a lot of minor changes to make it work in the way we needed it to, and a lot of additional comments to make it easier for others to understand the script in future years.

User Interface:

There were several pieces of UI that were implemented by the Art discipline of our team that weren't able to be added to the full functionality of the game. In future development, it would be recommended to add in these assets into the game and complete their functionality.

- **Tutorial:**
  - This game is intentionally a complicated system allowing for players to approach ghost hunting in different ways. This means that players also need to be able to learn the different tools at their disposal, as well as learning the different ghost types, ghost elements, and general phases of any ghost encounter. There is some development on this goal, but it has not been fully debugged or developed for the complete game.
- **Visibility Icon:**
  - The visibility Icon, which can be seen in the top left of the current official build of the game. This icon was intended to change from two different versions to show to the player if the ghost is visible or hiding.

Notable Issues:

Niantic Lightship:

Issue #1: API Overhaul

The 2024-25 version of the SPI-Glass Lab focused on finding the best software to build an AR game. Their dedicated work led us to using the Niantic Lightship API. This, by any metric, was a good decision.

However, Lightship was overhauled shortly after the completion of their project and rendered much of their work unusable during the development of this project. This severely hindered the 2026 development team from achieving all of the technical goals for the project, and forced a complete redevelopment of all the work the previous MQP had completed.

This issue is notable because there is no guarantee that the Lightship API will be overhauled again, possibly undoing the goals achieved during this phase of development. The solution that the team has found at this time is to not update the Lightship API beyond version 3.17 to maintain functionality of the project. This is achieved by relocating the package in Unity to the local files of the project, instead of following the normal process of caching the project.

Issue #2: Lack of Clear Documentation for Lightship

When developing using Lightship 3.17, the API as of writing does not have enough documentation to give a complete understanding of how to use it effectively. This issue is then compounded by the abundance of documentation for previous versions that are deprecated.

This meant for development that when trying to understand the operations of the API, one was more likely to find information on deprecated/reworked features that could not apply to the current version. This also had the effect of poisoning the responses of any large language model, when trying to more effectively collect resources, as the models cannot differentiate between information that is relevant and information that is out of date.

When following the tutorials provided by the Lightship team, it was found that details important to development were left out of the guides. This caused a number of issues and caused the development process to be marred with issues that could have been avoided if the guides were properly written.

The solution this team has found to somewhat resolve this issue is to read, debug, and comment the code in the package to see what is actually being accessed by the classes and functions being called. This solution works well since much of the API is not commented on or the comments are not descriptive enough. Any significant functions used by the project have been commented for future use.

Issue #3: Lack of responsive and helpful customer service

When researching the API further, this team has found that much of the customer service for Lightship is relegated to their discord server or their forums. These resources on the surface appeared to be sufficient to find answers. However, when reaching out on the discord server, questions were ignored or directed to the forums. This effectively nullified the usefulness of the discord, since it seemed that it couldn't be used to ask clarifying questions about the API. Additionally, the server had a low number of members and little to no management or moderation. An example of this is, a bot entered the server and began to repeatedly advertise a crypto-currency scam for days without intervention from any admin.

When asking the same questions on the forums, responses to questions were slow or non-existent. The first post made by the team received one response from someone on the development team asking for clarification, this response was after seven days of no interactions. When the issue was clarified, no response was given. The second post made by the development team has received no response at all or any other inkling of interest or support.

This issue repeatedly shows that the Niantic Development team cannot be trusted for finding information on their product and the community does not seem to be active enough to warrant consideration.

Issue #4: Silent Code Failures

When developing the project, the team ran into a recurring error that the NavMesh Agent made by Lightship would not path on the NavMesh (also provided by Lightship, and fundamental to the project). This was resolved at first, by removing both scripts from their respective game objects. This solution was not elegant or guaranteed, but there was no indication of what the issue that caused the problem was.

The issue was found after some time to be the NavMesh agent not properly receiving the NavMesh when trying to find a path to its destination. The Agent would have the mesh given to it as a variable when initialized. If the agent initialized before the mesh, this variable would remain null. Then, when setting the destination of the agent, the API would check if the mesh variable was null, and if it was, return and give no warning.

After discovering this issue, the team was able to make changes to the package to provide additional checking on if the mesh was set.

Issue #5: Project Relevant Features Cut

When developing the project, the team found that key features of Niantic's Lightship API were cut from the recent versions of the software. This includes their gps map system. This did not directly influence the immediate development of the project.

However, this issue continued to show that reliance on Lightship as the foundation of this project is risky. If major features like the gps map system that is heavily used in Pokemon Go,

the game that inspired this API, there is no way to guarantee that other major features could be deprecated as well.

Issue #6: Lacking DirectX 12 support

When developing the project on Windows, there were several small issues with using that operating system. One of which was that to use some of the simulation features that Lightship makes available, DirectX 12 could not be used, and the project settings had to be modified to use DirectX 11.

This issue was not difficult to resolve, but this issue highlights that Lightship's expectation that developers using their API will be only developing on a Mac operating system creates road blocks for developers wanting to use different operating systems. Niantic does not mention this in their documentation, only mentioning in one section that Windows development is experimental.

Apple:

Issue #1: Xcode requirement for mobile development

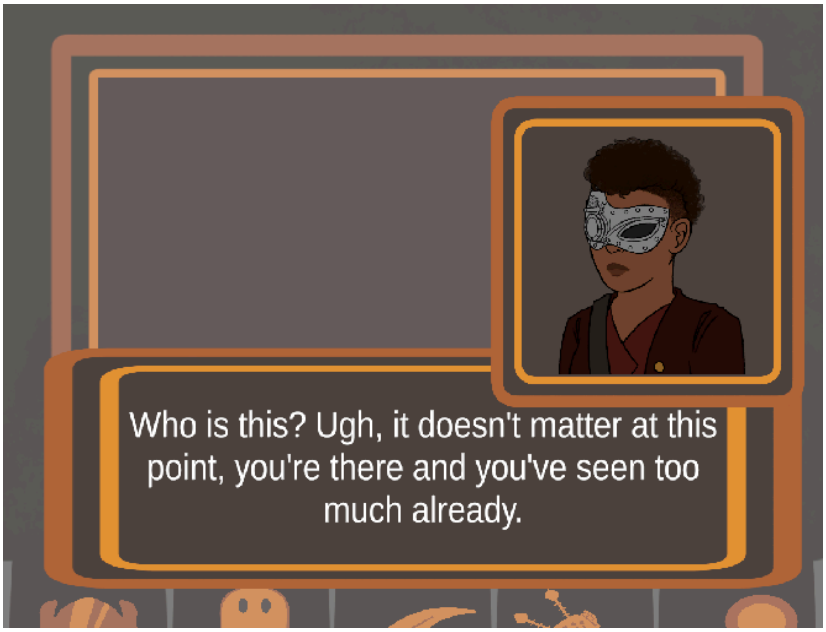
When developing the project, the team had to use a dedicated Mac device to build to iPhones. This added extra and almost unnecessary complications to the project. This restricted what we could easily test as the whole team was developing on Windows devices, meaning what was built onto a phone had to be at specific checkpoints. Additionally, when moving the project to a different operating system, some components of the project would break or not be correctly configured, requiring additional debugging that wouldn't have been necessary if it could be built directly from the development computers.

Once a routine was developed around the machine, issues became less frequent. However, the extra abstraction from the developed project made understanding what issues were arising from the devices and what could be done to resolve them overly difficult.

Future research should be done to find a way to build to iPhone directly from Windows devices through open source software, or virtual machines, to reduce development steps.

Narrative

The narrative elements of our project came into the play to guide the overall world and visual designs. Throughout the project, we worked closely to link the narrative elements into the tutorial design, to provide a means of guiding the player and introduce them to the world. In particular, the tutorial NPC, the Handler, was closely guided by decisions that connected her back to the organization as a whole, laying groundwork for further exploration into the lore of the world and providing future pathways for the player to engage with it.



Presently, narrative elements are communicated through a text box system, with a portrait for the speaker and space for her text to be displayed. Over time, this system will be expanded to provide stronger means of connecting the player to the world of the game.

The dialogue currently implemented within the vertical slice consists of our spirit tutorial as the main segment of story, while other pieces were written, this Spirit segment is the primary piece in the slice. The idea for the initial plot of the slice was for the player to be someone who saw the news advertisement arriving at the scene and found what seemed to be a state of disarray and chaos. After picking up a radio of some kind is tasked with helping Harper get control of the situation. The majority of our playable dialogue is this sequence, with barks and journal entries filling the rest of the slice's narrative load. The narrative, before being scoped down, includes encounters with the Wraith and Demon, as slowly but surely Harper would go from not really respecting your beginner's luck to seeing that you are a really proficient disperser that could help her uncover the secret of the founder and the higher ups.

Art and Visuals

3D Asset Creation

Each of the first 3 sprints were dedicated in part to going through the full 3D animation pipeline for each Ghost type. Each Ghost was first modeled, optimized and unwrapped in Zbrush, before being transferred to Autodesk Maya for material assignments, and Autodesk 3ds Max for rigging. Once a model reached this point our two artists worked on rigging and texturing the mesh in tandem. Rigging was done using 3ds Max's CAT (Character Animation Toolkit rigs) Rig plugin. Texturing was done in Adobe Substance Painter.

The tools followed a similar pipeline, instead being modeled and unwrapped in Maya before being textured in Substance.

Rigging and Animation

Autodesk 3ds Max's CAT rig plugin was utilized for the rigging process for both the Spirit and the Wraith primarily due to the ease of access to the system. CAT is very effective in making quick rigs that get the job done, but lacks the depth of traditional rigging methods for models. Any squash and stretch for the models, or movements like the mouth, would also need to be completed through morph targets, but this limitation worked out fine due to the lack of necessity in present animations. A major common issue would instead be encountered with joints, particularly the elbows. Since both models had a major focus on the arms in their designs, it became a balancing act in the overall rigging process to find a way to manage the arm that visually looked good. Ultimately, due to the shader system utilized for the models, it was determined that the majority of clipping issues would be nearly invisible, eliminating a portion of the necessity for more in-depth rigs for the arms. We found CAT to be an effective system in spite of these difficulties, as it made going from a model to a rig a very short process. Throughout the animation process, the rig would be tweaked as more problems came up, eventually culminating in the present day rigs for each model.

Another major part of the rigging process, particularly for the Spirit, was solving the issue of animating the tendrils. With the baseline CAT rig, the movement of each bone would need to be set individually, creating a very stiff look. This would end up being resolved with the spline IK solver, which allowed us to animate the tendrils by moving vertices that impacted the shape of a line, which the bones would follow along. This alone significantly sped up the animation process for the Spirit, allowing for complex movements with the tendrils to be completed in short amounts of times.

Shaders and Custom Lighting Model

The Ghosts and Tools final touches were applied in the Unity engine by applying Shaders and Post Processing effects. It was clear early on that we would be unable to get the cell shaded look we desired while using Unity's lit shaders and textures created in Substance Painter, as Unity would apply its realistic lighting model on top. We first tried to solve this by creating unlit shaders using Unity's shader graphs, researching transparency and toon shader techniques. The problem is that unlit shaders, since they ignore Unity's lighting model, are unable to receive scene lights or have shadows cast on them, removing the possibility of any dynamic lighting on our Ghosts or Tools. Additionally, we had trouble including texture maps other than the base color map into these custom models, notably our normal maps which were necessary to apply the detail baked from the Ghosts high poly meshes onto the low poly models. Additionally, we struggled to effectively combine the cell shading technique and transparency techniques in one graph.

We found that in order to combine multiple shader techniques and allow for a streamlined creation of new Ghost materials, we would need to create a modular custom lighting model using unity's node based shader graphs to bypass issues with Unity's default lighting, enabling greater flexibility. This system essentially does exactly the same thing as basic computer graphics

programs in game engines, just in a node based system. An additional benefit to building this system is that it creates a foundation for iterating of the shader and lighting settings in future years. Additionally this custom lighting model is less expensive than Unity's default lighting, improving overall performance, which is very important when developing for mobile devices. Our custom lighting model is largely based on Ben Cloward's *Custom Lighting Models* series of tutorials found on YouTube, diverging at points to accommodate our unique needs for Ghost transparency.

The foundation for our lighting model is the *BaseModularLights* shader subgraph. *BaseModularLights* is the assembly point for all of our other shader subgraphs, which each calculate an aspect of the lighting separately to later be combined into a single output. Where our shaders would normally feed texture map channels directly into the shaders master stack, and certain nodes would only be available in a Lit shader, that information is instead given to the *BaseModularLights* subgraph, where it calculates lighting and is then put in the Base Color node of the Unlit shader master stack, effectively circumventing Unity's lighting with our own.

Reflectance

The *Reflectance* subgraph interpolates between a higher and lower value based on a fresnel effect, simulating how less light is absorbed by a material allowing more light to reflect off the object towards the viewer. It then uses the metallic channel to interpolate between the previous output and the object's base color, making more metallic materials much more reflective. *Reflectance* returns a value that is later multiplied in the specularity and screen space ambient occlusion subgraphs, adjusting the strength of their reflections based on the player's perspective.

Specular

The Specular subgraph simulates the direct reflection of light from the surface by adding highlights to the area of the surface that reflects the most light. *BaseModularLights* currently uses the *Specular Blinn* subgraph, which gets the half angle of the view direction and the main light direction by adding them together and normalizing them. It then takes the dot product of the half angle and the surface normal, adjusts its power based on the roughness channel from the texture map, and is finally multiplied by the *Reflectance* subgraph output. We also created a specular phong subgraph that can easily be swapped in. Phong specularity doesn’t use the half angle of the view direction and main light.

Diffuse

The *Diffuse Lambert* subgraph uses the Lambert technique for diffuse lighting, which gets the dot product of the light direction and the surface normal to apply a scattered light on the lit side of the surface. There is a second *Diffuse Half-Lambert* subgraph available, which halves the dot product and adds 0.5, shifting the diffuse range from -1 -> 1, to 0 - 1 and wrapping light

around the back of the object a bit, resulting in a more stylized look. The current Ghost and object shaders use half-lambert.

Ambient Light / Global Illumination

The *AmbientSimpleSSAO* (screen space ambient occlusion) subgraph calculates the ambient diffuse and ambient specular lighting, allowing the scenes sky, ground, and horizon colors to illuminate the material.

The ambient specular is calculated by using a reflection probe node and multiplying it by the Reflectance. The ambient diffuse gets data from the *SSAO* subgraph, which uses a custom function to grab screen space ambient occlusion data that Unity already calculates under the hood.

The ambient diffuse takes the lesser value between the *SSAO* subgraph output, and the ambient occlusion channel provided by texture maps, whichever source provides the darker shadow is ultimately applied. Lastly this is multiplied with global illumination data at a given position in world space by using a Baked GI node, this masks global illumination in occluded spaces on the texture on top of any space occluded by calculations in world space. The ambient specular and ambient diffuse are added together, resulting in the final output.

Additional Lights

The outputs from the *Specular Blinn*, *Diffuse Half-Lambert,* and *AmbientSimpleSSAO*, subgraphs are all fed into the *Additional Lights* subgraph. *Additional Lights* uses a custom function to loop through all light sources found in the scene, and adds them together using the specular, diffuse, and ambient parameters it's given. Currently, each additional light uses the same color and shadow attenuation settings as the main light source in the scene, but this could be changed in the future to take unique parameters from each light source.

Assembly and Modularity

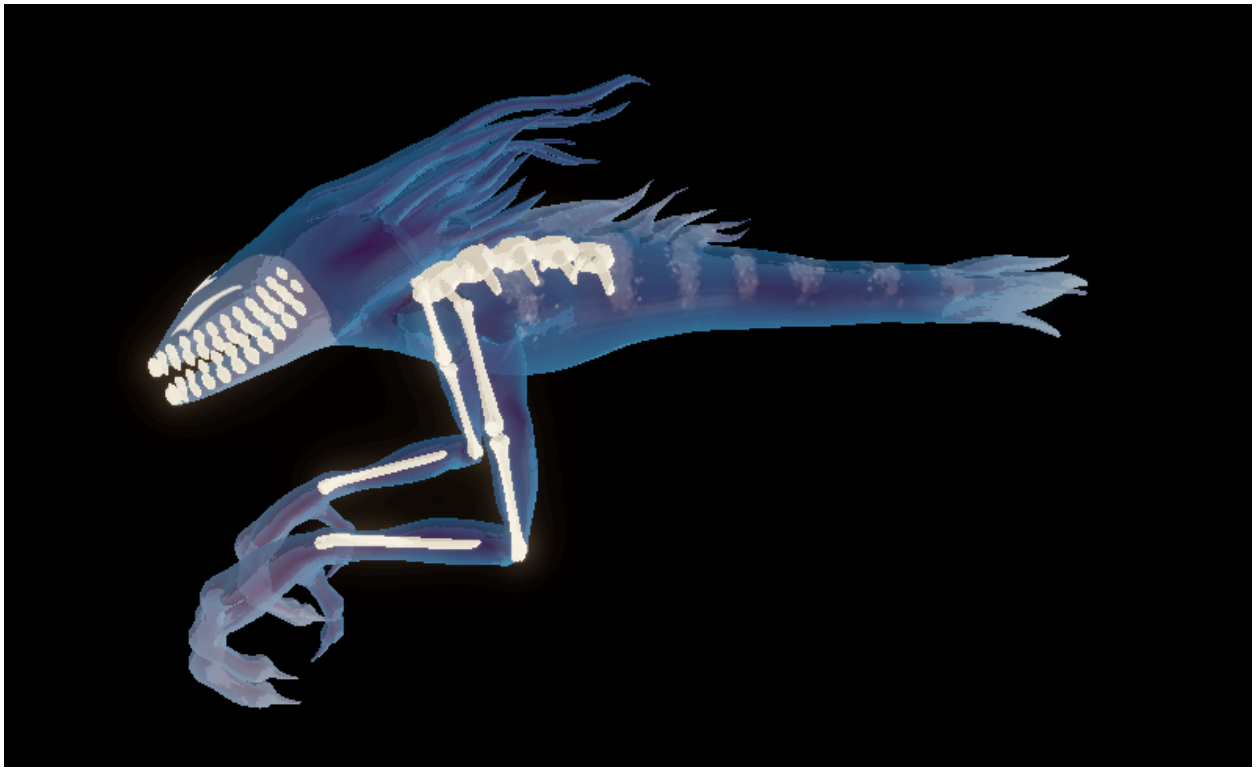
The *Additional Lights* output is combined with the outputs from the *Specular Blinn*, *Diffuse Half-Lambert,* and *AmbientSimpleSSAO* subgraphs, using add or multiply nodes where necessary. This is the final output of *BaseModularLights*, which in any given shader can be put in the base color channel of the shaders master stack. Each of the subgraphs can be swapped out for other variations, making it easy to change parameters and experiment. The *BaseModularLights sub* graph itself can be swapped out for variations based on a given shader's needs, building a library of highly customizable shaders that can be combined in unique ways.

We used this lighting model to create shaders for Tools and Ghosts. Both work on *ToonModularLights*, a variation of *BaseModularLights* that posterizes the final output, providing the cell shaded look we wanted. While the Tool shader puts the textures maps directly into the *ToonModularLights* subgraph, the Ghost_Base shader first multiplies a fresnel effect to the base color texture map, which is what the *ToonModularLights* subgraph uses as its basecolor. The

fresnel effect's power and color can be changed in the inspector, allowing us to easily create materials for each Ghost Element.

Spirit Final Render - Fire Element

Wraith Final Render - Water Element



Demon Final Render - Earth Element

Particles

Throughout the project, there were several effects that would come to mind as necessities for the gameplay loop, either in providing information to the player. All of the particle effects in this project were created using the Unity VFX Graph system, which made for a relatively quick and smooth process in a way that could be visually built out. The system gave us a significant amount of control over particle spawning, spawn methods, and how they behaved after spawning, making for a direct path for particle creation. It managed very well both with simple particles, such as our ambient dust and elemental particles, and in more complex variants- such as creating a water spray or the twirling dust of the vacuum.

The SUCK visual effect was created by utilizing a custom tornado shaped mesh. Two of these meshes are spawned by the VFX graph, one slightly larger than the other, both growing and spinning in space before shrinking at completion. A custom shader was created that animates noise on the mesh surface to give the impression of moving air. This effect is stronger on the smaller mesh compared to the larger, giving the tornado layers.

Audio

One of the most difficult yet challenges in working on a project like this as a one-term MQP is the “globally parallel development,” the idea that all facets of production will be worked on at once at the same time and level of progress. Whilst development timeline structure can of course vary from team to team, the traditional pipeline would be along the lines of “Designing a mechanic, then implementing that design, then creating the art and animations, then creating the audio.” On the other hand, due to both the time limitations of only having one-term and the makeup of our team, two dedicated artists and one designated audio person of the five members, we had to continuously develop art and audio assets for specific features whilst those same features were actively being conceived of, refined, and altered. Of course, this is in some ways a reflection of and preparation for the reality of a chaotic working environment, one where things can always change, features can get cut, characters can get reworked, et cetera. Nonetheless, this format extended that challenge to its absolute extreme.

This became particularly challenging for audio, specifically with regard to the sound effects, both in terms of editing and implementation. The two primary sets of sound effects the game needed are for the tools and the ghosts, and, whilst the ghosts were fairly set in stone in terms of what specific types of barks were needed and the different personality traits that needed to be conveyed, many of the tools were in a flux state a decent way into the project. Basic functionality like pulling up and shooting were easy to conceptualise, but some tools such as the EMF RADAR and the SPI-Glass had specific functionality added to them midway through production, whereas the Ouija Typewriter, the Salt Gun, and the Flamethrower Placeable all had to be scrapped for time after sound had already been edited together. However, the most extreme case of this is absolutely the Elemental Attack, an entirely new tool conceived midway through production, the exact specifications of which not settled until even later. To be clear, we came to

the decision to add this tool as a group as a way of logically solving a problem we had identified with our design. At the same time, the sound still needed to be created lest we fall behind, a challenge of either attempting to edit SFX earlier but without exact mechanics ironed out or waiting but having less time.

And all of this is also a sound design challenge. A sound effect can only be in the game if its corresponding host and mechanic is as well, so actually implementing all of the created SFX could only progress as far as tech had. Thankfully, we made the wise decision (no pun intended) to integrate Wwise, a platform specifically for video game sound design, into the project as early as possible, that way much of the sound design could effectively be set up external to the progress of the rest of the project. However, this could only go so far as, though the structure could be structurally implemented, whether or not that structure would actually hold couldn't be verified until much later. In other words, whilst being able to get some of it out of the way early was very helpful, having to do sound design without being able to hear most of the sound, compounded by the aforementioned design changes and the halted tech progress caused by the API, proved to be somewhat difficult.

With this in mind, overall output management also became a challenge. Having one person responsible for all of the audio is in many objective ways a benefit. On a group level it means that other members don't have to worry about creating audio assets amidst trying to finish other tasks, and that all facets of the audio are being created with a standard level of quality and experience, on top of essentially, on an individual level, providing dedicated time to cultivate of plethora of items to put in a portfolio in all the different fields: Sound effect editing, music composition, sound mixing, sound design, and even unexpectedly voice acting and voice directing. However, that also means that one person is in charge of managing sound effect editing, music composition, sound mixing, sound design, and, unexpectedly, voice acting and direction. In fairness, this is the type of challenge that's fun to embrace. Getting to be able to assume control of the audio identity of the game, to edit a common household noise into a RADAR beep and the squelching of ectoplasm or some simple mouth sound into radio static and a creaking radiator, to write four completely different arrangements of a piece of music that adapts to the encounter, to get to actually do fleshed out and comprehensive voice acting was an incredible opportunity. Nonetheless, it isn't any less daunting of a task when one person is responsible for making all of it come together.

Ultimately the best strategy for dealing with all of this ended up being taking on one discrete task at a time. Instead of strictly following the original plan of essentially trying to do all of the music, then all of the sound effects, followed by all of the implementation, it proved far more effective to go back and forth. To dedicate specific days to work on specific sound effects and music tracks as the primary focus. It also helped to extend further flexibility based upon productivity. Certain assets such as the vacuum SFX and the encounter music ended up being especially troublesome, and instances where it started becoming actively frustrating, it helped a lot to shift to another task for a while. Creating an outline of everything that needed to get done is in many ways a necessity, but exhibiting flexibility elicited the best results for the audio.

4 - Post Mortem

Compared to the original vision outlined in our design documents, SPI Glass's final playable is scoped down significantly from the original plan. The NaCL and Ouija Writer tools were both cut, alongside the Wraith and Demon's unique behavior and gameplay loop. Procedural assets and procedural mission creation were also cut. Other features including the SPI Glass and tutorial dialogue were developed but not fully implemented in the final build. SPI Glass consists of a Journal scene, which includes the title and main menus, and one AR mission, where players tackle the Spirit gameplay loop. The project was a major learning experience, and retrospectively we have identified areas that we would now approach differently given what we've learned.

Perhaps the biggest unanticipated complications to our project were related to using the updated Lightship API. Poor document code and deprecated features cost the team significant development time. Due to recent overhauls made by Niantic, the previous implementation of this game became deprecated. Additionally, this overhaul also deprecated all online documentation as fundamental systems were changed. With the lack of explicit documentation of features and functions of the API, the development team had to investigate the API's code, Identifying and fixing problems with Lightship held up the frontend feature development, and this in turn impacted other disciplines. Due to unique build requirements, we were also unable to test AR features in their intended environment until procuring a MacOS computer. As a result, all testing had to be simulated inside Unity for the first couple weeks.

Some of the roadblocks we encountered throughout development can be attributed to production and planning oversights. For one, some preproduction esque tasks bled into the production timeline, which if completed earlier would have streamlined asset development. In preproduction, we arrived at several decisions and agreements, but these weren't documented in a specific enough manner to avoid variance in interpretation among the team. This meant that early in production each discipline scrambled to clarify and adjust their designs. When design issues or contradictions were revealed, the given domain had to make changes that often affected dependencies and task prioritization of other domains, creating space for potential miscommunication.

Another area of struggle was managing meeting times between the relatively large team and contributors with conflicting schedules. While the core MQP team worked in tandem for the entirety of the one term MQP, supporting graduate students and 4099 contributors still operated on multi-course schedules. Though MQP students were available for most of the day, conflicts resulted in fewer chances for face to face interaction.

We have found that the multi-year nature of our project was underutilized in the planning phase. While a framework to later implement multiplayer features was planned, other aspects of our scope were treated as stand alone products, without considering intentional space for change. Later in development we did pivot to asset and system frameworks for future development, but considering our technical manpower, we believe a lot of time could have been saved by

intentionally setting a more limited scope, and explicitly focusing on creating building blocks earlier on, where we instead took that position out of necessity.

Recognizing these setbacks, these are our recommendations for future SPI Glass developers, one term MQPS, and multi-year projects.

Commit to being specific and granular in your initial designs and documentation. The inclination to set general guidelines and then improve designs as they are being created can sometimes be useful, but it often complicates coordination between the disciplines that need to build each part necessary for the intended design. While the initial design will expectedly contain issues, clearly laying out and committing to that design will streamline development, allowing you to playtest for short sights early on. The earlier you can test your game even if it's which also makes it easier to change.

For games with unique technical environments like augmented reality, aim to have more than one member who is familiar with the code base. Designate programmers to backend and frontend tasks, so that gameplay features can make progress separately from the development of the AR environment. By creating testing environments outside of the ones used to develop technological features like AR, allows for features to be developed not being dependent on the functionality of things like AR that may work inconsistently obstructing development otherwise.

Build executables and test early, even when there is not a lot to test. We learned why hands-on exploration of your game is required to find the areas that could use improvement and reveal design and interface issues. Many issues or bugs can be rendered invisible if not judged from the perspective of a player. Though getting different perspectives is important, internal playtests are majorly useful, and a crucial step. Related to this, prioritize paving a straightforward path for anyone to build and test on their device. This is specifically important for future developers on SPI Glass. Requiring building and testing to pass through one member wastes time, when each discipline is looking to test for different reasons. If like us, your project is simply unable to build early on, create paper prototype materials to roleplay gameplay, giving valuable insight in the meantime.

Do not neglect daily scrums and standups. Having a designated time each morning to make sure everyone knows what everyone else is progressing and intended next steps reduces the likelihood of misunderstandings. We found that scheduling weekly full team meetings with our advisors was helpful especially in the context of a one term MQP, and this was made possible by reserving space for the entire term way ahead of time. Do this as well for your daily scrums, this becomes even more helpful when in a position where contributors have conflicts with weekly full team meetings.

5 - Bibliography

Jimini Audio. (2026, March 7). *SPI-Glass Implementation Demo Reel* [Video]. YouTube. https://www.youtube.com/watch?v=ddfEM_mcjsI.

Jimini Audio. (2026, March 7). *SPI-Glass Original Soundtrack* [Video]. YouTube. https://www.youtube.com/watch?v=Mr-SS0e2nnE.

Jimini Audio. (2026, March 7). *SPI-Glass SFX Demo Reel* [Video]. YouTube. https://www.youtube.com/watch?v=ihalHy60JzI.

Jimini Audio. (2026, March 7). *SPI-Glass VA Demo Reel* [Video]. YouTube. https://www.youtube.com/watch?v=ELvnjKZASz0.

Jimini Audio. (2026, March 7). *SPI-Glass Voice Directed VA Demo Reel* [Video]. YouTube. https://www.youtube.com/watch?v=clxkssZGr18.

Jaliah Hippolyte. (2026, March 6). *SPIGlass Animation Demo Reel* [Video]. Youtube. https://www.youtube.com/watch?v=YeEUYLaApwY.

Niantic Lightship Developer Documentation — Niantic Lightship Documentation 1.0 documentation. (2023). Niantic.dev. https://niantic.dev/docs/

Niantic Spatial. (2026, February 13). NavMeshAgent is Fundamentally broken. Niantic Spatial SDK Community. https://community.nianticspatial.com/t/navmeshagent-is-fundamentally-broken/5672

Niantic Spatial. (2026, February 4). Additional Documentation for Lightship? Niantic Spatial SDK Community. https://community.nianticspatial.com/t/additional-documentation-for-lightship/5658/2

Ben Cloward. (2025, January 16) Custom Lighting Models Series [Video]. YouTube. https://www.youtube.com/watch?v=wI0YdkiduYY&list=PL78XDi0TS4lGnGa7L2X4o3UV-XYZEKNwj

Gabriel Aguiar Prod. (2019, June 27) *Unity Shader Graph - Tornado Shader Effect Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=Qyh9RPxeKcA&t=157s

Eric Wang_VFX Artist. (2025, December 8). *Unity Game VFX - Tornado Collection(Tutorial)* [Video]. YouTube. https://www.youtube.com/watch?v=dzXZIJ5MVww&t=173s

SPI Glass MQP. (2026 March 18) *SPI Glass*. Pinterest. https://pin.it/7Kabq3nAy

Oponn-1. (n.d.). *Oponn-1/unity-gesture-recognizer: A custom built gesture recognition system written for my game, maestro*. GitHub. https://github.com/Oponn-1/Unity-Gesture-Recognizer