Moonsighted

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

Authors:
Charlie Baldwin (IMGD BA)
Jasmine Duerk (CS, IMGD BA)
Kaamil Lokhandwala (CS, IMGD BA)
Dylan Valev (IMGD BA)

Advisors:
Farley Chery (IMGD)
Gillian Smith (CS, IMGD)

15 December 2021

Abstract

Moonsighted is a PC third-person hack-and-slash adventure about cultural and political strife in the face of technology and progress amongst moon-worshiping moths trapped in a cavern. Play as Mairu, a miner that harnesses the power of crystals known as ilarka, augmenting their body with arcane powers from the cave's precious natural resource. Explore dangerous depths to fight off cavernous critters to acquire more ilarka and help lead your people back to moonlight. This game portrays a fictional setting that depicts the dangers of unregulated and thoughtless progress of technology, where the player is put in the shoes of an unreliable narrator.

Moonsighted was developed in Unity engine for Windows, and can be played on the PC with a controller or keyboard and mouse. This report details the conception, creation, and polish of *Moonsighted's* story, art, design, and production.

Acknowledgements

We'd like to thank our advisors, Professors Farley Chery and Gillian Smith. With their input, critique, and experience, we were able to complete the project with the efficiency and quality needed for our short development time and remote workspace. We'd also like to thank all of our playtesters who provided feedback with their time with each prototype of the project.

Table of Contents

1: Introduction

This Major Qualifying Project is a game experience named *Moonsighted*, developed by four Interactive Media and Game Development majors at WPI. It was created during the academic B term 2021, a period of seven weeks.

Moonsighted is an action adventure game developed in the Unity game engine where the player character is a moth in a cave village. In this society, the moths worship the moon, and aim to escape the confines of the cave to reach it. In order to do so, the player fights enemies and collects crystals, providing power and special abilities to aid in their escape. However, this eventually leads to the village's downfall. The crystals taken by the moths are crucial in maintaining the structural integrity of the cave, resulting in the cave's collapse when mined. The primary gameplay loop consists of platforming to areas with crystal cores, defeating the creatures that protect those cores, and bringing back the mined cores to the village.

The project is a commentary on humans' destruction of the environment and its affects on the world. Difficult decisions have to be made to prioritize the needs of others while ensuring that the earth remains habitable and safe. Thus, one of our main goals with this project was to explore the relationship of the environment with humankind.

In *Moonsighted*, crystals are a source of power and light, greatly improving the lives of the moths and giving them hope to escape the cave. However, the moths disregard the effect that mining these resources would have on their home. The team wanted both sides of the argument for mining the crystals to seem reasonable and justified, mirroring situations in reality and portraying the idea that there isn't necessarily a correct answer. Narratively, our aim for the end of the game was to leave the player questioning their actions by having them assess whether their actions were wrong, despite the game's completion requiring them to destroy their home. This meant the player must grow an attachment to their current world, including the environment and the people.

Due to our short development period, our scope was small, meaning that our art and dialogue had to be rich enough that this attachment could happen in a short playtime. In addition to miscellaneous characters around the map, we created four named NPCs, or non-playable-characters, each with their own opinions on the village's future. The team also spent a significant amount of time on art and set-dressing so that the village and cave itself seemed like a place one would want to call their home.

By the end of the term, the team created a third-person Unity game to relay a message about human's destruction of nature. The game was tested on three separate occasions, and

learned about the balance of explicit directions and exploration, the importance of ambience and lighting in storytelling, and how essential controls and camera movement is in a third-person game.

This paper outlines the process of developing *Moonsighted*. Chapter 2 describes the background research and information gathered in preparation for development. Chapter 3 explains the design of the game, including lore about the world and main gameplay loop. Chapters 4 and 5 elaborate on the actual game creation, discussing technical implementation and art. Chapter 6 goes into the three playtests we ran for our game, how we prepared, and what we learned from each one. Chapter 7 concludes the paper and reflects on the outcomes of the project.

2: Narrative Inspiration

Humans have always created conflict with nature to reform, clear, and harvest resources for anthropocentric uses.

> "The romantic contrast between modern industry that "destroys nature" and our ancestors who "lived in harmony with nature" is groundless. Long before the Industrial Revolution, Homo sapiens held the record among all organisms for driving the most plant and animal species to their extinctions. We have the dubious distinction of being the deadliest species in the annals of life."
>
> **— Yuval Noah Harari, Sapiens: A Brief History of Humankind**

Moonsighted explores this tendency and is inspired by this theme. One of the concepts the team focused on to explore this tendency was the tragedy of the commons. The tragedy of the commons was conceptualized in 1833 by William Foster Lloyd, where many people of a community harvest a resource for short-term gain unsustainably (Spiliakos). This scenario ends in destruction of the land or the resource. A classic example of this scenario is overfishing, until there are not enough fish to repopulate the seas, and everyone ends up losing access to the fish (Spiliakos). The tragedy of the commons is rooted in a short-term reaction in relation to a natural resource. Using this as a basis of a moral dilemma, two sides of the narrative were created in the world of *Moonsighted.*

Different characters hold different views on the environment. There are three common viewpoints: anthropocentrism, biocentrism and ecocentrism. Anthropocentric views focus on humans alone, unfocused on the rest of the living world and environment. If an action helps humans, it is morally correct, regardless of the other animals or life it harms along the way (Barnhill). Biocentric views value all living organisms and biodiversity. Humans are viewed as an organism among many others on Earth, acting and interacting with them. An ecocentric view values abiotic factors such as river and water systems, as well as ecosystems within them (Barnhill). Biocentrism and ecocentrism have large overlaps and are often paired with one another, although even intelligent, long-term anthropocentrism has overlap with the other environmental viewpoints when taking into account all the environmental and biological services humans benefit from. Short-term anthropocentrism, or personal greed is what often leads to a tragedy of the commons.

Moonsighted draws on these concepts to create an underground area of moths, stuck in a cavernous area. Their "commons" are cave crystals called ilarka, and its power holds the underground together. Located in a dark tunnel above the moth's cavern village, the Teluna is said to lead to the surface, but the journey is dark and treacherous. A group of moths who interact with ilarka crystals obtain great strength and power, turning their skin and eyes a luminescent green. With their new power they believe they would be able to journey up the Teluna and reach the surface again. Thus, they are known as "moonsighted". The game takes place at a village of anthropomorphic moths, called the Etsuna. At the start of the game's narrative, the moths have discovered how to harvest underground crystals, or illarka shards, to become moonsighted. Before taking on the Teluna, they want to harvest as much illarka as possible in order to turn everyone moonsighted. That way, every moth would be able to complete the dangerous expedition and resurface. During this time, the player will witness conflicting opinions of the villagers regarding the destruction of the environment at the expense of becoming moonsighted.

3: Design

3.1: The World

The names and lore of the world were inspired by Basque mythology. The team chose Basque myths due to its emphasis and connection to the natural elements, building “upon the four natural elements, fire, earth, air and water.” (Bizkaia Talent). This mythology base aided in creating a culture for the game's world. There are many gods, goddesses and beings in this mythology to pull inspiration from. "Etsuna", for example is the name for village in *Moonsighted,* and it was created by combining the Basque word home, "Extea", and dark, "Iluna". We used this structure to create the world and name it with meaning.

Name	**Meaning**
Ilarka	Portmanteau of “ilargia” (moon) and “arroka” (rocks) - Google Translate
Onddargi	Portmanteau of “onddoa” (fungus) and “argi” (light) - Google Translate
Etsuna	Portmanteau of “etxea” (home), “sitsa” (moth), and “iluna” (dark) - Google Translate
Jonditsa	Portmanteau of “jondea” (people) and “sitsa” (moth) - Google Translate
Jondengoa	Portmanteau of “jondea” (people) and “lehenengoa” (first) - Google Translate
Uskerra	Shortening of “muskerra” (lizard) - Google Translate
Mairu	Derived from "Mairuak", a builder of tombs, standing stones (North American Basque Organizations, Inc.)
Odei	"Odei" refers to a deity or personification of

	storm clouds (North American Basque Organizations, Inc.)
Ila	From "Ilargi" and "Ile", the names of the moon (North American Basque Organizations, Inc.)
Olent	From "Olentzero" a surviving jentil (North American Basque Organizations, Inc.)
Tilak	From "Jentilak", giants that could throw stones (North American Basque Organizations, Inc.)
Iker	A Basque name meaning “visits” (Name Meaning Iker)

3.1.1: The Uskerra

The natural lizard-like fauna of the caves. They feed on the energy of the ilarka formations found throughout the cave system. As this is their only food source, they evolved differently from other forms of life, having only one large eye specialized to sense the ilarka energy. They also have a set of antennae growing from their heads that can detect subtle motions in the air, allowing them to know where other lifeforms are without seeing them.
These creatures are typically peaceful and historically have lived in relative harmony with the Jonditsa. However, when the Jonditsa began harvesting the ilarka, the Uskerra’s habitat was disrupted, and as a result grew defensive of their now dwindling food source.

As a defense mechanism, they are able to focus the ilarka they have consumed and reflect it back out through their eye, sending a violent blast of energy forward at threats, along with pouncing at nearby foes.

Figure 3.1.1a: The Uskerra

3.1.2: The Jonditsa

Anthropomorphic moths used to roam and fly on the surface. Long ago, the earth they rested on caved in, and sent a population deep below the surface. Void of light, they searched blindly through many dark and treacherous tunnels of the underground. The survivors, on the brink of insanity, found a dim light. Instead of their beloved moonlight, they instead found a grand network of luminescent mushrooms, called onddargi, which soon became their new home. Through many generations, the fallen moths expanded their new home by spreading mushroom spores and exploring the surrounding tunnels in order to find their original home, now a shadow, a myth, and a bedtime story.

The Jonditsa's only hope is a dark tunnel called the Teluna. Spiraling upwards with many branching paths, a downward draft emitting from its entrance is felt by the villagers. They believe the path to the surface lies up the labyrinth beyond the Teluna. To map its course, they have slowly expanded the area of the bioluminescent mushrooms into its tunnels, providing a stable and illuminated path up the winding tunnel. It is a slow and methodical process, but a necessary one in order to find their home.

Around the area, roots and mushrooms lead the way to ilarka formations - bright, beautiful, unbreakable crystal-moons. The discovery of ilarka's power came from an exploration team of Jonditsa. In one of the first expeditions of the cavern, a giant stalactite dropped on top of a monolithic ilarka core, releasing a blinding flash and cracking a small part of the crystal

core. To their surprise, the affected moths were not injured, but instead grew crystal formations on their arms and developed a green tint to their skin and eyes. This altered their genes, causing their children to inherit these traits. These moths, described as being moonsighted, were stronger and faster, and could use their crystals to harvest even more ilarka with ease.

Figure 3.1.2a: A moonsighted Jonditsa.

3.2: NPCs

The non-player characters of the Etsuna are placed around the village and the playable world, available to have conversations with Mairu, the player character. There are many NPCs, but four of them in particular have important roles in the game's narrative. Our goal was to display diverse opinions around the idea of becoming moonsighted.

Ila is a moth priestess, born moonsighted, but against the harvesting of ilarka shards to turn more moths moonsighted. This contradiction was the reason why she ultimately lost support from the rest of the villagers. Ila's late father was the previous village leader, and he led the people peacefully, but traditionally, as some view it. In his childhood, he was caught in an accident where a rock fell from the ceiling and blasted a crystal open. This caused an involuntary transformation, and became what was later to be defined as being moonsighted. However, Ila's father never viewed it as a gift. Ila is defined by her guilt of being moonsighted,

but she wishes for the people's trust. She has taken a place as a spiritual leader, though some believe her to be the rightful leader of the village.

Olent is an old moth who is not moonsighted. The destruction of the ilarka crystals pains him because he recognises they are integral to the cave systems. He is soft spoken, but humorous and clever with survival, and understands the environment better than anyone. Despite his knowledge about ilarka and its relationship to the environment, a heated argument with Odei drove Olent to leave the village. Now, he spends his time in the caverns, documenting everything he knows in a journal to document knowledge for future generations, so that past mistakes can be known. He lives ignoring his own regret, hoping his words will serve people after he passes.

Tilak is a young moth, ready to explore and very supportive of the ilarka movement. He also serves as the ilarka cook to serve upgrades to the player character. Tilak is the only NPC that must be interacted with, as he offers upgrades to the player character that are used to progress to each ilarka core. He is not moonsighted until after the second core is harvested. He is loud, positive and optimistic, and excited for his future of becoming moonsighted, believing that the moon on the surface will be found. Stuck taking care of the family business of cooking ilarka, he wishes he could be the one that mines.

Odei is Mairu's boss and the village leader. He is not moonsighted, giving him an initial appearance of being more relatable and representative of the people. Because of this, Odei has been steadily leading the village toward expansion, with a goal of getting everyone becoming moonsighted. Well spoken and respected but very opinionated and stubborn, his main rival is Ila, using the fact that she is moonsighted to argue that she no longer understands the people. He believes in equality in respect to being moonsighted, but is entirely consumed with the power it will provide.

3.3: The Player Character

Mairu is the player character and a moonsighted crystal miner, ambitious and ready to reach the surface. He feels the progress, the movement and excitement behind the moonsighted movement and is blind to its consequences. Since the player character is a moth prepared to dive deep into dangerous cavernous depths, the team wanted to capture both the lightness of a moth's movement and the impact of a warrior.

The evolution of combat follows a three-tiered system. As the player collects crystals from each crystal core, their tier will increase, furthering both the narrative and the player's skill

set. The player will start the game in the first tier. This equips the player with a three-hit attack sequence of lethal swings consisting of ilarka crystals extruding from their arms. Upon completing the sequence with the third and final attack, the player will perform a high-impact ground slam. This invites the player to strategize the timing of their three attacks, balancing two quick slashes and a third slow yet heavy hitting finisher.

In the second tier the player will unlock the ability to have an area of effect stun that will surround the player for a more defensive maneuver. Upon activation, the player will unleash stunning crystals that will instantaneously encircle them. Enemies caught in this encirclement will be stunned for a set amount of time, giving the player the opportunity to strategize an attack or perform a tactile retreat.

In the third and final tier, the player will be able to fire a projectile in the direction that they are facing. This allows the player to engage in battle in a very different way, since their kit in the first two tiers required close-up encounters to battle enemies. By adding the projectile, the player will feel the functional power of the crystals, complementing both gameplay and narrative of the importance of the ilarka crystal in the power it provides to whoever utilizes it. This tier will be the final.

3.4: Gameplay Loop

The gameplay loop was centered around the pro-ilarka protagonist, having the player feel the power of the crystals as they fly and battle effortlessly throughout the treacherous caverns in order to destroy ilarka cores and deliver them to the village. The player goes through three iterations of this loop, until the end, where the player's final blow delivers the death of everyone in the cavern's total collapse from environmental destabilization.

3.4.1: Movement and Combat

The player is tasked to navigate through a tunnel in a movement challenge, referred to as "platforming". This has the player thinking creatively and tactically as they have to jump, dash, and glide to edges of cliffs and pads of giant mushrooms in order to not drown in the caverns' watery depths below. By making the environment's platforming hazards organic, the player is able to tackle the space in the most natural way possible and remain immersed within the world and story. When the player has successfully platformed to the end of a tunnel, they reach an ilarka core, where a battle commences against waves of uskerra. The player must

survive by using the three hit combo attack sequence, dodging, and dashing, along with other abilities unlocked after obtaining more upgrades from Tilak with each harvest.

3.4.2: Crystal Collection

After the player has defeated every wave, they will be allowed to attack the ilarka core. Upon destruction, they will have successfully harvested the resource and begin to journey back to the village. The crystals remain unobtrusive to the player with no inventory or additional mechanic to the collection of the crystal itself. The team wanted to emphasize the destructive and violent nature of resource harvesting, and did not want to highlight or romanticize the process of manipulating a material to the point of forgetting where or how it came to be.

3.4.3: NPC Interactions

Most NPC, or non-player-character, interactions serve to drive the plot and provide more context and worldbuilding. As the player further destroys the caves, worried villagers will exclaim how the environment is decaying in tandem with others' galvanized controversial positions in the harvest-hungry neophiliacs. The only other interactions that alter the player's mechanics are Ila, who heals the player, and Tilak, who provides the player with essential abilities by feeding the protagonist ilarka-powered food sourced from their recent harvest. These abilities are needed in order to destroy blocked passages of additional caves. In doing so, the player has a reward and incentive to complete the loop, and they are also able to feel the positive effects of the ilarka, further justifying the narrative from an unreliable narrator's perspective.

4: Technical Implementation

We developed *Moonsighted* using the Unity game engine, because it allowed us to quickly iteratively prototype due to the engine's high level of control with C# scripting and thorough documentation. To manage and modularize the development process, we divided technical implementation into several managers related to key design components: player movement and combat, enemies, story, UI, and game state.

4.1 Game State

The Game State is a Unity scriptable object. A scriptable object exists in memory and can always be accessed in any scene, holding variables and preserving their changes as well. This makes it easy to access and easy to read from, especially when multiple objects depend on it's values. The Game State includes important data about the player such as health and the number of crystals destroyed. The UI will pull from much of the player data, and the story manager will pull from crystal data to determine environmental and story events.

4.2 Movement and Combat

Both movement and combat utilize Unity's built-in packages of the new Input System and Cinemachine. The former allows player input to be set up in a modular fashion, and the latter allows for heightened control of the main camera rendering what the player sees. The animation state machine built within Unity drives all of the visual communication of player input. We further expanded upon the animation state machine with a custom GUI tool for adjustable behavior and quick prototyping and testing. These are combined as a prefab, a built-in functionality in Unity that allows instances of objects to be saved and used anywhere within the project. In making a combined player and camera prefab, the team is able to prototype quickly within different scenes, allowing us to avoid conflicts with changes and experimentation.

4.2.1 Movement

Player movement is controlled from a third-person perspective where the player has full radial control of both movement and camera. The player's input is driven and detected through Unity's new Input System. This allows all detected input to share a custom-made Action Map, which handles any developer-categorized player actions, allowing for quick and easy setup and adjustments to player controls. For instance, as seen in Figure 4.2.1a, player movement is able

to be funneled through the *Player* Action Map. Thereafter, the Action Map categorizes the detected input under an Action, in this case, the *Movement* Action. With this set up, any kind of groupings of input can be categorized, as seen with keyboard input of WASD & arrow keys and controller joysticks. Such behavior is what allows the new Input System to be modular, as code only needs to reference the Action Map and Actions as opposed to long redundant lists of specific inputs.

Along with basic movement and jumping, the player is able to control the moth with high mobility by utilizing both air and ground dashes. After a player is airborne, they are able to press the jump input one additional time in order to flutter their wings and propel themselves forward, giving them tremendous movement in the air. Additionally, by holding down the jump input, the player is able to glide, holding themselves in the air for longer jumps and precise timing for where they want to land as they are able to survey the land from a strategic elevated viewpoint.

While on the ground, the player is able to dodge in three main directions (forward, left, and right) depending on which way they are facing and running towards. When triggering the input to dodge, the player will instantaneously rush towards one of the aforementioned directions, and will be able to dodge again after a short cooldown.

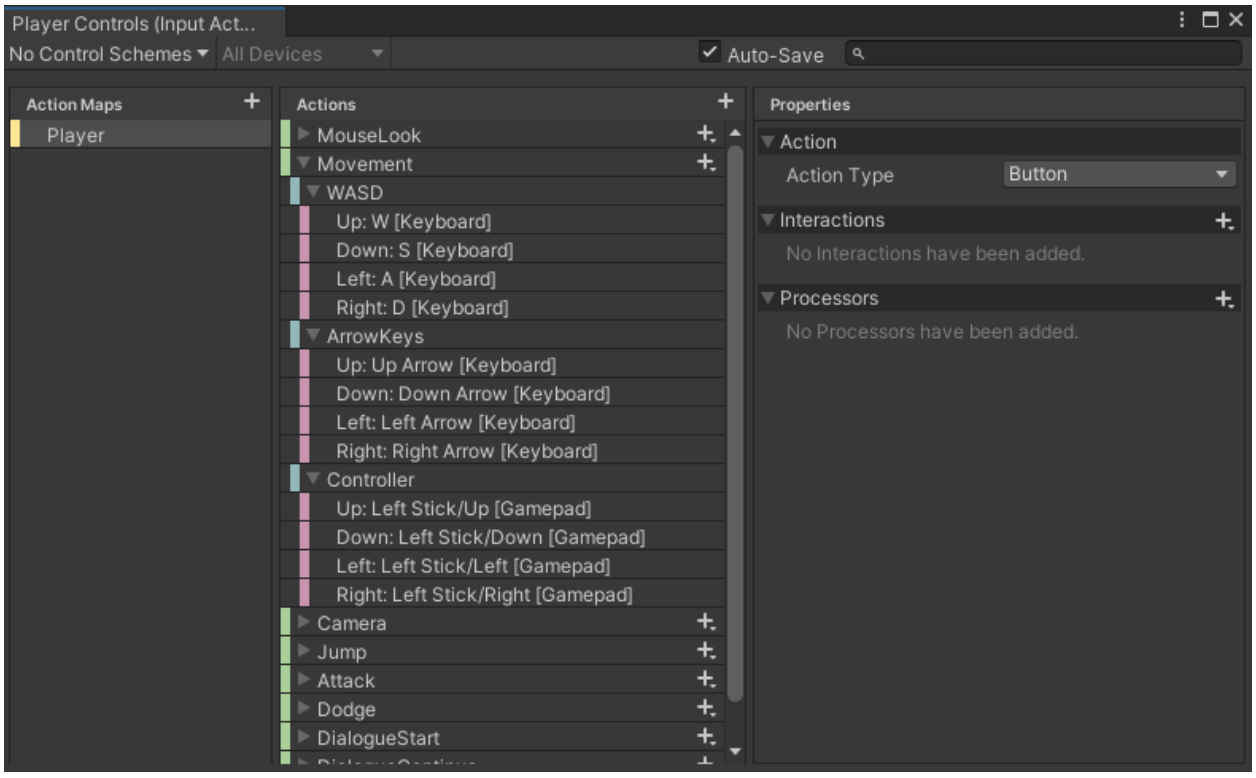


Figure 4.2.1a: Player Action Map

Additionally, because of the high-speed pacing of the game's combat, player visibility is of the utmost importance. It is crucial that the player is not only in control of their orientation, but also can rely on the camera to not hinder their performance. This is why Unity's Cinemachine was utilized, for it contains functionality for smooth camera rotation and interpolation between viewing angles. Additionally, Cinemachine's cameras contain functionality for collision detection. This allows the camera to not clip through the environment, and pulls it closer to the player if they are facing a wall or other static obstacles in the game scene. In doing so, the camera always maintains clear vision of gameplay, and minimizes the chance of the player being disoriented.

4.2.2 Combat

Combat follows a hack-and-slash paradigm paced with three hits per attack sequence. As the player progresses further into the game, they will unlock additional methods of combat to fight off enemies as they destroy more ilarka cores. The three-hit sequence and abilities are constantly monitored by Unity's built-in Coroutines, allowing for a separate function being called every frame. In doing so, we were able to ensure the game was able to update many systems at the same time without ruining the game's performance or pacing.

4.2.3 Animation State Machine

The animation state machine handles all of the animations that the player character should perform and smoothly blends them together. It determines this through both script and Unity's Animator component. By providing player-driven values through scripts into the Animator, things such as blend trees, transitions, and unique actions can be executed, as seen in Figure 4.2.3a.

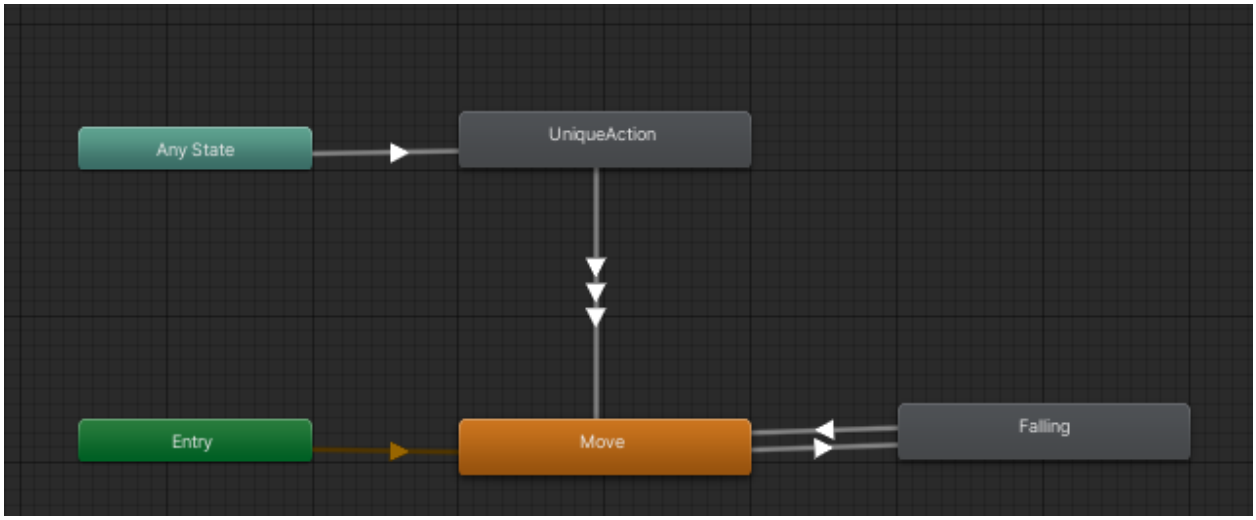


Figure 4.2.3a: Animation State Machine

The player is in a constant state of idle/movement. Because of this, a blend tree is utilized to smoothly interpolate between these states, as seen in Figure 4.2.3b. These states are smoothed by using a custom GUI that the team scripted (Figure 4.2.3c), allowing us to easily and quickly test different values to capture the best game-feel and player satisfaction when operating the player character. These values are also driven by attacking, giving the highest degree of control for easing in and easing out of attacks, both ground and aerial, allowing for the highest degree of control when running, jumping, attacking, and idling.

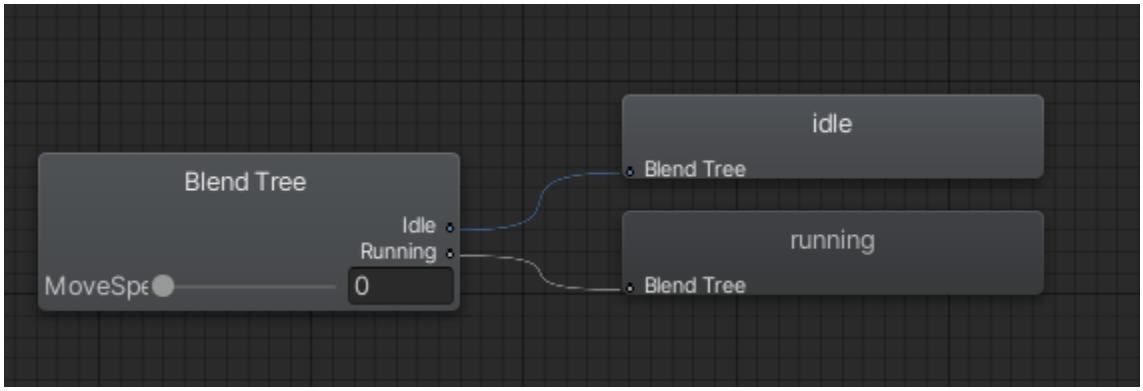


Figure 4.2.3b: Blend Tree for Idle and Run

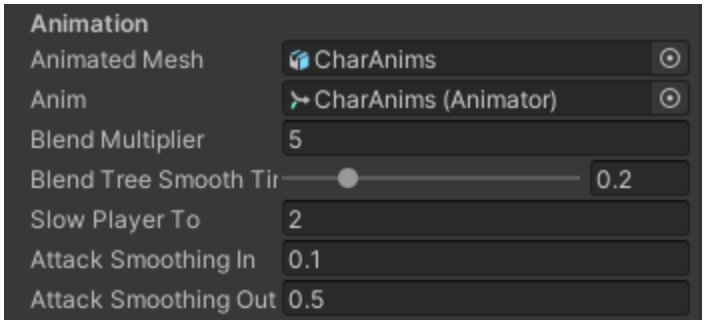


Figure 4.2.3c: State Smoothing GUI

4.3 Enemies

The enemies are a core part of the gameplay loop, posing as an obstacle for the player to overcome when mining crystal cores. In each zone that contains a crystal core, an initial wave of enemies will spawn as the player enters. A new wave will spawn when the prior one is defeated until the maximum number of waves (set in the code) is reached.

4.3.1: Scriptable Object

Each enemy area is defined in Unity through a scriptable object. This script contains information on each wave, including spawn points, number of enemies in each wave, and number of waves. This object also contains enemy stats, including health and damage on spawn, as seen in Figure 4.3.1a. Because there is one for each combat area, the team is able to tweak each zone's enemy information easily for balancing purposes and to increase the challenge as the game progresses.

Figure 4.3.1a: The Enemy Manager Scriptable Object

4.3.2: Enemy Prefab and AI

The enemy prefab is used to contain all information about an individual enemy. The prefab includes the enemy model and script. It is instantiated multiple times for each enemy in the zone. It contains the enemy's individual stats based on the scriptable object, and also tracks current health so that the enemy can be killed. The enemy script also contains AI functionality for movement and combat. Each zone has a baked navigation mesh that the AI uses to pathfind around the environment and to the player. Additionally, the enemy determines when to attack, and when to perform a melee or ranged attack, depending on its distance from the player and other factors such as obstacles in the way of the projectile's path. In a melee attack, the enemy will attempt to damage the area directly in front of it. There is an invisible collider parented to the enemy model in the prefab, and if this collider is hitting the player during the melee attack animation, the player will take damage. If the enemy chooses to perform a ranged attack, it will shoot a projectile, which is a model with a collider, towards the player. If this projectile collider hits the player, the player will take damage.

4.3.3: Spawner

In addition to the scriptable object, each combat zone has an enemy spawner. This reads in the scriptable object for that zone and uses it to generate the enemy waves. It also maintains information for wave control and knows when the player has finished the encounter

and can begin mining the crystal core. Whenever an enemy dies, a counter on the spawner keeps track, so it can reset the enemies and begin the next wave. Instead of instantiating the enemy prefabs for each wave, it initializes an amount of enemies equal to the maximum enemies that will be present in one wave.

4.4 Crystal Collection

Crystal cores are a prefab that contain a scriptable object. This contains the number of hits the crystal takes to be destroyed, the number of hits currently done on the crystal, and whether or not it has been destroyed. This information is used by the game state to determine the player's progress through the game (based on which crystals are destroyed).

4.5 Dialogue

Dialogue is a key component to making NPCs interactive and giving them life. The dialogue system was created to streamline the dialogue writing process so that the entire game can be written in a form similar to a play. This way, the entire team has access to the game dialogue document that can be quickly updated in the Unity game engine as a singular text document. To achieve this a few scripts have to work together to split up the written dialogue and distribute it to the correct triggers.

4.5.1: Parsing the Dialogue and Scene

The core of the dialogue system lies in parsing out a particular interaction, or a scene. Dialogue is a public class that takes a scene name and a file name. Upon initialization, this class will search the file for the scene and extract its contents as a list. The constructor to this class is shown in Figure 4.5.1a.

.

```
public Dialogue(string SCENE, string filename)
{
    txtAssets = (TextAsset)Resources.Load(filename);
    conversation = remLineBreak(txtAssets.text); //removes all line breaks
    sentences = conversation.Split(';'); //splits SCENES
    sentences = this.Process(SCENE);//splits lines and speaker tags in SCENE
```

```
}
```

Figure 4.5.1a: Dialogue Constructor

This class assumes the dialogue script is set up in a particular fashion, shown in Figure 4.5.1b.

```
SCENE2;
//PLAYER//]
Why are you staring up at the ceiling?]

//MOTH//]
If you watch closely, the mushrooms sway, don't they?]
The pulsing light, it's mesmerizing.]

//PLAYER//]
I'm glad you enjoy it, but our moon, our true home -]
It must be a thousand times better, right?]

//MOTH//]
Your eyes see the future. But mine see what is here for us now.];
```

Figure 4.5.1b: Sample Dialogue

Each title and block of text that makes up a scene is separated by a semicolon, and each character-line pairing within a scene is separated by a square bracket.

The speaker tag in the format //NAME// points to a character portrait and a text object to change the portrait and name.

4.5.2: Delivering the Dialogue

An NPC, or interactable, has a “DialogueStart” script that has a scene name attached to it. When a player is close enough to an NPC, a speaker icon pops up next to the NPC, signaling that they are interactable. As long as the scene name exists in the global dialogue script, the interaction will trigger a dialogue event. There also exists an option to type out the sentence character by character or have the dialogue all appear at once.

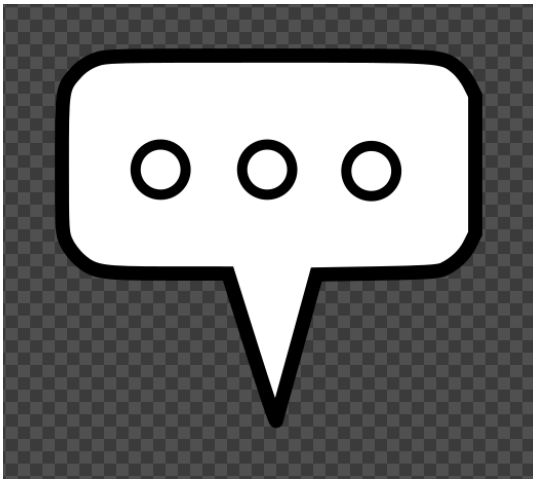


Figure 4.5.2a: Dialogue Icon

4.5.3: Characters

Each character has its own scriptable object that matches a name tag in the global dialogue script. Each scriptable object contains a portrait, a text color, and an array of sounds so that each character has a personalized feel. If the speaker tag does not find its associated object, it uses a default character object.

4.5.4: Dialogue Driven Events

To allow for dialogue and NPCs to affect the world, the dialogue script allows for function calls mid dialogue. Adding a colon to the speaker tag causes the script to run a parameter-less function in the StoryManager. If the function is not found, an error will be thrown. Below is an example of how it may look in the dialogue script.

```
 //:FullHealCharacter]
 //Ila]
I hope this healing will help you in your travels.];
```

Figure 4.5.4a: Dialogue Events, in the official game dialogue

4.6 Audio

4.6.1: Audio Manager

Unity's sound system requires an audio listener to receive sound and an audio source containing a sound file to play the sound. To consolidate these pieces to create a centralized sound system, the AudioManager was created. The AudioManager splits the game's sounds into four categories - action, ambient, dialogue and music. Each of these categories has its own audio mixer in the engine, so that different effects may be applied to each category of sound. For example, ambient noise might have more reverb than dialogue or action. The audio manager holds all the sounds that play, and is available for objects that are triggered to access it and play sounds from it. Figure 4.6.1a shows a view of the audio manager menu, and where different sounds can be inputted. Figure 4.6.1b shows a view of the different mixers in the Unity Engine.

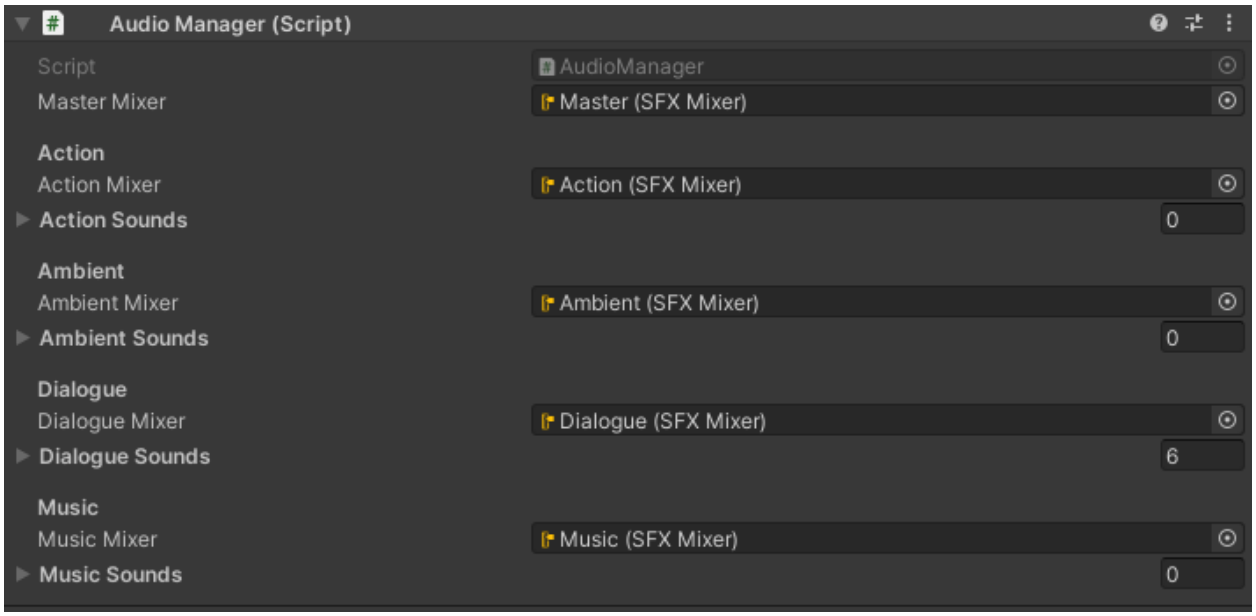


Figure 4.6.1a: Audio Manager

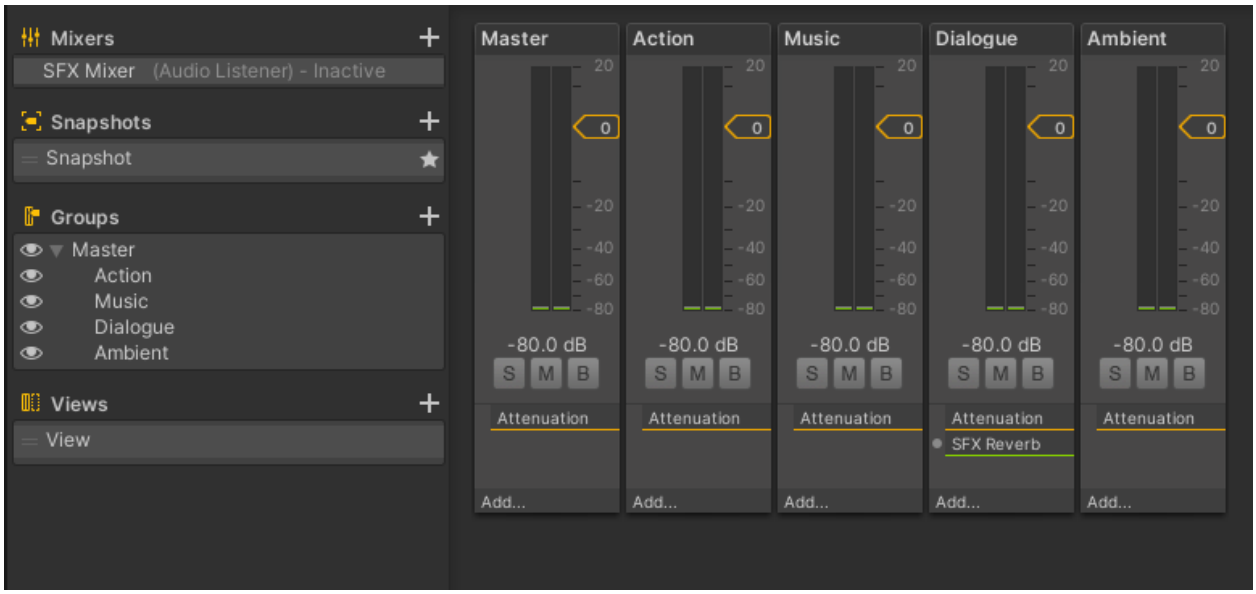


Figure 4.6.1b: Audio Mixers

4.6.2: Sounds

The team used an array of sounds from https://www.zapsplat.com/ and https://freesound.org/. Our custom sounds were the main soundtrack, played with a classical guitar and the moth sounds, made with foley and edited to the team's liking.

4.7 Story and Environment

Driving the story and environment is a central piece of *Moonsighted*. The StoryManager was created to be a strict, singleton manager that is in charge of all story-related events. The majority of its functions are private, but they can all be called with its "Execute" function:

```
public void Execute(string function)
{
	Invoke(function, 0);
}
```

Figure 4.7.1a: The StoryManager's core function

As shown in, one of the key uses of StoryManager is using the "Execute" function for dialogue based events. This function simply calls another function by using a string. This allows

the dialogue script to call existing functions in tandem with NPC interactions or other dialogue triggers.

To make play easy and straightforward, the StoryManager also fires tooltips that contain controls, new abilities, and combat cues. As the player speaks and interacts with the world, certain dialogue events and collisions can progress the story forward.

4.8 UI

The UI of moonsighted falls into three main sections: player state, tool tips and dialogue, and menus. The text assets in the project use Unity's TextMeshPro library. The library allows for bold, italics and color tags. In order to customize the project, the team created custom font assets for the dialogue, tooltips, and menus. The two fonts used were "Muskegon" and "Timeless". The UI backgrounds are a geometric fill colored in an earthy brown to match the moth-like theme. The background, shown in Figure 4.8a, was created in Inkscape, a free SVG software.

Figure 4.8a: UI Background

4.8.1: HUD

The heads up display shows the player's health in the form of crystals. When all of them break, the player dies. All the sprites are ordered in a grid-layout group, allowing for even spacing and accurate screen scaling. As the player takes damage, the HUD script senses a change in player health and updates the UI by replacing the full sprites with broken ones.

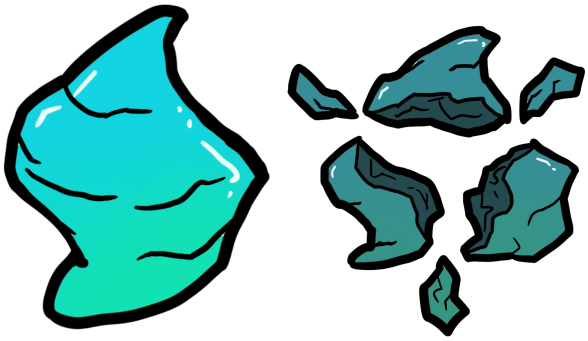


Figure 4.8.1a: Player Life Icons

4.8.2: Tooltips

The tooltips used DOTween, an imported library that allows for scripted tweened animations, in order to scale into and out of the scene when a notification is sent the player's way (DOTween). Each tip has a title and description. They appear in-game as shown in Figure 4.8.2a.

Figure 4.8.2a: Tooltip Example

The team downloaded free prompt sprites (for Xbox One controls specifically) from Those Awesome Guys (Xelu's). Then we made a sprite sheet of Xbox and keyboard icons using the Kavex/GlueIT: Simple SpriteSheet Tool (Kavex). Using the TextMeshPro library, the sprite sheet was connected with the tooltips UI, so that the sprites could be called in line, shown in Figure 4.8.2b.

```
Heads Up! Directions Below.
Press <sprite name="X"> then  <sprite name="Y">
```

Figure 4.8.2b: Tooltip Example text content

This makes the tutorial portions of the game unobtrusive and visually appealing. A "control" structure was created so that the sprites within the tooltips could have both keyboard and controller variants. The struct is outlined in Figure 4.8.2c, containing two strings to refer to controller sprite and keyboard sprites. This struct is used to define the controls and construct the notifications, so that they will change based on the GameState's control scheme, shown in the GetSprite function.

```
public struct Control {
    public string controller;
    public string keyboard;
}

Control Look = new Control("R_ANALOG", "MOUSE");
Control Move = new Control("L_ANALOG", "WASD");
Control Jump = new Control("A", "SPACE");
Control Attack = new Control("X", "LMOUSE");
Control Dodge = new Control("B", "SHIFT");
Control StartTalk = new Control("X", "E");
Control ContinueTalk = new Control("X", "E");
Control ExitTalk = new Control("B", "ESC");
Control Slam = new Control("Y", "CTRL");

public string GetSprite(Control control) {
    return $"<sprite name=\"{gameState.useController ? control.controller :
    control.keyboard}\">";
}


Notify("Welcome to Moon-Sighted.",
        $"{GetSprite(Move)} to move. {GetSprite(Look)} to look.\n
          {GetSprite(Jump)} to jump.\n
          {GetSprite(StartTalk)} to interact.\n
          {GetSprite(Dodge)} to dodge.\n");
```

Figure 4.8.2c: Notifications that switch between keyboard and controller

4.8.3: Dialogue

The UI for the dialogue contains some tweens as well, but much of the work is done by the character's scriptable objects and DialogueStart scripts to set the correct portrait, name, and dialogue content.

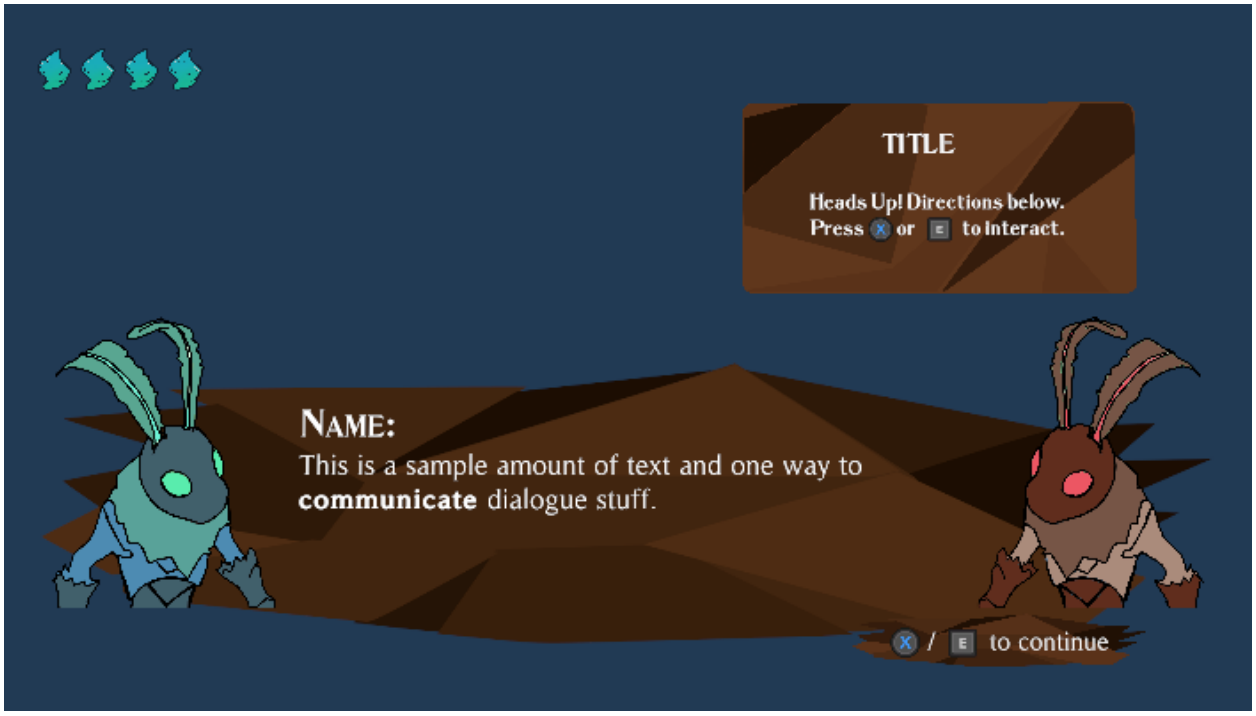


Figure 4.8.3a: Dialogue, Tooltip, and Health UI

4.8.4: Settings

The settings provide the player with a variety of tooltips, reminding them how to perform both basic mechanics and unlocked abilities. Additionally, the settings panel allows the player to pause the game and switch their control scheme from controller to keyboard, showing the inputs for each type of control. Finally, the player is able to quit the game and return to the main menu anytime they want.

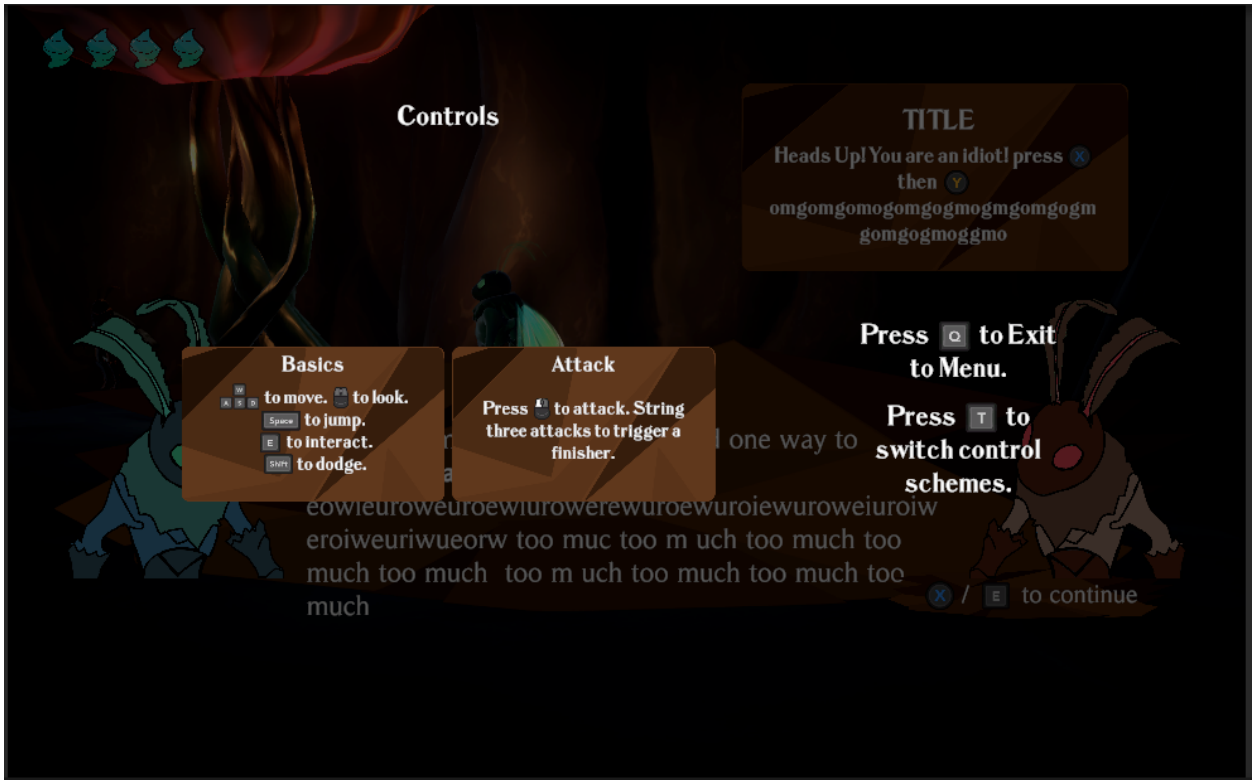


Figure 4.8.4a: Settings UI

4.8.5: Main Menu

The main menu is where the player arrives when they first open the game. In it, they are able to close the application, start a new game, and see the credits of who made the game and what their responsibilities were.

Figure 4.8.5a: Main Menu

4.9 Source Control

For source control, we decided to use Git, and hosted our project on Github. Our team already had some experience using Git, making it an easy choice for *Moonsighted*. When setting it up, we used the UnityYAMLMerge tool for smart merging.

```
[merge]
tool = unityyamlmerge

[mergetool "unityyamlmerge"]
trustExitCode = false
cmd = '<path to UnityYAMLMerge>' merge -p "$BASE" "$REMOTE" "$LOCAL" "$MERGED"
```

Figure 4.9a: Code snippet to add Unity's mergetool to .gitconfig

By adding the Figure 4.9a code snippet to each of our .gitconfig files for the project, Unity scenes and prefabs were able to be merged correctly and with less conflicts.

In addition to UnityYAMLMerge, we had other precautions in place to prevent merge conflicts. A majority of our scripts and assets were placed into prefabs, including the player character, NPCs, enemies, managers, and grouped environmental assets. This allowed us in many cases to edit the relevant prefabs without touching the scene itself. If two people needed to work in the same scene (for example, for building up the environment), the scene was duplicated and our team worked on the copies before merging manually. Lastly, we ensured that the scripts itself were isolated, and different utilities were split into different scripts. Thus, we could each edit different features without worrying about colliding with someone else. If same-script editing was necessary, we used Visual Studio Code live share, which allows teams to create a collaborative session where multiple people can edit the same script.

5: Art

Moonsighted is a third-person 3D game, so a majority of the art for the game included 3D characters, as well as environmental assets and props. However, there was also a significant amount of work done with regards to shading and lighting to give the game the mysterious vibe and style we wanted to achieve.

5.1: Characters

5.1.2: The Jonditsa

The primary characters in *Moonsighted*, the moth-like Jonditsa people, had to be given significant attention to ensure high quality in their design. In the story of the game, some of the moths are regular brown moths, while others have become ‘moonsighted’ and taken on a brighter green tone. To account for both of these varieties, we decided to just produce one Jonditsa model, and create two color variations, as seen in Figure 5.1.2a.

Figure 5.1.2a: Comparison of a 'moonsighted' moth (left) and a regular moth (right)

Despite the Jonditsa being moths, we needed them to have human-like proportions and body layout to simplify the animation process. This meant that we needed to make sure the moth features stood out, so that players would still see them as such. To do this, we put three

major materials on the Jonditsa's bodies: hard, shiny carapace armor on the chest, arms, and legs; softer, darker segmented flesh on the abdomen, head, hands, and feet; and light, fuzzy fur around the collar and on the forearms and shins. The carapace was essential for making the characters look like bugs, and the fur was the defining feature that made them appear specifically moth-like.

5.1.3: The Uskerra

The Uskerra are the main enemies the player fights in *Moonsighted*, although they are not the main antagonists. These reptilian creatures borrow elements from a few different species of animals, most notably axolotls, tigers, and dinosaurs such as T. rex. The axolotl is the most prevalent influence in the Uskerra's design, providing the iconic antennae. One unique feature we gave the Uskerra is their singular eye. In the setting, the Uskerra feed on the energy of the ilarka crystals, and so they've evolved a specialized crystalline eye that excels in detecting sources of ilarka energy. This look made the Uskerra visually interesting, as well as easier to spot against the dark cave backdrop. It also provided a logical source for the laser attack they use against the player.

Figure 5.1.3a: Moonsighted's main enemy, the Uskerra

5.2: Environment

We had a strong sense of what the world of *Moonsighted* would look like early on in the project, but until we started production we didn't know exactly what assets we needed to create to be able to put it together. In order to wrap our heads around this problem, we used modular kits to divide up our vision of the world, converting it from an abstract idea for a setting into a set of discrete objects we could model individually. The modular kit workflow works so well because we can then construct our level with each asset as a building block, using each piece both in ways we intended, as well as unconventional and creative ways. Our modular kit for this world consisted of three main subsets: the cave kit, the village kit, and other decorations.

5.2.1: Caves

Constructing the actual forms of the cave system that make up the world of *Moonsighted* was by far the trickiest challenge we had when designing our modular kits. Because caves can have such a wide range of shapes and sizes, it was hard to come up with a way to easily and effectively model them. Our first thought was to sculpt the entire cave system as one connected object with ZBrush, but this proved too daunting of a task for us. So, we broke down the caves into elementary parts. Our cave kit consisted of just a few pieces: an arching wall section, two types of floors, a cylindrical pillar, a ceiling, and a stalactite, as seen in Figure 5.2.1a. We were able to model each of these pieces individually, which was a much simpler task than modeling an entire tunnel. Once we imported them into Unity, they could be stuck together in countless configurations, allowing us to create whatever kinds of spaces we wanted, and to finely adjust the caves long after they were initially put together. Figure 5.2.1b shows an example of what the caves can look like when assembled from our kit. Given more time with our project, we would have created several more variations of each cave piece, as the limited breadth of our kit can make the caves appear somewhat monotonous at times.

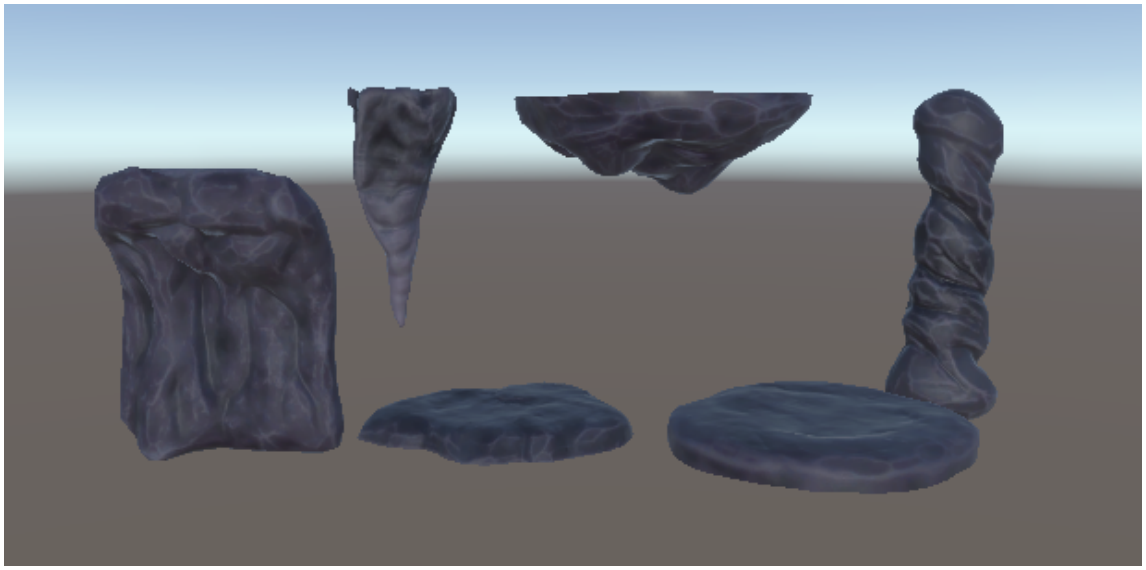


Figure 5.2.1a: Cave kit pieces (left to right): Wall, stalactite, floor #1, ceiling, floor #2, pillar

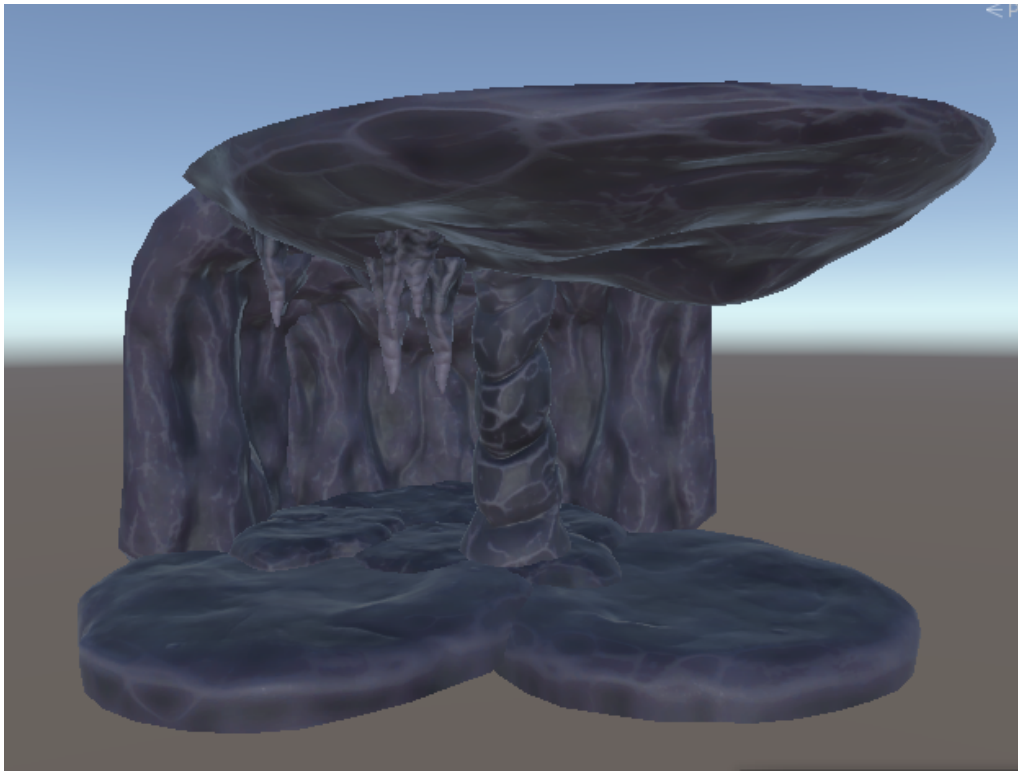


Figure 5.2.1b: Sample cave scene constructed from kit pieces

5.2.2: Village

The modular kit for the village was simpler than the caves. From our concept art, we knew exactly what models we needed to create in order to easily assemble our village area. Each house in the village was cylindrical in shape, and could be constructed from a few

separate cylindrical sections stacked on top of each other in various configurations. These sections included pieces like a ground-floor doorway piece, a fully enclosed wall piece, and a piece with a window and perch for moths to fly into. To supplement these core pieces, we also devised a few accessory pieces, including things like awnings, walls, draped fabric, and more. This was our plan in theory, however in practice it didn't end up working exactly like this. Instead of making each of these elements as a separate piece that could be sent to Unity individually, we instead created one complete house with each of these elements included, and created two other variations that had certain parts removed for variety. Figure 5.2.2a shows a complete house that includes all of the elements we created. This approach worked better for us due to time constraints; given enough time we would have been able to fully separate everything and create a proper modular kit, but this step had to be foregone to complete the rest of our tasks.

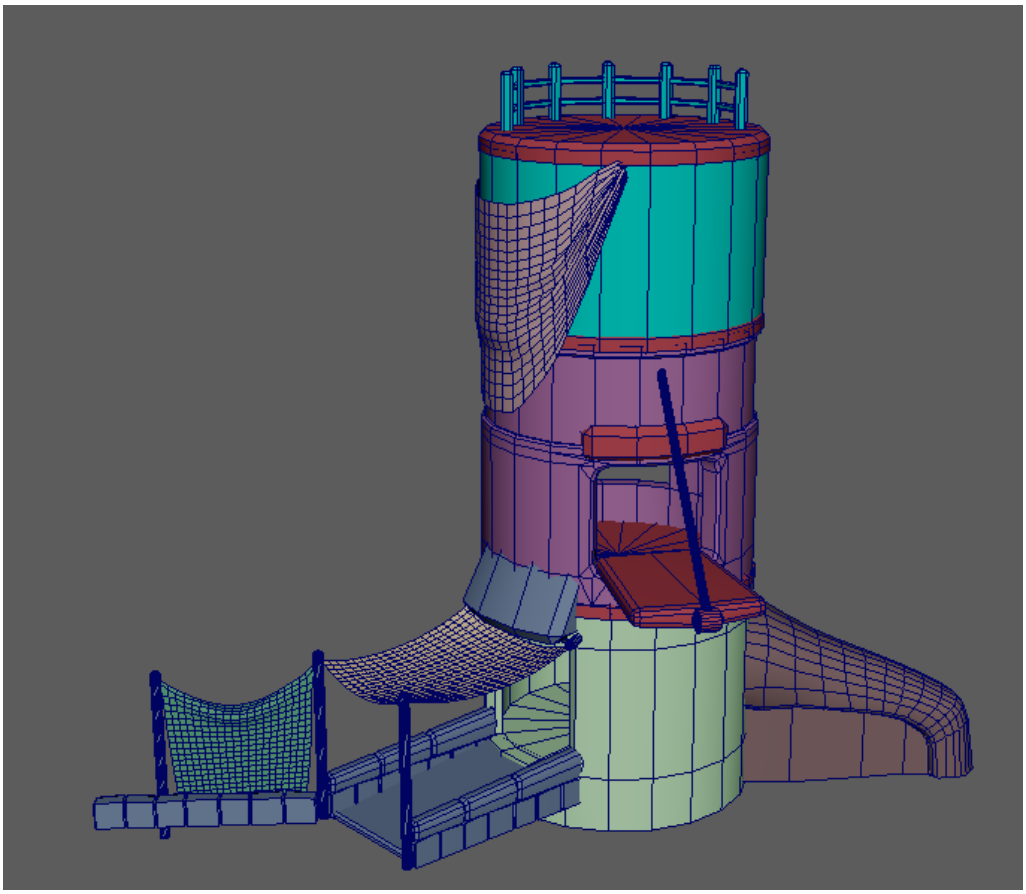


Figure 5.2.2a: Village house modeled in Maya, with all features included.

Figure 5.2.2b: Textured houses inside of the final scene.

5.2.3: Decorations

The last division of our modular kits was the various decorations we would use across the level, and was the most loosely defined of the three. This kit included anything else we wanted to have in our scene that didn't fit under the other two sections. This category included a wide variety of assets: the *onddargi* mushrooms (small and large variants), *ilarka* crystals, the *ilarka* cores, rocks and boulders, lanterns, and lore stones. Because this set of assets didn't have the same kind of cohesion as the other two kits, it was relatively simple to plan out. All we needed to do was make each individual piece and send them over to Unity, then we could scatter them around the scene as needed.

Figure 5.2.3a: Collection of decorative environmental assets we used in our scene.

5.3: Art Pipeline

As with any 3D game project, we needed to establish a clear pipeline for creating assets and implementing them in Unity. In the world of 3D art, a pipeline is a structured methodology for what software and techniques will be used for each asset, creating a clear pathway for taking an asset through the entire production process. While the pipeline can vary between different types of assets, and certain steps are occasionally skipped to save time, the plan still needs to be laid out as a reference point for everyone to use. Our pipeline comprised five distinct steps: concepting, modeling, texturing, animation, and shader creation.

5.4: Concepting

Creating concept art is an important first step when designing game assets. This involved creating 2D drawings of each of the models we plan to make, which can be used as reference when modeling later on. This step is crucial because it isn't generally advisable to

design an asset while modeling it. Drawing the design out in 2D is much easier as 2D art has a much quicker turnaround than 3D, allowing us to refine the design in a much shorter timeframe before moving on to modeling.

Before creating concept art, we drafted up moodboards to determine the style of the game, as well as to provide inspiration for the concepts for the environment and characters. We first used Pinterest to compile images in an organized board, before making a slideshow that explained our goals for art in more detail.

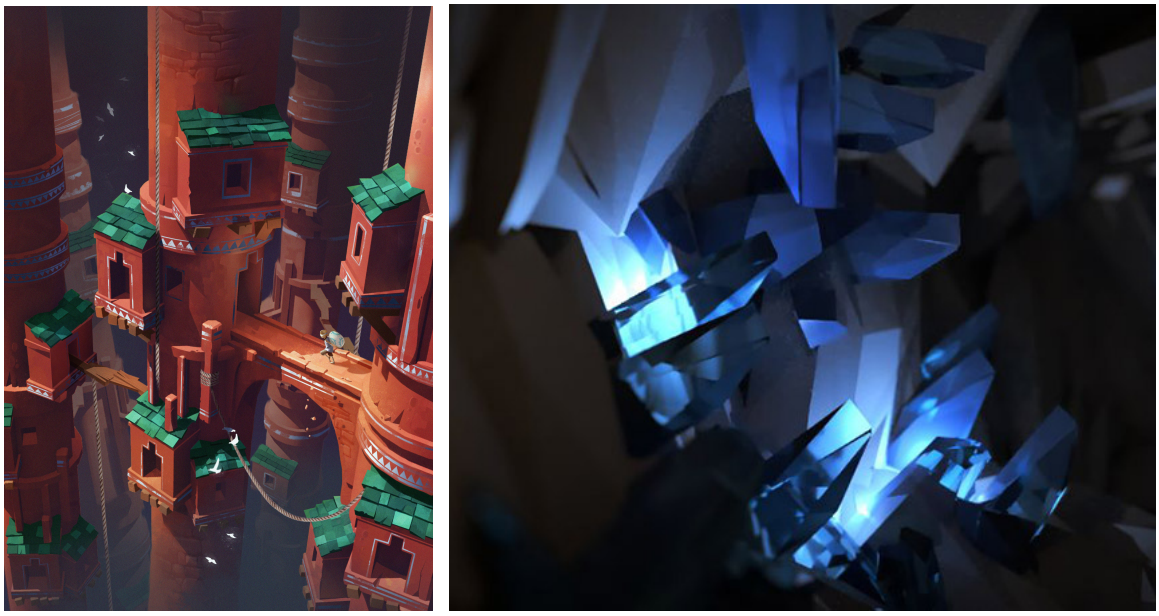


Figure 5.4a: Style Inspirations (Left: Concept Art by Elodie Mondoloni (Mondoloni), Right: Cave Lighting Demo by Pontypants (Pontypants))

Figure 5.4a shows an example of some of our moodboard images. The concept art on the left displays our goals for materials, with mostly flat colors and painted textures and accents for a stylized look. The lighting demo on the right demonstrated what we wanted to achieve lighting-wise, with a relatively dim environment contrasted by glowing, emissive focal points.

While we didn't create concept art for every single object that we included in our game, we did make some for each of the most crucial pieces. These assets included the *Jonditsa*, the *Uskerra*, the village buildings, the *ilarka* crystals, the *onddargi* mushrooms, and more. Assets like the cave kit pieces were not included, as the designs were simple enough that drawing them out beforehand would not do much to improve the modeling process later on.

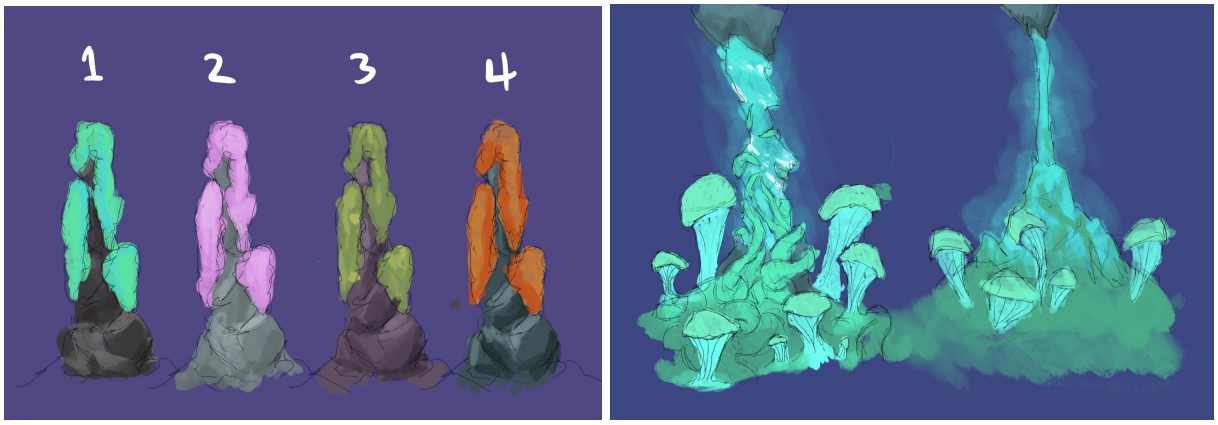


Figure 5.4b: Early concepts for crystal trees (right) and crystal core (left)

We had a lot of creative freedom with how to design the characters in our world, so it was tough to find a solid idea to start with. We knew we wanted to try and make the people some kind of anthropomorphic animal species, as they would be more visually interesting than humans. In order to find the animal we wanted to base our characters on, we first brainstormed a broad range of animals that could fit the setting. Because the game takes place inside a cave system, we came up with ideas for cave-dwelling animals, including salamanders, worms, and frogs. None of these really felt like they fit what we were going for, and eventually we settled on using moths as our inspiration. The thematic element of moths' attraction to light sources fit really well with our goals for the game. In real life, moths are drawn towards artificial light sources, often resulting in them harming themselves when they reach them. This is because their brains are wired to go towards the moon, in order to help them navigate at night. This introduces a really interesting theme in which the *Jonditsa* people are, in their hubris, striving for the crystalline *ilarka*, which would ultimately become their downfall.

In designing the actual characters, we wanted to really emphasize the moth-like features, while also keeping a fairly human-like frame to make animation and gameplay more straightforward. The key features we identified that needed to be present in order for players to identify the Jonditsa as moths included: wings, furry patches (especially around the neck), and the frond-like antennae. We added these features to a humanoid frame, and adjusted the limbs and torso to look more insect-like. From these details we settled on our final look for the Jonditsa, as seen in Figure 5.4d. We also wanted to differentiate the moon-sighted moths from their normal counterparts, so we drew inspiration from the real-life luna moth, which boasts a brilliant green wing pattern. This both aligns with the Jonditsa's affinity for the moon, and also visually alters them to look more like the crystals they consume.

Figure 5.4c: Luna Moth (Byrne)

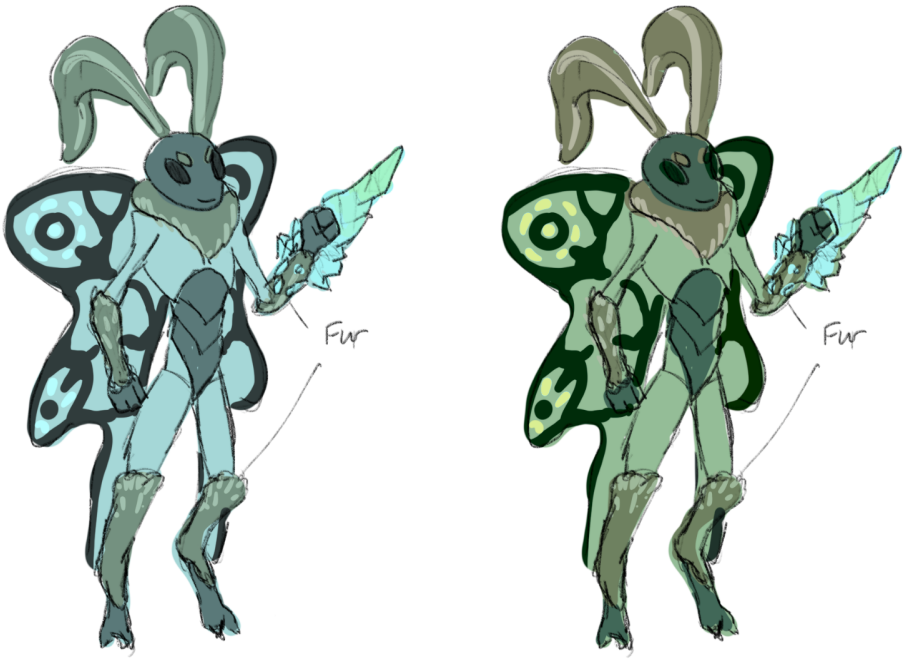


Figure 5.4d: Concept art for the player character, inspired by the luna moth

For the enemies of our world, we also wanted to go with some kind of animal, this time keeping the animal's original proportions to differentiate them from the moths. We ended up reusing one of our original ideas for doing a salamander/axolotl, and designed the *Uskerra* around that kind of anatomy. One of the big factors for deciding to use axolotls was the

antennae as they have a really interesting parallel with the moths' antennae. Axolotls and other cave salamanders are also usually blind, so we were able to introduce a property in which they have evolved to only seek out the crystals for their light, not needing to focus on other specific details in their surroundings. To emphasize this trait, we gave the *Uskerra* a singular large eye, glowing with the color of the *ilarka* crystals to emphasize their hunger for them.

Figure 5.4d: Concept art for the Uskerra enemies, inspired by the axolotl

5.5: Modeling

5.5.1: Blockout

To approach modeling, we decided to go with a blockout-based approach using the digital sculpting application ZBrush. This process entails using basic volumes, like spheres and cylinders, to construct the rough shape of the model. This allows us to get the major shapes and landmarks defined before diving into details. Using this workflow greatly increases our efficiency in modeling, as we slowly increase the detail in stages.

Figure 5.5.1a: Blockout of the Uskerra Model

5.5.2: Sculpting

The next stage of modeling is sculpting, also done with ZBrush. At this point, we use a function called DynaMesh on our blockout, which essentially connects all the parts together, and adds a much finer level of detail to the surface. With this added detail, we can use various brushes on the model to sculpt in the details we want. This is also usually done in two to three iterations, increasing the granularity of the DynaMesh surface each time. By the end of this stage, we have the model's 3D mesh looking exactly as we want it to look in-game, save for details we create in the texturing process.

Figure 5.3.2b: Finished detailed sculpt of the Uskerra model. The eye and antennae were added back during the retopology process.

5.5.3: Retopology

After we finished sculpting each model, we took it through an optimization process. The reason we have to do this is that the high-resolution models often have upwards of one million polygons on their surfaces, which is far too much for any game engine to handle. To remedy this, we do something called retopology, which involves taking a high-resolution 3D mesh and redrawing its surface polygons (or “topology”), maintaining the same broad forms from the original. We also must consider the edge flow of the model, which is a term for describing how cleanly the polygons wrap around the surface. This is important for animation, as when a body part of a creature rotates, it is directly manipulating whatever geometry is on the model. If the topology is not carefully laid out, the model might deform in strange ways when it moves.

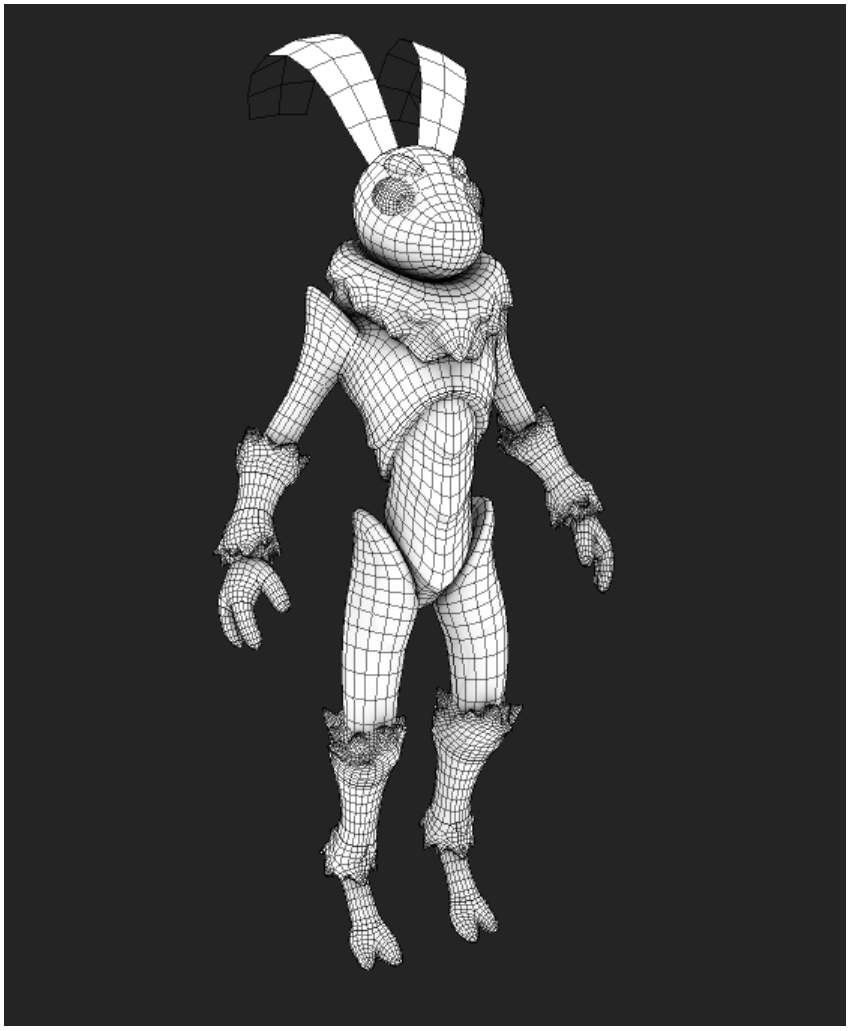


Figure 5.5.3a: Example of clean topology & edge flow on the Jonditsa model.

When retopologizing the characters for this project, we used two main techniques, depending on the model in question. The first one we used is called ZRemesher, and is a function of ZBrush that automatically retopologizes the model, taking influence from groups of polygons we designate on the surface. This method is somewhat unpredictable, and doesn't always produce a good enough result for us to use. However, in some cases it works perfectly fine. For the *Jonditsa* model, our initial ZRemesher tests went well enough for us to use that topology for the final model.

ZRemesher was generally our first choice for every model, but in cases where it didn't do what we wanted it to, we opted to go with our second technique, manual retopology. This method involves exporting the high-resolution out of ZBrush into another software package, which in our case was Autodesk Maya. Instead of using an algorithm to compute a clean

topology, we must draw out every polygon we wish the surface to have. This is more tedious, but it is significantly more likely to produce a result we can be satisfied with. In the case of the *Uskerra*, many attempts at using ZRemesher proved fruitless, so we had to instead opt to retopologize the model by hand.

5.6: Texturing

Texturing is the process through which detailed 2D images are wrapped around the surface of a 3D model, in order to enhance the appearance of the model in multiple ways. This process consists of a few key steps outlined below.

5.6.1: UV Unwrapping

In order to wrap a texture around a model, a UV map must first be created for that model. In short, a UV map is a representation of the surface of a model, unwrapped into a flattened net. Each pixel of the UV map corresponds to a point on the model's surface, so an image texture that is created for that specific UV map can provide detail for what color every point on a model's surface will be.

Like retopology, we had two main approaches to UV unwrapping, one being more automatic and the other being manual. The automatic route involved using ZBrush's built-in UVMaster feature, which allowed us to generate a set of UVs for a model with the click of a button. This approach worked great for things like rocks and mushrooms, whose shapes were more organic and simple. However, for other models like the *Jonditsa*, more thought had to be put in.

For assets like the *Jonditsa* and *Uskerra*, their detailed forms required manual UV unwrapping, as using UVMaster would result in a warped and stretched UV map. To do this, we sent the retopologized models over to Maya, and did the unwrapping there. This involved selecting which edges to use as seems, which defined the borders of each individual UV shell. Once every piece of the model that needed one had a unique shell, Maya was able to unwrap each shell individually. We then went in and laid out the shells in such a way that was both easy to read and used up as much of the UV space as possible. The end result of the UV unwrapping process for the main character can be seen in Figure 5.6.1a.

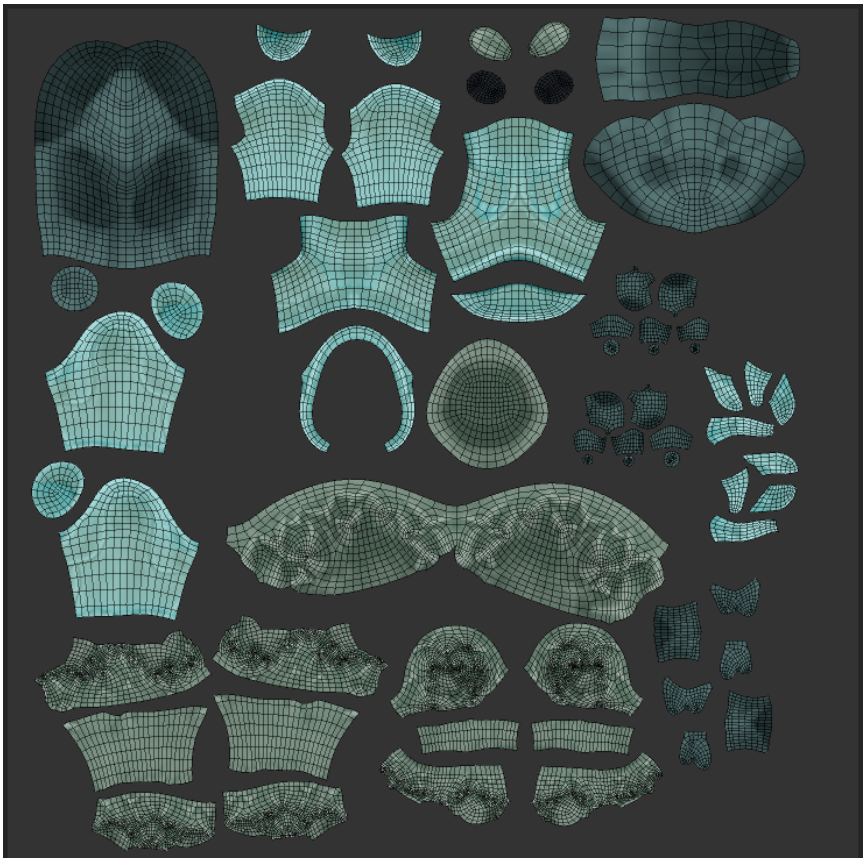


Figure 5.6.1a: Unwrapped UV map of Jonditsa model.

5.6.2: Texturing

Once a model has a good UV map, it's ready to enter the texturing process. Our primary tool for doing this was Substance Painter, a powerful tool that allows for texturing in multiple different channels simultaneously. The first part of creating textures for our models was baking, which takes the high resolution models created earlier during the sculpting process, and projecting them onto the surface of the low-resolution retopologized model. This creates what is known as a normal map, as seen in Figure 5.6.2a, which is a texture that informs the engine what angle each pixel on the model should reflect light at. In short, this fakes finer detail by communicating it through a texture, rather than actual geometry. For the most part, we only baked the models we were able to get to sculpting, which excluded some assets like the village buildings.

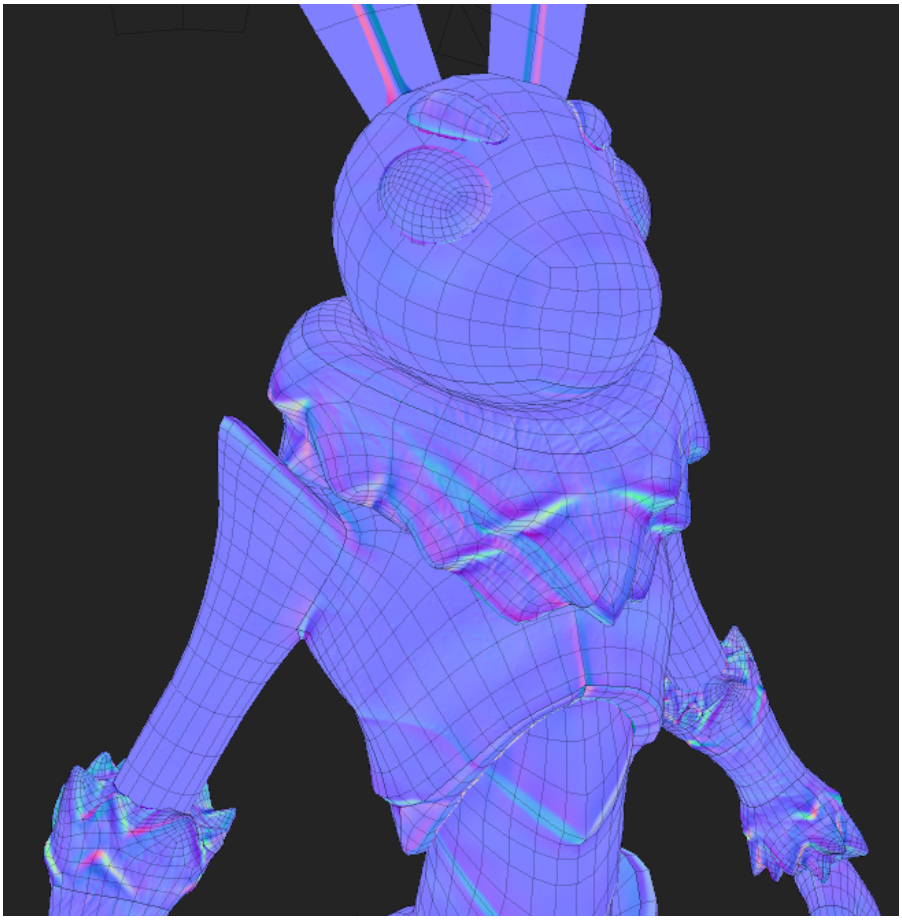


Figure 5.6.2a: Jonditsa model after baking. Note how the high resolution details have been transferred onto the low resolution model.

After baking, we used Painter's broad library of tools to both manually and procedurally create more surface details using various channels. The channels we used to paint with were as follows: Base Color, Roughness (how acutely the surface reflects incoming light), Metalness (how metallic the surface is), Height (the relative elevation of areas of the surface, allowing for indented or raised sections), and Emissive (how much an area glows, if at all).

The last step of this process is exporting. In order to export properly, we first had to settle on the export format we were using. Our game was made with Unity's Universal Render Pipeline, which has its own unique configuration for textures. Luckily, Substance Painter has presets for the output formats of many different destination packages, so it was as simple as selecting the preset for Unity URP and clicking export. We did make some slight changes to the name scheme of the export template in order to suit our organization needs. Our version of the preset gave the file a suffix based on what channel that texture represented. These included

“_BC” for Base Color, “_MS” for Metallic Smoothness, a combination of Metalness and Roughness unique to Unity, “_NM” for Normal, which also factored in additional Height information, and “_EM” for Emissive. Once exported, these textures could be dropped straight into a Unity material and applied to their respective models.

Figure 5.6.2b: Textures for the Jonditsa model. From left to right: Base Color, Metallic Smoothness, and Normal.

5.7: Animation

For animations, the team used a mix of hand animated and free motion capture from Mixamo. Initially we were going to use a motion capture suit to record and implement into the game, however, due to the COVID-19 pandemic, doing so proved difficult and would ultimately take too long for our project’s short development time.

5.7.1: Player Animations

All of the player’s animations for locomotion and combat were retrieved from Mixamo by uploading the player model and adjusting animation values for the specific movement we required. These animations were exported and brought into Maya, where animation clean up, reorientation, and skinning adjustments were made in order the animation looked as clean as possible on our player character before being imported into the project and animation state machine.

5.7.2: Enemy Animations

The enemy animations and rig were done within Maya after the model was completed. By making controllers for all of the limbs, the tail, and the mandible, the enemies had a high

degree of control for hand animation. These animations consisted of a walk cycle, melee and ranged attacks, and death.

Enemy movement was inspired by a lioness' movement and anatomy. We chose to use this as a reference because we wanted our enemies to have a more calculated and threatening feel, allowing them to prowl towards the player and deal very lethal precise attacks, encouraging the player to be nimble during combat.

5.8: Shaders

In order to communicate the aesthetics of ilarka's power, the team had to create stylized shaders which demonstrated the arcane power provided by the cavern's precious resource. To achieve this, the team utilized Unity's Universal Render Pipeline and Shader Graph, a node-based shader creation tool. This gave us a high degree of control of how to control the material in editor and in script.

5.8.1: Moonsighted Eyes & ilarka

In order to keep the appearance of ilarka cohesive, we used the same material for both the eyes of moonsighted moths and the crystals themselves. The material consists of a distortion technique of lerping between a texture's default UVs and a distorted variation where a gradient noise manipulates the base texture's UV. This process was contained within a subgraph (as seen in Figure 5.8.1a) to be used across other shaders and materials in order to easily maintain stylistic consistency across anything ilarka-based.

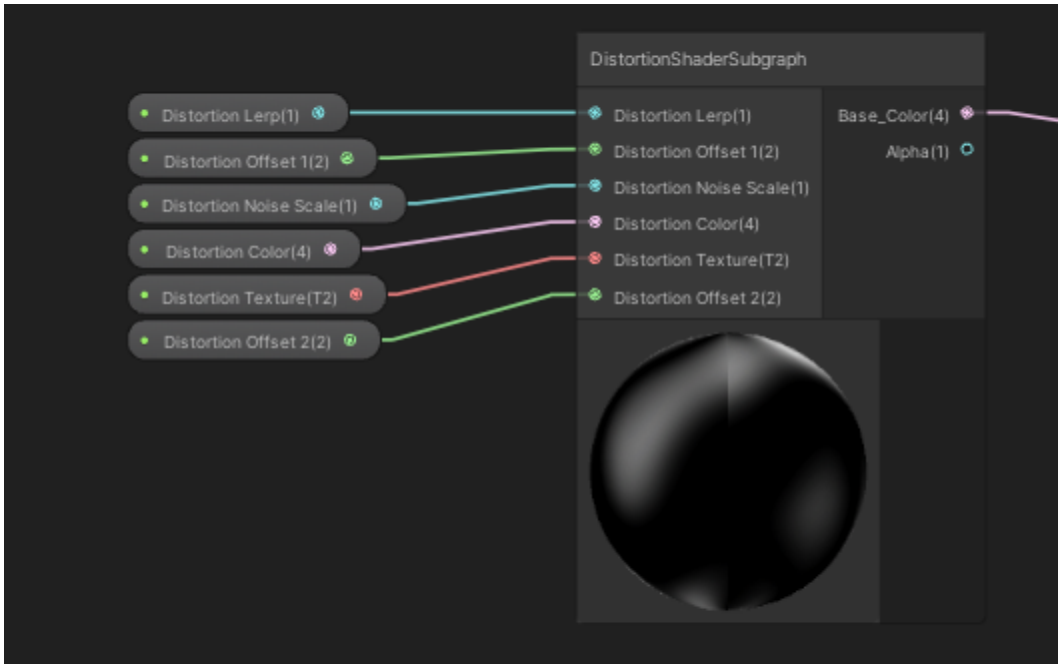


Figure 5.8.1a: Material Subgraph for ilarka-based Materials

5.8.2: Emissive Wings

Because the moth wings were actually rectangular planes, we used a transparent material to create wings in their actual shapes. This also made it easy to switch between regular and moonsighted moths through just a texture swap. Each wing had an upper and lower plane so that the wings could fold in or spread out (as shown in Figure 5.8.2a).

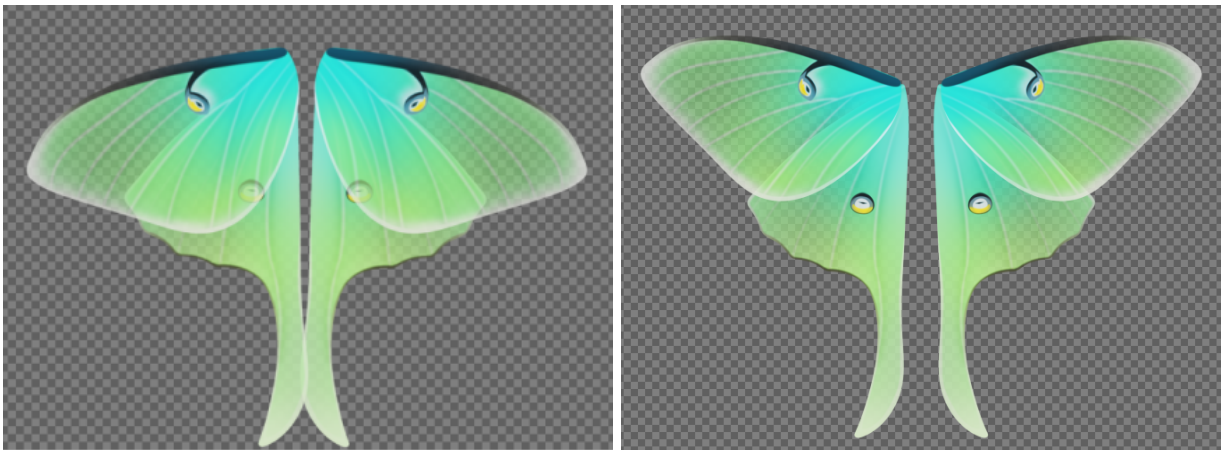


Figure 5.8.2a: Moonsighted Moth Wings (Two Orientations)

In the final material for the wings, the textures shown in Figure 5.8.2b were used for the diffuse map. Additionally, an emissive map was created for the moonsighted wings, which allowed the veins and edges of the wings to glow (as shown in Figure 5.8.2c).

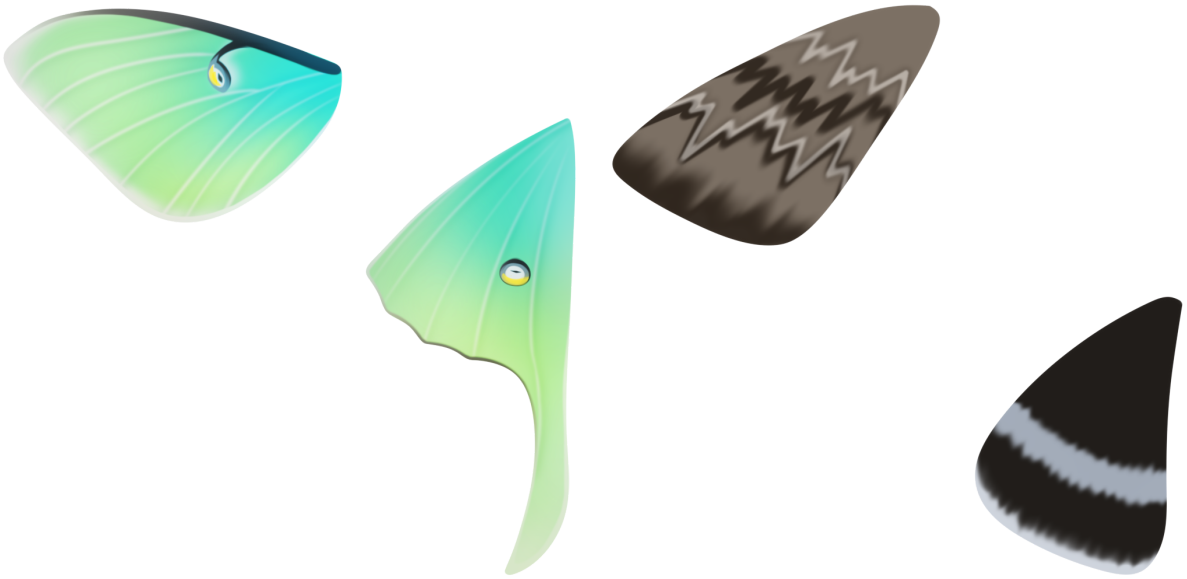


Figure 5.8.2b: Final Diffuse Texture for Moonsighted and Regular Moth Wings

Figure 5.8.2c: Emissive Map for Moonsighted Wings

In order to have the moonsighted moths' wings glow, we had to create a shader and materials specifically for controlling its emission in addition to the emissive map. Also, to reduce the amount of textures for a smaller project size, we added functionality for the texture to tile and mask itself to different sections, allowing us to use one texture for an entire wing set, as seen in Figure 5.8.2d.

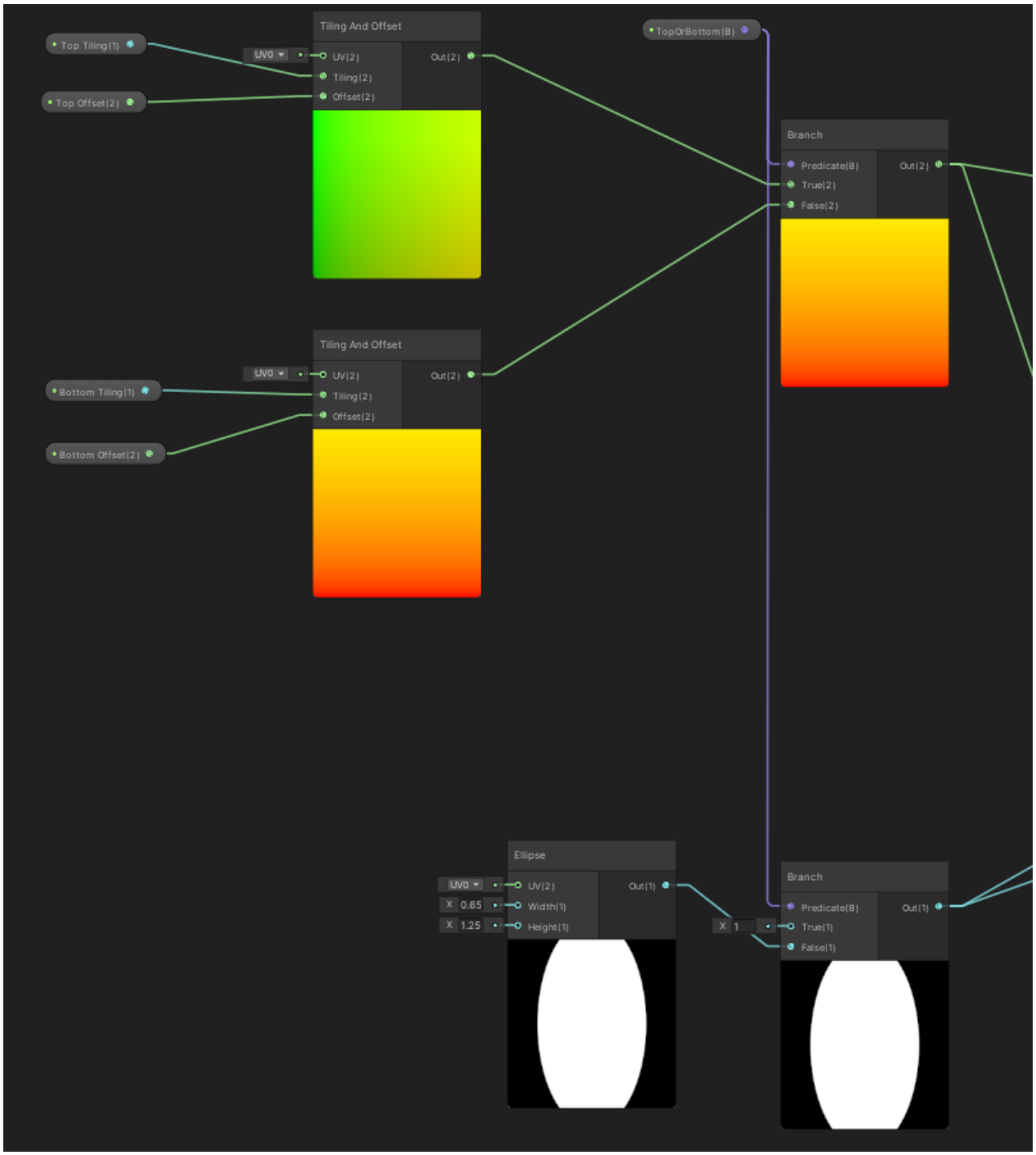


Figure 5.8.2d: Moth Wings Shader

Furthermore, in order to communicate to the player that they have used up their single air dash, we had to have the wings telegraph whether it is able to perform the action. We decided this was the best decision because the wings are what the player most prominently sees on the player character, along with the wings themselves being the subject of what was being communicated (their ability's use). This also reduced the need for additional UI, reducing our asset load and screen clutter, assisting with the immersion of the environment.

In order to control the emission's lerp, we had to access the properties of the wing's material (as seen in Figure 5.8.2e), and then use Unity's coroutines to update it alongside other script's update calls, making it less bearing on other scripts to monitor the interpolation of the material, leading to a more optimized and cleaner script as it can run through the while loop once, and then wait for other scripts to process that frame with a *yield return null* call.

```
/// <summary>
    /// toggles the emissivity of the wing's material
    /// </summary>
    /// <param name="active">if true turn emissive on. else turn emissive
off</param>
    /// <returns></returns>
    private IEnumerator EmissiveLerpCoroutine(bool active)
    {
        float currentEmissivePower = mat.GetFloat("_Emissive_Power");
        float totalTime = 0;

        while (totalTime <= emissiveLerpLength)
        {
            totalTime += Time.deltaTime;

            if (active)
            {
                // buffer for constant calls from player JumpHandler()
                if (currentEmissivePower != startingEmissivePower)
currentEmissivePower = Mathf.Lerp(0, startingEmissivePower, totalTime /
emissiveLerpLength);
```

```
            }
            else currentEmissivePower = Mathf.Lerp(startingEmissivePower, 0,
totalTime / emissiveLerpLength);

            mat.SetFloat("_Emissive_Power", currentEmissivePower);
            yield return null;
        }

        emissiveLerpCo = null;
        yield return null;
    }
```

Figure 5.8.2e: Script for Controlling Wing Material Emission

5.8.3: Water

To add a little more life to the cavernous underground, the team added water as the death zones.

Figure 5.8.3a: Water in Moonsighted

The water was created off of a 3D plane base with added shader effects. The water is created by three main elements: water color, normal map interference patterns, and vertex

displacement. The water color affects the plane's transparency and color. The normal map interference pattern creates a wave-like effect across the water by moving two different looping normal map textures over each other over time. As long as the material has enough smoothness, these normal maps create high-fidelity shadows and highlights that appear and disappear to create the illusion of movement. The vertex displacement gives the plane's vertices an up and down movement, so that the shores of the water or any object breaching the surface has visible waves oscillating on it.

5.9: Lighting

5.9.1: Lighting and Design

In a dark cavernous environment, lighting is key to making sure the player does not get lost. To light up the environment, there were fluorescent mushrooms, crystal structures, mushroom lamps, and backlit obstacles that emitted light in-game. Point lights were often used to accentuate their emissiveness to light up the environment. To lead the player to right spots within the cavern, the path was often lit up by green mushroom lamps, as seen in Figure 5.9.1a.

Figure 5.9.1a: Green Mushroom Lantern, leading the way

Initially, the lighting in-game contained very cool-temperature colors: blue for the small mushrooms, green-turquoise for the crystals, light-green for the luna moths, and a deep green for the lanterns. To offset this color imbalance, the team added large mushroom-trees to the

game, emitting a warm gentle orange light along with yellow objective ambient lights by obstacles blocking narrative beats.

5.9.2: Volumetric Light

The team believed that volumetric lights would add a high-quality polish to the depth of the lighting, so the team thought it would be a perfect fit for the shrine in *Moonsighted.* The problem with volumetric lighting is that it is very costly on the GPU, and not easily executed in Unity's Universal Render Pipeline (the render setting we used for *Moonsighted*). Instead, the team used Unity's shader graph to create a foggy light effect, using geometry shaped like a cone for the foggy volumetric body and a spotlight for the illumination itself. Additionally, we softened the edges of the foggy cone based on the camera position so that no clipping occurs. This resulted in a volumetric light-like effect without the GPU cost.

Figure 5.9.2a: Volumetric light effect at the shrine

6: Playtesting

To ensure this project's success, we had to take playtesting into account over the course of the term. Based on the schedule of the term, we decided to put together three playtesting sessions. The first would be on November 12, after three weeks of working on the game. This first session would allow us to get initial feedback on what we'd made up to that point, which

was crucial as we were the only ones to have played our game. This was necessary to lead into our second playtest, which was IMGD's Alphafest event.

Alphafest is a yearly project showcase held every B-term, where all in-progress MQPs are required to show the early build of their project. The event is held in-person, where participating teams can observe many users interacting with their projects live. With typical three-term MQPs, this is a way to get early feedback so projects can be polished before D-term's Showfest, where MQPs have to then showcase their final builds. Because we opted to do a one-term MQP, Alphafest instead came directly in the middle of our production timeline. By getting initial feedback out of the way the week before, we were able to iron out a majority of the major issues with our game. This allowed us to get much more specific and valuable feedback out of the much larger Alphafest.

6.1: Internal Playtest

6.1.1: Preparation

To prepare for our first playtest on November 12, we aimed to create a build that showcased all the features we developed up to that point. This included player movement, combat with enemies, and dialogue with NPCs. In ensuring that we had a playable vertical slice, we discovered many other smaller features that had to be implemented, including respawn, destroying rocks, and death if the player falls off a platform. Although we had the main features in already, bug fixing and completing these small tasks took us almost the entire week leading up to the playtest. By November 12, although our game worked in the Unity editor, we were getting mysterious bugs pertaining to all the player interactions. The most notable issue in the build was that the enemies targeted a random NPC instead of the player to walk to and attack. Although we resolved the issues a few days later, these build-only bugs prevented us from sending out a build for our playtest. Thus, playtesters instead were provided the in-editor version.

6.1.2: Running the Playtest

On November 12, we conducted our internal playtest. We could not send out builds, so we had playtesters play in person from our devices. Playtesters were first asked to fill out the IRB consent form, and then they were able to play the demo while we watched. By observing how they played through the level and asking them to speak their train of thought out loud, we

learned a lot about potential improvements that we would not know about with just the form. When they completed the demo, they then filled out the questionnaire we created.

6.1.3: Results & Findings

For our internal playtest, we had 6 participants who played *Moonsighted* and filled out the survey in Appendix C. The results of this survey is in Appendix D. Our main goal with this playtest was to find jarring issues and bugs for our demo we would display at Alphafest the following week. The main issue that participants commented on was the player camera. The camera had sensitive free rotation at all times, so players had trouble seeing what was in front of them, and this was further hindered by the jitteriness when clipping with cave walls or other objects. Additionally, the camera had no capability to look up or down, resulting in difficult platforming areas where the character could not actually look up to see their target platform to jump on. Because of these problems, 50% of users rated their control over their player as 1 or 2 on a scale of 1-5. It was clear that improving the camera was our main priority for our Alphafest build.

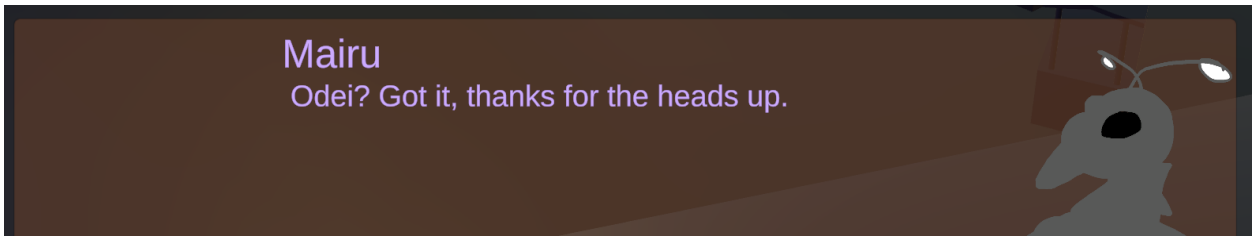


Figure 6.1.3a: Original Dialogue Box

Besides the camera, there were other smaller issues that had to be addressed. The dialogue was confusing due to the layout of the UI, which made it unclear that one of the moths speaking was the player himself. Also, the purple text color was not very readable. The players would most often skip reading the exchanges with NPCs, which demonstrated we needed both UI improvements and more concise dialogue to promote actually reading it.

6.2: Alphafest

6.2.1: Preparation

To prepare for our alphafest build, the team had to review and improve our internal playtest build on a time crunch. The most concerning problem was the build bug, where enemies would attack the NPC instead of the player! Upon a player falling, the death event would not trigger properly.

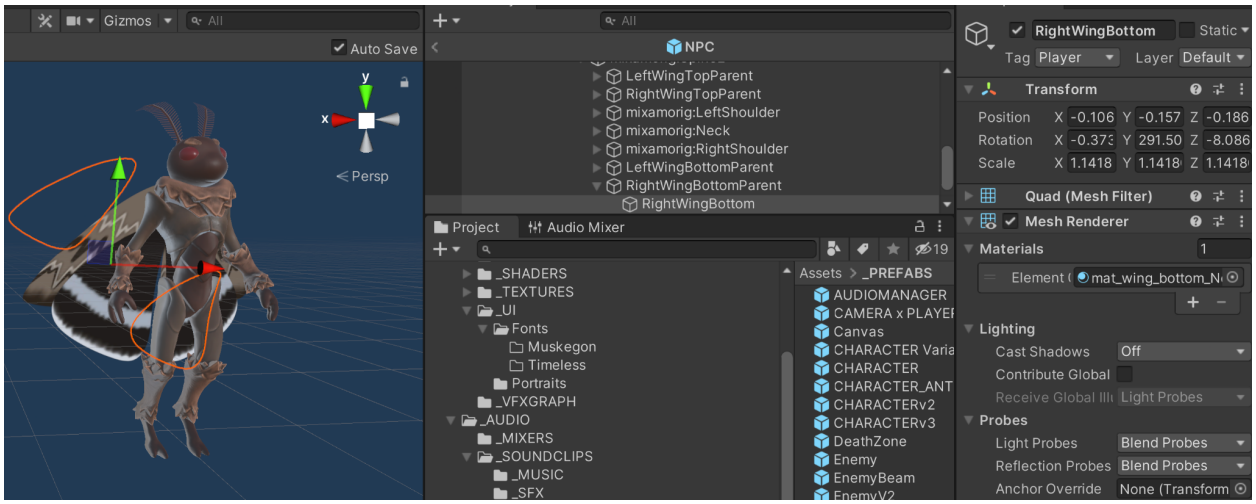


Figure 6.2.1b: The moth wings each had the 'Player' tag.

The culprit turned out to be the moth wings! The enemies searched for a 'Player' tag, and the NPC wings had that tag as well. We tracked down this bug by using a 'DebugtoGUI' script, which prints errors to the scene. Player death was not working, because the wings were being treated as the player instead of the real one. To fix these problems, objects made a more direct reference to the player, or used Unity's FindObjectofType instead of FindObjectWithTag so that no wings would be selected.

Once this main bug was fixed, the team moved onto overhauling the camera, so that it would follow the player with reducing clipping. For inspiration, we opened up third person games like *The Witcher 3* to compare cameras. A hybrid solution where both the camera direction changes player direction, and player direction changes the camera direction. This kept the camera behind the player, and prevented it from clipping through walls. Next, a huge amount of responsive Mixamo animations and sound effects were implemented so that operating Mairu felt more real and engaging.

To deal with tutorial parts and story beats, a player notification system was added, and the dialogue and UI visuals were overhauled. During this development work, our lead artist created huge rock-like structures in Zbrush, and arranged and set up the Alphafest scene to make the play experience feel more like a cave and less like a void.

Figure 6.2.1b: From Dark Void to Improved Cave

6.2.2: At the Event

To set up, the team brought four devices, one from each member, at a designated booth to showcase at. Two of these devices contained a build of the application, allowing for two playtesters at a time. The other two devices held a survey, shown in Appendix E, for the playtesters to fill out after completing the demo. Additionally, the team was able to assess live reactions and think-aloud commentary as useful feedback for improvements to the game, providing an additional level of natural criticism as we saw how the players came to their conclusions in the survey. After each playtester completed the demo and its respective survey, the team sanitized the devices with disinfectant wipes.

6.2.3: Results & Findings

Sixteen people playtested and filled out our survey for *Moonsighted* at Alphafest. The full results are shown in Appendix F. At this event, we got a significant amount of valuable feedback both through the form and by watching people play and discussing the game in person.

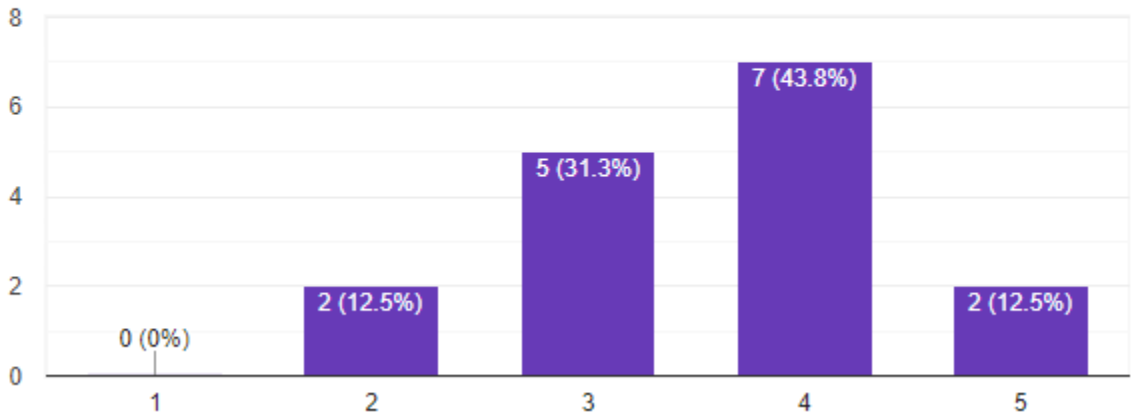


Figure 6.2.3a: Rate the level of control you had over your player (with 1 = No Control, 5 = Very Fine Control)

Fluid movement is crucial for the foundation of our game, and we wanted to ensure that the player's movement matched the game feel. Multiple players mentioned floatiness, some of which considered it a positive while others disliked it. The floatiness contributed to the idea that the player is a moth, but some testers preferred the combat to look more hard-hitting. Additionally, many players discussed how a real moth would be more fluttery and erratic, while the controls seemed more human-like. Overall, only 56% of participants said they had good control over their character, so changes had to be made to improve the ease and intuitiveness of character movement.

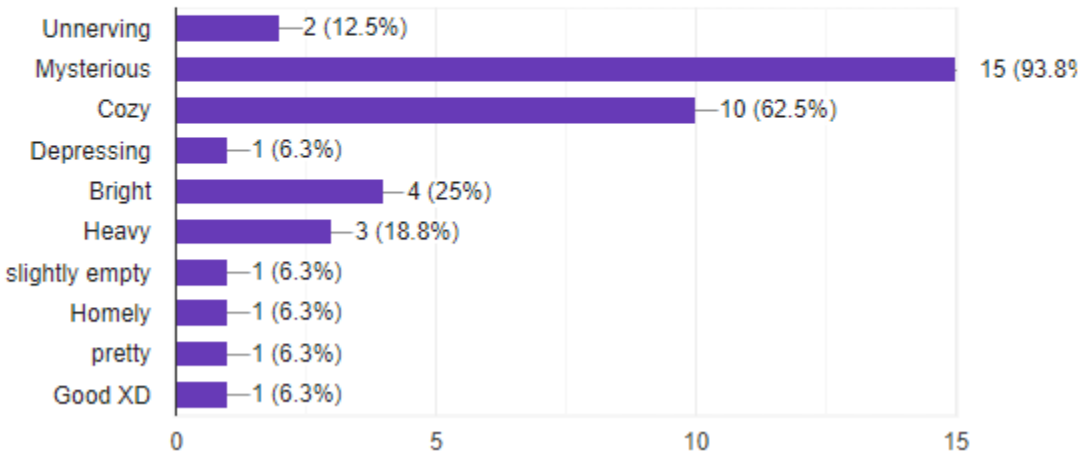


Figure 6.2.3b: How would you describe the ambience of this environment? Check all that apply.

Players started the game in a central village hub. We achieved our goals for the vibes of this area, with 94% of users citing the environment ambience as mysterious, and 63% saying it was cozy. However, some players were confused with their orientation, and did not know where to go or what to do. We attributed this to a lighting issue due to the cavern's stylistic dark and moody lighting being too dark. To address this, we added more point lights to key areas so that the player would more easily know their waypoints. The village also presented confusion with its NPCs, with players not knowing which NPCs to talk to in order to progress the story. This was caused by our initial layout for NPCs, as they were all standing side-by-side where the player starts, leading the team to place those interactable characters in a more natural order in their path to leave the cave, doubling as another natural waypoint and path for the players to orient themselves on where to go and what to do. Players also did not complete the entire conversation an NPC had to offer, leading us to believe that the dialogue UI should remind the player with a symbol of the input to proceed in order to say that there is another sentence.

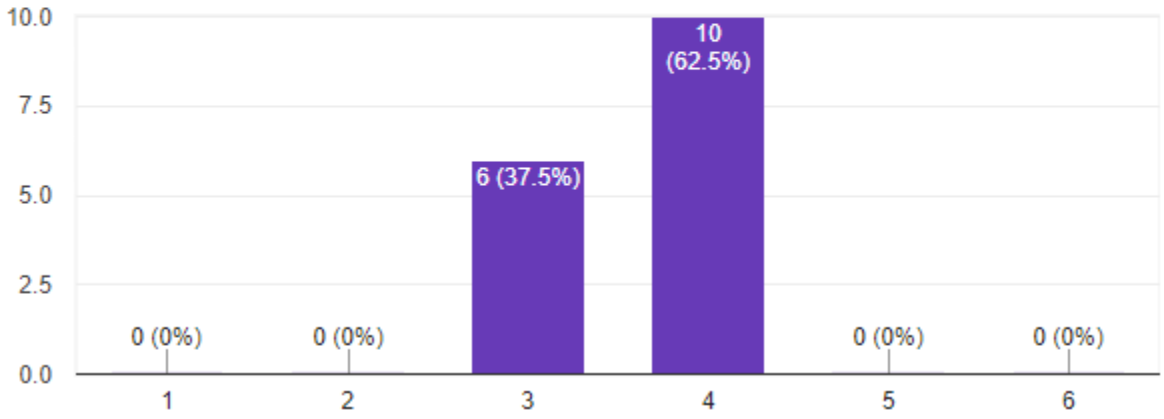


Figure 6.2.3c: Rate the player speed (with 1 = Too Slow, 6 = Too Fast)

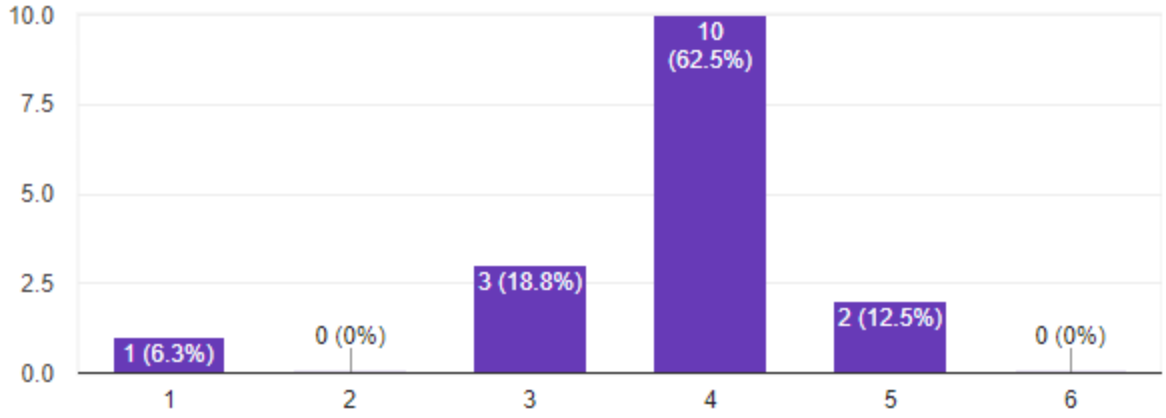


Figure 6.2.3d: Rate the player's aerial dash (with 1 = Too Slow, 6 = Too Fast)

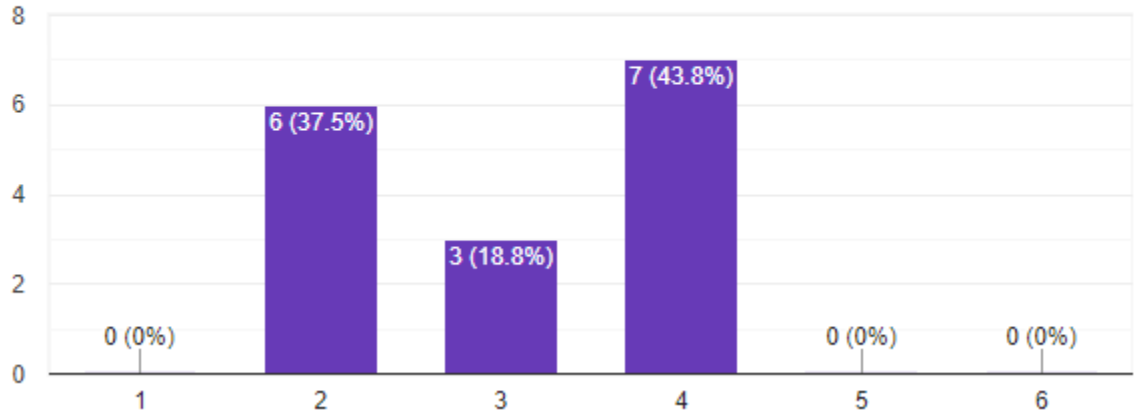


Figure 6.2.3e: Rate the player's attack speed (with 1 = Too Slow, 6 = Too Fast)

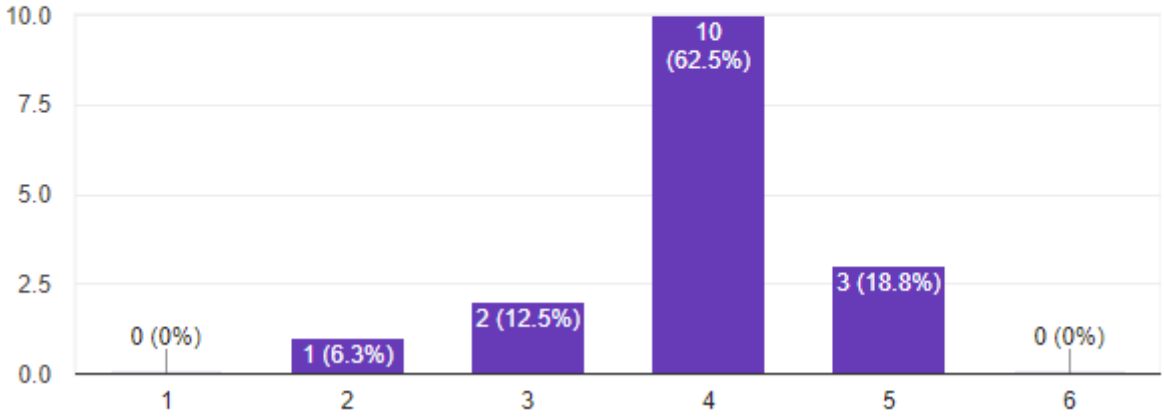


Figure 6.2.3e: Rate the feel of the player's jump (with 1 = Too Stiff, 6 = Too Floaty)

A large aspect of the game being assessed during playtesting was how combat felt for the player. Since our game is following the hack-and-slash paradigm with gameplay, it was crucial that we had the fighting to not only feel good, but also make sense to the player. Throughout several playtests, the players demonstrated excitement and intrigue into the three-hit combo system, especially enjoying the impact of the third and final hit in the combo sequence. However, as they began to use the attacks more within combat, issues began to reveal themselves. Most glaring were the animations. Because of the team's decision to make the player's state machine have built-in modularity for any animation call on the player, it made animation calls to attacks become jarring, as input (often spammed by the players in moments of high action) would cancel certain attack animations and not smoothly transition to the next.

This made combat an eyesore for some playtesters, and was also confusing as the movements were not consistent.

Additionally, the playtesters had a difficult time with visibility during combat, frequently suggesting that the camera should be able to lock onto the nearest enemy, demanding less of the player and making their combat experience smoother. To improve upon this, the team also noted that the camera was too low, significantly reducing player visibility and preventing them from looking around the entire environment. Players also remarked how the camera control was too sensitive for both colliding in walls and via player input, and requested to have the sensitivity be reduced along with behavior for the camera recenter itself over time. To complement this telegraphing of clarifying what was happening in combat, we also received many suggestions to add both visual and audio feedback as to when the enemies and player take damage and collide with one another.

To amend these changes, the team made another animation state machine handler for combat animations in order to maintain the immersion and satisfactory game feel all throughout combat. Additionally, we added both sounds and impact visual effects for when the player strikes enemies and for when they take damage from enemies.

In general, the playtest ran very smoothly and did not have any game-breaking bugs. However, there were a few quality of life issues that we discovered through the playtest. The dialogue box was too high up on some screen aspect ratios, which covered the player and NPCs. Additionally, there were odd situations where the player could stand on sloped surfaces that were not meant for traversing, including some of the sloped walls of the cave. That meant that sometimes if they failed platforming, they could stand on the walls underneath the platforms, which was not what we had intended. There were also issues with walking up stairs because it was not a perfect slope. Despite these problems, the game was completely playable from start to finish, which met our goal for this playtest.

6.3: Final Playtest

6.3.1: Preparation

To prepare for the final playtest, the team had to scale from a solid playtest demo to a complete game, while fixing the quick and essential problems from the Alphafest demo. The main problems the team was able to target was ensuring the dialogue box stayed at the correct height, adding sound and visual effects when players took and dealt damage, and making tooltips more specific to reduce confusion. The camera and combat, although important aspects

of the game, took too much time to redo along with scaling the game, so combat and camera movement remained the same as from alphafast. Expanding the environment, completing the gameplay loop to include three crystal cores, and an ending cutscene were the team's largest tasks.

Completing the gameplay loop required a full script for NPCs and game events to correctly trigger the environmental changes and the ending sequence. We wrote the game event functions into the StoryManager, and wrote the dialogue game event timing into the game's official dialogue. The team also added a variety of UI elements. This included adding a main menu to start, quit, show credits, and set control scheme, adding a settings menu to display controls, switch control schemes, and navigate to main menu, and finally adding additional tooltips UI for explicit game directions. The team added a minor tweak to the dialogue box as well to include a prompt to continue the dialogue interaction.

Environmentally, the game required two more crystal cores, tunnels leading up to them, lighting, and set dressing with interactable or aesthetic objects. The team split up the work to speed this process up. One member was modeling, two members made a tunnel to a crystal core with the modular cave kit, and the last member set dressed the scene with aesthetic objects. The objects used in this final level setup included aesthetic mushrooms and rocks, as well as functional healing crystals, platforming mushrooms, deadly water, and interactable lorestones and NPCs.

6.3.2: Running the Playtest

In order to conduct the playtest, the team sent out a final build of the game with a survey attached via email. The pool was selected from the team’s undergraduate institution, Worcester Polytechnic Institute, for anyone to play. We did this because we wanted to survey as much as possible as quickly as possible for the final iteration of the game, assessing whether or not the team was successful in their end goals of the project. Additionally, the institution’s student body conducts several playtests for varying projects throughout the year, meaning that playtesters match our target demographic of young individuals who are familiar with and play video games.

6.3.3: Results and Findings

Our final playtest had eight playtesters, who experienced the game from start to finish and filled out a survey shown in Appendix G, who's full results are displayed in Appendix H. Between Alphafest and our final playtest, our main goal was to complete the building and laying out of the cave system, allow the whole narrative to be experienced, and an ending cutscene to

demonstrate the outcome of the player's actions. Thus, we aimed to see whether the story and end especially made sense now that the whole game was put together. According to our survey, we achieved our goal in that regard. When asked what happened at the end, 75% of playtesters acknowledged that all the moths had died or that the cave system had collapsed because of the crystal mining. Only one user said that they did not understand the ending. When asked how the ending made them feel, 63% of the players said that they felt sad about the outcome. The others said that they didn't feel much or that it wasn't that impactful. Thus, while most people understood the story, there is more work to be done to make the player feel a connection to the moths and the environment.

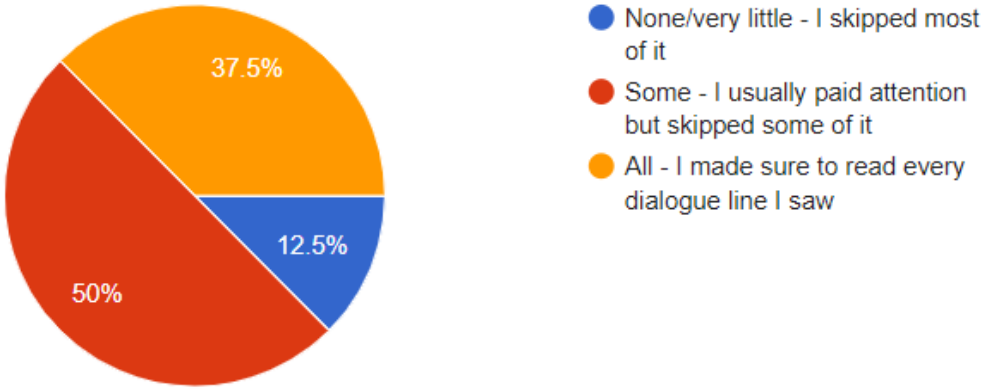


Figure 6.3.3a: How much dialogue did you read?

In addition to just the ending, we wanted to ensure that the player character's decisions made sense within the environment. Our main method of communicating the story was through dialogue from the player himself, other moths, and lorestones placed around the map. When asked how much dialogue they read, 50% of users stated they read some, but skipped some of it, and 38% said they read all of the dialogue they saw (as shown in Figure 6.3.3a). Because the lore was reinforced multiple times through different dialogues, we hoped that reading some of the dialogue was still enough for us to portray the story. Thus, 88% of the users reading at least some of the dialogue was considered a success.

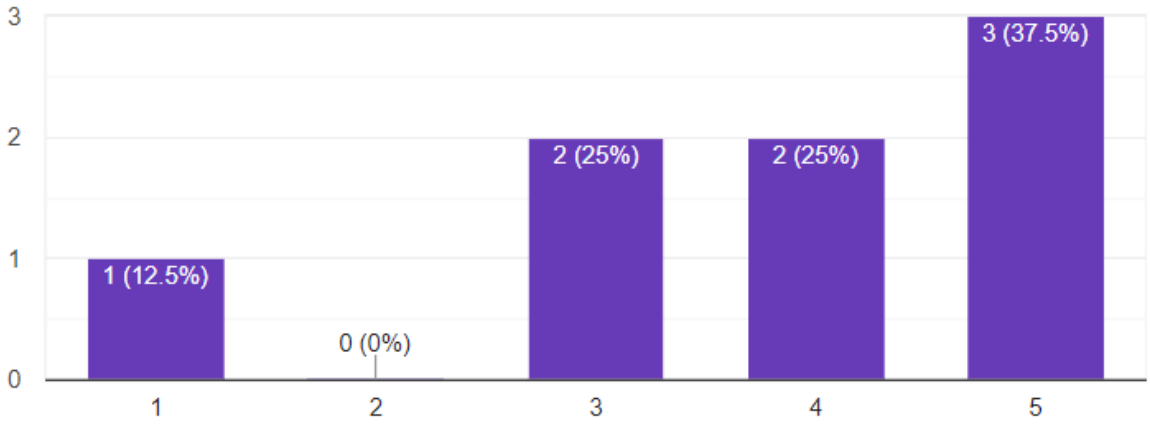


Figure 6.3.3b: How well did you understand your character's goals and motivations while playing (with 1 = Not At All, 5 = Understood Completely)?

This was further examined in our survey when we asked how well the playtesters understood Mairu's (the player character) goals (as shown in Figure 6.3.3b). 63% had a good understanding of their motivations, and only one playtester had no idea. While the outcome of Mairu's choices was clear, the reasons behind his decisions were less obvious, and could be better elaborated in the dialogue to reinforce the idea that his intentions were to help the village, despite the fact that he ended up destroying it.

Figure 6.3.3c: Dialogue Bug

Fortunately, this playtest did not uncover any game-breaking bugs. However, there were a few problems that could be solved for a final polished version of the game. The most apparent bug was in one of the dialogues with Olent, where a typo in the script caused half of his

dialogue to be overlaid with gibberish spoken by a lorestone (as seen in Figure 6.3.3c). Luckily, it was easy to find this typo in the dialogue document and fix it.

7: Conclusion

The entire development of this project started and ended within 7 weeks. Due to this timeline, the team's priorities and development cycle for prototypes, design, art, and programming had to be incredibly iterative and decisive. To do so, the team was dedicated to a project scope we all believed we could achieve. We wanted enough time to create a game with a clear gameplay loop, narrative, and polish to the immersive space in order to demonstrate that shorter gameplay experiences with meaningful play can be created in a feasible amount of time.

7.1: What Went Right

There were many things the team was proud of accomplishing in our incredibly quick development cycle. The team was especially happy with the aesthetics. We were able to create high quality modeled and textured characters and environments, which were complemented with clean animations and moody colorful lighting. We were able to use the power of the Unity game engine's Universal Render Pipeline, Visual Effects Graph, and many other built-in post-processing effects to accomplish such polish for visual effects, particles, volumetric light and water, and emissive materials. User feedback from playtesting also believed that the game's aesthetics were appealing and immersive.

Another notable accomplishment was being able to create a game that had a full narrative arc with a start, middle, and end, directly tied to the core gameplay loop and driven by the player's actions. The team wanted to emphasize the appeal and deceit of an unreliable narrator by having them literally play as one and drive their ultimately deadly decisions. We were pleased to include a final cutscene that stripped the player of their control and power, as their environmental damage was beyond repair, complementing our intent with the medium.

The team was incredibly effective in equal work distribution, respecting work hours and budgeting time to relax for mental wellness, and also successfully managing in a predominantly independently led remote project. To have only a digital barrier and our skill to rely on daily, the feat of completing an entire narrative action game with a core gameplay loop and ending was an accomplishment the team was very happy with.

7.2: What Went Wrong

Due to the uncommon circumstances of the on-going COVID-19 pandemic, short development cycle, and unforeseen medical and mental emergencies occurring within the team,

there were many setbacks and obstacles to completing a cohesive game experience in our 7 week development cycle.

One of the main challenges of the project was creating a character controller that had a smooth “game-feel”. The team wanted the player’s movement and combat to feel fluid and responsive, granting them the maximum amount of control when navigating and fighting throughout the level. As we prototyped and tested out the aforementioned mechanics, we were able to quickly learn the difficulty of mastering gamefeel, and struggled throughout the entirety of the development time trying to get it to feel as smooth as possible for the player. The unresolved gameplay issue existed mostly with the camera and player combat, both of which went hand-in-hand. By having the camera be entirely operated by the player (with no automated assistance for its control), the player felt occasionally claustrophobic and confused when caught in a tense and tight space fighting enemies.

Additionally, the team would have liked to have utilized more Basque culture into the world building and artwork for the game world. However, the team did not have enough time to conduct the proper amount of research without appropriating the culture, it’s people, and aesthetics. Because of this we pivoted our emphasis from Basque cultural aesthetics to Basque mythology, where we felt it was more feasible to properly research and draw inspiration from.

Furthermore, due to the need of some members of the team to isolate due to COVID-19, the team was unable to utilize equipment and engage with certain pipelines, such as using a motion capture suit to record all of the moth and enemy animations for the game. As a result the team had to pivot our art pipeline to gathering and cleaning up mixamo animations. This also impacted and slowed many days of development, as sick members struggled with recovering physically and mentally.

7.3: Future Work

One of the biggest changes needed for the game is improvements to the camera. Even though we understood that the camera was too sensitive and hard to maneuver throughout development, it was a lower priority when compared to actually getting all the content into the game. In the future, the team would ideally make the camera more responsive and smooth for the player to improve the game feel, as well as adjust its angles so it is easier for them to see where they are navigating and fighting.

In addition, the team would have liked to have fleshed out the combat more with an enemy lock-on, making it even easier for players to control the player and clearly see what they

are doing. This feature was requested by playtesters, but was out of scope in the short time frame of our project. While combat was a core mechanic in our game, we did not change it much after it was initially developed, moving onto other key parts such as story and art implementation. Thus, there are many small tweaks and improvements to be made to combat with regards to balancing, increasing difficulty after each core, and game feel. Specifically, the player's projectile ability was difficult to use, and needs a larger effect and perhaps a longer range. Also, while the functionality to change the stats of enemies (such as health and damage) and wave information (such as number of waves and enemies per wave), these features were not actually used, as we ran out of time to do proper testing of different stats.

The team also spent a large amount of time developing the world and lore for the game. By putting in more lore stones, characters, and conversations, we could have more room for commentary on the various viewpoints to the environment, along with filling the world with more immersion for the player to get attached to.

8: References

Barnhill, David. “Key Terms in Environmental Philosophy — Faculty/Staff Sites.” *University of Wisconsin Oshkosh*, Board of Regents of the University of Wisconsin System, 20 Sept. 2010, www.uwosh.edu/facstaff/barnhill/490-docs/thinking/terms.

Bizkaia Talent. “Basque Country, a land of myths and legends.” Bizkaia Talent, 2021, www.bizkaiatalent.eus/en/pais-vasco-te-espera/senas-de-identidad/vasco-tierra-leyendas.

Byrne, Colm. “Luna Moth.” *Easy Science for Kids*, Orion Systems, https://easyscienceforkids.com/luna-moth/. Accessed 13 Dec. 2021.

“DOTween - Get Started.” DOTween, 2021, dotween.demigiant.com/getstarted.php.

Harari, Yuval. Sapiens: *A Brief History of Humankind*. Reprint, Harper Perennial; Reprint edition, 2018.

Kavex. “GitHub - Kavex/GlueIT: Simple SpriteSheet Tool.” GitHub, 2016, github.com/Kavex/GlueIT.

Mondoloni, Elodie. “EQOO Concept Art.” *Twitter*, Twitter, 8 Nov. 2020, https://twitter.com/elomondoloni/status/1325421526796165120?s=12. Accessed 13 Dec. 2021.

“Name Meaning Iker | Baby Names Meaning | Kidadl.” Kidadl, 2021, kidadl.com/baby-names/meaning-of/iker.

North American Basque Organizations, Inc. “Mythology and Legends.” *North American Basque Organizations*, North American Basque Organizations, Inc., 2013, nabasque.eus/mythology.html.

Pontypants. “Cave System with Refractive Blue Crystals.” *Sunday Sundae*, Sunday Sundae, 18 Aug. 2018, https://sundaysundae.co/how-to-make-low-poly-look-good/. Accessed 13 Dec. 2021.

Spiliakos, Alexandra. “Tragedy of the Commons: What It Is & 5 Examples | HBS Online.” *Business Insights - Blog*, President & Fellows of Harvard College, 6 Feb. 2019, online.hbs.edu/blog/post/tragedy-of-the-commons-impact-on-sustainability-issues.

“Xelu’s FREE Controller Prompts.” Those Awesome Guys, 2021, thoseawesomeguys.com/prompts.

9: Appendices

Appendix A. Dialogue Scripts

Dialogue Scripts is the content of the game's official dialogue script, which lives as a text file in the project. This file uses a custom made language for string parsing, and it contains all dialogue information in *Moonsighted*. It is parsed to separate individual scenes and scene titles using semicolons. Each scene's content is separated by a square bracket, "]". Note the speaker tags within a scene are preceded with "//", and a game event is preceded by "//:". Each section in bold is added for organizational purposes only. In game, only the scene title and the content underneath it matters. See more details about code parsing in section 4.5 of the main report.

;Core 0;

;C.0.ODEI;

//Odei]

Ah, there's my favorite miner! I'm still counting all the ilarka shards from your last haul.]

Thanks to your work, we'll be able to make the whole town moonsighted in no time!]

Maybe then some of these freeloaders will be strong enough to actually help us out. . .]

//Mairu]

Glad to be of help.]

//Odei]

Hah, that's the spirit!]

Speaking of which, I've got a lead on a new ilarka deposit near town, meaning I got another job for you.]

Head out through the main cave, look for a large ilarka core, and bring it back here.]

//Mairu]

Sure thing, boss.]

//Odei]

Before you go, why don't you go pay Ila a visit over at her shrine first?]

Despite all of her ravings about how these crystals are killin' us, her healing powers are a great way to prepare for a job.]

Can't stand it, but that's why I got you to take the beating instead! Hah! Alternatively you can destroy smaller ilarka crystals for a little boost of health as well.]

;C.0.ODEI+;

//Odei]
You find that ilarka core yet?]
//Mairu]
Still working on it, boss.]
//Odei]
Alright, well, let's get a move-on. If you're having a tough time, try stopping by Tilak's shop. He might have something that could help.]

;C.0.ILA;
//Ila]
Welcome Mairu. How goes your harvesting of those blasted crystals?]
//Mairu]
Quite well, actually. The more ilarka we collect, the closer we are to finally escaping this place.]
//Ila]
The Teluna will always be there, we will ascend it and find the moon in time. We don't need to rush it with the moonsight.]
Our ancestors had patience, and so can we.]
//Mairu]
Well, you're moonsighted too, Ila. What's not to like about this power?]
//Ila]
A blessing for you, but a curse for me. I'm glad you're happy with it.]
I take it you're preparing for another expedition?]
//Mairu]
Right you are. I'd appreciate a little patching up before heading out.]
//Ila]
Of course. . .]
//:FullHealCharacter]
Watch yourself out there, it's dangerous past the walls of our Etsuna.]
//Mairu]
Right. Take care Ila.]

;C.0.TILAK;
//Tilak]

Eyy Mairu! How's it hangin'?]
//Mairu]
Hey, not too bad Tilak! What're you cooking up today?]
//Tilak]
Oh, just the usual onddargi stew. Those little 'shrooms pack quite the punch of flavor!]
Although. . . I'm lookin' to make another batch of ilarka candy. More and more folks are looking to become moonsighted, and I gotta satisfy 'em.]
Speaking of which, are you and Odei bringin' in a new shipment of ilarka soon? I can't meet those demands without those crystals.]
//Mairu]
Yep, I'm just about to head out on another outing.]
//Tilak]
Alright! Just bring those crystals back to me and I can get cooking. I'll even give ya a lil' extra to boost your moonsight powers!]
Actually, ya know what? I've got a bit of candy left over, why don't you take this now as an advanced thank-you?]
//:ActivatePlayerDash]
//Mairu]
Thanks, Tilak. I'll be back with that ilarka before you know it.]

;C.0.OLENT;
//Olent]
Ah, hello Mairu. That was quite the entrance there.]
//Mairu]
Olent? The town hasn't seen you in months!]
//Olent]
Yes, I do regret not visiting more. But I've found a peaceful life out here among the flora and fauna.]
You might learn to enjoy it too, if you spent more time enjoying this place than you do destroying it.]
//Mairu]
I can't rest easy here in these caves knowing that our true home lies up there, on the surface, under the moonlight.]
You of all people should know that.]

//Olent]
I may be old, but I've never seen the surface. The Etsuna is all I've known, same as you.]
And if I've learned anything, it's that we're better off trying to celebrate this beautiful place.]
Not risking our lives trying to find a home we've never known!]
//Mairu]
Alright, you can keep wandering around picking onddargi, I'll be looking for ilarka and actually getting things done.]
//Olent]
As you wish. If you must go on, please do mind the Uskerra. They're peaceful creatures, but the ilarka are their food source.]
If you keep consuming it, they're not going to be happy. . .]

;C.0.NPC1;
//Villager]
One of Odei's moonsighted, eh? You're lucky to have gotten it so soon.]

;C.0.NPC2;
//Villager]
Ila likes to pretend that she understands the common moth's issues.]
//Mairu]
She's pretending?]
//Villager]
Of course! Her father was the village leader, and she inherited his moonsight.]
She was born powerful. Odei's right. She'll never truly know the struggle.]

;C.0.NPC3;
//Villager]
I love staring at the slow sway of the onddargi. What a peaceful glow they give off.]
//Mairu]
Not as good as the moon.]
//Villager] How can you know when you've never seen it?]
//Mairu]
I just know.]

;C.0.NPC4;
//Villager] I remember your gentle eyes, before you were moonsighted. I don't know what I'm looking at anymore.]
//Mairu] I'm the same as always. Just stronger. You should get moonsighted too.]

;Core 1;

Themes, llarka doubt, ilarka craze, sides becoming more polarized, npc comments on mushrooms dimming, seasonal sadness type beat, start of an artificial winter

;C.1.ODEI;
//Odei] Keep that ilarka coming! This is what I like to see!]
//Mairu] When can we explore the Teluna? I think we're ready now.]
//Odei] Not so fast. We need the entire town moonsighted first.]
To feel the power of the ilarka coursing through my veins. . . I cannot wait.]
//Mairu] Why not become moonsighted now?]
//Odei] No no no, Mairu. That would be a bad image. Then I'd be no better than Ila.]
I need these non-moonsighted moths knowing I'm on their side, right?]

;C.1.TILAK;
//Tilak] Look at you, the hero of Etsuna! Another crystal for Mairu. Let me give it a look.]
Oh, wow. This one's powerful, lots of raw ilarka energy. Let me cook it up.]
//:ScreenFade]
Alright, deals a deal. Some ilarka for the miner, and the rest for the village.]
I don't mean to brag but this looks to be some of my finest work. Eat up!]
//:ActivatePlayerProjectile]
//Mairu] Delicious. Thanks, Tilak.]

;C.1.ILA;
//Ila] Another crystal destroyed, hmm?]
//Mairu] Sure thing. We're making progress, it's a good thing Ila.]
//Ila] I have an uneasy feeling about all this, Mairu. It's almost as if I hear the onddargi's whispers fading.]

//Mairu] Change is always uncertain. Sometimes you have to take a chance.]
Without moonsight, we won't make it up the Teluna. Our true home, Ila, is up there. This is all temporary.]
//Ila] This, the Etsuna, is our home, Mairu, and I want to make sure it survives.]
//Mairu] And I want a future for us. We don't belong down here, and the ilarka are our way out.]
This is our best chance to leave. It's impossible without the moonsight.]
Not that you would know. You've never had to live without it.]
//:FullHealCharacter]
//Ila] You bear the gift of moonsight, yet you're still blinded by arrogance.]
I'll always be here to heal you, but you've clearly made up your mind. Tread with care.]

;C.1.OLENT;
//Olent] You're quite the explorer, aren't you?]
//Mairu] And the Etsuna's best miner.]
//Olent] Right, the best at destroying this world.]
//Mairu] Why are you even out here?]
//Olent] Have you heard the story of how the town's leadership came to be?]
//Mairu] I was young, but I know of it. Odei promised progress and provided for the people.]
He cared about all of us and we made him our leader.]
//Olent] That's what they tell, yes. But I always believed Ila should have taken the village on, an unpopular opinion.]
//Mairu] Ila is not a leader. She never even asked for power.]
//Olent] And that is why she would be the best fit. Odei never cared for our people.]
The only thing he cares about is power and control. It is a pity you cannot see that.]
//Mairu] Someone had to do something. He's bringing the moonsight to everyone, so we can finally leave this place.
//Olent] And why must we be moonsighted to do so?]
I am an old moth, Mairu. My time to act has passed. I wish you wisdom in your path.]
It is easy to be swept up in the excitement of change. I pray it does not consume you.]
As always, mind the Uskerra.]

;C.1.NPC1;
//Villager]

When I was small, I felt the onddargi give such a powerful glow. It seems they've dimmed.]
//Mairu]
Could be your eyes. You're getting old, after all.]
//Villager] Perhaps. . .]

;C.1.NPC2;
//Villager] My brother tried exploring up the Teluna not too long ago. He wasn't moonsighted, and he almost died.]
It is downright wrong to have any regular moth to do that. We all need moonsight!]
The fact that some people resist such an obvious change gets my blood boiling.]
//Mairu] It is our only option to progress.]

;C.1.NPC3;
//Villager] I don't know why, I've been feeling sad. Our home has darkened.]
//Mairu] The underground is not meant for us. But we will move forward soon.]
//Villager] I'm not sure I want to. The Etsuna is my home. But it cries today. . .]

;Core 2;

;C.2.ODEI;
//Odei] Time is drawing near, Mairu. With these crystals, the preparations to convert the entire village are being made.]
Then we will begin to scale the Teluna. The birth of a new era!]
//Mairu] An exciting time to be alive.]
//Odei] Right you are! A birth of new equality, power and possibilities for all moths.]
Don't let up, we're almost there.]
//Mairu] What of the dimming onddargi? I've heard talk that we could run out of food.]
//Odei] I'm not worried! We'll be out of here soon enough, and have plenty to eat under the moonlight!]
Time to finish the job and move on! There's one ilarka core left, go ahead and do what you do best.]
//Mairu] Right, of course.]

;C.2.TILAK;

//Tilak] Notice anything different?]

//Mairu] Tilak! This is amazing, look at you! Moonsighted! How is it?]

//Tilak] It's amazing! I feel alive, like a brand new moth. All thanks to you and your last find.]

And it seems like you got more for me. I never thought things could go this well.]

//Mairu] Doesn't seem like everyone's so happy.]

//Tilak] You mean Ila and the others? They're just a bit unnerved at the changes.]

Maybe Ila's feeling less powerful with all the new moonsighted moths, hah!]

Anyway, let me take a look at your new find . . .]

She's a beauty, probably your best find yet. This one might knock you out, you ready?]

//Mairu] As I'll ever be. Let's see it.]

//:ScreenFade]

//Tilak] Here's your share. Careful with that one.]

//:ActivatePlayerSlam]

//Mairu] Woah. Ok, ok. . .]

//:ScreenFade]

//Tilak] Wow, you were out for a while. But you're absolutely glowing! Go test out your new powers.]

;C.2.ILA;

//Mairu] Hey Ila, I -]

//:FullHealCharacter]

//Ila] I did my job. You're free to go.]

//Mairu] Everyone's going to be moonsighted soon, Ila. At least pretend to be happy about it.]

//Ila] I'll share the village's happiness, but my heart mourns our home.]

//Mairu] I know things have been tense between us, but . . .]

I hope we can be friends again in the new world, on the surface.]

//Ila] At this point, I truly wish for things go as planned, Mairu.]

There's no turning back now. I hope it's all worth it.]

;C.2.OLENT;

//Olent] You cannot deny that a dark shroud has surrounded the Etsuna.]
//Mairu] I don't. That is why we must finish the job and leave.]
//Olent] Your job is one of destruction. That is why the ondarggi cry out. The backbone of the caves, the Etsuna, our home.]
//Mairu] We will finish what we started, and the Etsuna will be a memory of the past once we reach the moon. It's time to return to the surface, Olent. We are going to leave.]
//Olent] No, Mairu. This is my home, and yours too. I will stay here until the bitter end.]
//Mairu] So be it. Starve alone, Olent. You were always so stubborn.]
//Olent] That makes two of us, doesn't it?]

;C.2.NPC1;
//Villager] Time to get out of here. It's been far too long we've been stuck in this hole.]
//Mairu] I agree. I've always wondered if we have family on the surface that never fell down here.]
//Villager] They would be living the dream.]
//Mairu] Seems like we'll be joining them soon!]
//Villager] I can't wait!]

;C.2.NPC2;
//Villager] The onddargi mushrooms look so gray. Our home is being destroyed. And you're all blind to it. I'm not joining this movement.]
//Mairu] You're wrong. We're moving to better things. If you don't become moonsighted you'll just be left behind.]
//Villager] I'll take my chances.]

;C.2.NPC3;
//Villager] True equality is on the horizon! We will all be moonsighted soon!]

;Miscellaneous;

;OBSTACLE1;
//]<smallcaps> Beyond these tunnels are long stretches and cliffs, Deadly pools of water that will drown the strongest. The gift of flight may aid in your travels.]

;OBSTACLE2;
//]<smallcaps> A rock wall lays ahead, impassable by many. A high velocity projectile may break it open.]

;OBSTACLE3;
//]<smallcaps> A rock wall lays ahead, strong and firm. A mighty burst may clear the way.]

;LSTONE1;
//] <smallcaps> Boundless Years Past, Here Descended The First Moths, The <B>Jondengoa</B> </smallcaps>]
<smallcaps> Lost to their mother Moon, a home was made from the depths of the Etsuna</smallcaps>]
<smallcaps> They gazed up through the Teluna, pining for the light they once knew</smallcaps>]
<smallcaps> Their spawn, the Jonditsa, would never know its beauty</smallcaps>]

;LSTONE2;
//]<smallcaps> The prismatic wonder of the ilarka, the window to a lost world</smallcaps>]
<smallcaps> The Jonditsa held the untold majesty of their ancestors in their clutches</smallcaps>]
<smallcaps> Providing the sliver of hope for regaining what was never had</smallcaps>]
<smallcaps> Would it ever have been theirs to take</smallcaps>]

;JOURNAL1;
//Journal of Iker, the Explorer] This expedition has been a great success! I've learned essential details about the local flora and fauna.]

It seems both the reptilian Uskerra and the fungal onddargi rely on ilarka crystals to sustain themselves.]
The onddargi derive their beautiful glow from them, and the Uskerra use them as their primary food source.]
Truly remarkable, that this one crystal can sustain an entire ecosystem!]

;JOURNAL2;
//Journal of Iker, the Explorer] Studying the structure of these giant ilarka cores has revealed some fascinating results about this place!]
Not only do these crystals support the local wildlife, but they also seem to be structurally integral to this cave system.]
These magnificent, lustrous spires are responsible for holding the roof over heads. Who would've thought?]

Appendix B. Informed Consent Agreement Form

Below is the informed consent agreement that each playtester read and signed before beginning the playtest.

Investigator: Farley Chery

Contact Information: fjchery@wpi.edu, +1 (781) 718 9559

Title of Research Study: Moonsighted

Introduction: You are being asked to participate in a research study for Moonsighted. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation. This project is for Moonsighted, an action based video game that communicates the importance of ecosystems and our natural environment. The final product is an application that runs on a PC.

Purpose of study: The Googoo Berry Dilemma MQP is aimed to create a PC third-person adventure about cultural and political strife in the face of technology and progress amongst moon-worshipping moths trapped in a cavern. By having the user playtest the game, they will find bugs for the developers to fix to ensure that future iterations of the gameplay experience can be as functional as possible. Furthermore, the user will provide feedback on game design decisions and provide the developers with insight on how fun and playable the gameplay experience was, in order to make future gameplay experiences more enjoyable for future audiences.

Procedures to be followed: After you sign the Informed Consent Agreement (below), you will be instructed to download the game software provided via the following link (https://drive.google.com/drive/folders/14wJdjvFpSlmEQTk_ilhKxoRX4L-rK_cO?usp=sharing). You will need to be on a desktop with the Windows operating system in order to use the application. This will direct you to a Google Drive folder containing a zipped executable of the software and a text file explaining how to use the application that you must download. A Google account is not required to download the application and instructions. If prompted with a warning stating that you are unable to scan the file for viruses, click “DOWNLOAD ANYWAY”. Once downloaded, you are able to close out of the internet tab as it is downloaded on your personal

computer to which thereafter you can go into your files' downloads and extract the zipped download. Once extracted, enter the files and double click the executable file titled "Moonsighted.exe". Upon opening the game, you will playtest for a timed 15 minutes. After completing the game, you will be directed to another Google Form for a post evaluation survey in order to characterize aspects of your subjective experience and to solicit suggestions for improving the game experience. If the participant wishes to remove the application, they can find the folder destination of where they downloaded the application and extracted it to, select the file, and delete it.

Risks to study participants: There are no foreseeable risks associated with this research study.

Benefits to research participants and others: You will have an opportunity to enjoy and comment on a new application under active development, and assist the developers in making a more player-friendly gameplay experience for future players in future iterations of the game. Additionally, you will be able to keep the software installed if you want to.

Record keeping and confidentiality: Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to confidential data that identify you by name. Any publication or presentation of the data will not identify you. There will be no data collected from files downloaded.

Compensation or treatment in the event of injury: There is no foreseeable risk of injury associated with this research study. Nevertheless, you do not give up any of your legal rights by signing this statement.

For more information about this research or about the rights of research participants, **or in case of research-related injury**, **contact the Investigator listed at the top of this form**. You may also contact the investigators Dylan Valev (davalev@wpi.edu), Jasmine Duerk (jlduerk@wpi.edu), Charlie Baldwin (cgbaldwin@wpi.edu), or Kaamil Lokhandwala (kslokhandwala@wpi.edu). Alternatively, you can contact IRB Manager (Ruth McKeogh, Tel. 508 831- 6699, Email: irb@wpi.edu) and the Human Protection Administrator (Gabriel Johnson, Tel. 508-831-4989, Email: gjohnson@wpi.edu).

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

Participant Signature: _____ Date: _____

Participant Name (please print): _____

Researcher Signature: _____ Date: _____

Appendix C. Internal Playtest Survey

Below are the internal playtest survey questions, the first of the playtests.

Does player movement feel representative of mothlike movement? *

Mark only one oval.

	1	2	3	4	5	6	
Doesn't feel like a moth at all	◯	◯	◯	◯	◯	◯	Feels very mothlike

What aspects of player movement made it feel or not feel mothlike? *

_____

Rate the level of control you felt over the player character *

Mark only one oval.

	1	2	3	4	5	
Very little control, not intuitive/easy to grasp	◯	◯	◯	◯	◯	Very fine level of control, easy to use

What did you dislike about the controls? *

_____

_____

_____

_____

_____

What did you like about the controls? *

_____

_____

_____

_____

_____

Rate the player speed *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the player's aerial dash *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the player's attack speed *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the feel of the player's jump *

Mark only one oval.

	1	2	3	4	5	6	
Too stiff	◯	◯	◯	◯	◯	◯	Too floaty

Did you find yourself running or dashing more? Which felt more fun? *

How would you describe the ambience of this environment? Check all that apply. *

Check all that apply.

- [ ] Unnerving
- [ ] Mysterious
- [ ] Cozy
- [ ] Depressing
- [ ] Bright
- [ ] Heavy
- [ ] Other: _____

Do you feel a powerful weight to the player's three-hit-combo? If so, why? If not, why not? *

_____

What potential player abilities would you want to see added to the game? Please select whichever options seem fun to you, as well as any additional ideas you may have. *

Check all that apply.

- [ ] Multiple dashes
- [ ] Damaging dashes
- [ ] Earthbending
- [ ] Shooting a energy blast projectile
- [ ] Enhanced blades
- [ ] Aerial dodge maneuver (loop-de-loop)
- [ ] Other: _____

Was the dialogue engaging, or did you find yourself skipping it?

_____

_____

_____

_____

_____

Rate the difficulty of fighting the enemies.

Mark only one oval.

	1	2	3	4	5	
Easy	◯	◯	◯	◯	◯	Hard

What was the most engaging part of your playtest experience?

Mark only one oval.

- ◯ Environment (Exploration)
- ◯ Dialogue and Story
- ◯ Platforming
- ◯ Combat
- ◯ None of them were particularly engaging

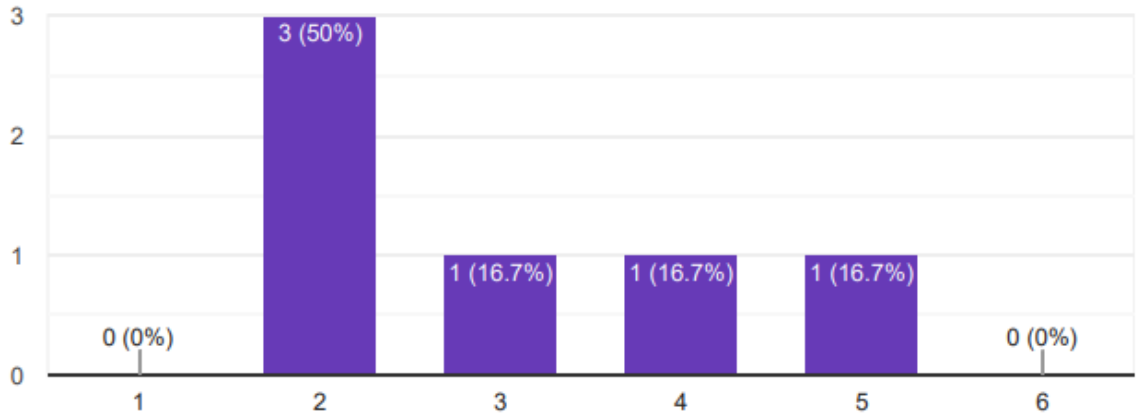
Note any gameplay difficulties and/or bugs that affected your playing experience.

Appendix D. Internal Playtest Survey Results

Here are the results of the first playtest, including graphs and short responses.

Does player movement feel representative of mothlike movement?

6 responses



What aspects of player movement made it feel or not feel mothlike?

6 responses

I imagine the mothlike movement as more fluttery and airborne. Being low to the ground just reminded me of a human.

light on feet

the flight and ease of turning made me feel the sporadic flight pattern of a month

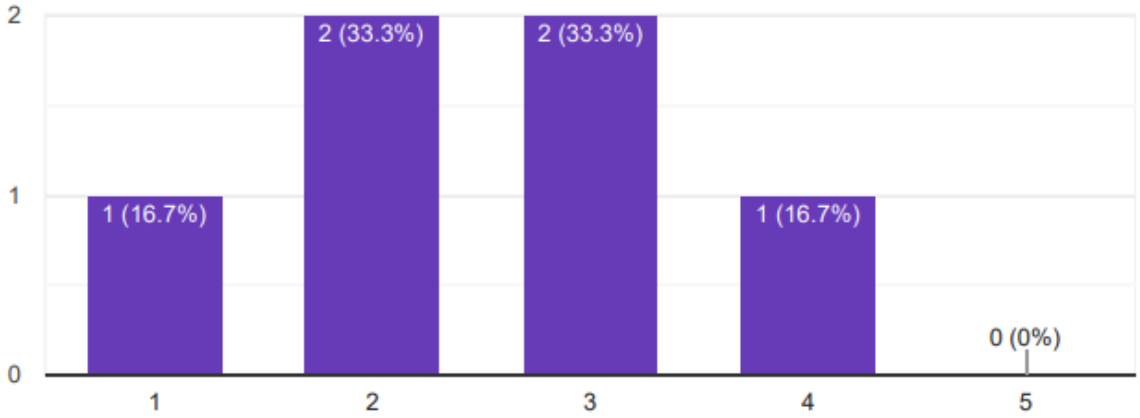
Moths generally move pretty erratically, tho i don't think that translates well into an actual game unless it was some form of controlled chaos mechanic or just erratic animations.

I am a moth man tho so idk

The camera, the glide is clunky

Rate the level of control you felt over the player character

6 responses



What did you dislike about the controls?

6 responses

Most of the motion felt controlled, but the dashing felt unsteerable and overly fast.

camera is wonky

nothing

The jump was very unexpectedly high. I could not look down, which made the platforming part a bit difficult. I had the instinct to be zoom with the scroll wheel , but that's prob just a personal thing.

I jump so high I do not see where I am going to land. The camera also wiggles sometimes and is a little hard to control.

again the camera

What did you like about the controls?

6 responses

Not much. They were satisfactory but not enjoyable.

jumping and dashing

The controls were not super easy to use, but I liked that cause I don't think a moth would be very in control of itself.

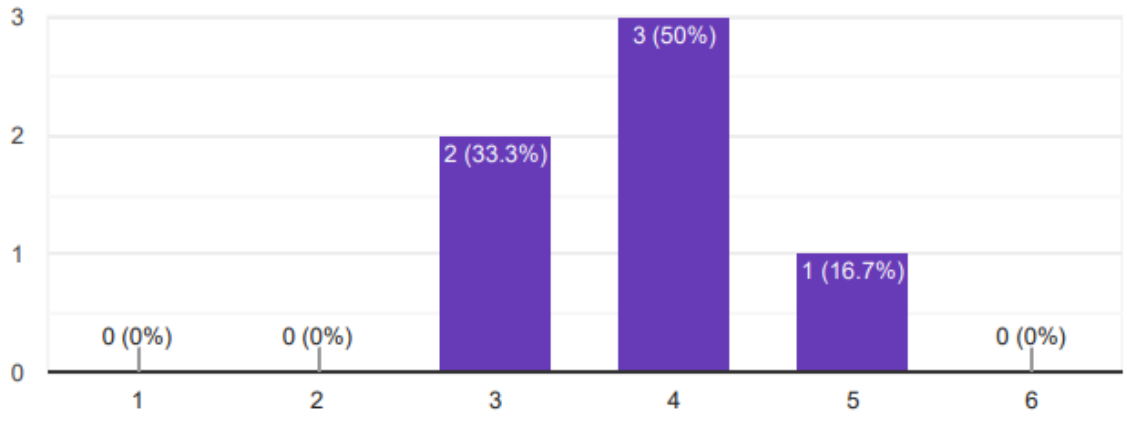
The combat felt fluid, while there wasn't any accompanying hit noises, the combat didn't feel cheap. There were no "Oh i totally hit that fucker" type mishaps.

I like WASD and E and click those are pretty default and self explanatory.

the dash and interact features were on intuitive keys

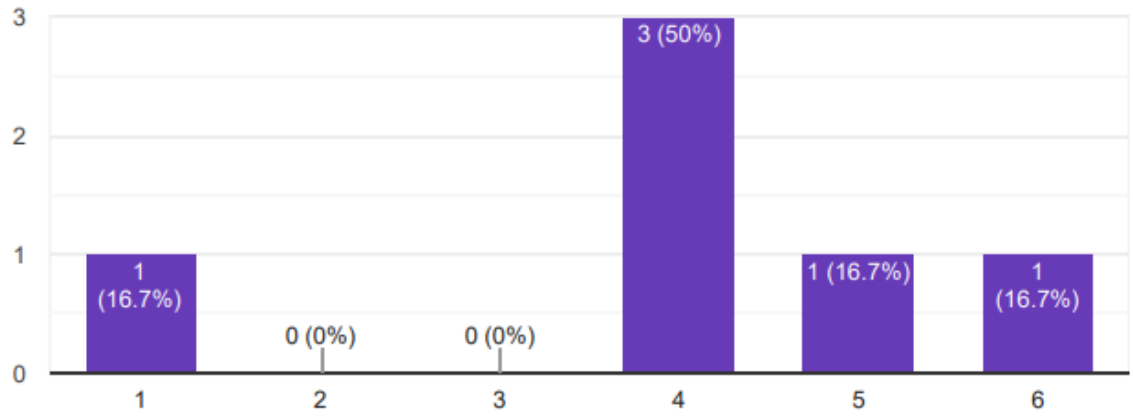
Rate the player speed

6 responses



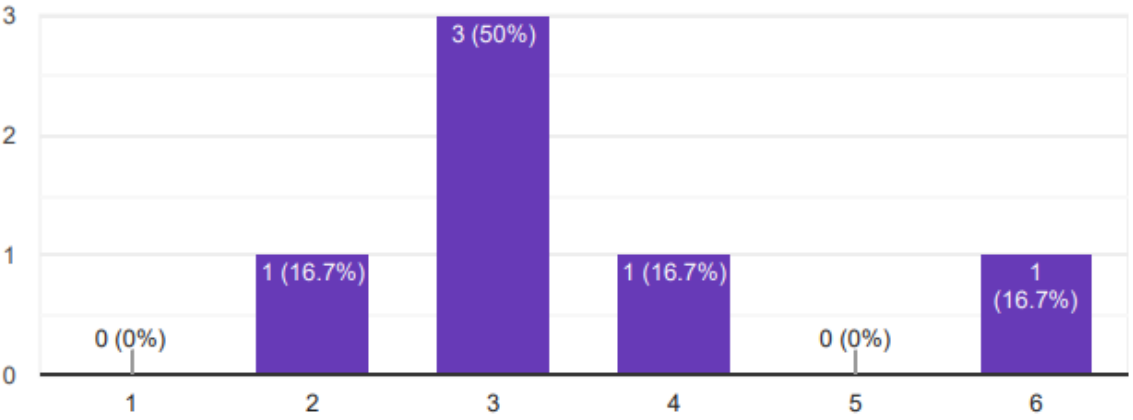
Rate the player's aerial dash

6 responses



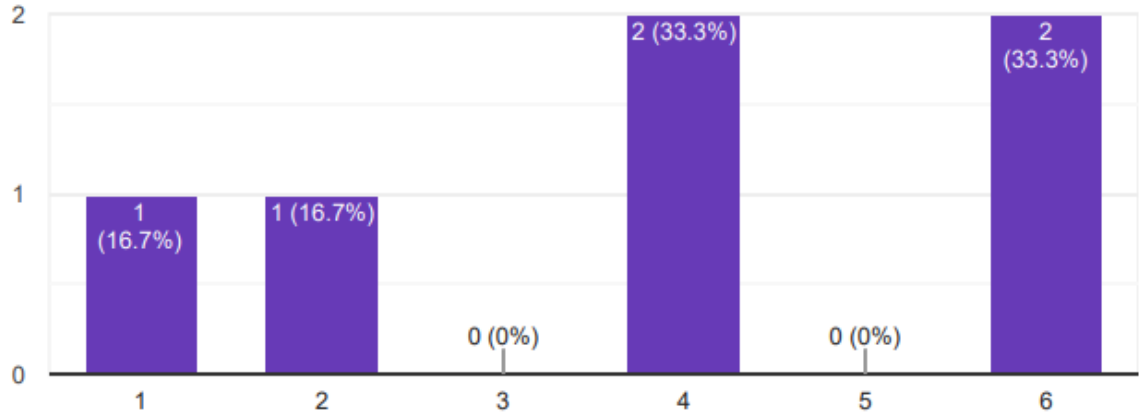
Rate the player's attack speed

6 responses



Rate the feel of the player's jump

6 responses



Did you find yourself running or dashing more? Which felt more fun?

6 responses

Running. Dashing wasn't viable.

dashing was more fun

dashing, it was more fun

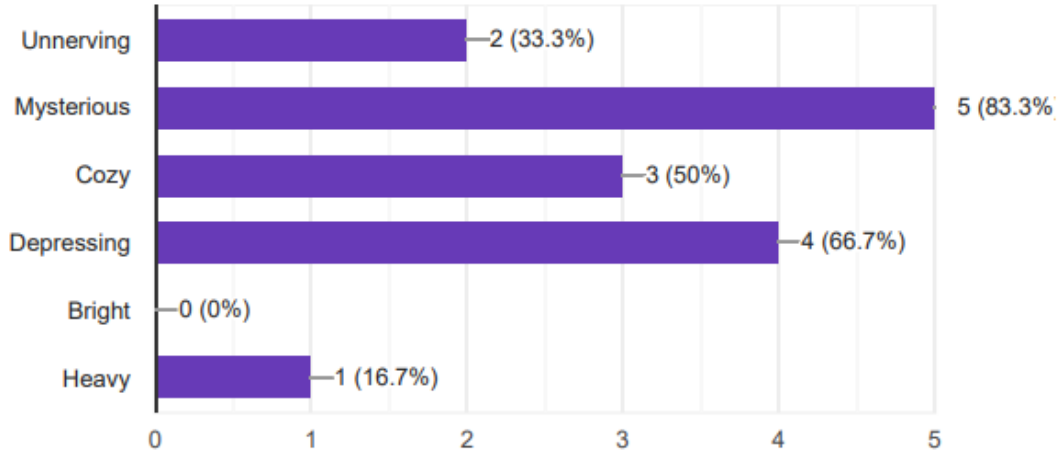
I just ran and jumped, I didn't realize there was an air dash until I was told,

Running, I didn't even know there was an aerial dash until I took this quiz LMAO

running because the dashing felt too unusable

How would you describe the ambience of this environment? Check all that apply.

6 responses



Do you feel a powerful weight to the player's three-hit-combo? If so, why? If not, why not?

6 responses

Visually I can see the weight and power, however the camera perspective was a bit far back in terms of viewing it. Not having sound effects also subtracts.

No, add sfx

I thought that the attacks were a bit hard to aim, so I never achieved a three hit combo

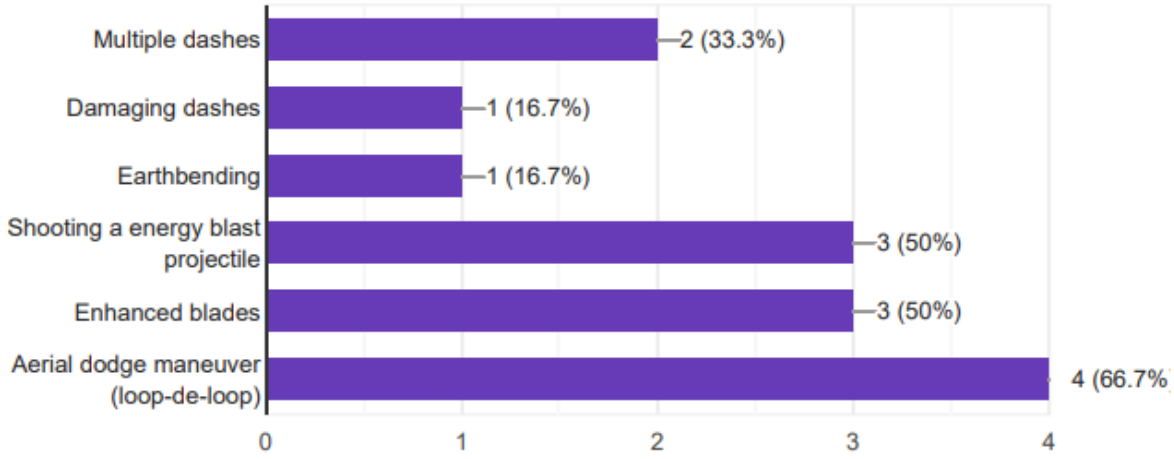
It felt somewhat impactful, audio feedback and necessary juice is probably needed tho.

No. No sound or any other indicator other than sparkles.

no because there was no sound and no indicator as to how far an attack reached or if you hit or not, or what the 3rd attack in the combo did

What potential player abilities would you want to see added to the game? Please select whichever options seem fun to you, as well as any additional ideas you may have.

6 responses



Was the dialogue engaging, or did you find yourself skipping it?

6 responses

The dialogue was interesting, but wording was confusing at times.

Yes, differentiate speakers

the dialog was a bit hard to follow since I had few clues to tell if it was my player talking

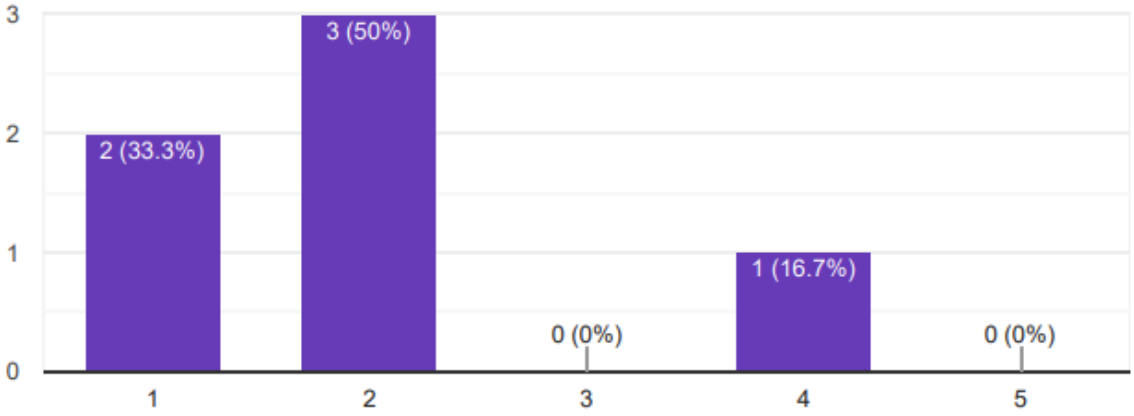
I found my self skipping a lot of it honestly, tho I was intrigued by the guy who called me a fucker.

WHy is it purpel

relatively engaging

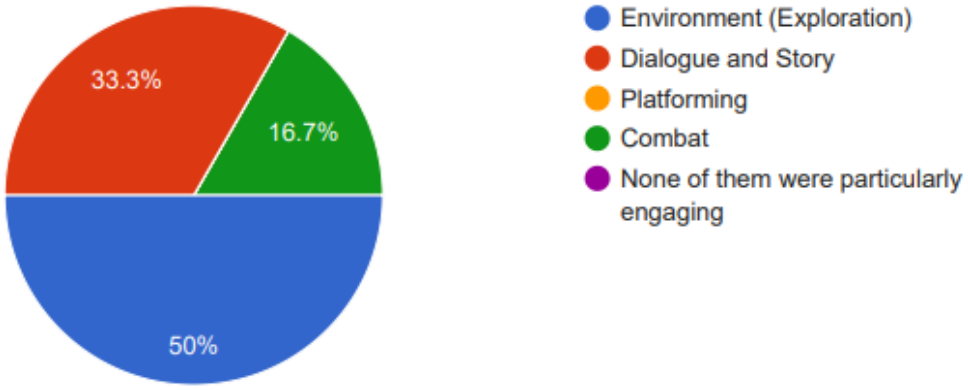
Rate the difficulty of fighting the enemies.

6 responses



What was the most engaging part of your playtest experience?

6 responses



Note any gameplay difficulties and/or bugs that affected your playing experience.

6 responses

Combat was confusing, as enemies were a bit hard to see, and it was difficult to tell when one was killed. I believe I took damage after enemy death by being too close to their bodies, yet I'm unsure as to why.

Fix camera,no bugs

none found

I respawned oddly when killing an enemy and dying at the same time.

Ila bugging fr why I just stand on them tho

camera.

Appendix E. Alphafest Survey

Below are the internal playtest survey questions, the first of the playtests.

Does player movement feel representative of mothlike movement? *

Mark only one oval.

	1	2	3	4	5	6	
Doesn't feel like a moth at all	◯	◯	◯	◯	◯	◯	Feels very mothlike

What aspects of player movement made it feel or not feel mothlike? *

_____

Rate the level of control you felt over the player character *

Mark only one oval.

	1	2	3	4	5	
Very little control, not intuitive/easy to grasp	◯	◯	◯	◯	◯	Very fine level of control, easy to use

What did you dislike about the controls? *

_____

_____

_____

What did you like about the controls? *

How did you feel about the camera? *

Rate the player speed *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the player's aerial dash *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the player's attack speed *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the feel of the player's jump *

Mark only one oval.

	1	2	3	4	5	6	
Too stiff	◯	◯	◯	◯	◯	◯	Too floaty

Did you find yourself running or dashing more? Which felt more fun? *

_____

How would you describe the ambience of this environment? Check all that apply. *

Check all that apply.

- [ ] Unnerving
- [ ] Mysterious
- [ ] Cozy
- [ ] Depressing
- [ ] Bright
- [ ] Heavy

Other: ☐ _____

Do you feel a powerful weight to the player's three-hit-combo? If so, why? If not, why not? *

_____

What potential player abilities would you want to see added to the game? Please select whichever options seem fun to you, as well as any additional ideas you may have. *

Check all that apply.

- [ ] Multiple dashes
- [ ] Damaging dashes
- [ ] Earthbending
- [ ] Shooting a energy blast projectile
- [ ] Enhanced blades
- [ ] Aerial dodge maneuver (loop-de-loop)

Other: ☐ _____

Was the dialogue engaging, or did you find yourself skipping it?

Rate the difficulty of fighting the enemies.

Mark only one oval.

	1	2	3	4	5	
Easy	◯	◯	◯	◯	◯	Hard

What was the most engaging part of your playtest experience?

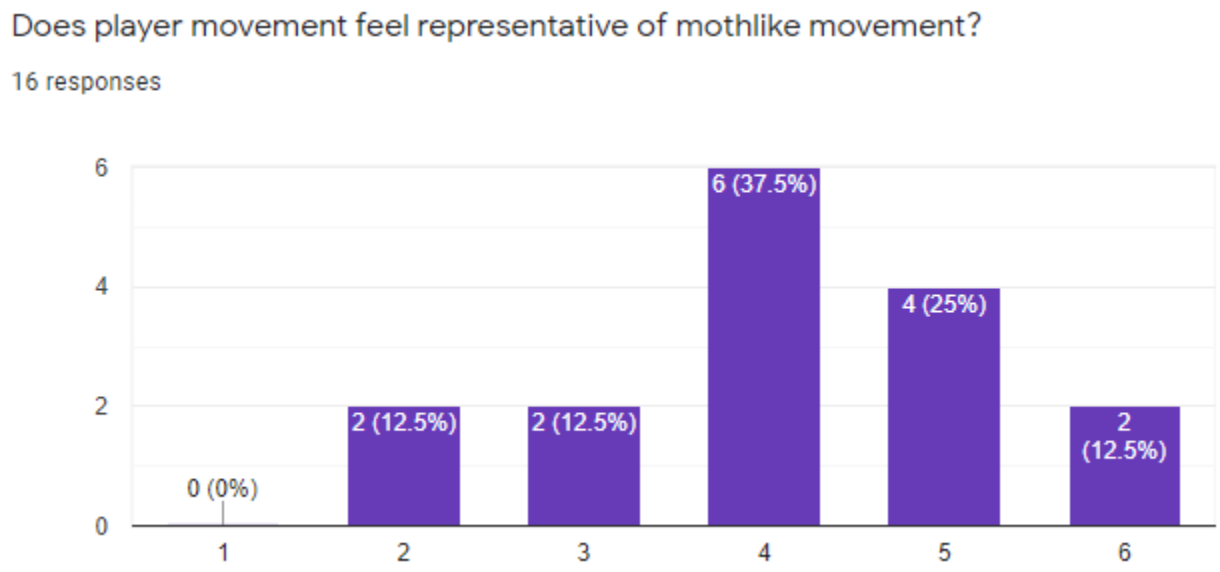
Mark only one oval.

- ◯ Environment (Exploration)
- ◯ Dialogue and Story
- ◯ Platforming
- ◯ Combat
- ◯ None of them were particularly engaging

Note any gameplay difficulties and/or bugs that affected your playing experience.

Appendix F. Alphafest Survey Results

Here are the results of the second playtest, including graphs and short responses.



What aspects of player movement made it feel or not feel mothlike?

16 responses

I liked being able to fly forwards

The flight ability and the slow freefall speed

The design and the platforming

i imagine moths moving in a fluttery impulsive way

From the perspective of a *warrior*-like moth, I think this move set makes sense. Having the wings be a significant part of gliding and shifting around the map feels smooth and intuitive.

the models were very well done and the glowing wings and bright antennae were very cool and mothlike

floaty

It feels natural

the speed of the looking movement was a bit fast

air dash and spontaneous movement felt moth like, as well as the emphasis on light

Moths dont walk or airdash

The glide felt moth like, but only horizontally

it just felt like normal running movement. the flying dash was cool, but a hover or something would emphasize the moth stuff

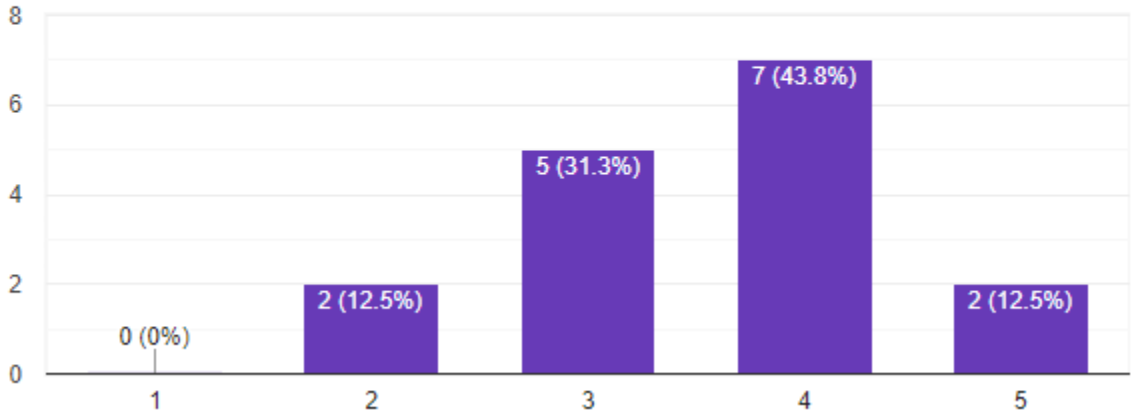
Sprinting feels less moth-like and more human

Jumping and walking felt smooth, but moths usually move really erratically, so it didn't feel very mothlike.

The air dash was very fluttery, but it a lot of the movement felt linear instead of how moths kinda go all around

Rate the level of control you felt over the player character

16 responses



What did you dislike about the controls?

16 responses

The controls went away rather quickly

The attack chain is a bit too slow

I didnt like turning the character

felt somewhat limited

I felt like some of my movements (especially the B dash) overshot on several occasions. At the moment this is OK since the areas are large, but in more confined spaces it would be valuable to have tighter control over how far you jump.

This is probably just because I'm bad at games but I had trouble controlling the camera. when going down things it wouldnt tilt down and I would clip through the floor, and sometimes through walls

nothing

The discontinuous of actions - like lack of combos

look speed

dodge felt a bit hard to control, attacks could have used more emphasis

wanted a ground dash

The moth's momentum kind of stops entirely when the dash is over, which feels a bit stilted

Could not move perspective up

Had to fight the camera all the time

Overall, the character spent too much time in animations, which led to being locked into swinging my weapon even if I jumped, which looked weird. Also, it took a while to fall during the platforming section.

A touch floaty

What did you like about the controls?

16 responses

I liked being able to jump forward and nyooom

The air dash covers a really long horizontal distance

They lent themselves well to player freedom

i like the buttons chosen for the actions

I appreciate the variety of movements as it stands. Dashing and gliding around feel very promising, and moth-like indeed.

otherwise good

lot of control

It is very natural and fine tuned

everything else :D

movement felt good, air dashing was very fun

liked the air dash

I love the idea of dashing around, and with some fine grain control and more freedom it could be very expressive. The walk cycle also makes walking feel good!

dash in the air was fun and the triple attack combo is Pleasing at the end

The jump, the dashing, and the attack felt nice

I enjoyed the dash mechanic and the three-hit combo, even if the controls were a bit floaty for my liking.

Nice and responsive

How did you feel about the camera?

16 responses

A little drifty and clippy

It's good but it'll be better if I can lock on to an enemy mid fight

It felt decent for the most part but it was hard to handle in tight spaces as it kept jerking around and clipping into objects

i didnt like how low it was and how much i had to micromanage it

The camera definitely needs a lot of work. Camera work is always a huge pain and I don't quite know what I'm talking about, but something between smooth automatic movements with more snappy player-controlled movements seems like a good balance to strike.

hard to move but again, im bad at games

little weird at the top

The camera is fine, but if resetting is allowed (like press right stick), it would be very helpful

i liked the 3rd person view

camera controlled well (slight drift but that seems to be the controller)

should be higher/angled more

The camera is too low at the default height to accurately gauge jumps. It could also move in more when it collides with a wall or building

grrr why no look up :(

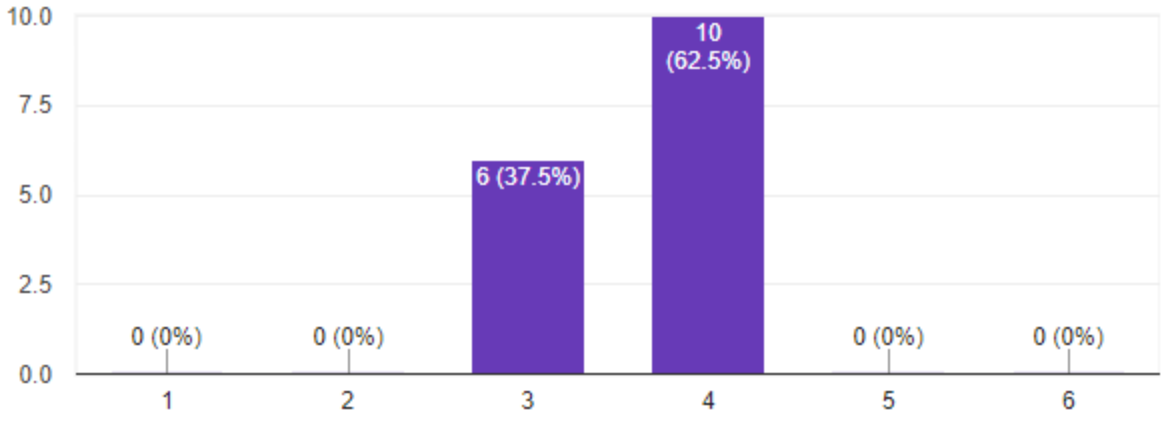
Bad

It moved when I turned slightly, which made it challenging to line myself up with the platforms to jump to and also keep the camera in the position I liked.

Nice and responsive

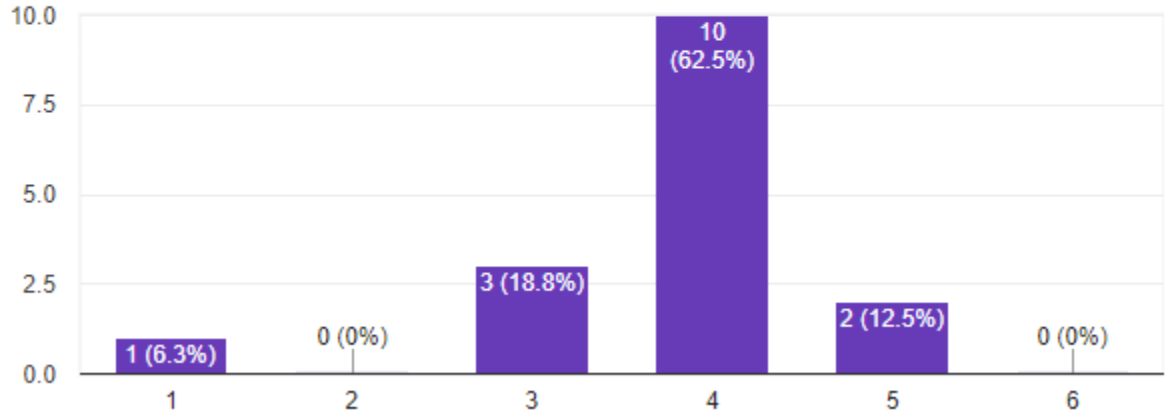
Rate the player speed

16 responses



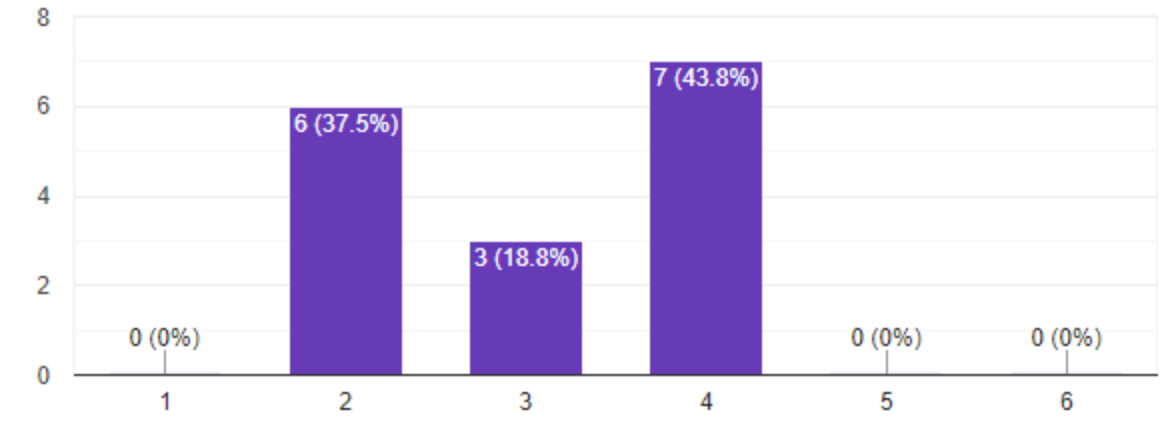
Rate the player's aerial dash

16 responses



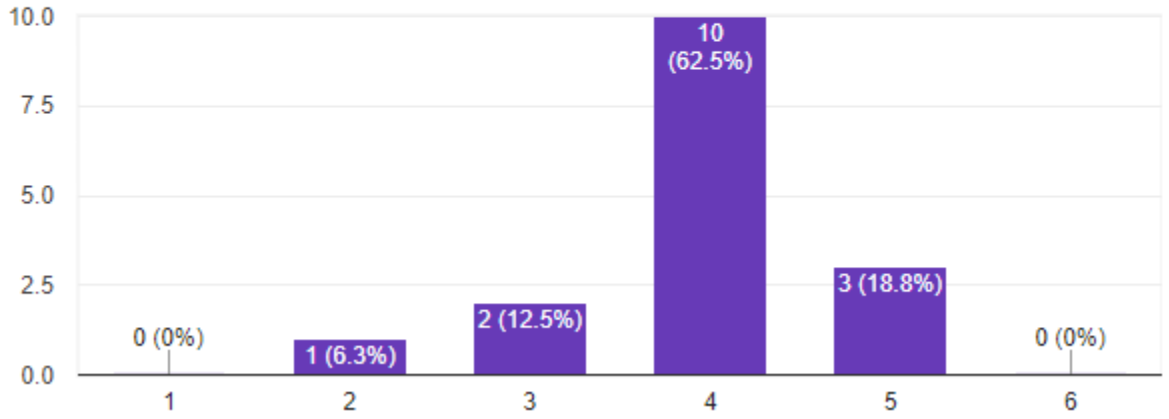
Rate the player's attack speed

16 responses



Rate the feel of the player's jump

16 responses

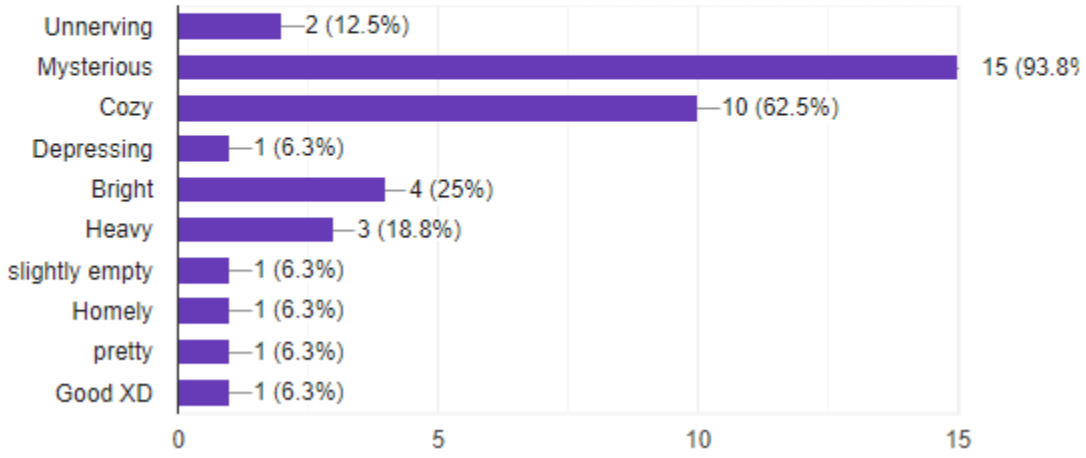


Did you find yourself running or dashing more? Which felt more fun?

16 responses

How would you describe the ambience of this environment? Check all that apply.

16 responses



Did you feel a powerful weight to the player's three-hit-combo? If so, why? If not, why not?

16 responses

It was powerful and satisfying to get a nice combo in

Not really, since it's a bit too slow by combo standards

The three hit combos sounds are unbalanced and dont match what's going on screen

i didnt notice it

The third hit does feel pretty nice, if a bit floaty on the lead in. Having the little particle effect and clashing sound definitely sells the impact.

nope, didnt notice that

yes

Yes. Sound + special effect

yes, it's really nice to show off the finisher animation

not particularly, didn't feel too different from just a normal attack

yes, the final animation and sound help make it feel strong

In some ways- enemies don't react, so it doesn't feel like there's much impact. The slow speed of the final attack also feels awkward, and the first 2 swings could use some emphasis

I never felt like i needed the three-hit-combo to finish any enemies and it didnt feel like i knew when it was actually hitting the enemy

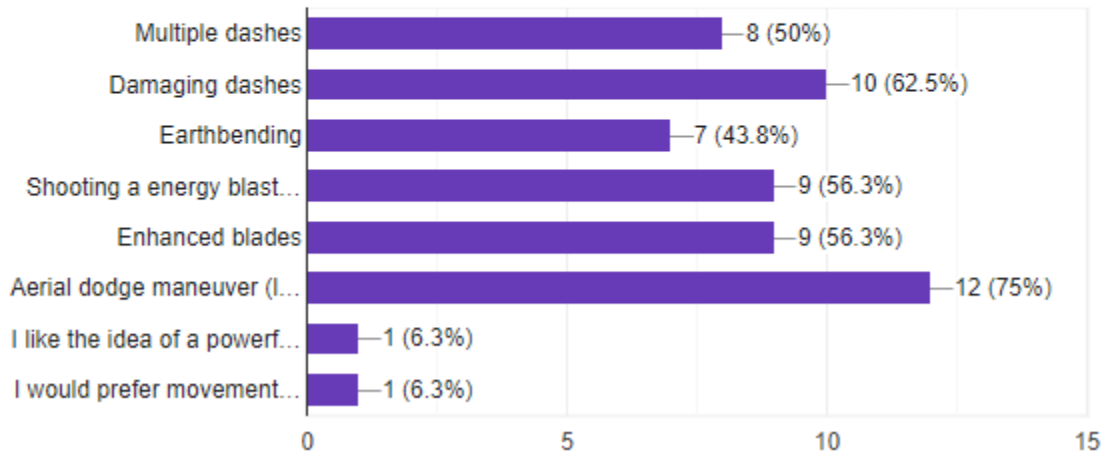
Yes. I like the effects, but the animation felt a bit stiff

The final hit of the attack gave a slight sense of weight, but it was a bit too floaty overall, and there wasn't any reaction from enemies when they got hit.

No, there was very little feedback

What potential player abilities would you want to see added to the game? Please select whichever options seem fun to you, as well as any additional ideas you may have.

16 responses



Was the dialogue engaging, or did you find yourself skipping it?

16 responses

Engaging!

It is engaging with the worldbuilding involved in them

Somewhat but I didnt have time to read it

i liked the lore around it although admittedly i did skip it to find out my motivation first and would have likely engaged with it later

I like the dialogue! I did usually skip the animating part, but I did try my best to absorb the details of the world. That being said I also appreciate that the conversations are relatively short, and don't overstay their welcome.

engaging and informative

50/50

engaging

I found it engaging enough to not skip

i was skimming it

Yes! It’s a bit of a terminology dump so it’s a lot to take in but I’m interested in the mystery of the world

skip! there were some lore words i didnt care about. I didnt feel an stakes in the world, so I didnt care about the characters

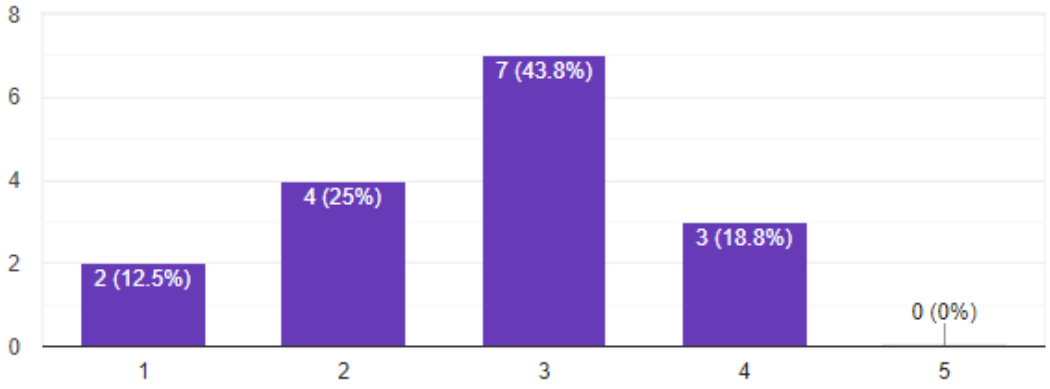
Glossed over it for the most part

A lot of the dialogue used terms I wasn't really familiar with or invested in, so I tried to read it but didn't follow most of it.

I read most of it, if only skimmed

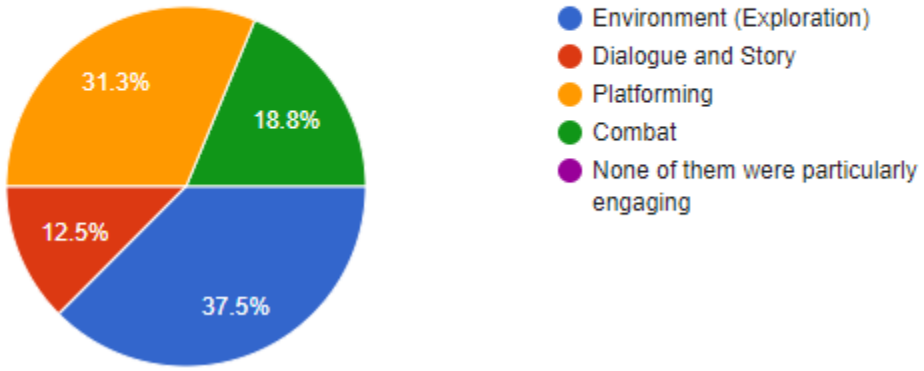
Rate the difficulty of fighting the enemies.

16 responses



What was the most engaging part of your playtest experience?

16 responses



Note any gameplay difficulties and/or bugs that affected your playtesting experience.

6 responses

Camera clipped through the pillars at the start

There isn't any audio cue when I lose health.

in games i like to avoid enemy attacks, i didnt feel like there was an engaging way to dodge and attack at the same time

The monsters kept disappearing and then reappearing, i wasn't sure if that was intentional or not

n/a

It was hard to see the platforms below me during the platforming section/keep myself traveling straight through the air when jumping between platforms.

Appendix G. Final Playtest Survey

Below are the final playtest survey questions, the last of the playtests.

Rate the level of control you felt over the player character *

Mark only one oval.

	1	2	3	4	5	
Very little control, not intuitive/easy to grasp	◯	◯	◯	◯	◯	Very fine level of control, easy to use

What did you dislike about the controls? *

What did you like about the controls? *

How did you feel about the camera? *

Rate the player speed *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the player's aerial dash *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the player's attack speed *

Mark only one oval.

	1	2	3	4	5	6	
Too slow	◯	◯	◯	◯	◯	◯	Too fast

Rate the feel of the player's jump *

Mark only one oval.

	1	2	3	4	5	6	
Too stiff	◯	◯	◯	◯	◯	◯	Too floaty

How would you describe the ambience of this environment? Check all that apply. *

Check all that apply.

- [ ] Unnerving
- [ ] Mysterious
- [ ] Cozy
- [ ] Depressing
- [ ] Bright
- [ ] Heavy
- [ ] Other: _____

Do you feel a powerful weight to the player's three-hit-combo? If so, why? If not, why not? *

_____

Which ability did you like to use the most and why?

_____

_____

_____

_____

_____

Was the dialogue engaging, or did you find yourself skipping it?

_____

_____

_____

_____

_____

Rate the difficulty of fighting the enemies.

Mark only one oval.

	1	2	3	4	5	
Easy	◯	◯	◯	◯	◯	Hard

What was the most engaging part of your playtest experience?

Mark only one oval.

- ◯ Environment (Exploration)
- ◯ Lore and Story
- ◯ Platforming
- ◯ Combat
- ◯ None of them were particularly engaging

How much of the dialogue did you read?

Mark only one oval.

- ◯ None/very little - I skipped most of it
- ◯ Some - I usually paid attention but skipped some of it
- ◯ All - I made sure to read every dialogue line I saw

How well did you understand your character's goals and motivations while playing?

Mark only one oval.

	1	2	3	4	5	
Not at all	◯	◯	◯	◯	◯	Completely understood

What do you think happened at the end?

How did the ending make you feel?

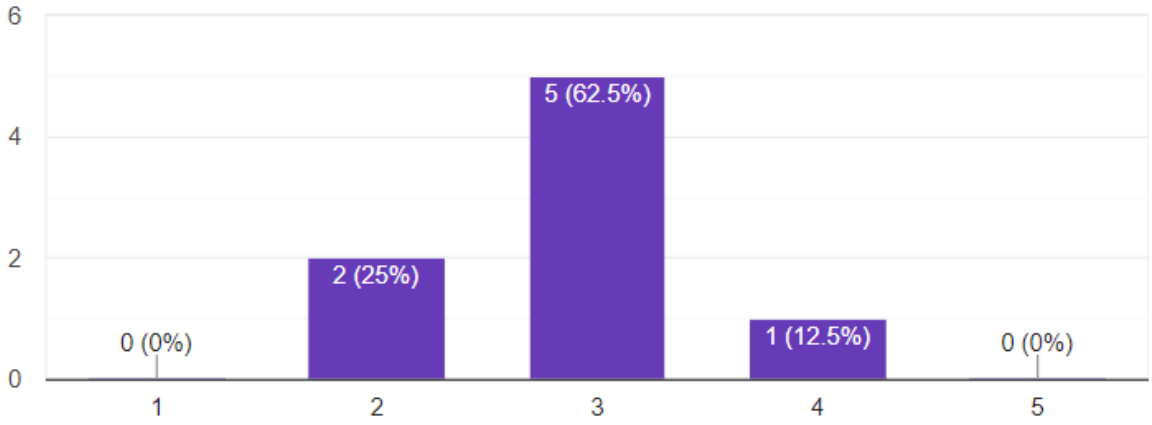
Note any gameplay difficulties and/or bugs that affected your playing experience.

Appendix H. Final Playtest Survey Results

Here are the results of the third and final playtest, including graphs and short responses.

Rate the level of control you felt over the player character

8 responses



What did you dislike about the controls?

8 responses

Too floaty didnt feel like there was any weight

The mouse controls/camera controls were a little funky

Camera was difficult to control at times. Also the air dash was always a risk to use as I would be propelled so far so quickly.

the camera was a little too sensitive, I think slowing it down would make the controls a bit more fluid

Camera, left control is an awkward key on some computers

mid-air dash could feel a bit too quick, often was hard to determine how far it would go

The free camera was too sensitive, which made it difficult to line up

The camera felt clunky and it was somewhat hard to tell when an attack hit an enemy

What did you like about the controls?

8 responses

It was quick

The keys and buttons were straight forward and good

The attacks were fluid and intuitive.

I liked the jump/dash combo, they were very easy to use

i like jump and air dash being on the same button

everything felt tight and responsive

The jump and jump dash were cool and easy to understand.

The movement felt fun

How did you feel about the camera?

8 responses

Too finicky kept moving around and wasnt easily controlled. made me nauseous

It was hard to control and very "loose"

Very sensitive

It moves a bit too fast and I should be able to tilt it to see things above me.

Camera should be able to move up and down, camera should be dampened, when you hit a wall the camera adjusts automatically, when double jumping I feel like moving towards where the camera is facing would be more intuitive (same with attacks)

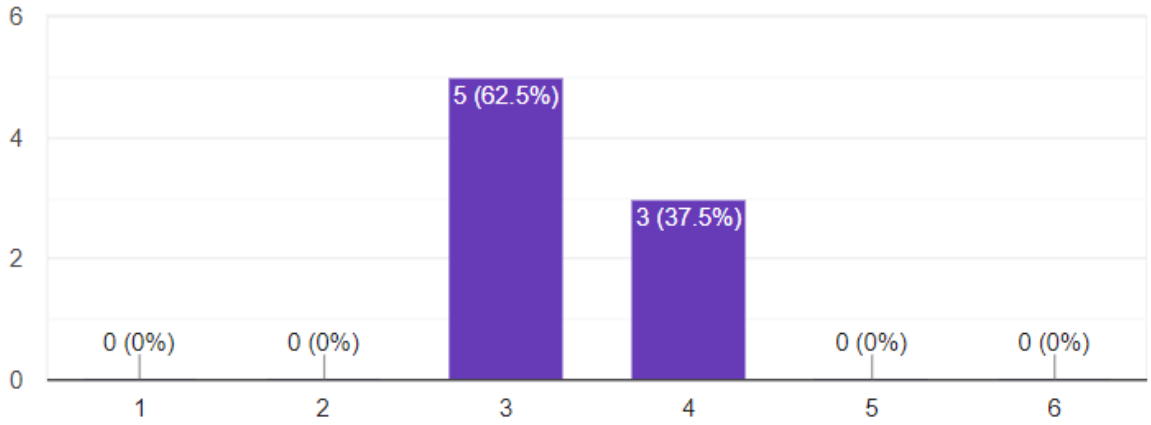
camera moved very quickly along with player, often felt as if it was fighting against how I was trying to move the camera

Too sensitive, would like a control for that

Camera was hard to use

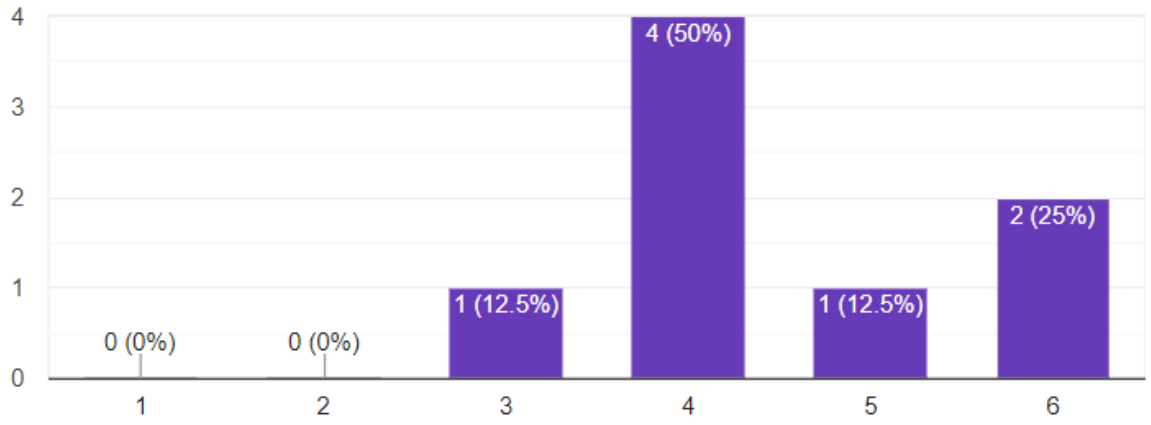
Rate the player speed

8 responses



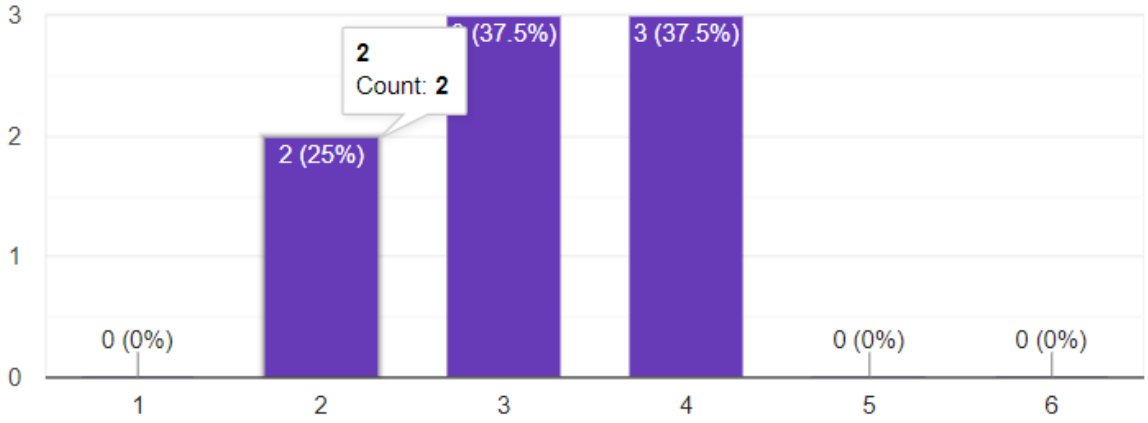
Rate the player's aerial dash

8 responses



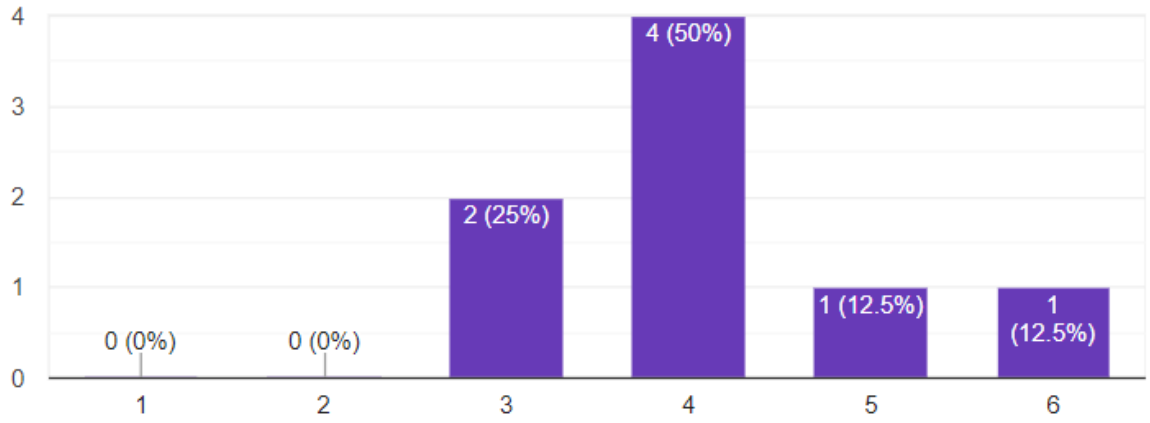
Rate the player's attack speed

8 responses



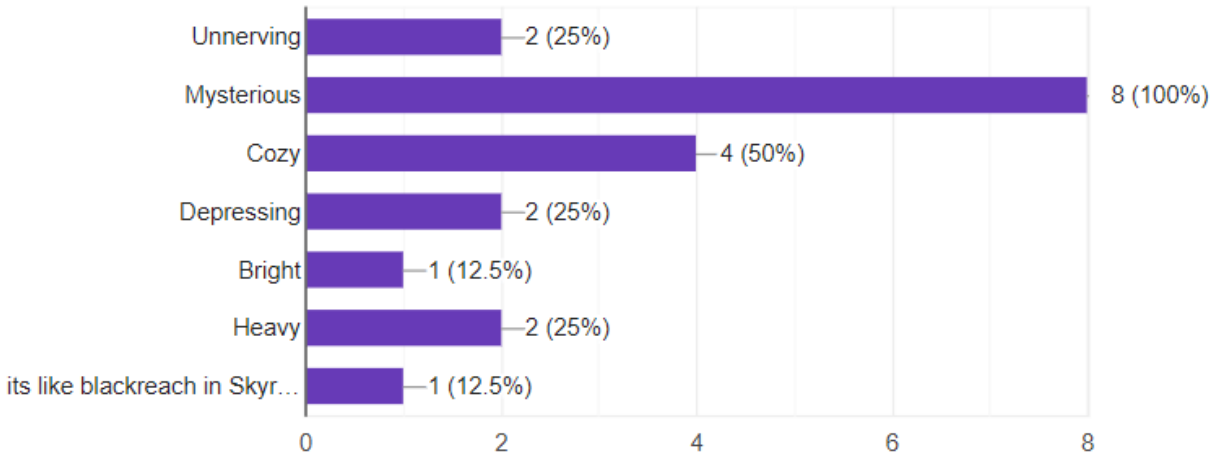
Rate the feel of the player's jump

8 responses



How would you describe the ambience of this environment? Check all that apply.

8 responses



Do you feel a powerful weight to the player's three-hit-combo? If so, why? If not, why not?

8 responses

No just felt like he floated on punch a mattress

I don't understand this question but it was very nice :)

I do feel a powerful weight.

I felt like the individual attacks were weighty, but the projectile didn't feel like a projectile, mores a ball of light.

no because visually it is hard to tell the effect of each individual hit

I liked how it looked visually, though I feel it could have had more weight if the attacks knocked enemies back a bit

The third hit is too slow to be useful, with the small range

It didn't seem worth it to use the combo

Which ability did you like to use the most and why?

8 responses

The ground spike one. Killed in one hit

flying because its pretty

The ground pound was the coolest.

The explosion was easily my favorite ability cause it looked really neat.

the aoe one because it killed them all instantly

While it was a bit hard to control, I enjoyed using the mid-air dash and zipping around

Air dash

I used the dash a lot

Was the dialogue engaging, or did you find yourself skipping it?

8 responses

I skipped it

It was good! I was a little confused at first but I think it gets the story across

It was engaging for the start, but most of the dialogue covered the similar themes such that you could get the gist while skipping through by the end. It was enjoyable.

it was engaging, but the names made it somewhat hard to tell what was whaat in the game. Maybe some visual aids would help?

i skimmed it mostly but did not skip it, so somewhat engaging

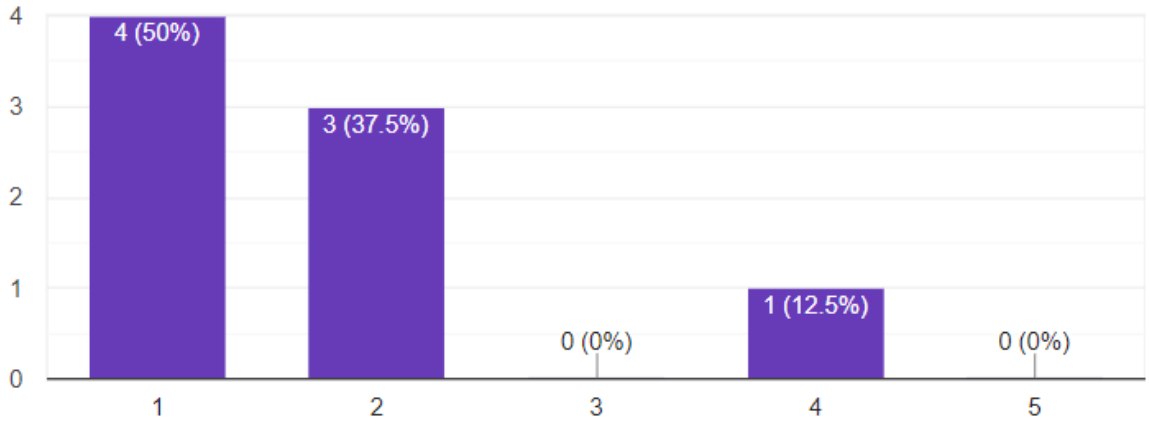
I read all the dialogue, while most was engaging I did feel some of the conversations lasted a bit long

I didn't skip much, which is a plus as I'm a skipper. I thought it was about the right amount of dialogue per person

I read it but would have skipped it normally

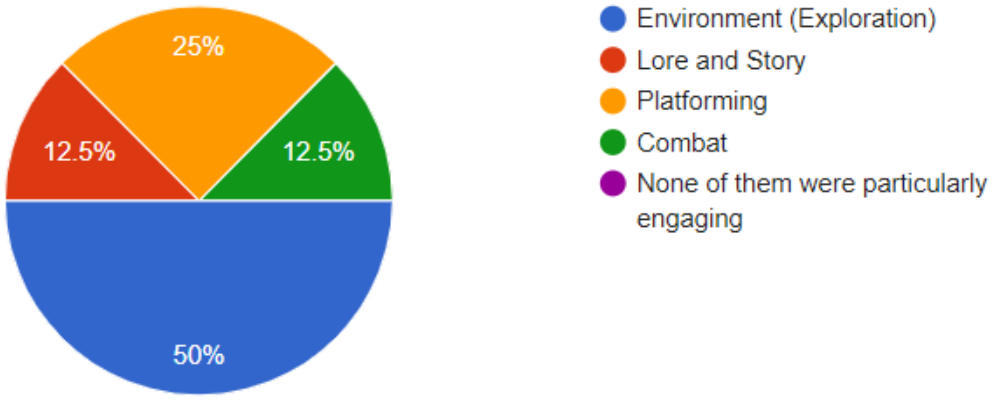
Rate the difficulty of fighting the enemies.

8 responses



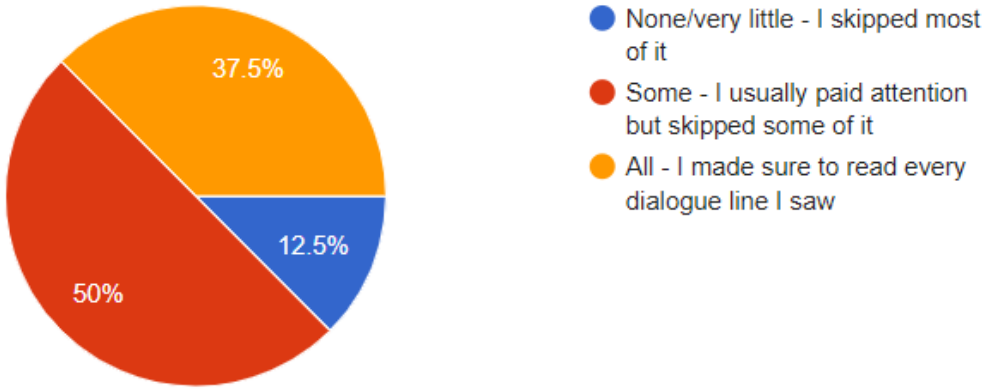
What was the most engaging part of your playtest experience?

8 responses



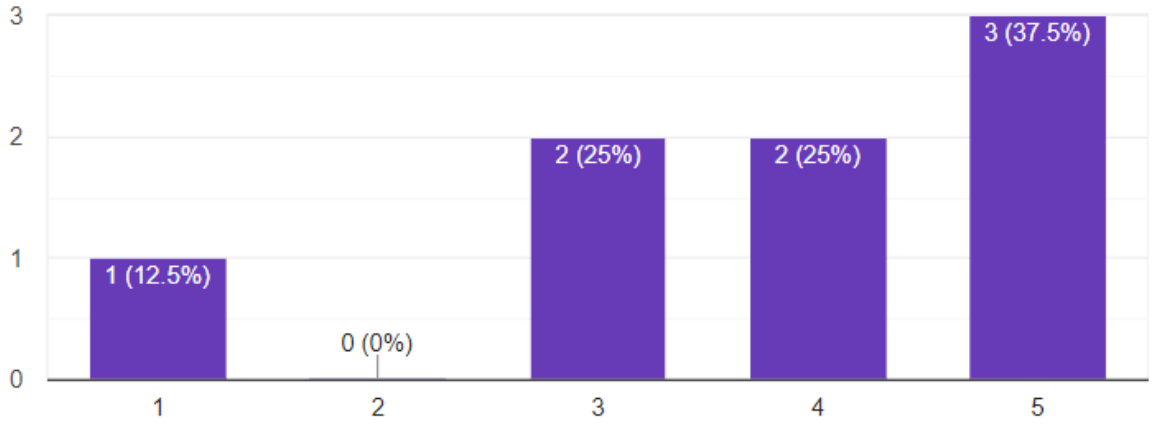
How much of the dialogue did you read?

8 responses



How well did you understand your character's goals and motivations while playing?

8 responses



What do you think happened at the end?

8 responses

No idea

everyone dies):

Bruh the ceiling caved in they dead as hell. Any survivors would evacuate out the hole at the top but like all the collapse shots showed the moths getting crushed : (

in their quest to become moonsighted, the moths destroyed their own home because they expended resources that were vital to their survival

the crystals were holding up the cave and when the last one was taken it all collapsed

definitely seems to be part of a larger story

The miner destroyed all the support crystals and brought down the cave system.

Everyone died

How did the ending make you feel?

8 responses

idk

saddened

Regretful cause they were fools. Ah well, don't do meth kids.

I could see it coming but it made me feel sad because there was nothing I could do about it.

nothing because I already knew what was going to happen

curious about what happens next, but didn't leave too much of a lasting impact

Sadge. I thought he was supporting the community by bringing them food and exploring their world, but really he was just destroying it. Perhaps its a metaphor for technology advancement.

Sad that everyone died

Note any gameplay difficulties and/or bugs that affected your playing experience.

8 responses

If you hold down jump after dashing it continues to let you dash making it very easy to fly though the platforming. Also killing yourself teleported you back so it was easy to get back after finishing the monster.

good job guys :) it looks very nice

Camera, the dash was too quick. It was a little difficult to find out exactly where to go at first because the lighting was so dark. I also thought the collapsed spires were still collectible at first.

I clipped through a wall once but other than that it was very polished

you can see through walls if you go next to them, dialogue bug with olend(?), also in the 2nd to last stage there isn't a good way to climb back up

Not really a major issue, but I played with a Duelshock 4 and having the controller instructions use Xbox symbols/layout made me have to stop and think for a sec :p

I'm a bad jumper...

The dash sometimes hit the ground mid dash and went way farther than I was expecting and I died