Head in the Clouds

Important note: this submission contains a URP project and build (primary submission for grading), an HDRP project and build showcasing the original lighting, and a video playthrough of the HDRP build in case HDRP does not work on the grading computer. Enable "Prefer External GPU" before running projects or builds.

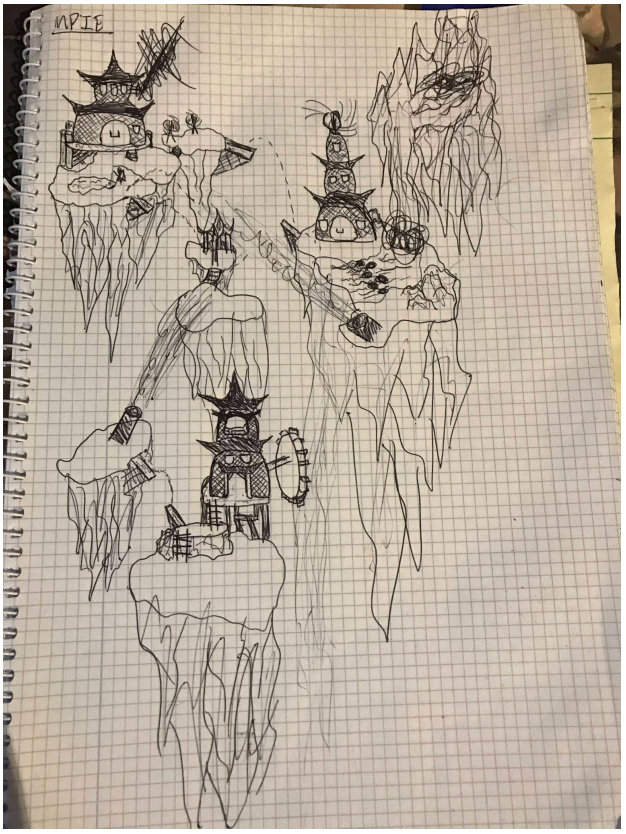
Title screen

Head in the Clouds is a first-person game about an author delivering books to his fellow villagers. The game takes place in a small floating village inspired by *The Legend of Zelda: Skyward Sword*'s Skyloft. As the player glides and boosts between islands, they realise that nobody actually wants to read their book. The protagonist, however, has his "head in the clouds" and believes everyone will love it.

View from the Windy Islands

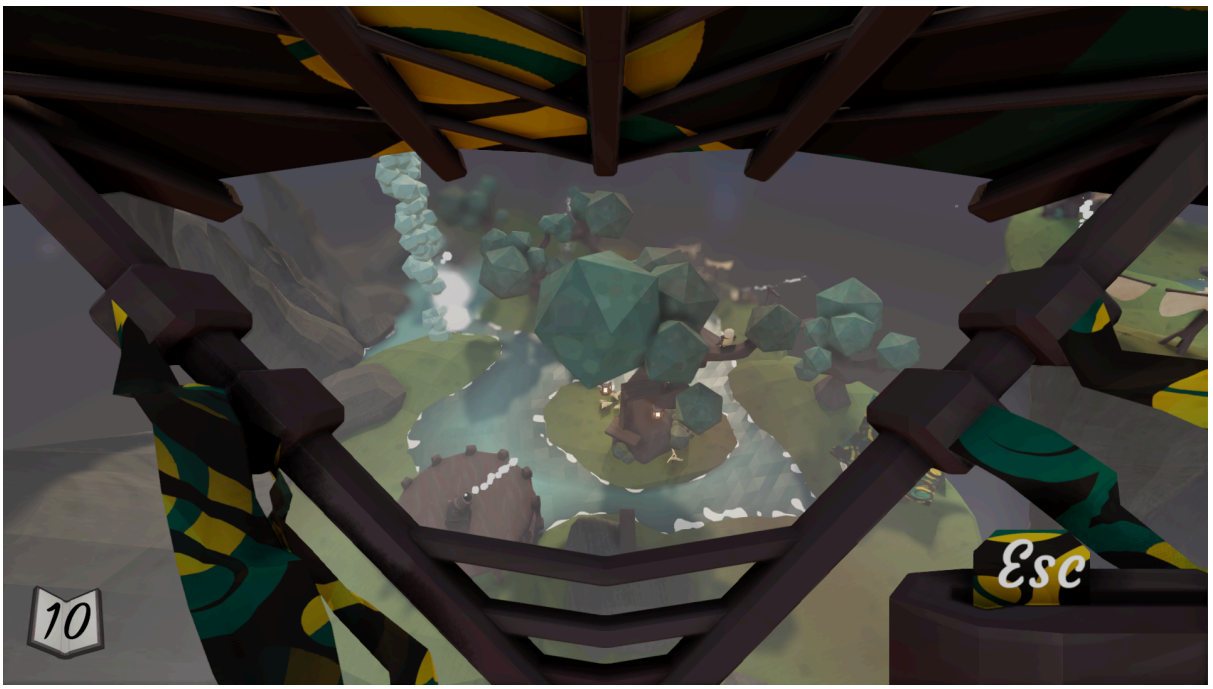
My goal for this project was to create an open, atmospheric, and calm environment that could still be completed in a very short time without stress. I wanted a highly polished experience with fitting music, sound effects, atmospherics, and in-world UI. I have developed many projects where I planned to implement a dynamic soundtrack but never got around to it, so I decided to treat it as a minimum viable product feature for this assessment. I set out to use HDRP to fully execute the quality I wanted. I intended to include a satisfying movement system to complement the atmosphere. I wanted a clean main menu that displayed the game's environment and lighting. I also made it my goal to try new techniques in 3D modelling such as rounding low-poly edges on soft objects.

Background



An early sketch of the village

I faced difficulty settling on an idea for this project. My greatest challenge was the five-minute limit; I wanted to create an open environment but could not guarantee the player would finish in time. On a previous project, I created a floating island map to allow players to manoeuvre around while still limiting the areas they could occupy. I applied this approach to keep the player moving without placing them on a linear track. From lower islands, players need to use air cannons to boost higher. Once higher up, players can then glide down to other islands. This movement system supplied adequate, open-ended gameplay while keeping the player's options limited.



Gliding above Bog Island

The glider is a staple piece of the game's identity. I wanted gliding between islands to create a similar feeling to paragliding down from a tower in *The Legend of Zelda: Breath of the Wild* or gliding across wide gaps in *Ori and the Blind Forest*. The depth of field and volumetric fog are important in keeping the small environment feeling bigger than it is; I drew on similar implementations in games like *Yoshi's Crafted World* and *Supraland*. Other atmospherics were inspired by *Journey*, *Hollow Knight*, and *LEGO Builder's Journey*.

Bog Island

As many of my recent projects have been fast-paced and intense, I wanted to design a more serene experience for this project. Deciding to go with an airy theme, I created windmills, air-blasting cannons, floating islands, fog in lower areas, a glider, and breezy music. The music was heavily influenced by the trilling violins in Divine Beast Vah Medoh's soundtrack from *The Legend of Zelda: Breath of the Wild*. I recreated these with synths since I did not have access to my violin at the time of writing the soundtrack; this ended up adding more character to the song.

```
//music that plays while gliding or air cannon boosting
2 references [redacted] 23 hours ago | 1 author, 2 changes
private void MidairMusic()
{
    //melody, trill chords
    audioSettings.SetTrackFade(0, true);
    audioSettings.SetTrackFade(1, true);
    audioSettings.SetTrackFade(2, false);
    audioSettings.SetTrackFade(3, false);
}



//music that plays when on the ground
1 reference [redacted] 23 hours ago | 1 author, 2 changes
private void GroundedMusic()
{
    //trill chords, low chords and bass, drums
    audioSettings.SetTrackFade(0, false);
    audioSettings.SetTrackFade(2, true);
    audioSettings.SetTrackFade(3, true);
}
```

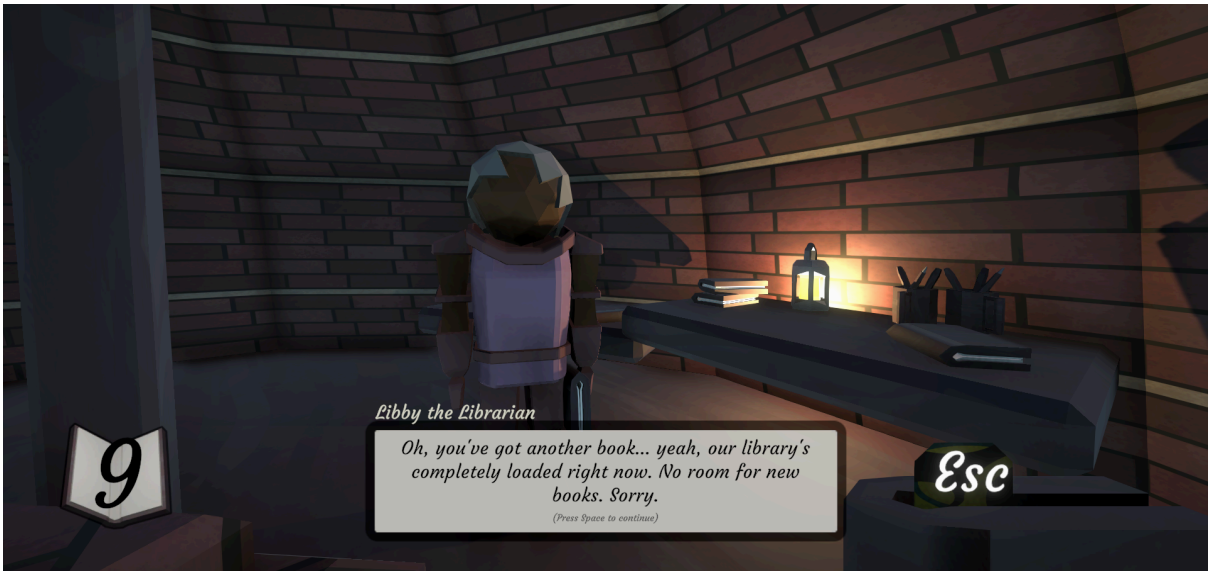
```
Unity Message | 0 references [redacted] 23 hours ago | 1 author, 3 changes
private void Update()
{
    //fade in
    if (state == fadeState.FADEIN)
    {
        //fade in by fadeSpeed amount
        audioSource.volume += fadeSpeed * Time.deltaTime;
        if (audioSource.volume >= maxVolume)
        {
            //stop fading
            SetState(fadeState.PLAYING);
        }
    }

    //fade out
    else if (state == fadeState.FADEOUT)
    {
        //fade out by fadeSpeed amount
        audioSource.volume -= fadeSpeed * Time.deltaTime;
        if (audioSource.volume <= 0.0f)
        {
            SetState(fadeState.SILENT);
        }
    }
```

Music tracks fade in and out dynamically

To emphasise the lightness of gliding, I remove the lower-end tracks from the music and add a melody and wind when gliding. This was inspired by how the music changes throughout the flying bikes scene in *E.T.: The Extra-Terrestrial*. The tracks for lower chords, bass, and drums return on landing, making being grounded feel heavier. I have wanted to create a dynamic soundtrack for most games I have developed, so finally implementing one was very rewarding.

"You know nobody likes your books."

The game was initially quite wholesome to match the peaceful, atmospheric village. However, as I added dialogue, I decided the characters would dislike the protagonist and his books. This added a nice twist to the game and prompted me to name it *Head in the Clouds*, as the protagonist is ignorant of the whole village disliking him and his books. The name also ties in with the village floating above the clouds and supports the airy theme.

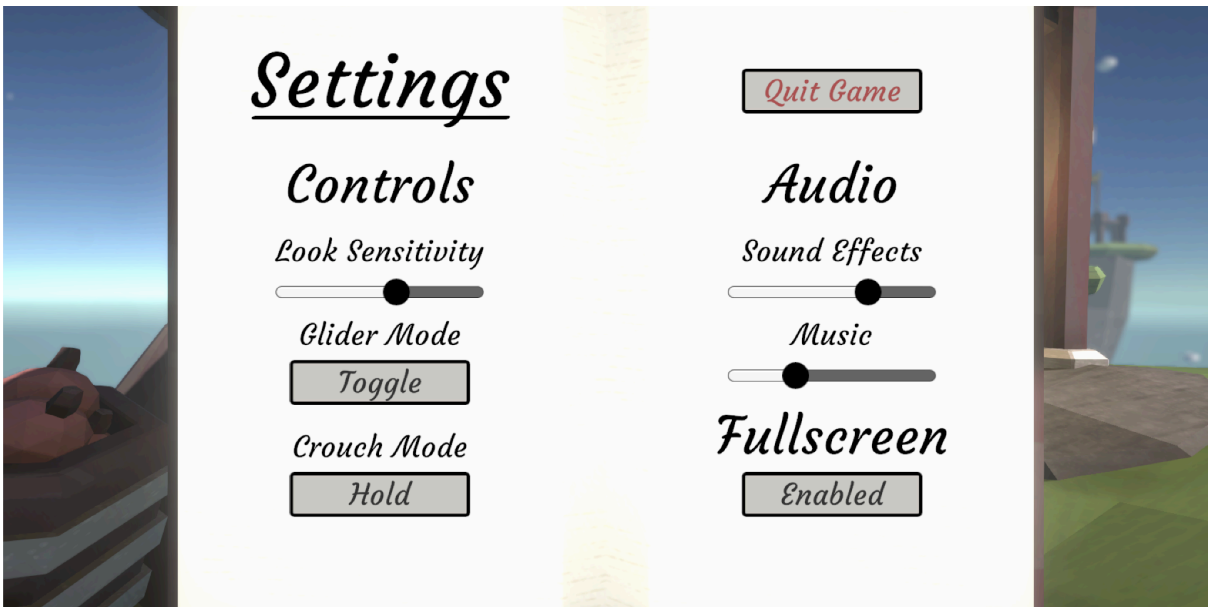


"Loaded" Library seems to have quite a bit of free space

The greater purpose of the game is to show how much care artists put into their work without guaranteed reciprocation from consumers. The protagonist is intentionally ignorant of the lack of return on his books to stay motivated. This theme was partially inspired by spending all of winter break working on a (hopefully) five-minute game.

Even the protagonist knows that time is of the essence

The gameplay is intentionally very simple to prevent a difficulty curve that could take too much time to learn. The mechanics are few, so I made the most of what I had by placing the problem-solving aspect into reaching certain islands. Initially, I had planned to implement fewer, more challenging characters: a sleeping chieftain whose book had to be snuck onto his desk, a builder named Fortunato who accidentally built himself inside his house (inspired by Edgar Allen Poe's *The Cask of Amontillado*), and a child playing hide and seek on a confusing wooded island similar to the *Legend of Zelda* franchise's Lost Woods. These would have been more engaging, but would each take more variable amounts of time and could not guarantee a five-minute playthrough.



Immersive settings menu inside a book

I designed the settings menu to be in line with the world. Rather than creating a pop-up menu, I added a book in the bottom right corner of the screen with a bookmark labelled “Esc.” On pressing Escape, the book opens and the settings UI appears with it. This was inspired by Mark Brown’s video on immersion in Metro Exodus (Game Maker’s Toolkit, 2019).

Air cannon tutorial

I also placed tutorials on in-world signs as necessary rather than showing a pop-up onscreen. This keeps them eye-catching and permanently available while also out of the player's way. I placed the tutorials for gliding and air cannon boosting in their first relevant locations to slowly feed the player information without overwhelming them.

Island title

I did use pop-ups for island titles and character title cards, but these are also area-specific and unobtrusive. Island titles appear upon landing on an island to orient the player. The first island does not initially show its title to prevent information overload and give more emphasis to the next island's title.

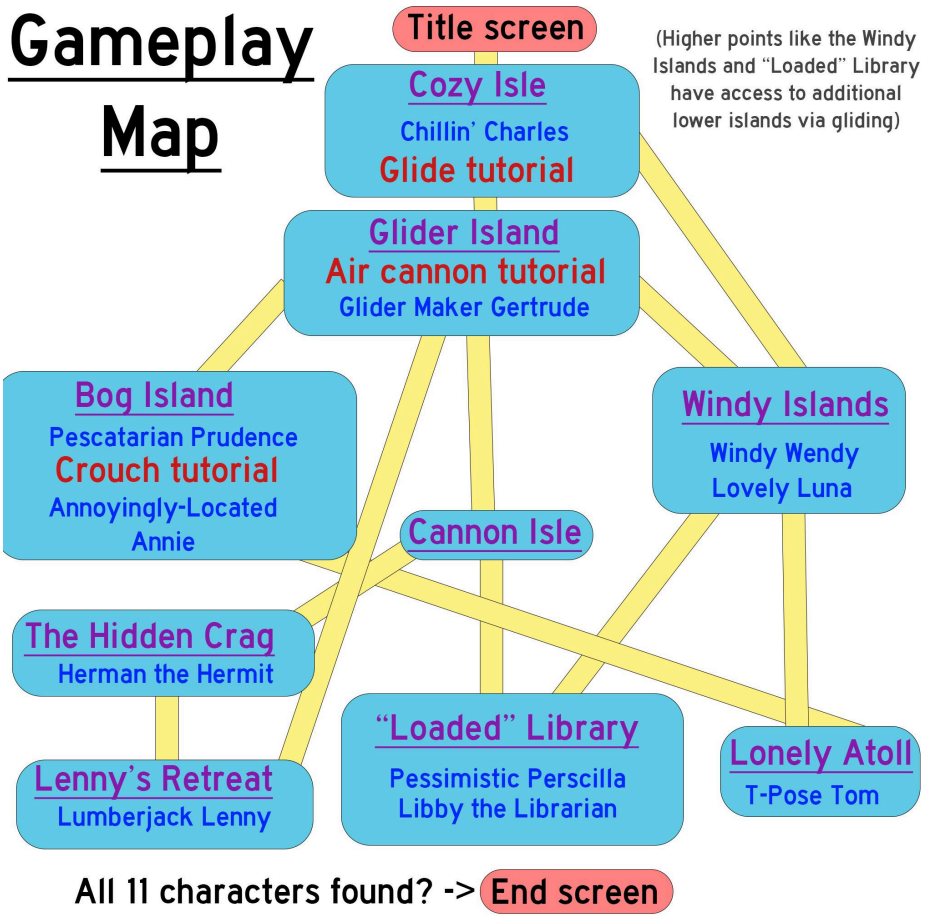
Experience

Map of the game world

The game opens to a title screen featuring a selection of shots from around the village. Pressing space will give the player control of the character and instruct them to deliver books to the villagers. Right away, the player can offer a book to the first character and begin to realise the villagers may not want the books after all. They proceed to a diving board-like plank jutting off the island beside a sign teaching them how to glide.

Gliding to Glider Island

As the player glides, the music changes to a melody and the soundtrack's trademark trilling synths. Cozy Isle, the starter island, is intentionally farther from the rest to promptly give the player the longest glide in the game. This way, the player becomes accustomed to the gliding experience and can watch as Glider Island slowly comes into view. Here, the player learns to use air cannons, one of which is immediately available to boost the player to another character's house.

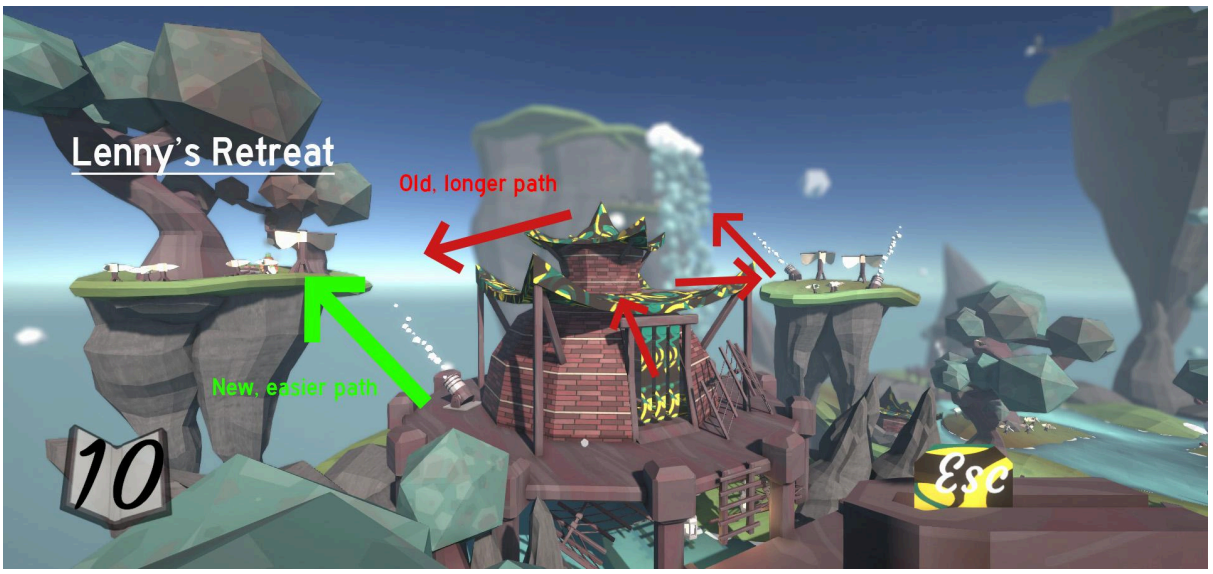


Paths the player can take

From this point, the game is fairly nonlinear. The player can now visit islands as they choose. However, there is a ladder inside the aforementioned house, so players are most likely to climb to the top, glide to Cannon Isle, and take an air cannon to The Hidden Crag or "Loaded" Library. Other players may head to the adjacent Bog Island. There are characters to find on almost every island.

Annoyingly-Located Annie lives up to her name

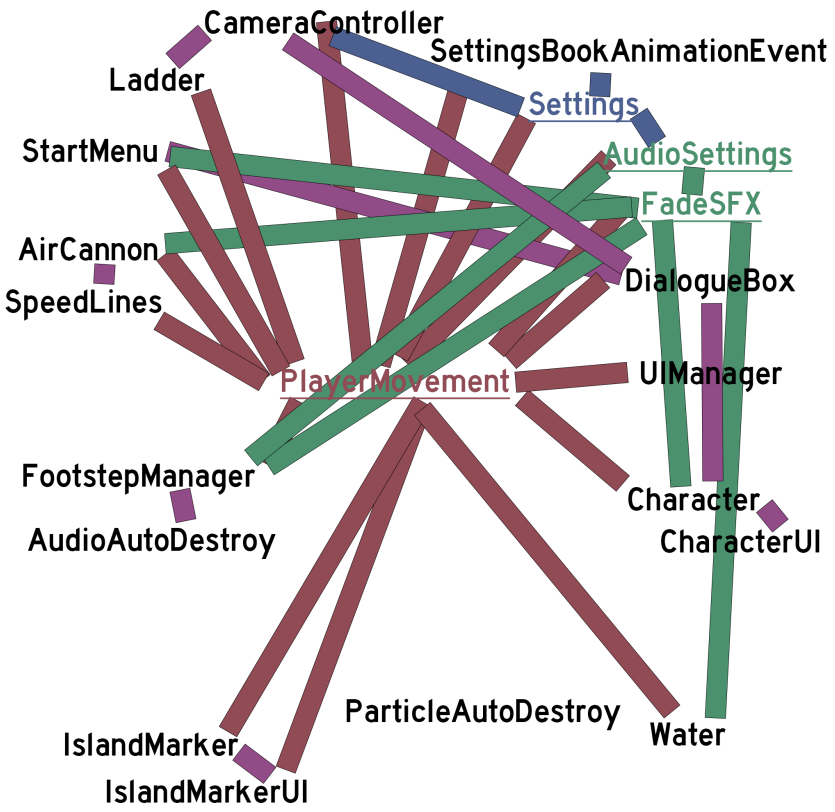
There are 11 characters in total, with two essentially being freebies. The rest must be found through exploration. Some characters are trickier than others. One is sitting atop a tree on Bog Island. Lenny's Retreat is partially hidden behind Glider Island. However, safeguards are in place to prevent the player from missing a character. If they bypass the first character on the starter island, they can return via an air cannon on the Windy Islands.



Ease of access to Lenny's Retreat

Across development, I added more air cannons than necessary to ensure difficult areas could be easily reached. Lumberjack Lenny is difficult to find, so I added another air cannon on Glider Island for ease of access. "Loaded" Library used to only be reachable from Cannon Isle, but can now be boosted to from the Windy Islands. These air cannons were added to shorten playtime and would be removed in a longer-form release.

Technical Implementation



References between scripts

This project was my biggest dive into Unity and C# yet and had a bit of a learning curve. I developed a first-person controller with Unity's Character Controller component, a dialogue system, a camera setup for the main menu and end screen, a class for managing audio fading in and out, in-world titles for characters, island titles that fade in and out, an immersive settings menu, and a footstep sound manager that plays random footsteps depending on the surface the player walks on.

```
void UpdateCrouch()
{
    //same system as gliding for toggle vs hold
    //except this time there's an additional check "tryingToStand"
    //this is further explained in CanStand()

    if (Input.GetButtonDown("Crouch"))
    {
        if (!crouchToggle || (crouchToggle && !isCrouching))
        {
            tryingToStand = false;
            TryCrouchStart();
        }

        else
        {
            tryingToStand = true;
            TryCrouchStop();
        }
    }

    else if ((Input.GetButtonUp("Crouch") && !crouchToggle) || tryingToStand)
    {
        tryingToStand = true;
        TryCrouchStop();
    }
}
```

```
//i prevent the player from uncrouching into an object by constantly checking above them when they want to uncrouch
//if anything's inside the area their head would be, they can't uncrouch
//once they're in the clear they automatically uncrouch unless they started intentionally crouching again
1 reference | [redacted] 2 hours ago | 1 author, 3 changes
bool CanStand()
{

    //check above the player's head (their height if they were standing)
    if (Physics.Raycast(transform.position, charController.transform.up, halfHeight * 1.5f) == true)
    {
        return false;
    }

    return true;
}
```

Players cannot uncrouch under a low ceiling, but will automatically uncrouch upon leaving

The first-person controller is the longest script I wrote and references most other scripts. It manages basic movement as well as gliding, boosting with air cannons, and crouching, among many smaller tasks like interacting with characters. When the player tries to uncrouch in an area they could not normally stand in, CanStand() continuously raycasts above their head to know when they can automatically uncrouch. Footsteps are also quieter and slower during crouched movement. Players cannot crouch in water to prevent the camera from clipping under the water.

```
private void OnTriggerEnter(Collider other)
{
    switch (other.gameObject.tag)
    {
        //entered water
        case "Water":
            overlappedWaterPlanes.Add(other.gameObject);
            isInWater = true;

            TryCrouchStop();

            break;

        //entered an air cannon's boost trigger
        case "Air Cannon":
            airCannonTimer = airCannonMaxTime;
            overlappedAirCannon = other.gameObject;
            airCannonMove = AirCannonBoost(overlappedAirCannon);

            MidairMusic();
            boostSound.audioSource.Play();

            speedLines.SetActive(true);
            speedLinesScript.SetOpacity(1.0f);
            break;

        //entered a character's interaction area
        case "Character":
            nearCharacter = other.gameObject;
            nearCharacterScript = nearCharacter.GetComponent<Character>();
            break;

        //entered the area around an island
        case "Island Marker":
            overlappedIslandMarker = other.gameObject.GetComponent<IslandMarker>();
            break;
    }
}
```

OnTriggerEnter() uses tags to differentiate between GameObjects

PlayerMovement checks for trigger colliders to achieve various effects: water planes slow the player and create splashing particles, air cannons boost the player and then transition back to default player movement, characters have pop-up title cards when nearby, and islands show their names upon landing. Triggers on ladders detect the player and take control of their movement. Water plane triggers leave looping splashing sounds in the player's wake.

```
public void LandingParticles()
{
    Vector3 landingParticlesSpawn = transform.position;
    landingParticlesSpawn.y -= halfHeight;  //place them at the character's feet. crouching is never an option when this function is called so halfHeight will always be the same
    Instantiate(landingParticles, landingParticlesSpawn, Quaternion.Euler(-90.0f, 0.0f, 0.0f));
}
```

```
1 reference [redacted] 1 day ago | 1 author, 3 changes
private void TryIslandMarker()
{
    //if on an island the player hasn't already just seen the title for
    if (overlappedIslandMarker != null && lastIslandMarker != overlappedIslandMarker.text)
    {

        //IslandMarkerUI islandMarkerUI = Instantiate(islandMarkerUIPrefab);
        //islandMarkerUI.transform.SetParent(canvas.transform);
        //islandMarkerUI.SetText(overlappedIslandMarker.text);

        islandMarkerUI.gameObject.SetActive(true);
        islandMarkerUI.SetText(overlappedIslandMarker.text);

        //update lastIslandMarker so we can check against it
        //the player can now go back to the previous island and will see its title again
        lastIslandMarker = overlappedIslandMarker.text;
    }
}
```

```
float GetJumpHeight(float airCannonTimerAlpha)
{
    //immediately snap to ground when crouching
    if (justStartedCrouching == true)
    {
        justStartedCrouching = false;
        verticalVelocity = -100.0f;
    }


    //prevent the player from immediately falling supa dupa fast when air cannon boost ends
    else if (airCannonTimerAlpha > airCannonVerticalVelocityFalloffTime)
    {
        verticalVelocity = 0.0f;
    }


    //regular Y movement
    else
    {

        //prevent bouncing
        if (isGrounded && verticalVelocity < 0)
        {
            verticalVelocity = 0f;
        }
```

```
void UpdateGrounded()
{
    //charController knows isGrounded before this script does
    //so when they disagree, we can use it for events for leaving the ground and landing

    if (!charController.isGrounded && isGrounded)
    {
        OnLeftGround();
    }

    else if (charController.isGrounded && !isGrounded)
    {
        OnLanded();
    }

    //now that the events are done they should be equal
    isGrounded = charController.isGrounded;


    if (isGrounded)
    {
        //this timer is for coyote time for gliding and jumping
        //mainly helps with slopes so the character walks naturally downhill
        groundedTimer = 0.2f;
    }

    if (groundedTimer > 0)
    {
        groundedTimer -= Time.deltaTime;
    }
}
```

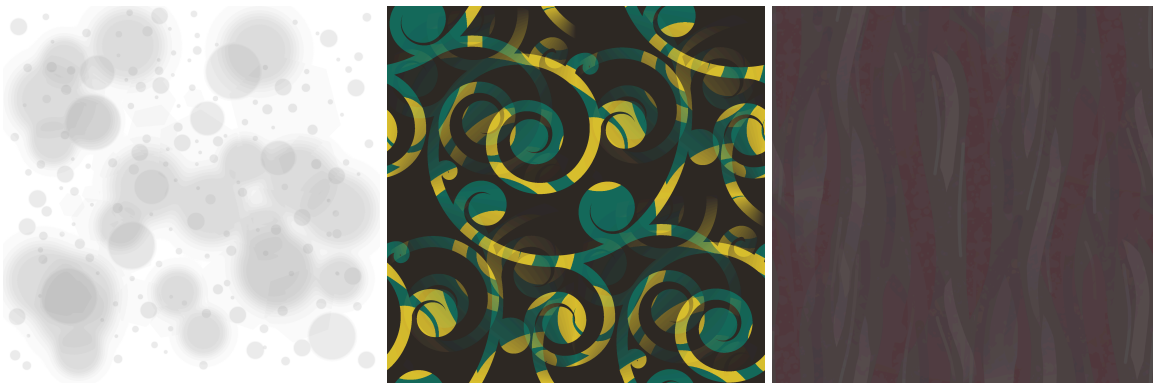
```
1 reference [redacted] 1 day ago | 1 author, 1 change
private void OnLeftGround()
{
    respawnPosition = transform.position;
    footstepManager.SetMovingGrounded(false);
}
```

Non-repetitive island titles, snappy crouching, bounce prevention, coyote time, and respawn locations

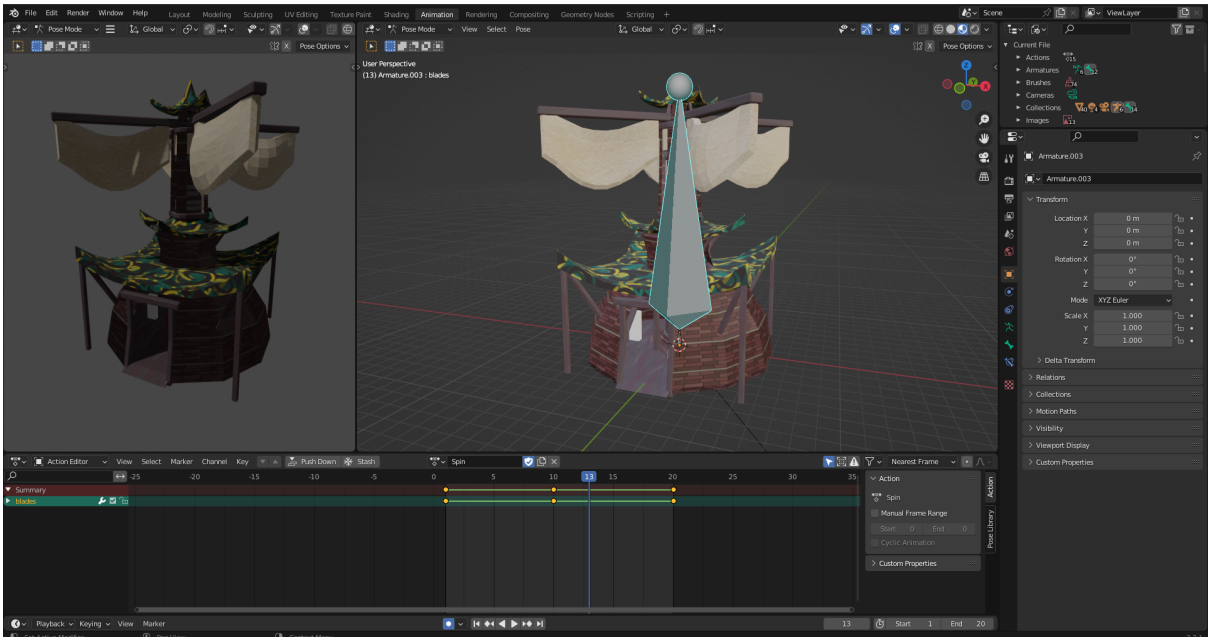
There are many smaller checks in place for polish. Landing particles only play when the player hits the ground hard enough. After air cannon boosting, the player's gravity is reset to prevent an unintuitive drop to the ground. Whenever the player leaves the ground, their last grounded location is saved in case they fall too far and need to respawn. The most recent island title is stored so it does not reappear unless the player sees a different one first.

HDRP volumetric fog, lens flare and dirt, and depth of field

I dedicated a large chunk of my time to learning lighting with HDRP to ensure an in-depth lighting setup with features like lens flare, lens dirt, and volumetric fog. The lighting of this project, in its HDRP state, was one of the project's core focuses. I found it fairly easy to work with, adding components and tweaking values until I had the atmosphere I wanted. However, I later learned that HDRP is incompatible with the computers that the project would be graded on. I rebuilt the lighting to the best of my ability in URP, though URP lacks many of HDRP's features.

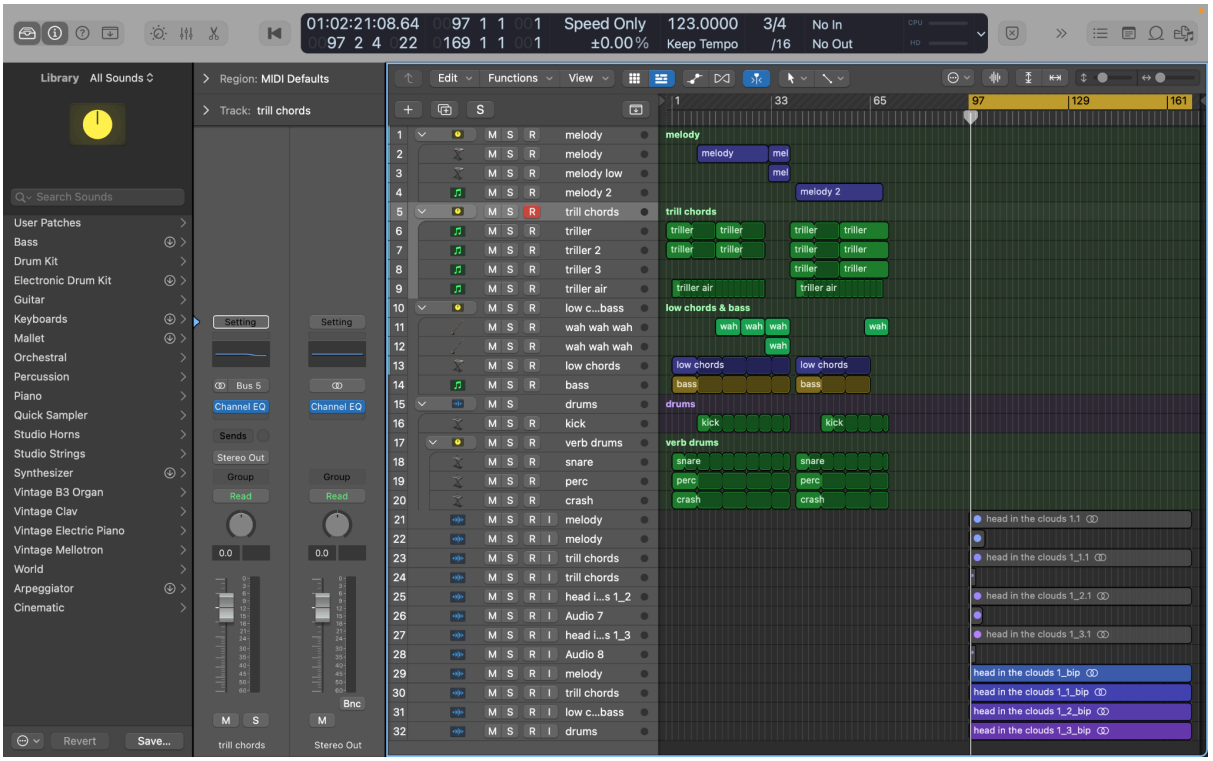


Textures (from left to right): Lens dirt, spiral cloth, wood



Windmill house animation in Blender

I created all assets for this project save for fonts, the default Unity skybox, and a handful of shader and code chunks from tutorials (referenced in-editor). I modelled and animated 3D models in Blender, then used Illustrator, Photoshop, and GIMP to create seamlessly tileable textures.



Soundtrack in Logic Pro

The music was created in Logic Pro using synths like Massive and Alchemy then bounced into four seamlessly-looping tracks to be faded in and out at runtime. The sound effects are a mix of recordings and synthesised noises. These were made in a variety of ways: water splashing from my hand in a bowl of water, grass crunching from a pitched-down recording of squeezing a plastic water bottle, waterfalls from synthesised white noise, the glider opening and closing from an umbrella, windmills creaking from opening a creaky door and thumping a metal stool, and wind from breathing.

Reflection

The player's first disappointing taste of the villagers

This project certainly met its goals. I am satisfied with the map design techniques I learned and the Unity and C# experience I gained during development. Creating character for the game was a great development of the project; I have typically done this solely through world design in the past, but the ironic theme of the game and dialogue brought its character out. I am very pleased with the level of polish I resulted with, though more quality of life improvements and edge case fixes could be added. The main menu is one of my favourite parts of the game; the moving camera adds polish and draws the player in. The dynamic soundtrack is another of my favourite features. It turned out much better than I expected in combination with gliding and dialogue.

Transparent smoke particles blur into the background due to depth of field bug

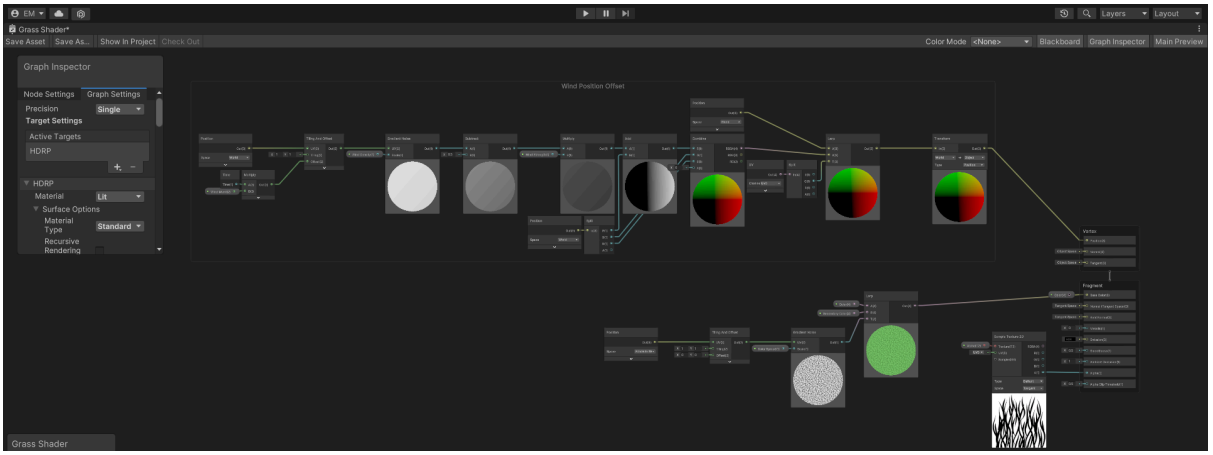
As for sound design, I believe I could have done a better job mixing audio and ensuring balance between sounds like waterfalls and footsteps. Modelling-wise, I could have pushed myself harder to detail in more depth, though the new techniques I used worked as intended. I also wish I had spent longer developing the style of the game as the map is quite visually similar to some of my past games. The level could have been developed with more foresight. Tricky objectives are not viable in a five-minute experience, which I should have known to avoid. The fixes I made later in development may do the job but would be better replaced with smarter level design. The depth of field can also cause issues with particle effects and transparent textures, visible in campfires' smoke particles and the air cannon boost's speed lines.

HDRP (top) vs. URP (bottom)

The greatest challenge of this project was changing the render pipeline in the last few days. The lighting is one of this project's cornerstones; the game feels somewhat bland and lifeless without the work I put into the HDRP build. Had I thought ahead and planned around my target device rather than jumping straight into HDRP, I could have improved the quality of the URP version. Another challenge was designing for a time constraint; I often found myself limiting my ideas to ensure the game would be short enough.

Gliding to Bog Island

Without being limited to five minutes of gameplay, I could expand the village and add puzzles the player must complete to reach villagers. I could also redesign the map to include islands that are tougher to reach and in-world hints that lead the player to characters. I would have liked to include an item management system to aid puzzle design. A character trapped in a shed could need a key owned by another character, who has mysteriously gone missing and must be hunted down through clues from the townsfolk.



Unused grass shader

I would have also liked to include grass and flowers on the islands. After fully implementing a grass shader, I found Unity does not allow the use of its foliage system on anything other than Unity terrain, meaning I would have to hand-place each pane of grass or code or purchase a foliage system. More 3D models and materials would have been nice as well; there is only one unique tree in the game and houses look very similar. In a concept sketch, I planned to add a watermill under one of the waterfalls but never got around to it. I would have loved to add fauna in the form of fish, groups of birds, and perhaps a large fantastical whale or serpent flying around the islands. A custom skybox and greater functionality for the in-game map would also be great features for future development.

Word count: 2739

References

E.T.: The Extra-Terrestrial (1982). [Film]. United States: Universal Pictures.

Game Maker's Toolkit. (2019). *Why Metro Exodus is so Immersive*. [Video]. Available at: https://youtu.be/8geGHbWIMXA?t=178 [Accessed 5 January 2023].

LEGO Games (2021). *LEGO Builder's Journey*. [Video game].

Moon Studios (2015). *Ori and the Blind Forest*. [Video game].

Nintendo (2017). *The Legend of Zelda: Breath of the Wild*. [Video game].

Nintendo (2011). *The Legend of Zelda: Skyward Sword*. [Video game].

Nintendo (2019). *Yoshi's Crafted World*. [Video game].

Poe, E. A. (1846). The Cask of Amontillado. In *Godey's Lady's Book*.

Supra Games (2019). *Supraland*. [Video game].

Team Cherry (2017). *Hollow Knight*. [Video game].

Thatgamecompany (2012). *Journey*. [Video game].