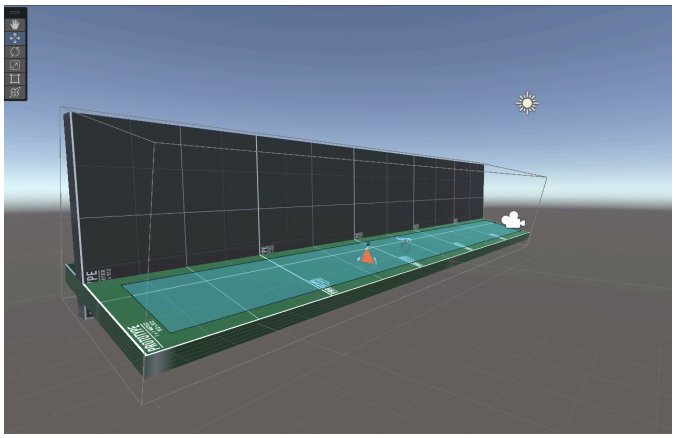
Core Gameplay - Just Strut

Design Questions Asked:

- Can we modify the feel of the player movement so walking vertically feels as quick and snappy as the horizontal movement?
- How can we improve the experience of traversing the environment in ways that aren't purely mechanical?

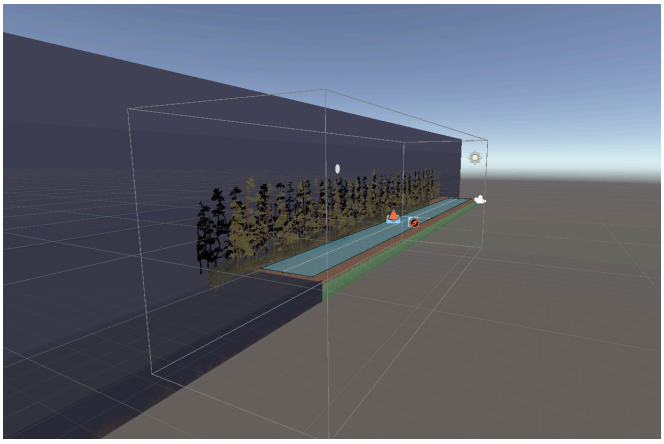
Visual Evidence:

Old LoFi Prototype Movement



https://youtu.be/gWSpXzVueWg

New Hi-Fi Prototype Movement



https://youtu.be/I6b8HjiOJg4

Description:

In the screenshots and video above, you can observe the changes done to the movement of the player. Originally, as seen in the LoFi movement, the terrain was originally slanted to provide an illusion of depth that was in line with our perspective. The vertical movement was slowed down and a tad delayed due to gravity still influencing the player as they were essentially being elevated up and down a ramp.

In our HiFi build, the terrain has been un-tilted, now requiring all of the sprite assets to instead tilt on their pivot points. The only exception to this is the floor, which should already reflect its accurate depth with the camera. Because it's not on a ramp anymore and because of some changes to make stopping more abrupt, stopping and going are now much more snappy. Outside of the mechanical changes, we've added more flourish to the experience of walking around by adding footstep sound effects.

Knowledge Gained:

1. DON'T DO SLANT PHYSICS
2. IT'S AWFUL

- But seriously, working with physics objects and gravity on a slanted surface where the slant its meant to feel like it's flat is shockingly difficult.
- Working with a flat surface with normal physics while just having sprites tilt toward the camera is much nicer to work with.
- We also decided it would be best to not have the physics colliders and hit detection tilt with the camera
    - All of that should stay vertical and proper
    - The sprites and *visual* aspects of the game should be the only thing tilting

Core Gameplay - Cheat Keys

Design Questions Asked:

- What buttons would we even need to test our game's tools?
- Should they all be placed into a single script or should they be spread out and placed into existing scripts where they make sense with their correlating cheats?
- How many do we add?
- What buttons can we use that playtesters won't accidentally hit?

Visual Evidence:

```
// If V is pressed, change the resolution state
if(Input.GetKeyDown(KeyCode.V)){
    resolutionState++;
    updateScreenRatio();
}

// If G is pressed, toggle Fullscreen
if (Input.GetKeyDown(KeyCode.G)){
    fullscreen = !fullscreen;
    updateScreenRatio();
}
```

```
if(Input.GetKeyDown(KeyCode.L)){
    TakeDamage(10);
}
if (Input.GetKeyDown(KeyCode.Semicolon)){
    Heal(10);
}
```

Video Demonstration of Cheat Keys:
https://youtu.be/-FVknhIJWeE

Description:

We've had more ideas for cheat keys since this point but these are the early ones that we had come up with (plus the resolution one for the Hi-Fi Deployment prototype report). When we initially started testing the combat and blocking mechanics we needed a way to test these things out, even when in the absence of an enemy to fight against. So we have two buttons dedicated to taking damage and healing up from damage.

The taking damage button was the first one that we put into the project so we could specifically test out the blocking mechanic before we had any enemies to attack the player with. Then, of course, the heal button came right afterwards to make sure the Hunter (player character) didn't die in the process. We have since also added a button to spawn enemies and a button to reset the scene.

Knowledge Gained:

- Putting in dev buttons for testing things is easy… so do it more!
- Doing cheat keys makes both in-game testing super easy and makes playtesting go more smoothly when you can change things mid-game.
    - One of the things that slowed early development was having to restart the scene for things to reset
- Even if these buttons end up being accidentally hit later on, you can always remap them to something else
    - or make them into a multi-button input (like pressing O first to access the cheats hidden on buttons that would normally do nothing)