Game Name: Ecto

Description Short: Jump around floating islands, collect bizarre weapons, and fight off the destructive shadow government that wants to mine for rare crystals.

Description Long: Ever walk around the woods, wielding a cool stick you found on the forest floor? Now what if you were a wizard?.. on a floating island? FIGHTING THE SHADOW GOVERNMENT?? Travel across randomly generated floating worlds, exploring and finding bizarre, stick-gun style weapons, powered by crystals, flowers and mushrooms!

Genre: FPS, Adventure, Rouguelike

In its current state, Ecto is mostly a framework of what random floating island generation could be. Islands have connection points and are placed together at random like puzzle pieces. Once the island positions are set, each island will randomly arrange its physical features, such as trees, logs and vegetation. I tried using this same system to randomly spawn in items, but that turned out to be too messy, which brings us to the next step.

Object Generation - after the islands have been laid out, it spawns in objects and items at SpawnPositions (which are collected as a list on each individual island). Right now, it is only actually used to place a glowing orb at the furthest island - the rudimentary "goal" of the level. Eventually this system will be expanded to be responsible for spawning in anything that is important to gameplay (not just for fun/visuals).

Randomness - right now, everything is evenly random. There are no rarities and most notably, no spawning "recipe". To expand on this system, there needs to be a list of islands/objects that MUST spawn. Then for the remaining islands, just pick randomly.

Game Loop - Explore a random level, find the orb, fight enemies, go through the portal, repeat.

Combat - Currently, it just spawns in waves of missiles. I'd like there to be more interesting AI. Something like space cop cars that do figure 8s around the player and shoot the missles.

Missing features of the games identity:

I'd like to focus more on the exploration part of the game. Having a plethora of interactable objects randomly spawning in. I would like to make designed mini-puzzles or interactions that give you cool items as a reward.

Examples:

- Click and drag a rock out of the way to reveal a crystal
- Click and drag to snap branches off until it reveals a stick weapon in the middle
- Pick up flowers and mushrooms to attach to your weapons.

Here are all the notable classes and their responsibilities :

- *GameManager.cs* - Loading/Deleting Levels, Spawning/Deleting Players, GameState (for multiplayer only- framework exists but I'm not using it right now)
- *LevelManager.cs* - Manager for the current game - controls the generator and the enemy spawner, fall trigger, moves the level in the battle state
- *LevelGenerator.cs* - Generates the random levels
- *EnemySpawner.cs* - Spawns in waves of enemies
- *Enemy.cs* - **Should be renamed to *DamagableNode.cs*** because its a Node3D implementation of *IDamagable.cs* using *Health.cs*
- *IDamageable.cs* - Interface for things that can be damaged
- *Health.cs* - self explanitory
- *Player.cs* - Movement, Health
- *PlayerInventory.cs* - Picking up / Dropping Items
- *PlayerInteraction.cs* - The interaction system
- *RandomObjectPicker.cs* - picks a random selection of children to keep / delete.
- *Island.cs* - Holds connectionPoints, Playerspawn, bounds of island
- *IslandObject* - Not sure if i like the name, but it is a type of thing that can be spawned onto a *SpawnPosition.* Right now, I only use it for the Orb spawning.
- *Interactable.cs* - An object the player can interact with using *PlayerInteraction.cs* - Has a position to grab, has properties for if its clicked / being clicked, has an optional property for a required item (a "key" item you need to hold)
- *Item.cs* - An object that can be picked up by *PlayerInventory.cs*. Has a reference to the corresponding Pickup scene
- *InteractablePickup.cs* - An interactable that has a reference to an *Item* packedScene, which gets picked up by *PlayerInventory.cs.*
- *UI.cs* - everything UI

Most of the code is specific to this game, but there are a few reusable ones. *IDameageable.cs*, *Health.cs* and *Utility.cs*

I used the observer pattern with the UI, and it made separation of concern much better than my previous games.

It’s easy to add new items and interactables, because you can inherit the class.