FOREST CHALLENGE

HARRISON SHOWMAN

OVERVIEW

In my game, the player starts out in a forest-like village kind of environment. It is in third person, and the player can move around exploring the area. They will notice that there are three gated off areas with see-through barriers preventing them from going to each. To unlock the gated areas, you must collect gems. The first gem is in the main area right next to the barrier on the left side, and if the player walks over it to collect it, the barrier in front will disappear, allowing the player to access the first additional area. By completing the challenge within each area, the player can get the subsequent gems. The player can keep track of how many gems they have in the bottom left of the screen the whole time. The goal of this game is to unlock all the gated areas by completing the challenges, thus unlocking the entire world.

Edits After Draft 1

The goal of the game is the same as the walking simulator; collect gems to unlock the next levels and beat all 3 levels to beat the game. The one shift in the goal of the game is that there is a speed aspect now; the game now has a timer running from when you start the first level until you finish the 3rd one. The challenge is to beat your best time playing through the game. I also changed it so that there are 2 gems in front of each level that only appear after beating the previous level (already there for level 1), and collecting the gems that have newly appeared is what takes down the barrier into the next level (there are now 6 total gems). The NPCs I added into the game also adds some personality; each one is kind of like a gatekeeper of each level.

MECHANICS

The player moves around by using WSAD. However, “A” and “D” only rotates the player, whereas “W” and “S” move the player forward and backward. The player can jump with the spacebar and can roll by right clicking. There are multiple triggers as well. Each gem is a trigger that destroys itself when hit by the player. Additionally, in the first section unlocked, each platform landed on is an additional trigger and a message will show up, telling the player they are on the right track.

Edits After Draft 1

Instead of the triggers in level one, I have NPCs walking around for each level and when you walk up to them, they tell you about what state of the game you are in. They each walk in a random direction within a defined area. The first 2 levels are just jumping on platforms to get to the gem at the top. In the first level, one of the platforms moves up and down, and in the second level, all 4 platforms move up and

down but at different speeds and different height boundaries which makes it more difficult. The third level is by far the most advanced; when you walk up to the 6 tiles on the floor, they light up green in a random order, and then go back to white. You then need to click the tiles in the same order, and if done correctly, the final gem will appear, and you finish the game. If you click out of order, it will make a new random order and repeat the process. Another big addition is the timer; the nature of the game makes it feel like a speed run type of game, where no level is too difficult but beating your time is what makes it hard.

I also added an audio soundtrack that loops throughout the entire gameplay to make the game feel more alive and engaging.

Work so far

I've created the world that my game takes place in, all with a low-poly feel to it. I made use of a prefab for a little kid that had animations attached to it that I use. I also use various assets that provide the houses/rocks/trees/ground that are in the world. I've added in a few NPCs that continuously do their idle animation (In the future, my goal is that they will say something if approached). I've also added rock barriers to provide limitations in the world. I've only finished the first level of the game, but it meets all the requirements for walking simulator.

So far, I have implemented the triggers as mentioned, and have got player/camera movement working. I set up my own animation controller, where I had to set up the transitions and set parameters that trigger these transitions.

In the next week, I will add the next two levels in and continue to improve upon what I have so far. My stretch goal is to make the levels more complex (memory game would be great), but I was running into player movement and camera issues that were making it complicated even just doing the platform jumping, so I kept it simple for the sake of this assignment.

I've made the following scripts from scratch so far:

- GameManager.cs
    - Keeps track of how many gems have been collected and deactivates barriers accordingly. Also keeps track of what platform the player is on in the first level to show the correct message
- CollisionBehavior.cs
    - Added to all of the triggers in the world, and takes action depending on what the trigger is supposed to do, making use of GameManager.cs
- PlayerMovement.cs
    - Implements how the player moves around the world, and also includes activating the animation transitions when certain buttons are pressed
- CameraMove.cs
    - Makes it so the camera that always stays behind the player, allowing the player to look up and down a restricted amount as well

Edits for Element Collector

I ended up using the default 3rd person controller and camera because it was taking too much time to make those scripts from scratch myself. The biggest changes I made was getting the 3 levels complete.

I have made the following scripts supporting the gameplay:

- CollisionBehavior.cs
    - Same behavior as for walking simulator
- GameManager.cs
    - Still keeps track of gem count and takes down barriers accordingly. Now it also keeps track of the timer, and the game state, which affects what the NPCs dialogue is when you approach them.
- NPC_Behavior.cs
    - Makes the NPCs walk around randomly in a defined area. Detects if the player is nearby and if so, it stops walking, faces the player, and a dialogue box pops up that is based on the current game state.
- Dialogue.cs
    - Holds the actual dialogue lines that the NPCs could say, and an IEnumerator that makes the dialogue box disappear after 3 seconds. These methods are called by NPC_Behavior so it knows what dialogue box to show.
- GridGame.cs
    - Used for the third level. When you start the level, a random order from 1-6 is created and the tiles light up in this random order. Then, when the player presses enter, the tiles return to their original color. Then, the player has to click the tiles in the same order as they lit up. If done correctly, the final gem appears that allows the player to finish the game. If done incorrectly, the game restarts and a new random order is created and the process is repeated.
- UpDownPlatform.cs
    - Makes platforms move up and down with a specified minimum and maximum height and movement speed.
- RotatingGem.cs
    - Simple script that just makes the gems rotate slowly.