Final Project Report

Hrg Sara, 12440404

Horror survival video game – ***Hush***

Game Overview

It has been, for a longer time now, my wish to be able to create a horror genre video game in 3D. Having a requirement to use procedural generation in this project, I decided to go with a classic "survive the maze" scenario where exactly PCG would be the most useful. Having to leave the maze and survive as being the only objective of this game, awareness of the player is the core thing that drives the gameplay. Being fan of the good old Slenderman game, I decided to also create an SCP like monster that would exist somewhere in the maze and behave solely based on that awareness. Awareness of a player is less forgiving as danger level rises, and player should use the audio cues to figure out how to keep that danger level as low as possible, and either calm the monster, or make it follow them all over the maze.

PCG

Not having enough experience with doing 3D art, I thought PCG would also save me a lot of time and energy when it comes to building the whole world as well as making it *look* good. In one of the earliest stages of implementing this game, I decided to go with Recursive Backtracking algorithm which ended up being the perfect choice for a horror game. I created script which would generate a 2D grid of cells and then with the usage of depth-first search (DFS) backtracking method, I implemented an actual maze carving.

I also made sure the maze has an exit placement needed for the win condition, and decided to place it at a random spot, as long as it was somewhere on the outer wall. Entrance, however, stayed at 0,0, which is also an exact spot for the player spawn.

For the 3D look, I created a separate scene of only one block which would just generate as much as possible during the run time until it creates the maze.

After a successful playthrough with a "dummy" player I saw that there were no impossible layouts of the maze and the exit was spawned properly. Additionally, the maze had a different layout on every run, and it was just enough complicated to make a perfect horror game run, marking it as a successful PCG maze implementation. What was left to do is the enemy-player logic, which was much more complicated to do than the maze.

Player logic

Multiple systems are important for the player – enemy logic to function properly. Player awareness is the key to winning. Having that idea in mind, I wanted their actions to impact the gameplay, which is why I implemented:

Movement system: player has unlimited walking and very limited sprinting.

Stamina system: sprinting drains stamina, walking recharges it. Completely drained stamina locks the sprint, while at 35% recharge sprint is unlocked again.

Noise and danger system: threat reacts to how long you have been loud. Low danger makes enemy anchors far away and slowly while higher danger triggers it to anchors closer and in a fast manner. Basically, if you sprint, danger rises, detection increases.

In conclusion, awareness is the core design idea of the game, and I based the gameplay on these questions: What does the player currently see? Is the enemy in that view? How long has the enemy been observed? Did the player look away?

Enemy logic

When it comes to my enemy creature, I created it in the form of a logical presence that always exists, rather than having a physical body that teleports. The idea was, for it to always have a place, but only sometimes have a body. That said, I made my enemy live in different states.

1. Enemy exists as an anchor. The enemy is assigned to one of many pre-generated anchors, and that anchor changes based on player's style of play. Basically, enemy re-anchors itself faster and closer to the player, if a player makes too much noise.
2. It is close, but you don't see it. This is where I used directional audios and calculated distance between player and the enemy. For the far distance I used low ominous music, for the closer, mid one I used that same music plus breathing sound effect, while for the very close range I made breathing louder. Enemy is still not visible, but you can conclude where he is if you follow the audio cues. He stays anchored, unseen but very much localized.
3. Manifestation. This is where a static 3D model of an enemy actually manifests and spawns on a chosen anchor. Important rule here is that he spawns at the anchor that is active at that time. The other condition is that it manifests once the player enters the near range area.
4. Observed countdown. This one is triggered when enemy is in the camera view and ray cast of the player confirms it. Here the player has a split of a second to react – turn away and survive, keep looking and get punished. Player agency becomes the most important thing again.

5. Punishment for looking. Punishment and a jump scare only happen under the previously mentioned conditions. In case player manages to run away from it, enemy logic continues.

Environment and extra stuff

Since I wanted this game to induce an uncomfortable and uneasy feeling while also creating tension, keywords I was led by were creepy, dark and foggy. I played a lot with lighting and environment system that Godot engine provides, but it ended up being a very challenging task that I kept changing until the export time.

At the end I decided to stay with foggy floor area, dim night light and the coolest part of the overall aesthetics – rain. CPU particles came in handy for that.

When it comes to textures, 3D enemy model and audio, I put respective websites/platforms and tagged people who I borrowed it from on my Itch.io page.

Things I learned

You really stop being scared after you iterate through the maze 10000000 times.