Programming learning Journal

October 19, 2021

After completing the basic framework for a procedural walking system I realised that I had overlooked an important part regarding the offset of the legs and making them step in sequence. I struggled to find a solution to the problem after coming back since I hadn't read the code in a week and it was difficult to understand what needed to be changed.

I tried setting the foot's downward raycast to originate from below the foot itself, rather than anchored to the body, but this led to both feet getting stuck in their initial positions. I believe that this is the way forward but I need to rework the way that the foot target positions are calculated. I need to ensure that the foot target positions are only changed when the foot gets within range. To do this, I need to ensure that the ray is only cast at the start, and successively each time the foot's downward ray reaches the target.

I noticed that my system was working only by accident, and that the legs were snapping to the target positions as soon as they were defined, which was not how it was meant to work. I wanted the targets to appear ahead of where the feet would snap, rather than where the feet were currently snapped. This would require a great deal of reworking of the script.

I came to the realization that I needed a third raycast, down from the body to detect when the body was in range of the foot target. This would then snap the feet and update the foot target when in range.

November 01, 2021

Here's what needs to happen:

Get "newpos" by raycasting down from a position above each IK target and ahead by stepdist

Raycast from hips to ground to detect when in range of the "newpos"

November 02, 2021

Ideas for packages: Set of puzzle objects/obstacles

Pads that require certain tagged objects to be placed/consumed

Using unity events to trigger functions in other scripts (use door/elevator in example)

Puzzle elements can be disabled and require an event to fire to become usable

Power routing? Lasers, mirrors and recievers.

November 16, 2021

While making a laser reflection simulation, I came across an interesting problem. Since I want this to be a reusable component to be used in any situation where a laser might be needed, I wanted it to be able to trigger a variety of functions. I wanted any laser to be able to trigger any number of different responses from different objects, but with unity's events system I would have to manually add each script to its respective laser. This would mean that two different lasers might not be able to trigger the same event.

My proposed solution is to come up with some method of identifying functions that should be run when a gameobject is hit by a laser, such as a specific name or class. I settled on using “SendMessage” and using a standardised name for any functions that should be triggered by a laser.