# Using the forward vector for motion

A screen shot of a computer code

Description automatically generated

To give a character motion in a 3D space, you need to give it a velocity value, just like in 2D. Here, however, there are three dimensions, with the addition of the z axis. So, to make our character move, we simply assign a value to the vector’s z value if the ‘forward’ button is being pressed. If it isn’t, the velocity will go back to nothing, stopping all motion. The reason I multiply by -1 is because the character was moving backwards. I’m not sure if this was a flaw in my code, but the multiplication seemed to fix it.

# How rotation in 3D works

A black background with white text and green text

Description automatically generated

I’m sure there are a hundred ways of doing rotation in 3D, but here we used the InputEventMouseMotion event. I checked if the event was the mouse motion, and if it was, the relative y value of the motion is added to the rotation value of the camera. The problem with this, however, was that the camera would move WAY too fast, so I slowed it down to 5% of the original speed. That helped remedy the problem, but it’s still a bit fast. There must be a better way to do 3D rotations.

# Complexities of setting up an object for 3D motion

The biggest complexity of setting up an object for 3D motion, I think, is understanding that there’s now a third dimension that needs to be calculated. The x and y axis always existed, and we knew how to calculate the vectors for that, but now we need to consider the z axis as well. We also now have to consider any rotations that may be made. If the camera rotates 30 degrees, the player’s movement should reflect that, and the player should rotate as well. This is especially true if the player is in motion. When the camera turns while the player is walking, the player should turn too.

# Advantages and disadvantages of the area3D method of interacting with objects.

A screenshot of a video game

Description automatically generated

Advantages:

* Just like in 2D, using an Area3D allows us to easily create interactive prompts. If *something* is in the area, it’s easy to send a signal to a script and make something happen.
* It also allows developers to make something happen whether the player knows it or not. For example, if a player walks through the doorway of, say, a dungeon, an area3D can be used to signal that the player is in the room. Then the game controller can spawn in whatever needs to be spawned.

Disadvantages:

* One drawback of using an area3D as a method of interacting with objects in the game is that the player just has to be in the area, they don’t have to be looking at the object that’s interactable. So, if you have a button, the player *should* have to look at it to press it. The problem is, with an Area3D, they don’t have to look at it. They just need to be in the space.