

Thomas Zugrav – Learning Journal

14/12/2024

A minor issue I encountered today when trying to implement a sprinting functionality into my game prototype was that I would receive some strange jolting effect whenever I would test to see if my sprinting worked.

I realised shortly that I had used the wrong Check to see if my LeftShift was being held or pressed once. I had used (Input.GetKeyDown(KeyCode.LeftShift)) to check for the key being held because I was used to using GetKeyDown instead of just GetKey when dealing with inputs.

After some testing and re looking at my code I realised this error and changed it to the right check which fixed my issue and made my sprinting smooth!

```
Vector3 move = transform.right * x + transform.forward * z;

if (Input.GetKey(KeyCode.LeftShift))
{
    controller.Move(sprintSpeed * Time.deltaTime * move);
}
else
{
    controller.Move(speed * Time.deltaTime * move);
}
```

20/12/2024

Today I ran into an issue when updating my “fetch item” script in my RPG prototype game. I am making this script to be able to finish my fetch quest for my game.

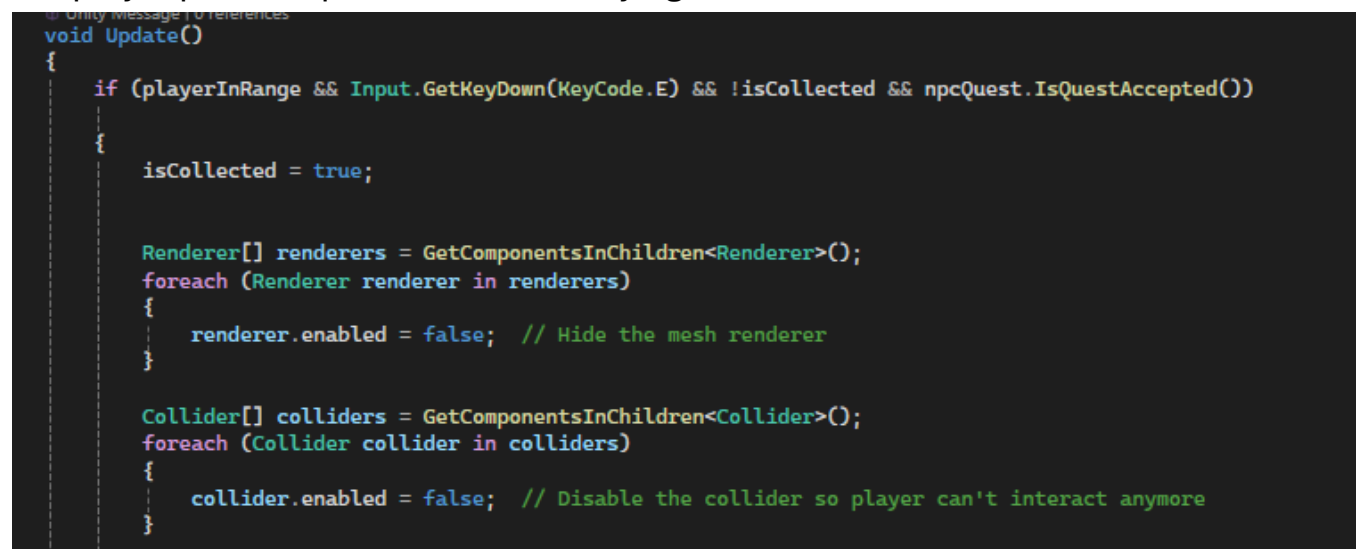
This script is attached to an object that is meant to be picked up by the player to hand in to an NPC.

The issue I encountered is that when the player would pick up this item the quest tracker and the prompt for picking it up wouldn't update. In my code,

after the player picks the item up they are supposed to be greeted with a message saying, “You have now collected the item!” and an update to the quest tracker guiding the player to return to the npc none of which was actually happening when testing.

After some deep thought on why this was happening I thought, maybe the TextMeshPro component couldn’t be accessed any longer after the player picks the item up as the item is destroyed. I looked this up and that is in fact true that when a object is destroyed all of it’s components become inactive.

So a quick fix for this was to instead hide the mesh and the collider when the player picks it up instead of destroying it. I did this like so:



```
void Update()
{
    if (playerInRange && Input.GetKeyDown(KeyCode.E) && !isCollected && npcQuest.IsQuestAccepted())
    {
        isCollected = true;

        Renderer[] renderers = GetComponentsInChildren<Renderer>();
        foreach (Renderer renderer in renderers)
        {
            renderer.enabled = false; // Hide the mesh renderer
        }

        Collider[] colliders = GetComponentsInChildren<Collider>();
        foreach (Collider collider in colliders)
        {
            collider.enabled = false; // Disable the collider so player can't interact anymore
        }
    }
}
```

Before, in the update I simply had a function to destroy the object when the player is in it’s collider and presses E. This way makes it so the messages do update.

28/12/2024

Today I ran into another issue when deciding to continue working on my fetch item script. This time with an issue regarding a displayed message not clearing after using “Invoke”.

I had used Invoke before to clear messages in UI so I thought I would try it again to clear a message the player receives after picking up the item however, no matter what I would set the f value for the invoke it would never clear. I started with 3f so it should clear the message after 3 seconds which it never did.

I never really found out the reason why this was happening but since I couldn't get it to work I thought I might as well just use a different method of clearing the text which would be a Coroutine.

I added the name space "System.Collections;" to my script so that I could access Coroutine and simply replaced my invoke code with a StartCoroutine.

```
        DisplayMessage("Item collected! Now return to NPC.");  
        if (npcQuest != null)  
        {  
            npcQuest.MarkQuestComplete();  
        }  
  
        StartCoroutine(ClearMessageAfterDelay(2.4f));  
    }  
}
```

And at the bottom of my script added a IEnumerator so that the coroutine would work.

```
private IEnumerator ClearMessageAfterDelay(float delay)  
{  
    yield return new WaitForSeconds(delay);  
    ClearQuestMessage();  
}  
}
```

In the end I got the outcome I was trying to achieve even though it took a couple extra steps. I'm still confused on why the invoke didn't work but I will try to look more into it.

