Particle effect activates when player leaves the tunnel

When the player passes through a trigger, a particle effect is enabled for a period of time.

1) To begin, place some basic objects for the player to walk around on, then add the player controls (for this demo I simply used a preset FPS controller).

2) Create a cube and place it on a section of the floor, on the above demo I placed it inside of a tunnel, where a separate behavior teleports the player to another location. Turn off the Mesh Renderer in the object's properties, and tick "Is Trigger" in the box collider.
3) Create a particle system and place it in the scene (For this demo I had it in front of the tunnel that the player exits from once teleported).

4) Create an empty object in the scene and make it a parent with the particle system in the hierarchy. This is due to particle systems not including "SetActive" functions, so assigning the function to the empty object will disable anything that it's parenting.

5) Add this code to the Trigger:

```
public GameObject WP;

void Start () {
    WP.SetActive(false);
}

private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.tag == "Player")
    {
        WP.SetActive(true);
        StartCoroutine (turnoff ());
    }
}

IEnumerator turnoff()
{
    yield return new WaitForSeconds(2f);
    WP.SetActive(false);
}
```

6) Create a “GameObject” variable. The parent object will be assigned to this.

5) In the Start event, type “(GameObject).SetActive(false);”. This will disable the particle system from the start of the scene.

6) Create an “OnTriggerEnter” event and type the if statement, “if (other.gameObject.tag == “Player”)”. This will make the trigger only activate if the object that passes through has the tag “Player”, which needs to be assigned in the player model’s properties.

7) Inside the if statement, type “WP.SetActive(true);” and “StartCoroutine (turnoff());”. When the player enters the trigger, this will enable the particle system. The second piece of code acts as an alarm for a function to activate at a specific time. In this case it will be the amount of time before the particle system is disabled again.

8) To define the timer, you will need the “IEnumerator” event. Next to the event you must also include the method name, which needs to be the same as the name of the coroutine.

9) Inside of the “IEnumerator” event, type “yield return new WaitForSeconds(2f);”. This will call how many seconds of the scene the function will wait for until the below code becomes active, which is “WP.SetActive(false)”.