Challenge 4
Soccer Scripting

Challenge Overview:
Use the skills you learned in the Sumo Battle prototype in a completely different context: the soccer field. Just like in the prototype, you will control a ball by rotating the camera around it and applying a forward force, but instead of knocking them off the edge, your goal is to knock them into the opposing net while they try to get into your net. Just like in the Sumo Battle, after every round a new wave will spawn with more enemy balls, putting your defense to the test. However, almost nothing in this project is functioning! It’s your job to get it working correctly.

Challenge Outcome:
- Enemies move towards your net, but you can hit them to deflect them away
- Powerups apply a temporary strength boost, then disappear after 5 seconds
- When there are no more enemy balls, a new wave spawns with 1 more enemy

Challenge Objectives:
In this challenge, you will reinforce the following skills/concepts:
- Defining Vectors by subtracting one location in 3D space from another
- Track the number of objects of a certain type in a scene to trigger certain events
- Using Coroutines to perform actions based on a timed interval
- Using for-loops and dynamic variables to run code a particular number of times
- Resolving errors related to null references of unassigned variables

Challenge Instructions:
- Open your Prototype 4 project
- Download the "Challenge 4 Starter Files" from the Tutorial Materials section, then double-click on it to Import
- In the Project Window > Assets > Challenge 4 > Instructions folder, use the resources as a guide to complete this challenge
<table>
<thead>
<tr>
<th>Challenge</th>
<th>Task</th>
<th>Hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hitting an enemy sends it back towards you</td>
<td>When you hit an enemy, it should send it away from the player</td>
</tr>
<tr>
<td>2</td>
<td>A new wave spawns when the player gets a powerup</td>
<td>A new wave should spawn when all enemy balls have been removed</td>
</tr>
<tr>
<td>3</td>
<td>The powerup never goes away</td>
<td>The powerup should only last for a certain duration, then disappear</td>
</tr>
<tr>
<td>4</td>
<td>2 enemies are spawned in every wave</td>
<td>One enemy should be spawned in wave 1, two in wave 2, three in wave 3, etc</td>
</tr>
<tr>
<td>5</td>
<td>The enemy balls are not moving anywhere</td>
<td>The enemy balls should go towards the “Player Goal” object</td>
</tr>
</tbody>
</table>

**Bonus Challenge**

<table>
<thead>
<tr>
<th>Task</th>
<th>Hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>The player needs a turbo boost</td>
</tr>
<tr>
<td>Y</td>
<td>The enemies never get more difficult</td>
</tr>
</tbody>
</table>
**Challenge Solution**

1. In PlayerControllerX.cs, in OnCollisionEnter(), the `awayFromPlayer` Vector3 is in the opposite direction it should be.

```csharp
Vector3 awayFromPlayer = transform.position -
other.gameObject.transform.position;
= other.gameObject.transform.position -
transform.position;
```

2. In SpawnManagerX.cs, the `enemyCount` variable is counting the number of objects with a “Powerup” tag - it should be counting the number of objects with an “Enemy” tag.

```csharp
void Update() {
    enemyCount = GameObject.FindGameObjectsWithTag("Powerup Enemy").Length;
    ...
}
```

3. In PlayerControllerX.cs, in the OnTriggerEnter() method, you need to initiate the `PowerupCooldown` Coroutine in order to begin the countdown process.

```csharp
private void OnTriggerEnter(Collider other) {
    if (other.gameObject.CompareTag("Powerup")) {
        ...
        StartCoroutine(PowerupCooldown());
    }
}
```

4. In SpawnManagerX.cs, the for-loop that spawns enemy should make use of the `enemiesToSpawn` parameter.

```csharp
for (int i = 0; i < 2-enemiesToSpawn; i++) {
    Instantiate(enemyPrefab, GenerateSpawnPosition(), ...
}
```

5. In EnemyX.cs, the `playerGoal` variable is not initialized - initialize it in the Start() method.

```csharp
void Start() {
    enemyRb = GetComponent<Rigidbody>();
    playerGoal = GameObject.Find("Player Goal");
}
```
**Bonus Challenge Solution**

**X1** To add a turbo boost, in `PlayerControllerX.cs`, declare a new `turboBoost` float variable, then in `Update()`, add a simple if-statement that adds an “impulse” force in the direction of the focal point if spacebar is pressed:

```csharp
private float turboBoost = 10;

void Update() {
    ...
    if (Input.GetKeyDown(KeyCode.Space)) {
        playerRb.AddForce(focalPoint.transform.forward * turboBoost, ForceMode.Impulse);
    }
}
```

**X2** Add the Smoke_Particle prefab as a child object of the focal point (next to the camera), then in `PlayerControllerX.cs`, declare a new `turboSmoke` particle variable and assign it in the inspector.

**X3** In `PlayerControllerX.cs`, in the if-statement checking if the player presses spacebar, play the particle

```csharp
if (Input.GetKeyDown(KeyCode.Space)) {
    playerRb.AddForce(focalPoint.transform.forward * turboBoost, ForceMode.Impulse);
    turboSmoke.Play();
}
```

**Y1** In `SpawnManagerX.cs`, declare and initialize a new public `enemySpeed` variable, then increase it by a certain amount every time a wave is spawned:

```csharp
public int enemyCount;
public float enemySpeed = 50;

void SpawnEnemyWave(int enemiesToSpawn) {
    ...
    waveCount++;
    enemyCount += 25;
}
```
In EnemyX.cs, declare a new spawnManagerXScript variable, get a reference to it in Start(), then set the enemy's speed variable to your new enemySpeed variable

```csharp
private GameObject playerGoal;
private SpawnManagerX spawnManagerXScript;

void Start() {
    enemyRb = GetComponent<Rigidbody>();
    playerGoal = GameObject.Find("Player Goal");
    spawnManagerXScript = GameObject.Find("Spawn Manager").GetComponent<SpawnManagerX>();
    speed = spawnManagerXScript.enemySpeed;
}
```

To test, make the speed variable in EnemyX.cs public and check the enemies’ speed when they are spawned in different waves