unity

4.4 For-Loops For Waves

Steps:

Step 1: Write a for-loop to spawn 3 enemies

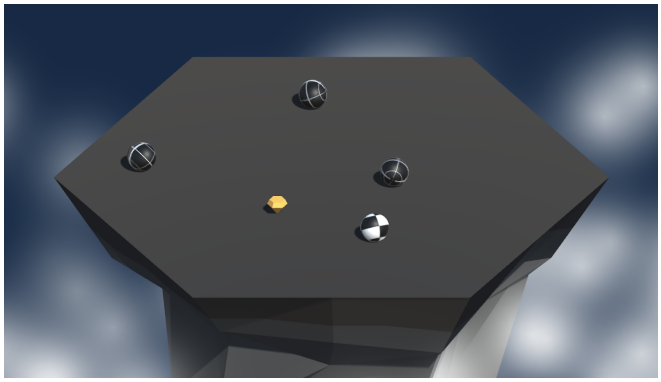
Step 2: Give the for-loop a parameter

Step 3: Destroy enemies if they fall off

Step 4: Increase enemyCount with waves

Step 5: Spawn Powerups with new waves

Example of project by end of lesson



Length: 60 minutes

Overview: We have all the makings of a great game; A player that rolls around and rotates the camera, a powerup that grants super strength, and an enemy that chases the player until the bitter end. In this lesson we will wrap things up by putting these pieces together!
First we will enhance the enemy spawn manager, allowing it to spawn multiple enemies and increase their number every time a wave is defeated. Lastly we will spawn the powerup with every wave, giving the player a chance to fight back against the ever-increasing horde of enemies.

Project Outcome: The Spawn Manager will operate in waves, spawning multiple enemies and a new powerup with each iteration. Every time the enemies drop to zero, a new wave is spawned and the enemy count increases.

Learning Objectives: By the end of this lesson, you will be able to:
- Repeat functions with For-loops
- Increment integer values in a loop with the ++ operator
- Target objects in a scene with FindObjectsOfType
- Return the length of an array as an integer with .Length

Step 1: Write a for-loop to spawn 3 enemies

We should challenge the player by spawning more than one enemy. In order to do so, we will repeat enemy instantiation with a loop.

1. In SpawnManager.cs, in ***Start()***, replace single **Instantiation** with a **for-loop** that spawns 3 enemies
2. Move the for-loop to a new ***void SpawnEnemyWave()*** function, then call that function from ***Start()***

- **New Concept:** For-loops
- **Don't worry:** Loops are a bit confusing at first, but they make sense eventually. Loops are powerful tools that programmers use often
- **New Concept:** ++ Increment Operator

```
void Start() {
  SpawnEnemyWave();
  for (int i = 0; i < 3; i++) {
    Instantiate(enemyPrefab, GenerateSpawnPosition(),
    enemyPrefab.transform.rotation); } }

void SpawnEnemyWave() {
  for (int i = 0; i < 3; i++) {
    Instantiate(enemyPrefab, GenerateSpawnPosition(),
    enemyPrefab.transform.rotation); } }
```

Step 2: Give the for-loop a parameter

Right now, SpawnEnemyWave spawns exactly 3 enemies, but if we're going to dynamically increase the number of enemies that spawn during gameplay, we need to be able to pass information to that method.

1. Add a parameter ***int enemiesToSpawn*** to the ***SpawnEnemyWave*** function
2. Replace ***i < __*** with ***i < enemiesToSpawn***
3. Add this new variable to the function call in ***Start()***: ***SpawnEnemyWave(___);***

- **New Concept:** Custom methods with parameters
- **Tip:** GenerateSpawnPosition returns a value, SpawnEnemyWave does not. SpawnEnemyWave takes a parameter, GenerateSpawnPosition does not.

```
void Start() {
  SpawnEnemyWave(3); }

void SpawnEnemyWave(int enemiesToSpawn) {
  for (int i = 0; i < 3 enemiesToSpawn; i++) {
    Instantiate(enemyPrefab, GenerateSpawnPosition(),
    enemyPrefab.transform.rotation); } }
```

Step 3: Destroy enemies if they fall off

Once the player gets rid of all the enemies, they're left feeling a bit lonely. We need to destroy enemies that fall, and spawn a new enemy wave once the last one is vanquished!

1. In Enemy.cs, **destroy** the enemies if their position is less than a **-Y value**
2. In SpawnManager.cs, declare a new ***public int enemyCount*** variable
3. In ***Update()***, set ***enemyCount = FindObjectsOfType<Enemy>().Length;***
4. Write the **if-statement** that if ***enemyCount == 0*** then ***SpawnEnemyWave***

- **New Function:** FindObjectsOfType

```
void Update() {
  ... if (transform.position.y < -10) { Destroy(gameObject); } }

<------>
public int enemyCount

void Update() {
  enemyCount = FindObjectsOfType<Enemy>().Length;
  if (enemyCount == 0) { SpawnEnemyWave(1); } }
```

Step 4: Increase enemyCount with waves

Now that we control the amount of enemies that spawn, we should increase their number in waves. Every time the player defeats a wave of enemies, more should rise to take their place.

1. Declare a new ***public int waveNumber = 1;***, then implement it in ***SpawnEnemyWave(waveNumber);***
2. In the if-statement that tests if there are 0 enemies left, **increment waveNumber** by 1

- **Tip:** Incrementing with the ++ operator is very handy, you may find yourself using it in the future

```
public int waveNumber = 1;

void Start() {
  SpawnEnemyWave(3 waveNumber); }

void Update() {
  enemyCount = FindObjectsOfType<Enemy>().Length;
  if (enemyCount == 0) { waveNumber++;  SpawnEnemyWave(1 waveNumber); } }
```

Step 5: Spawn Powerups with new waves

Our game is almost complete, but we're missing something. Enemies continue to spawn with every wave, but the powerup gets used once and disappears forever, leaving the player vulnerable. We need to spawn the powerup in a random position with every wave, so the player has a chance to fight back.

1. In SpawnManager.cs, declare a new ***public GameObject powerupPrefab*** variable, assign the **prefab** in the inspector and **delete** it from the scene
2. In **Start()**, **Instantiate** a new Powerup
3. Before the ***SpawnEnemyWave()*** call, **Instantiate** a new Powerup

- **Tip:** Now that we have a very playable game, let's test and tweak values

```
public GameObject powerupPrefab;

void Start() {
  ... Instantiate(powerupPrefab, GenerateSpawnPosition(),
  powerupPrefab.transform.rotation); }

void Update() {
  ... if (enemyCount == 0) { ... Instantiate(powerupPrefab,
    GenerateSpawnPosition(), powerupPrefab.transform.rotation); } }
```

Lesson Recap

New Functionality	• Enemies spawn in waves • The number of enemies spawned increases after every wave is defeated • A new power up spawns with every wave
New Concepts and Skills	• For-loops • Increment (++) operator • Custom methods with parameters • FindObjectsOfType