Game Programming

Lecture IV
Spawn Manager

Izzet Fatih Senturk

Spawn Manager Setup

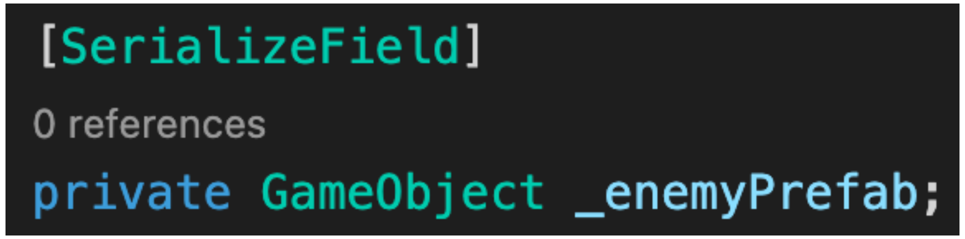
- Spawn enemies continuously while the game is running
  - Spawn enemies randomly with a time interval
- Spawn manager is not a physical object that we can see
  - But a behavior of spawning objects
- Create an empty game object: “Spawn_Manager”
- Create a new C# script: “SpawnManager_sc”
  - Add the new script to the game object

Coroutines

- How to delay events in Unity?
- When you call a function, it runs to completion before returning
  - Any action taking place in a function must happen within a single frame update
  - How to reduce opacity of an object gradually?
- Coroutine
  - Like a function but has the ability to pause execution and return control to Unity
  - Then continue where it left off on the following frame

Spawn Manager

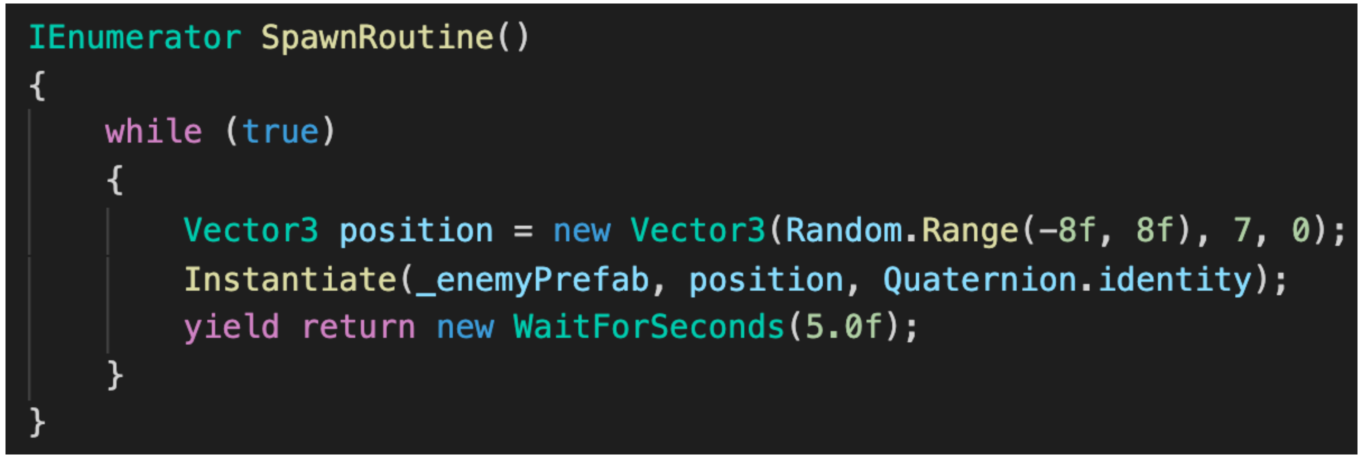
- Enemy is instantiated from “Enemy” prefab
  - Define the prefab variable in “SpawnManager_sc”
- Delete “Enemy” object from the Hierarchy window
- Set the ”Enemy” prefab variable of the “SpawnManager_sc”



```
[SerializeField]
0 references
private GameObject _enemyPrefab;
```

Instantiate Enemies

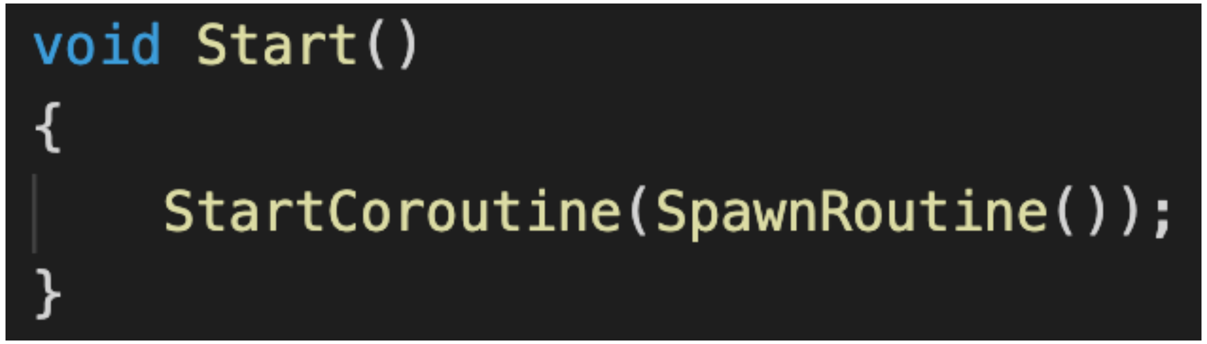
- Identify the “Enemy” position
  - Set x randomly between -8 and 8
  - Y is set to top of the screen
- Instantiate the “Enemy”
- Wait for 5 seconds before instantiating another one



```
IEnumerator SpawnRoutine()
{
    while (true)
    {
        Vector3 position = new Vector3(Random.Range(-8f, 8f), 7, 0);
        Instantiate(_enemyPrefab, position, Quaternion.identity);
        yield return new WaitForSeconds(5.0f);
    }
}
```

Start the Coroutine

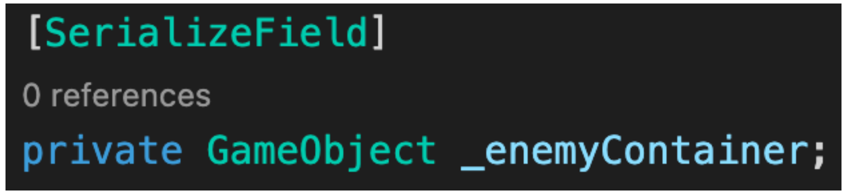
- A special method to start coroutines: StartCoroutine
- Call the StartCoroutine method by
  - StartCoroutine("SpawnRoutine")
  - StartCoroutine(SpawnRoutine())



```
void Start()
{
    StartCoroutine(SpawnRoutine());
}
```

Tidy up Spawning

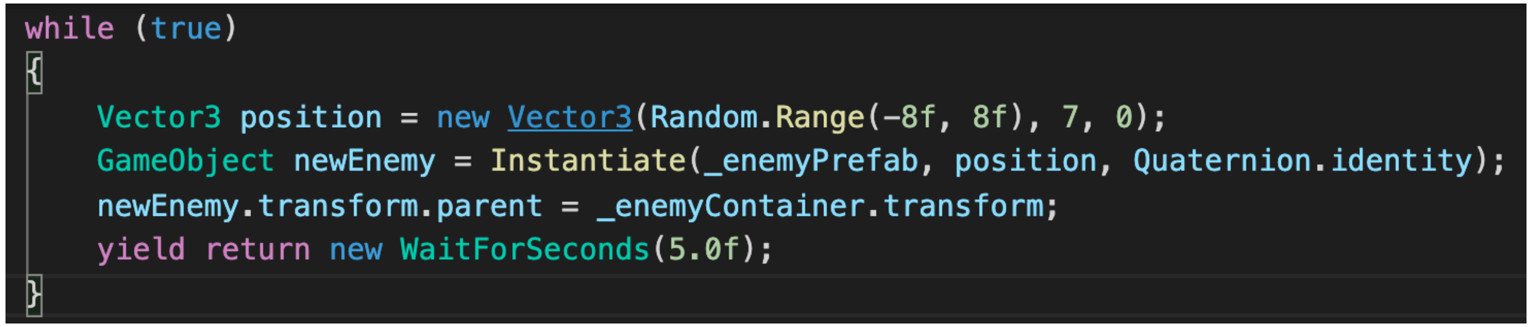
- The number of “Enemy” objects is increasing in the Hierarchy
  - Organize ”Enemy” objects for a better debugging
  - Make the “Enemy” objects child of “Spawn_Manager”
- Under “Spawn_Manager”, create an empty object: “Enemy_Container” as a container for “Enemy” objects
- In “SpawnManager_cs” create a variable for the container
  - Drag the “Enemy_Container” on the variable



```
[SerializeField]
0 references
private GameObject _enemyContainer;
```

Tidy up Spawning

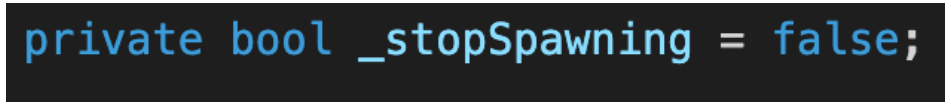
- Set the “Enemy” objects as the child of the “Enemy_Container”
  - Assign the parent of the new “Enemy” objects



```
while (true)
{
    Vector3 position = new Vector3(Random.Range(-8f, 8f), 7, 0);
    GameObject newEnemy = Instantiate(_enemyPrefab, position, Quaternion.identity);
    newEnemy.transform.parent = _enemyContainer.transform;
    yield return new WaitForSeconds(5.0f);
}
```

Stop Spawning

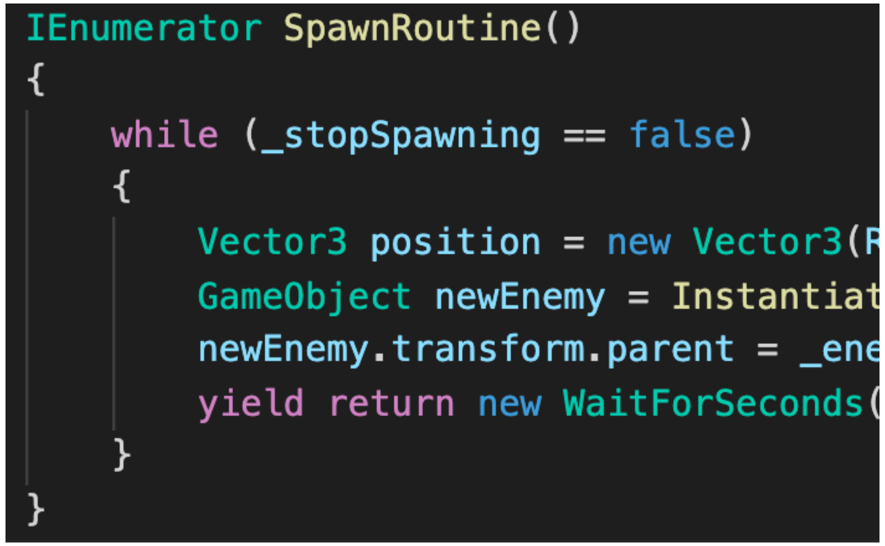
- Stop spawning new “Enemy” objects when the “Player” dies
- Script communication between “Player_sc” and “SpawnManager_sc”
- Create a Boolean variable in “SpawnManager_sc” to check if we need to stop spawning



```
private bool _stopSpawning = false;
```

Stop Spawning

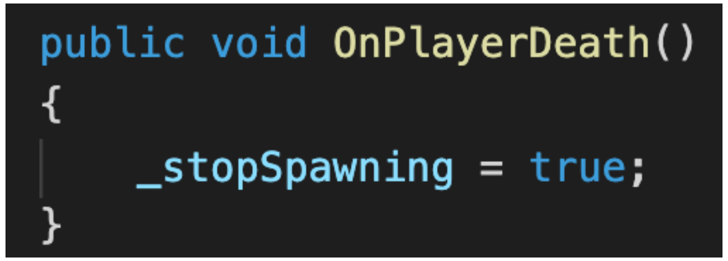
- Update the while condition of the spawn coroutine



```
IEnumerator SpawnRoutine()
{
    while (_stopSpawning == false)
    {
        Vector3 position = new Vector3(R
        GameObject newEnemy = Instantiat
        newEnemy.transform.parent = _ene
        yield return new WaitForSeconds(
    }
}
```

Stop Spawning

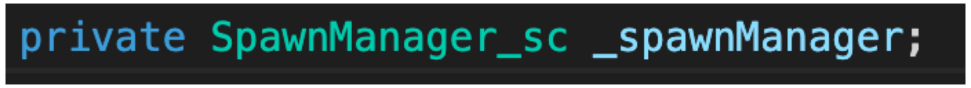
- Define a public function to set the Boolean control variable true
  - Exit infinite while loop consequently



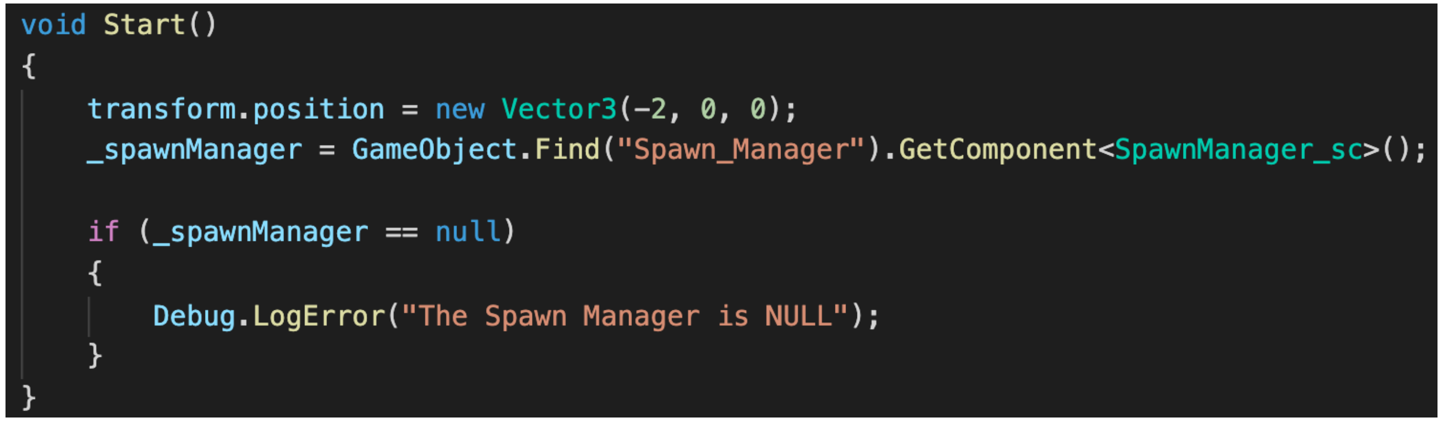
```
public void OnPlayerDeath()
{
    _stopSpawning = true;
}
```

Script Communication

- ”Player_sc” has to find “SpawnManager_sc” to communicate with it
  - Create a variable for spawn manager in “Player_sc”
  - Find the “SpawnManager_sc” script with Find function
    - Use GetComponent function to find the corresponding component



```
private SpawnManager_sc _spawnManager;
```

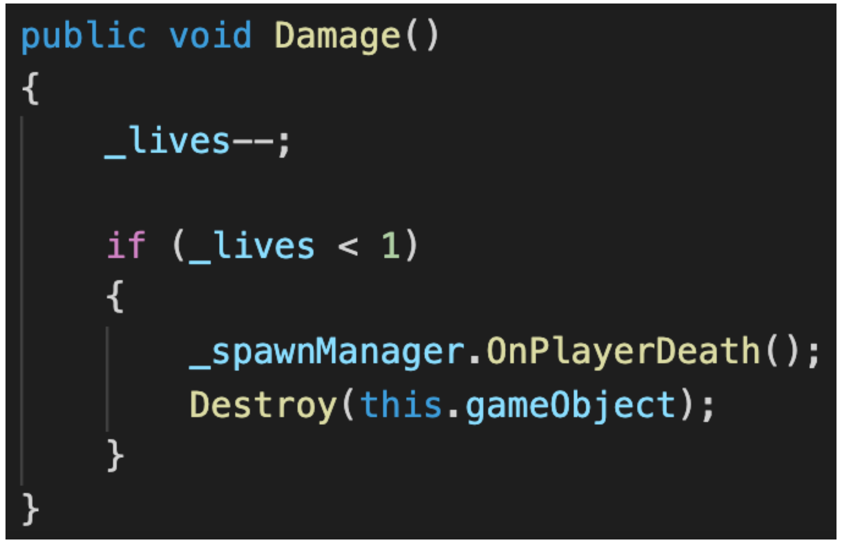


```
void Start()
{
    transform.position = new Vector3(-2, 0, 0);
    _spawnManager = GameObject.Find("Spawn_Manager").GetComponent<SpawnManager_sc>();

    if (_spawnManager == null)
    {
        Debug.LogError("The Spawn Manager is NULL");
    }
}
```

Script Communication

- Update the damage function



```
public void Damage()
{
    _lives--;

    if (_lives < 1)
    {
        _spawnManager.OnPlayerDeath();
        Destroy(this.gameObject);
    }
}
```