Game Programming

Lecture VI
Triple Shot

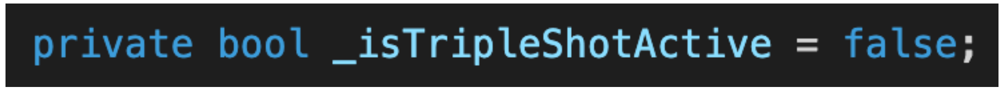
Izzet Fatih Senturk

Create the Triple Shot Prefab

- Add the Laser to Hierarchy and then make two more copies
  - Position them accordingly
    - Same y values and negative x value for the wings
- Create an empty object: Triple_Shot
  - Make 3 lasers the child of the new object
- Make it a prefab
  - Delete it from the Hierarchy

Triple Shot Behavior

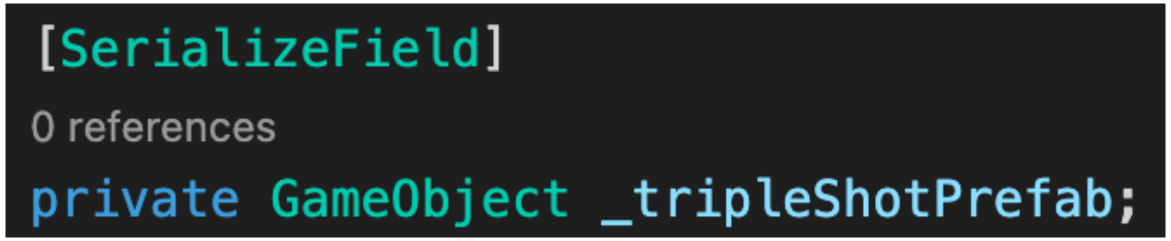
- Develop logic to decide when to use the triple shot
  - When we collect a special sprite notify the user that triple shot is enabled for a certain period of time
  - Enable triple shot and then disable after timeout
- Define a Boolean variable for the triple shot activation



```
private bool _isTripleShotActive = false;
```

Triple Shot Behavior

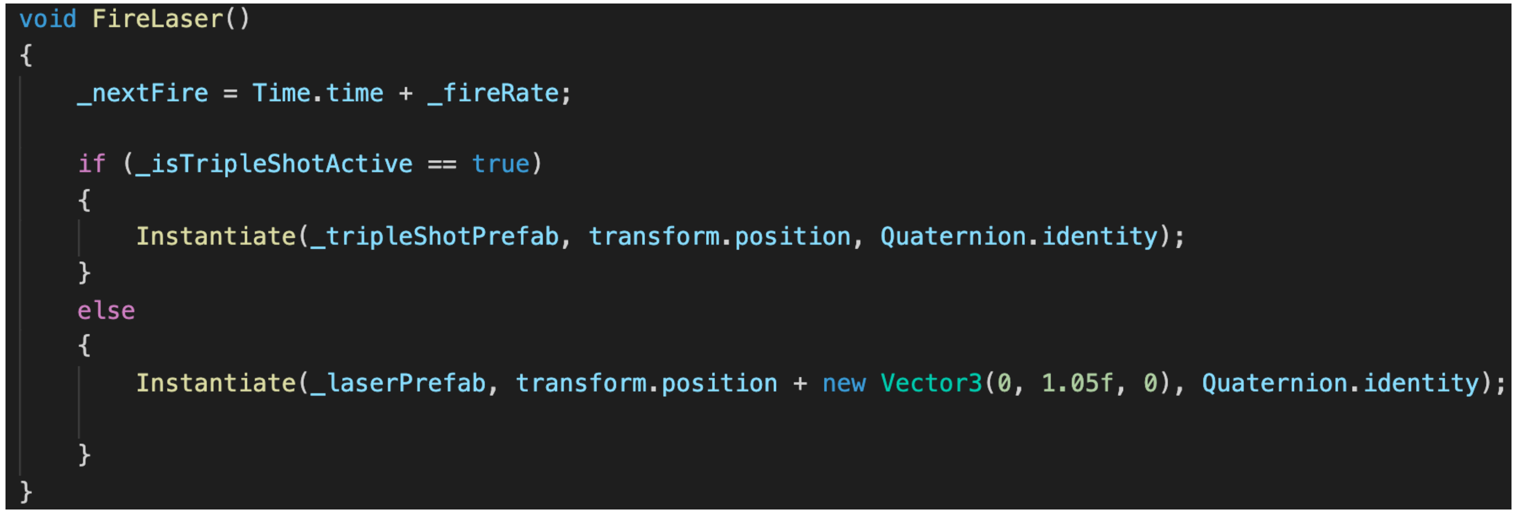
- Define a variable for the game object of the triple shot prefab



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

Triple Shot Behavior

- Update FireLaser function to consider triple shot



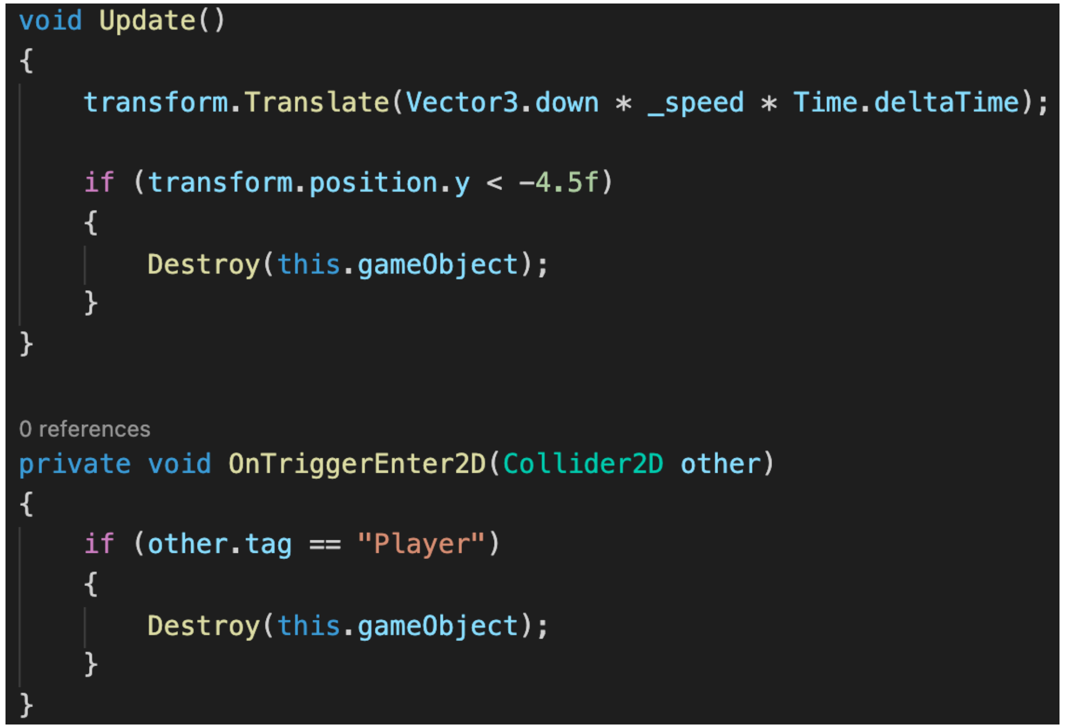
```
void FireLaser()
{
    _nextFire = Time.time + _fireRate;

    if (_isTripleShotActive == true)
    {
        Instantiate(_tripleShotPrefab, transform.position, Quaternion.identity);
    }
    else
    {
        Instantiate(_laserPrefab, transform.position + new Vector3(0, 1.05f, 0), Quaternion.identity);

    }
}
```

Triple Shot Powerup Behavior

- Add the logic to enable triple shot
  - Upon collecting the powerup sprite
- Create an object that we can collect
  - Take the first frame of the triple shot from sprites folder and drag to Hierarchy window
- Rename it to Triple_Shot_Powerup
- Scale down to 0.5 0.5 0.5
- Add a Circle Collider to be able to collect it
  - Set isTrigger true
- Add RigidBody2D
  - Set gravity scale to 0
- Set the sorting layer: “Foreground”
- Create a new C# script: Powerup_sc
  - Add the script to the object
- Make it a prefab

Triple Shot Powerup Behavior

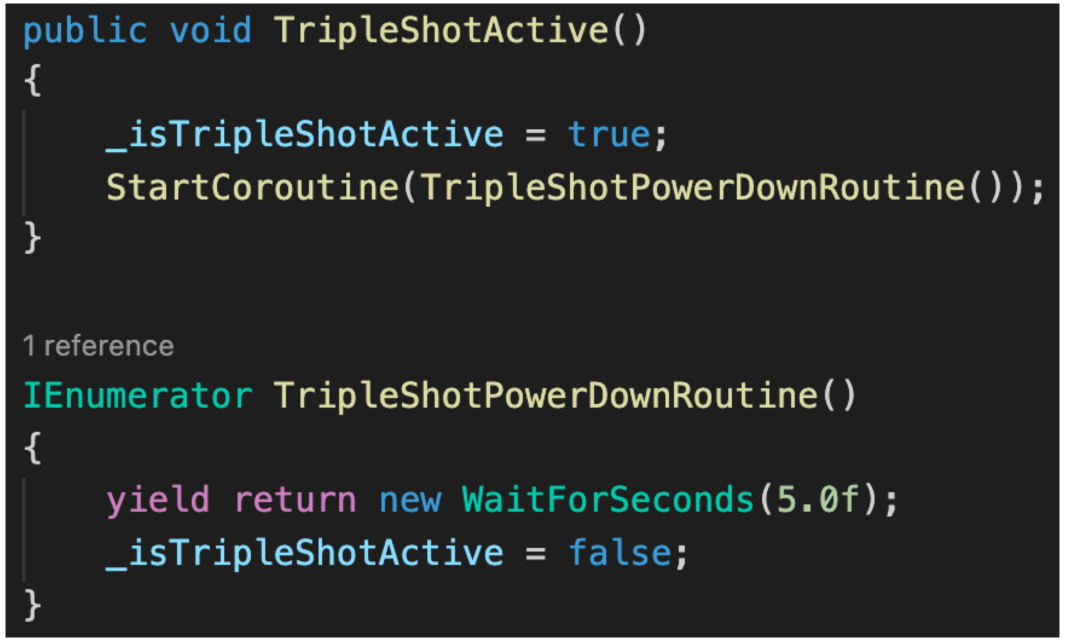
- The logic
  - Move down at a speed of 3
  - When we leave the screen, destroy the is object
  - If collides with the player (only collectible by the player), enable triple shot and destroy this object



```
void Update()
{
    transform.Translate(Vector3.down * _speed * Time.deltaTime);

    if (transform.position.y < -4.5f)
    {
        Destroy(this.gameObject);
    }
}

0 references
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        Destroy(this.gameObject);
    }
}
```

Triple Shot Powerup Behavior

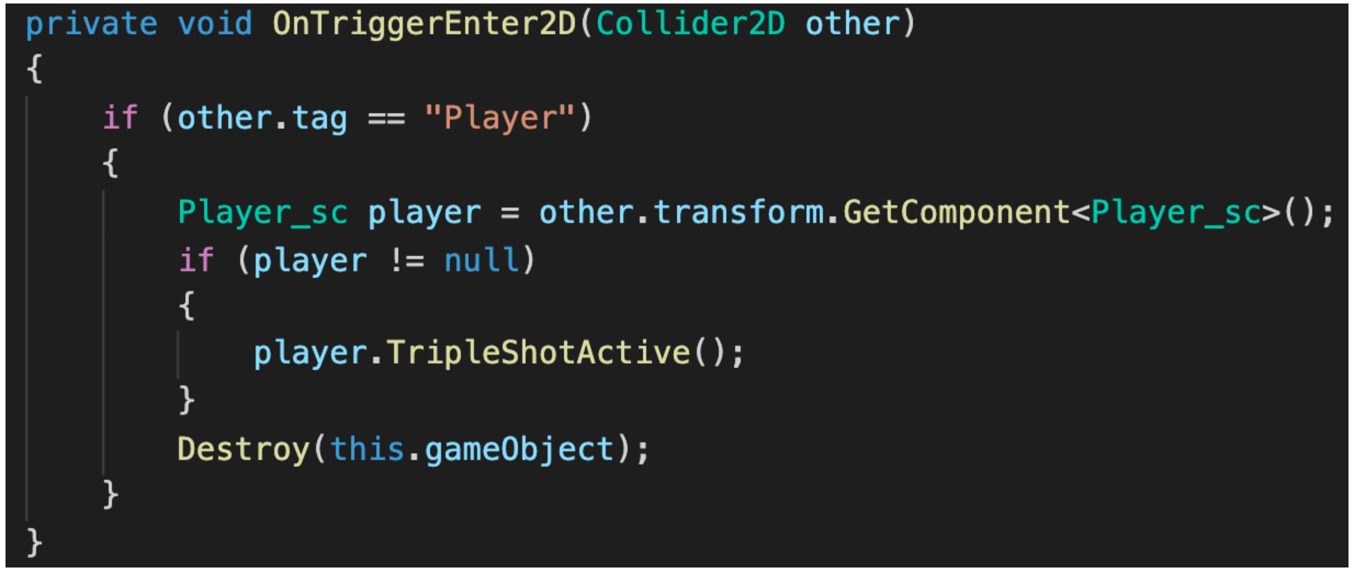
- Script communication to enable triple shot and then disable it after timeout
  - For player, define a public method to set triple shot active



```
public void TripleShotActive()
{
    _isTripleShotActive = true;
    StartCoroutine(TripleShotPowerDownRoutine());
}

1 reference
IEnumerator TripleShotPowerDownRoutine()
{
    yield return new WaitForSeconds(5.0f);
    _isTripleShotActive = false;
}
```

Triple Shot Powerup Behavior

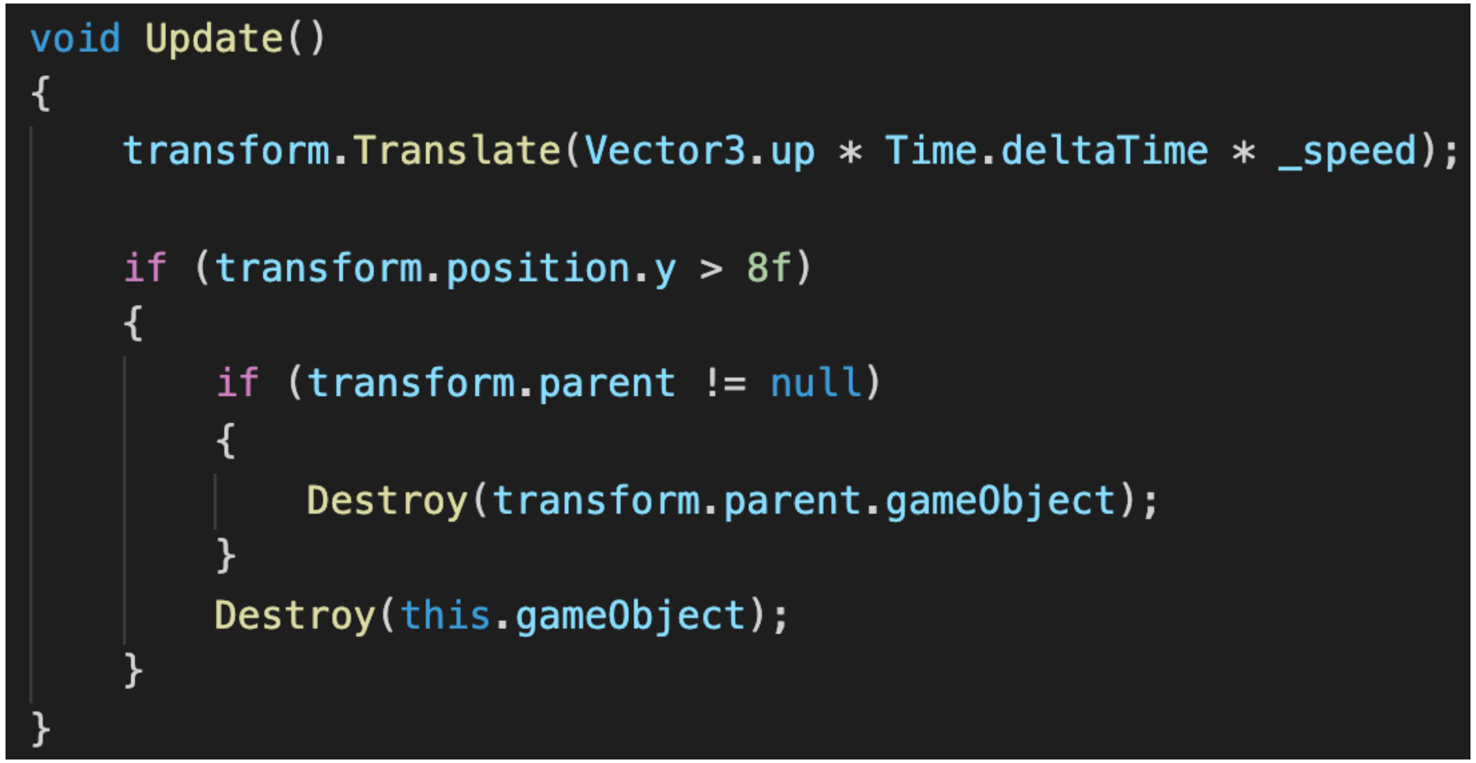
- Script communication to enable triple shot
  - Find the script component
  - Call the activate method



```
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        Player_sc player = other.transform.GetComponent<Player_sc>();
        if (player != null)
        {
            player.TripleShotActive();
        }
        Destroy(this.gameObject);
    }
}
```

Destroy Parent Triple Shot

- Check if the object has a parent (triple lasers have one!)
  - Delete the parent first if there is



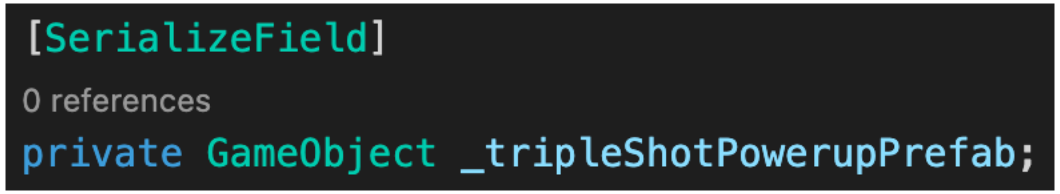
```
void Update()
{
    transform.Translate(Vector3.up * Time.deltaTime * _speed);

    if (transform.position.y > 8f)
    {
        if (transform.parent != null)
        {
            Destroy(transform.parent.gameObject);
        }
        Destroy(this.gameObject);
    }
}
```

Animate Triple Shot Powerup

- Before creating the animation select the Game Object that you want to create the animation for
  - Select Triple_Shot_Powerup
- Go to Window -> Animation -> Animation
  - Dock this next to the console
- Create an animator and an animation clip
  - Create a new folder for animations under assets: "Animations"
  - Save the new animation in the "Animations" folder: "Triple_Shot_Powerup_anim"
- Make sure the object is selected
  - Enable key frame recording mode
  - Select frames in triple shot powerup folder and drag them to frame
  - Play the animation to test it
  - Stop record mode
- In the animations folder two files are recorded. One of them is the controller file
  - Controller is added automatically to the animation component for the game object
  - We can open controller to see inside

Spawn Triple Shot Powerup

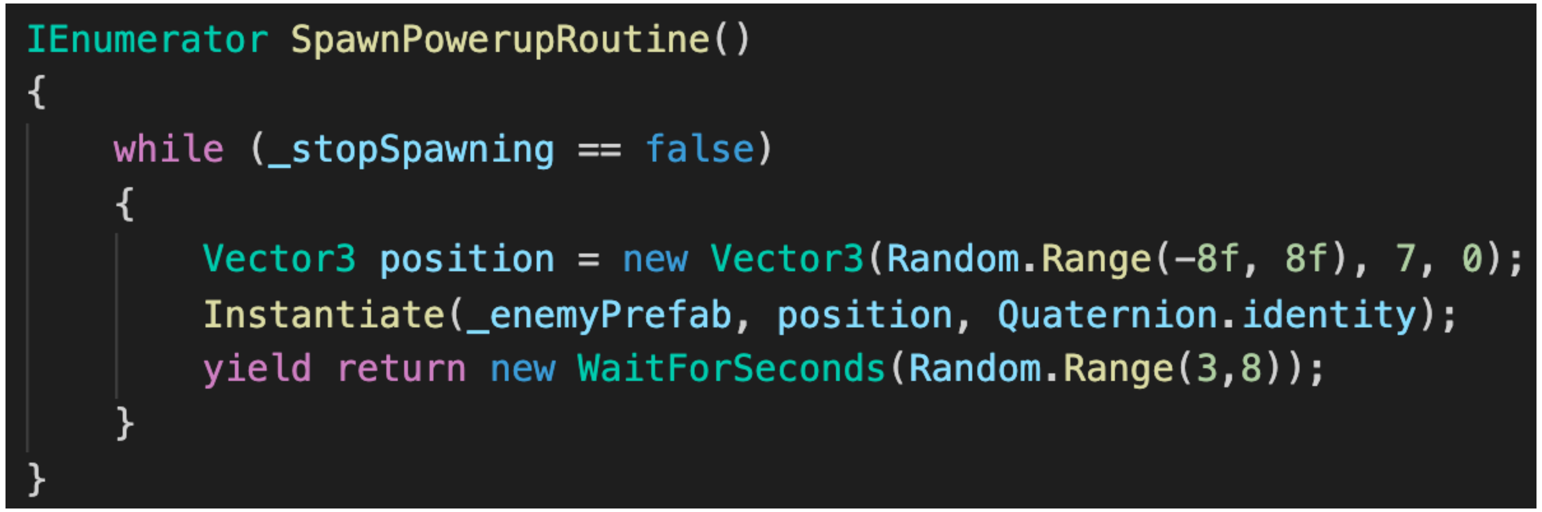
- Modify Spawn Manager to include the Triple Shot Powerup
- Create a variable for the Triple Shot Powerup prefab
  - Set the new field in the Inspector



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

Spawn Triple Shot Powerup

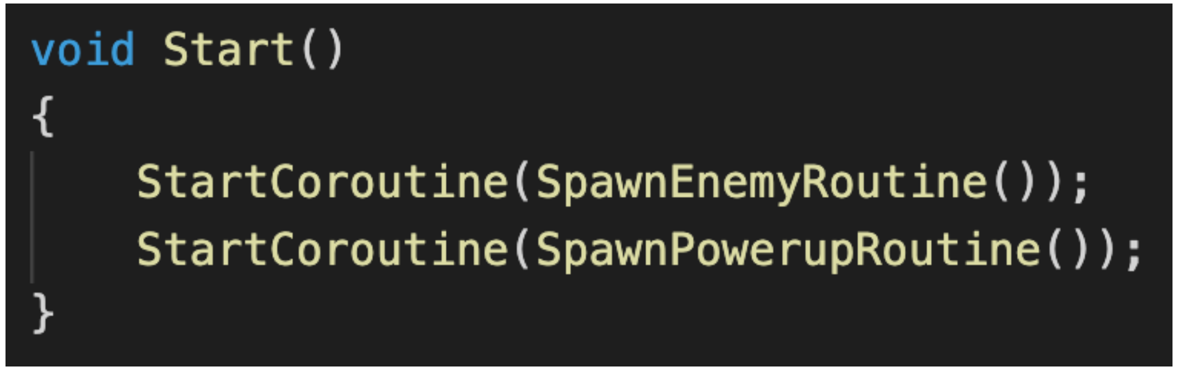
- Change the existing routine name from “SpawnRoutine” to “SpawnEnemyRoutine”
- Create a new routine for the powerup: “SpawnPowerupRoutine”



```
IEnumerator SpawnPowerupRoutine()
{
    while (_stopSpawning == false)
    {
        Vector3 position = new Vector3(Random.Range(-8f, 8f), 7, 0);
        Instantiate(_enemyPrefab, position, Quaternion.identity);
        yield return new WaitForSeconds(Random.Range(3,8));
    }
}
```

Spawn Triple Shot Powerup

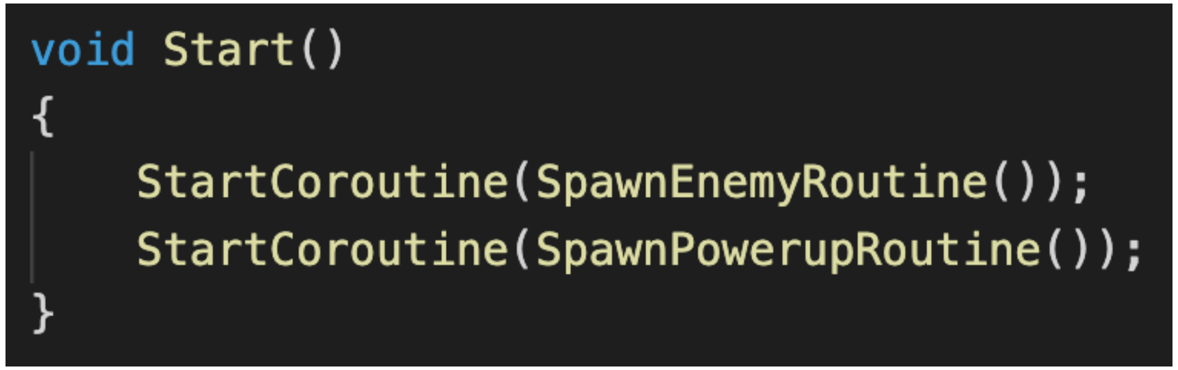
- Start the “SpawnPowerupRoutine”



```
void Start()
{
    StartCoroutine(SpawnEnemyRoutine());
    StartCoroutine(SpawnPowerupRoutine());
}
```

Cleanup and Organization

- More powerups will be added to the game
- To organize powerups create a new folder under Prefabs: “Powerup”
  - Add Triple Shot Powerup prefab to the new folder



```
void Start()
{
    StartCoroutine(SpawnEnemyRoutine());
    StartCoroutine(SpawnPowerupRoutine());
}
```