

# Collision Decisions + GAME OVER

## 1. Make a new method to spawn animals

- In **SpawnManager.cs**, create a new **void *SpawnRandomAnimal()* {}** function beneath ***Update()***
- Cut and paste the code from the **if-then statement** to the **new function**
- Call ***SpawnRandomAnimal();*** if **S** is pressed

## 2. Spawn the animals at timed intervals

- In ***Start()***, use ***InvokeRepeating*** to spawn the animals based on an interval, then **test**.
- Remove the **if-then statement** that tests for **S** being pressed
- Declare new ***private startDelay*** and ***spawnInterval*** variables then playtest and tweak variable values

## 3. Add collider and trigger components

- Double-click on one of the **animal prefabs**, then **Add Component > Box Collider**
- Click **Edit Collider**, then **drag** the collider handles to encompass the object

- Check the “**Is Trigger**” checkbox
- Repeat this process for each of the **animals** and the **projectile**
- Add a **RigidBody component** to the projectile and uncheck “use gravity”

## 4. Destroy objects on collision

- Create a new **DetectCollisions.cs** script, add it to each animal prefab, then **open** it
- Before the final **}** add an **OnTriggerEnter** function using **autocomplete**
- In **OnTriggerEnter**, put **Destroy(gameObject);**, then test
- In **OnTriggerEnter**, put **Destroy(other.gameObject);**

## 5. Trigger a “Game Over” message

- In DestroyOutOfBounds.cs, in the **else-if condition** that checks if the animals reach the bottom of the screen, add a Game Over message: **Debug.Log(“Game Over!”)**
- Clean up your code with **comments**
- If using Visual Studio, Click *Edit > Advanced > **Format document*** to fix any indentation issues (On a **Mac**, click *Edit > Format > Format Document*)

## 6. Lesson Recap

### New Functionality

- Animals spawn on a timed interval and walk down the screen
- When animals get past the player, it triggers a “Game Over” message
- If a projectile collides with an animal, both objects are removed

### New Concepts & Skills

- Create custom methods/functions
- InvokeRepeating() to repeat code
- Colliders and Triggers
- Override functions
- Log Debug messages to console

Pizza\_gun

```

using UnityEngine;

public class Pizza_gun : MonoBehaviour
{
    [SerializeField]
    public float Weapon_Bullet_Projectile_Speed = 60.0f;
    [SerializeField]
    private float topBound = -60;

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime * Weapon_Bullet_Projectile_Speed);

        if (transform.position.z < topBound)
        {
            Destroy(gameObject);
        }

    }
    private void OnTriggerEnter(Collider other)
    {
        Destroy(gameObject);
        Destroy(other.gameObject);
    }
}

```

## Spawn\_manager

- ```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SpawnManager : MonoBehaviour
{
    public GameObject[] animalPrefabs = new GameObject[10];
    [SerializeField]
    private float spawnRangeX = 20.0f;
    [SerializeField]
    private float spawnPozY = -20.0f;

```

```

[SerializeField]
private float startDelay = 5f;
[SerializeField]
private float spawnInterval = 2f;
// Start is called before the first frame update
void Start()
{
    // invoke repeating, will make the method or void def function with name "nameInQuotes"
    // it will call that function after , delay, then chozen interval
    InvokeRepeating("SpawnRandomAnimal", startDelay, spawnInterval);
}

// Update is called once per frame
void Update()
{
    if (Input.GetKeyDown(KeyCode.S))
    {
        SpawnRandomAnimal();
    }
}
void SpawnRandomAnimal()
{
    // Select a random index within the array range
    int random_animal_Index = Random.Range(0, animalPrefabs.Length);
    Vector3 spawnPos = new Vector3(Random.Range(-spawnRangeX, spawnRangeX), spawnPozY, -
30);
    // Instantiate the selected animalPrefab
    Instantiate(animalPrefabs[random_animal_Index], spawnPos,
        animalPrefabs[random_animal_Index].transform.rotation);
}
}

```

- movement

```

using UnityEngine;

public class Movement : MonoBehaviour
{
    [SerializeField]

```

```

private float speed = 10.0f;
[SerializeField]
private float xRange = 10.0f;
[SerializeField]
private float horizontalInput;
[SerializeField]
public GameObject ProjectilePrefab;
// Start is called before the first frame update
void Start()
{

}

// Update is called once per frame
void Update()
{

    horizontalInput = Input.GetAxis("Horizontal");
    // Calculate the desired position based on the input
    Vector3 desiredPosition = transform.position + Vector3.right * -horizontalInput * Time.deltaTime
* speed;
    // Clamp the desired position within the x range
    float clampedX = Mathf.Clamp(desiredPosition.x, -xRange, xRange);
    desiredPosition = new Vector3(clampedX, desiredPosition.y, desiredPosition.z);
    // Move the object to the clamped position
    transform.position = desiredPosition;
    if (Input.GetKeyDown(KeyCode.Space))
    {
        // shoot pizza
        Instantiate(ProjectilePrefab, transform.position, ProjectilePrefab.transform.rotation);
    }
}
}

```

## Animal Movement

```

using UnityEngine;

public class AnimalMovement : MonoBehaviour

```

```
{

    public float Animal_Movement_Speed = 50.0f;
    public float lowerBound = 50;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.forward * Time.deltaTime * Animal_Movement_Speed);
        if (transform.position.z > lowerBound)
        {
            Destroy(gameObject);
            Debug.Log("In the vernacular of your people... \n Game Over!");
        }
    }
}
```

•

---

Revision #1

Created 23 May 2023 21:47:11 by naruzkurai

Updated 18 June 2023 09:54:37 by naruzkurai