unity

2.4 Collision Decisions

Steps:

Step 1: Make a new method to spawn animals

Step 2: Spawn the animals at timed intervals

Step 3: Add collider and trigger components

Step 4: Destroy objects on collision

Step 5: Trigger a "Game Over" message

Example of project by end of lesson

Length: 50 minutes

Overview: Our game is coming along nicely, but there are are some critical things we must add before it's finished. First off, instead of pressing S to spawn the animals, we will spawn them on a timer so that they appear every few seconds. Next we will add colliders to all of our prefabs and make it so launching a projectile into an animal will destroy it. Finally, we will display a "Game Over" message if any animals make it past the player.

Project Outcome: The animals will spawn on a timed interval and walk down the screen, triggering a "Game Over" message if they make it past the player. If the player hits them with a projectile to feed them, they will be destroyed.

Learning Objectives: By the end of this lesson, you will be able to:
- Repeat functions on a timer with InvokeRepeating
- Write custom functions to make your code more readable
- Edit Box Colliders to fit your objects properly
- Detect collisions and destroy objects that collide with each other
- Display messages in the console with Debug Log

Step 1: Make a new method to spawn animals

Our Spawn Manager is looking good, but we're still pressing S to make it work! If we want the game to spawn animals automatically, we need to write our very first custom function.

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

- **New Concept:** Custom Void Functions
- **New Concept:** Compartmentalization / Abstraction

```
void Update() {
  if (Input.GetKeyDown(KeyCode.S)) {
    SpawnRandomAnimal();
    int animalIndex ... (Cut and Pasted Below) }}

void SpawnRandomAnimal() {
  int animalIndex = Random.Range(0, animalPrefabs.Length);
  Vector3 spawnpos = new Vector3(Random.Range(-xSpawnRange,
    xSpawnRange), 0, zSpawnPos);
  Instantiate(animalPrefabs[animalIndex], new Vector3(0, 0, 20) spawnpos,
    animalPrefabs[animalIndex].transform.rotation);
 }
```

Step 2: Spawn the animals at timed intervals

We've stored the spawn code in a custom function, but we're still pressing S! We need to spawn the animals on a timer, so they randomly appear every few seconds.

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

- **Tip:** Google "Repeating function in Unity"
- **New Function:** InvokeRepeating

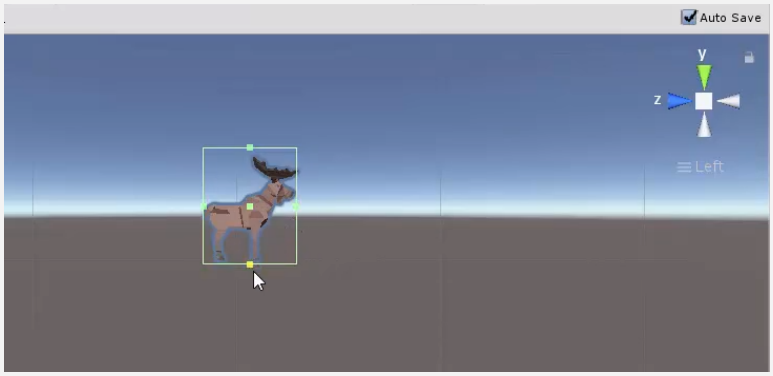
```
private float startDelay = 2;
private float spawnInterval = 1.5f;

void Start() {
  InvokeRepeating("SpawnRandomAnimal", startDelay, spawnInterval); }

void Update() {
  if (Input.GetKeyDown(KeyCode.S)) {
    SpawnRandomAnimal(); } }
```

Step 3: Add collider and trigger components

Animals spawn perfectly and the player can fire projectiles at them, but nothing happens when the two collide! If we want the projectiles and animals to be destroyed on collision, we need to give them some familiar components - "colliders."

1. Double-click on one of the **animal prefabs**, then *Add Component > Box Collider*
2. Click **Edit Collider**, then **drag** the collider handles to encompass the object
3. Check the "**Is Trigger**" checkbox
4. Repeat this process for each of the **animals** and the **projectile**
5. Add a **RigidBody component** to the projectile and uncheck "use gravity"

- **New Component:** Box Colliders
- **Warning:** Avoid Box Collider 2D
- **Tip:** Use isometric view and the gizmos to cycle around and edit the collider with a clear perspective
- **Tip:** For the Trigger to work, at least one of the objects needs a rigidbody component



Step 4: Destroy objects on collision

Now that the animals and the projectile have Box Colliders with triggers, we need to code a new script in order to destroy them on impact.

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

- **New Concept:** Overriding Functions
- **New Function:** OnTriggerEnter
- **Tip:** The "other" in OnTriggerEnter refers to the collider of the other object
- **Tip:** Use VS's Auto-Complete feature for OnTriggerEnter and any/all override functions

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

Step 5: Trigger a “Game Over” message

The player can defend their field against animals for as long as they wish, but we should let them know when they’ve lost with a “Game Over” message if any animals get past the player.

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

- **New Functions:** Debug.Log, LogWarning, LogError
- **Tip:** Tweak some values to adjust the difficulty of your game. It might too easy!

```
void Update() {
  if (transform.position.z > topBound)
  {
    Destroy(gameObject);
  } else if (transform.position.z < lowerBound)
  {
    Debug.Log("Game Over!");
    Destroy(gameObject);
  }
}
```

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