unity

3.2 Make the World Whiz By

Steps:

Example of project by end of lesson

Length: 70 minutes

Overview: We've got the core mechanics of this game figured out: The player can tap the spacebar to jump over incoming obstacles. However, the player appears to be running for the first few seconds, but then the background just disappears! In order to fix this, we need to repeat the background seamlessly to make it look like the world is rushing by! We also need the game to halt when the player collides with an obstacle, stopping the background from repeating and stopping the obstacles from spawning. Lastly, we must destroy any obstacles that get past the player.

Project Outcome: The background moves flawlessly at the same time as the obstacles, and the obstacles will despawn when they exit game boundaries. With the power of script communication, the background and spawn manager will halt when the player collides with an obstacle. Colliding with an obstacle will also trigger a game over message in the console log, halting the background and the spawn manager.

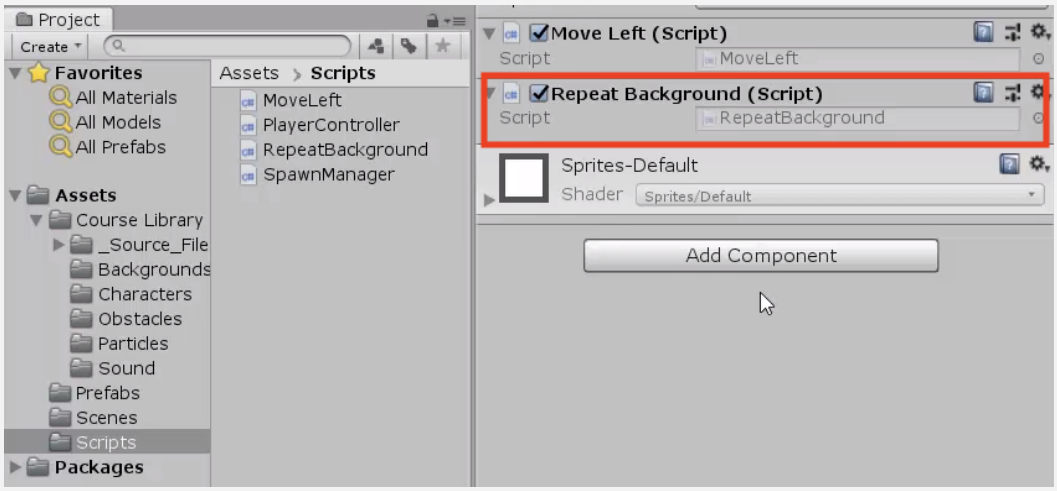
Learning Objectives: By the end of this lesson, you will be able to:
- Use tags to label game objects and call them in the code
- Use script communication to access the methods and variables of other scripts

Step 1: Create a script to repeat background

We need to repeat the background and move it left at the same speed as the obstacles, to make it look like the world is rushing by. Thankfully we already have a move left script, but we will need a new script to make it repeat.

1. Create a new script called **RepeatBackground.cs** and attach it to the **Background Object**

- **Tip:** Think through what needs to be done: when the background moves half of its length, move it back that distance



Step 2: Reset position of background

In order to repeat the background and provide the illusion of a world rushing by, we need to reset the background object's position so it fits together perfectly.

1. Declare a new variable ***private Vector3 startPos;***
2. In ***Start()***, set the ***startPos*** variable to its actual starting position by assigning it ***= transform.position;***
3. In ***Update()***, write an **if-statement** to reset position if it moves a certain distance

- **Don't worry:** We're setting it at 40 for now, just to test basic functionality. You could probably get it right with trial and error… but what would happen if you changed the size?

```
private Vector3 startPos;

void Start() {
  startPos = transform.position; }

void Update() {
  if (transform.position.x < startPos.x - 50) {
    transform.position = startPos; } }
```

Step 3: Fix background repeat with collider

We've got the background repeating every few seconds, but the transition looks pretty awkward. We need make the background loop perfectly and seamlessly with some new variables.

1. Add a **Box Collider** component to the **Background**
2. Declare a new ***private float repeatWidth*** variable
3. In ***Start()***, get the width of the **box collider**, divided by 2
4. Incorporate the ***repeatWidth*** variable into the **repeat function**

- **Don't worry:** We're only adding a box collider to get the size of the background
- **New Function:** .size.x

```
private Vector3 startPos;
private float repeatWidth;

void Start() {
  startPos = transform.position;
  repeatWidth = GetComponent<BoxCollider>().size.x / 2; }

void Update() {
  if (transform.position.x < startPos.x - 50 repeatWidth) {
    transform.position = startPos; } }
```

Step 4: Add a new game over trigger

When the player collides with an obstacle, we want to trigger a "Game Over" state that stops everything In order to do so, we need a way to label and discern all game objects that the player collides with.

1. In the inspector, add a "Ground" tag to the **Ground** and an "Obstacle" tag to the **Obstacle prefab**
2. In PlayerController, declare a new ***public bool gameOver;***
3. In ***OnCollisionEnter***, add the **if-else statement** to test if player collided with the "Ground" or an "Obstacle"
4. If they collided with the "Ground", set ***isOnGround = true***, and if they collide with an "Obstacle", set ***gameOver = true***

- **New Concept:** Tags
- **Warning:** New tags will NOT be automatically added after you create them. Make sure to add them yourself once they are created.
- **Tip:** No need to say gameOver = false, since it is false by default

```
public bool gameOver = false;

private void OnCollisionEnter(Collision collision) {
  isOnGround = true;
  if (collision.gameObject.CompareTag("Ground")) {
    isOnGround = true;
  } else if (collision.gameObject.CompareTag("Obstacle")) {
    gameOver = true;
    Debug.Log("Game Over!"); }
}
```

Step 5: Stop MoveLeft on gameOver

We've added a gameOver bool that seems to work, but the background and the objects continue to move when they collide with an obstacle. We need the MoveLeft script to communicate with the PlayerController, and stop once the player triggers gameOver.

1. In **MoveLeft.cs**, declare a new ***private PlayerController playerControllerScript;***
2. In ***Start()***, initialize it by finding the **Player** and getting the PlayerController component
3. Wrap the **translate method** in an **if-statement** checking if game is not over

- **New Concept:** Script Communication
- **Warning:** Make sure to spell the "Player" tag correctly

```
private float speed = 30;
private PlayerController playerControllerScript;

void Start() {
  playerControllerScript =
  GameObject.Find("Player").GetComponent<PlayerController>(); }

void Update() {
  if (playerControllerScript.gameOver == false) {
    transform.Translate(Vector3.left * Time.deltaTime * speed); } }
```

Step 6: Stop obstacle spawning on gameOver

The background and the obstacles stop moving when gameOver == true, but the Spawn Manager is still raising an army of obstacles! We need to communicate with the Spawn Manager script and tell it to stop when the game is over.

1. In **SpawnManager.cs**, get a reference to the **playerControllerScript** using the same technique you did in MoveLeft.cs
2. Add a condition to only instantiate objects if ***gameOver == false***

```
private PlayerController playerControllerScript;

void Start() {
  InvokeRepeating("SpawnObstacle", startDelay, repeatRate);
  playerControllerScript =
  GameObject.Find("Player").GetComponent<PlayerController>(); }

void SpawnObstacle () {
  if (playerControllerScript.gameOver == false) {
    Instantiate(obstaclePrefab, spawnPos, obstaclePrefab.transform.rotation);
} }
```

Step 7: Destroy obstacles that exit bounds

Just like the animals in Unit 2, we need to destroy any obstacles that exit boundaries. Otherwise they will slide into the distance… forever!

1. In **MoveLeft**, in ***Update()***; write an if-statement to **Destroy** Obstacles if their position is less than a ***leftBound*** variable
2. Add any **comments** you need to make your code more **readable**

- **Tip:** Reference your code from MoveLeft

```
private float leftBound = -15;

void Update() {
  if (playerControllerScript.gameOver == false) {
    transform.Translate(Vector3.left * Time.deltaTime * speed); }

  if (transform.position.x < leftBound && gameObject.CompareTag("Obstacle")) {
    Destroy(gameObject); } }
```

Lesson Recap

New Functionality	• Background repeats seamlessly • Background stops when player collides with obstacle • Obstacle spawning stops when player collides with obstacle • Obstacles are destroyed off-screen
New Concepts and Skills	• Repeat background • Get Collider width • Script communication • Equal to (==) operator • Tags • CompareTag()
Next Lesson	• Our character, while happy on the inside, looks a little too rigid on the outside, so we're going to do some work with animations