Challenge 4
Obstacle Interactions

Exam Objectives Covered

- Select the appropriate properties, scripts, and components of GameObjects for required tasks
- Obtain a defined result by using Unity API methods, given Unity's API documentation

Activity Overview: Implementing Events for Obstacle Interactions

Welcome to the next challenge in your SkiFree-inspired game development journey! Now that you have functional obstacles, it's time to implement event-driven interactions to create dynamic responses when the player collides with obstacles. In this session, you'll use Events and callback functions to handle obstacle collisions and trigger various behaviors.

Step 1: Understand and Set Up Events

Events are crucial for decoupling complex systems and rapidly prototyping new behaviors. If you're not familiar with Events, refer to the Support document for this session.

Step 2: Set Up the Event System

Create an event system to handle player collisions with obstacles.

Instructions:

1. Define the Event:
    - Create a new script, ObstacleEventManager, and define an event for handling player collisions with obstacles.
    - The event should trigger necessary actions upon collision.
2. Trigger the Event in Obstacle Class:
    - Modify your Obstacle class to use the defined event. When a collision with the player is detected, invoke the event.

Step 3: Implement Knockback Functionality

Create a component to handle the knockback effect on the player.

Instructions:

1. Create Knockback Component:
    - Create a new script, PlayerKnockbackHandler.
    - Implement functionality to knock the player back for a short period and disable player control during this time.
2. Subscribe to the Event:
    - In your PlayerKnockbackHandler, subscribe to the obstacle collision event you defined earlier.
    - When the event is triggered, execute the knockback logic.

Step 4: Play Sound on Collision

Ensure a sound plays when the player collides with an obstacle.

Instructions:

1. Create Sound Handler Component:
    - Create a new script, CollisionSoundPlayer.
    - Implement functionality to play a designated sound clip when a collision event occurs.
2. Subscribe to the Event:
    - In your CollisionSoundPlayer, subscribe to the obstacle collision event.
    - Play the sound when the event is triggered.

Step 5: Integrate Components

Ensure all components work together seamlessly when a collision occurs.

Instructions:

1. Test Components Individually:
    - Test the PlayerKnockbackHandler and CollisionSoundPlayer components individually to make sure they respond correctly to the event.
2. Test Combined Functionality:
    - Test the entire system to ensure the player is knocked back, control is disabled during knockback, and the sound plays upon collision.

Good luck with implementing events! This exercise will help you master event-driven programming and modular design in Unity, enhancing your capacity to create responsive and flexible game systems. Happy developing!