Challenge 3
Basic Obstacles

Exam Objectives Covered

- Explain how to use prefabs in a scene
- Describe the process and outcomes for changing a nested prefab or prefab variant
- Explain the differences between basic inheritance and interfaces

Activity Overview: Adding Obstacles to the Ski Slope

Welcome to the next challenge in our SkiFree-inspired game development! Now that the player can effectively navigate the slope, it’s time to introduce obstacles to add some challenge and dynamic behavior to our prototype. In this session, you'll focus on creating obstacle prefabs and writing functionality to handle player collisions.

Step 1: Create a Rock Prefab

Utilize the provided art assets to create a basic rock obstacle for the ski slope.

Instructions:

1. Locate Art Assets:
    - Navigate to Assets > Models to find the rock art assets.
2. Create Rock Prefab:
    - Create a new GameObject in your scene and apply the rock model.
    - Adjust its properties (e.g., collider, material) to suit the game's visual and collision requirements.
    - Save this GameObject as a prefab named RockPrefab in your project.

Step 2: Create a Prefab Variant

Create a variation of the rock prefab using different visual assets.

Instructions:

1. Duplicate Rock Prefab:

- Duplicate RockPrefab to create a new prefab variant.

2. Apply Different Visual Assets:
   - Swap out the visual assets (e.g., texture, model) with different ones found in the Models folder or other provided assets.
   - Save this variant as RockPrefabVariant.

Step 3: Write a Base Obstacle Class

Develop a base class that all obstacles can inherit from to handle player collisions.

Instructions:

1. Create Obstacle Script:
   - Create a new C# script named Obstacle.
   - This base class should include functionality to register collisions with the player.
2. Implement Collision Detection:
   - Within the Obstacle class, implement a method to detect collisions with the skier and log or register these events.

Step 4: Extend the Obstacle Class

Create a specific type of obstacle that removes itself upon collision with the player.

Instructions:

1. Create Extended Obstacle Class:
   - Create a new C# script named RemovableObstacle that inherits from the Obstacle class.
2. Implement Self-Removal:
   - Override the collision method to include functionality that removes the obstacle from the scene upon collision.

Once you've implemented and tested these features, your game will have functional obstacles to challenge the player and add depth to the gameplay.

Good luck with adding obstacles! This exercise will help you develop skills in prefab creation, class inheritance, and collision handling in Unity. Happy developing!