Projet : Intégrer la physique de base dans le jeu et gérer les collisions entre le joueur et les obstacles

Exercice Projet : Intégrer la Physique de Base dans le Jeu et Gérer les Collisions entre le Joueur et les Obstacles

Dans cet exercice, vous allez intégrer les composants physiques de base dans un jeu Unity. L'objectif est de gérer les collisions entre le joueur et les obstacles tout en appliquant des forces physiques simples (comme la gravité et le mouvement), et en réagissant à ces collisions de manière appropriée.

Étapes de l'Exercice

1. Créer la Scène du Jeu

1. **Créer un terrain de jeu simple :**
   - Créez un **Plane** dans la scène pour servir de sol.
   - Redimensionnez-le si nécessaire pour qu'il couvre une zone de jeu suffisamment grande.
2. **Ajouter des obstacles :**
   - Créez plusieurs **Cubes** ou autres objets géométriques (ex : **Sphères**, **Cylindres**) pour servir d'obstacles.
   - Positionnez-les de manière aléatoire ou sous forme de plateforme.
   - Ajoutez un **BoxCollider** ou **SphereCollider** à chaque obstacle (sauf si vous les définissez comme triggers).

2. Créer le Joueur

1. **Créer un personnage joueur :**
   - Créez un objet **Cube** ou **Capsule** pour représenter le personnage du joueur.

- Ajoutez un **Rigidbody** au personnage pour que les forces physiques (comme la gravité) l'affectent.
- Si nécessaire, activez l'option **Is Kinematic** pour contrôler manuellement le mouvement sans forces physiques.

2. **Ajouter des contrôles de base pour le joueur :**
   - Utilisez un script pour permettre au joueur de se déplacer. Par exemple, vous pouvez contrôler le mouvement avec les touches **WASD** ou **Flèches directionnelles**.

3. Gérer la Gravité et les Collisions

1. **Activer la gravité :**
   - Assurez-vous que l'option **Use Gravity** est activée sur le **Rigidbody** du joueur pour que la gravité affecte l'objet.
2. **Ajouter un Collider pour la détection de collision :**
   - Ajoutez un **BoxCollider** ou **CapsuleCollider** au personnage pour détecter les collisions avec les obstacles.

4. Script pour Détecter les Collisions et Réagir

1. **Créer un script pour gérer les collisions :**
   - Ajoutez un script qui détecte les collisions entre le joueur et les obstacles et qui applique une réaction (par exemple, afficher un message ou faire rebondir le joueur).

Voici un exemple de script pour détecter les collisions entre le joueur et un obstacle :

```
using UnityEngine;

public class PlayerCollision : MonoBehaviour
{
    // Cette méthode est appelée quand une collision avec u
n autre objet commence
    void OnCollisionEnter(Collision collision)
    {
        // Vérifie si l'objet collidé est un obstacle
        if (collision.gameObject.CompareTag("Obstacle"))
```

```
        {
            // Afficher un message de collision dans la con
sole
            Debug.Log("Le joueur a heurté un obstacle !");

            // Appliquer un effet au joueur (par exemple, r
ebondir ou appliquer une force)
            Rigidbody rb = GetComponent<Rigidbody>();
            rb.AddForce(Vector3.up * 500); // Appliquer une
force de rebond vers le haut
        }
    }
}
```

1. **Ajoutez le script au joueur :**
    - Attachez ce script à votre objet **Joueur**.
    - Assurez-vous que les obstacles dans la scène ont le tag **Obstacle** afin que le script puisse identifier les collisions avec eux.

5. Ajouter des Effets Visuels et Sonores (Optionnel)

Pour rendre le jeu plus interactif et amusant, vous pouvez ajouter des effets visuels et sonores lors des collisions :

- **Effet de particules** : Ajoutez un système de particules pour simuler une explosion ou un impact lorsque le joueur touche un obstacle.
- **Son de collision** : Utilisez un clip sonore pour émettre un bruit lors de la collision avec un obstacle.

Exemple de script pour jouer un son de collision :

```
using UnityEngine;

public class PlayerCollision : MonoBehaviour
{
    public AudioClip collisionSound;  // Référence au son d
e collision
    private AudioSource audioSource;
```

```
    void Start()
    {
        audioSource = GetComponent<AudioSource>();
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Obstacle"))
        {
            // Jouer le son de collision
            audioSource.PlayOneShot(collisionSound);

            // Appliquer une force de rebond
            Rigidbody rb = GetComponent<Rigidbody>();
            rb.AddForce(Vector3.up * 500);
        }
    }
}
```

6. Tester le Jeu et Ajuster la Physique

- Lancez la scène et testez les interactions entre le joueur et les obstacles.
- Vérifiez que le joueur réagit correctement aux collisions (par exemple, qu'il rebondit ou que l'objet est détruit si cela est nécessaire).
- Ajustez la **masse**, le **drag**, et les autres propriétés du **Rigidbody** pour obtenir un comportement physique réaliste.

7. Ajouter des Obstacles Dynamiques (Optionnel)

Vous pouvez ajouter des obstacles mobiles (par exemple, des plateformes qui se déplacent ou des ennemis) pour rendre le jeu plus intéressant. Utilisez des **Rigidbody** avec des mouvements programmés ou des animations pour faire bouger ces obstacles.
