Introduction aux moteurs physiques : gravité, forces, collisions

Les moteurs physiques dans les jeux vidéo permettent de simuler des phénomènes naturels et mécaniques afin de rendre l'interaction avec le monde du jeu plus réaliste. Dans ce cours, nous allons aborder les concepts de base des moteurs physiques dans **Unity**, tels que la gravité, les forces et les collisions.

---

1. Qu'est-ce qu'un moteur physique ?

Un moteur physique est un système qui gère la simulation des lois physiques dans un environnement de jeu. Il calcule les interactions entre les objets du jeu (comme les personnages, les objets ou les éléments du décor) en fonction des lois de la physique, telles que la gravité, les forces, et les collisions. Le moteur physique permet ainsi d'appliquer des comportements réalistes, comme le mouvement, la rotation, les rebonds ou les chocs.

Unity inclut un moteur physique intégré qui utilise principalement deux systèmes :

- **PhysX** (pour la 3D) : Un moteur physique développé par Nvidia, utilisé dans Unity pour simuler des mouvements réalistes, des collisions et des interactions.
- **2D Physics** : Utilisé pour les jeux en 2D, ce moteur est basé sur un système physique adapté à des jeux où la physique est simplifiée.

---

2. Les Composants Physiques dans Unity

Pour travailler avec la physique dans Unity, nous utilisons plusieurs composants. Les plus importants sont :

- **Rigidbody** : Ce composant permet à un objet de réagir à la physique. Il applique la gravité, les forces, et gère les collisions.
- **Collider** : Un **Collider** définit la forme physique d'un objet afin que la physique puisse calculer les collisions. Il peut être un **BoxCollider**, **SphereCollider**, **MeshCollider**, etc. Un **Collider** est nécessaire pour détecter les collisions, mais il n'applique pas de forces.

Rigidbody

- **Rigidbody** permet à un objet d'être affecté par les forces physiques (comme la gravité, les impulsions ou les forces externes).
- Un objet avec un **Rigidbody** sera affecté par les lois de la physique d'Unity (chutes, mouvements, rotations, etc.).
- Si un objet n'a pas de Rigidbody, Unity ignore la physique pour cet objet, et il restera statique ou se déplacera uniquement selon le script du développeur.

---

3. Gravité

La gravité est la force qui attire un objet vers le bas. Dans Unity, elle est gérée par le **Rigidbody** et peut être configurée de différentes manières.

La Gravité par défaut de Unity

- Par défaut, Unity applique une gravité de **9.81 m/s²** (ce qui correspond à la gravité terrestre). Cela fait tomber les objets vers le bas de la scène.
- Vous pouvez activer ou désactiver la gravité d'un objet en modifiant le paramètre **Use Gravity** dans le composant **Rigidbody**.

Personnaliser la Gravité

1. Pour changer la force de la gravité globale dans votre jeu, allez dans **Edit** > **Project Settings** > **Physics** et ajustez la valeur de la **Gravity**.
2. Vous pouvez aussi ajuster la gravité de manière individuelle pour chaque objet en désactivant **Use Gravity** et en appliquant une force manuellement avec des scripts.

Exemple pour appliquer une gravité personnalisée via un script :

```
Rigidbody rb = GetComponent<Rigidbody>();
rb.AddForce(Vector3.down * customGravity, ForceMode.Acceler
ation);
```

---

4. Forces

Les forces sont utilisées pour simuler des interactions physiques, telles que les poussées, les tirages ou les impacts. Dans Unity, vous pouvez appliquer des forces aux objets via leur **Rigidbody**.

Types de Forces dans Unity

- **AddForce** : Applique une force sur un objet. Cela peut être une force constante ou une impulsion.

  Exemple pour appliquer une force de poussée :

  ```
  Rigidbody rb = GetComponent<Rigidbody>();
  rb.AddForce(Vector3.forward * 10f);  // Force vers l’ava
  nt
  ```

- **AddTorque** : Applique une force de rotation, faisant tourner un objet autour de son centre de masse.

  Exemple pour appliquer une force de rotation :

  ```
  Rigidbody rb = GetComponent<Rigidbody>();
  rb.AddTorque(Vector3.up * 50f);  // Rotation autour de
  l'axe Y
  ```

- **AddExplosionForce** : Applique une force d’explosion qui affecte un objet en fonction de sa proximité avec le point d'explosion.

---

5. Collisions

Les collisions sont un aspect essentiel de la physique dans les jeux vidéo. Elles permettent aux objets de réagir lorsqu'ils entrent en contact les uns avec les autres.

Les types de Colliders

- **BoxCollider** : Un **Collider** en forme de boîte. C’est le plus simple et le plus performant.
- **SphereCollider** : Un **Collider** en forme de sphère. Il est utile pour les objets ronds.
- **CapsuleCollider** : Utilisé pour les objets cylindriques ou les personnages.
- **MeshCollider** : Utilisé pour des formes complexes, mais moins performant que les autres. Il peut utiliser la forme réelle d’un objet pour les collisions.

Détection de Collision

Unity offre deux types de détection de collision : **Collider Trigger** et **Collider Normal**.

1. **Collider Normal** : Ce type de collision génère une réaction physique (par exemple, un objet rebondit après avoir frappé un autre).
2. **Collider Trigger** : Ce type de collision n’interrompt pas le mouvement des objets, mais permet d'exécuter des actions lorsque deux objets se touchent. Vous pouvez utiliser des événements comme **OnTriggerEnter** ou **OnTriggerExit** pour détecter ces collisions.

Exemple de code pour détecter une collision

```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Enemy"))
    {
        Debug.Log("Le joueur a touché un ennemi !");
    }
}
```

Collision entre deux objets

Les objets ayant des **Rigidbody** et des **Colliders** détecteront automatiquement les collisions. Lorsque la collision se produit, Unity applique des réactions physiques : les objets peuvent se repousser, rebondir ou se stopper, selon leurs propriétés physiques.

---

6. Forces et Collisions Avancées

Interaction entre objets avec des Rigidbody

Les objets équipés de **Rigidbody** réagiront aux collisions en fonction de leur **masse**, de la **vitesse** et des **forces appliquées**. Par exemple, un objet lourd se déplacera moins facilement qu'un objet léger lorsqu'une force est appliquée.

Comportement des objets en cas de collision

Vous pouvez personnaliser le comportement des objets en utilisant des matériaux physiques appelés **Physic Materials**. Ceux-ci définissent la friction et le rebond d'un objet lorsqu'il entre en collision avec un autre objet.

Exemple de code pour appliquer un **Physic Material** :

1. Créez un **Physic Material** dans votre projet (clic droit dans le projet > Create > Physic Material).
2. Ajustez les propriétés comme la friction ou le rebond dans l'**Inspector**.
3. Appliquez-le à un objet via le **Collider** dans l'**Inspector**.

---

7. Conclusion

Les moteurs physiques sont essentiels pour créer des environnements interactifs et réalistes dans les jeux vidéo. Unity offre un ensemble complet d'outils pour gérer la gravité, les forces, et les collisions, permettant de simuler des comportements physiques complexes de manière simple et intuitive. Que vous développiez un jeu 3D ou 2D, comprendre comment utiliser ces composants vous aidera à créer des expériences de jeu plus immersives et dynamiques.

Points Clés à Retenir :

- **Gravité** : Appliquée par défaut dans Unity, elle peut être ajustée pour chaque objet.
- **Forces** : Utilisez `AddForce` et `AddTorque` pour appliquer des forces de déplacement et de rotation.
- **Collisions** : Utilisez des **Colliders** pour détecter les contacts entre objets et réagir en conséquence.