Les objets physiques 3D

Rigidbody

Pour qu'un gameObject soit soumis aux lois de la physique 3D, il suffit de lui ajouter un composant « Rigidbody ». Le gameObject pourra alors être affecté par la gravité et pourra interagir « physiquement » avec les autres gameObjects dans votre monde.
Ajout d'un Rigidbody : « AddComponent/Physics/Rigidbody »

Les Colliders

Les composants de type Colliders permettent à Unity de détecter les collisions entre les gameObjects. Les Colliders peuvent être appliqués aux rigidbodies et aux obstacles. Pour qu'une collision entre deux objets soit détectable, les deux objets doivent avoir un Collider et au moins un des deux objets doit être rigidbody.
Ajout d'un Collider: « AddComponent/Physics/BoxCollider » ou autre type de collider.

Conseils

- Il ne faut pas ajouter le composant rigidbody à un parent ET à un enfant. De façon générale, seul le parent doit être rigidbody alors que ses enfants ne le sont pas. Les enfants peuvent cependant tous avoir la composante collider.
- Pour regrouper ou attacher des objets rigidbody ensemble, il est préférable d'utiliser des « joints » physiques.
- Il ne faut jamais redimensionner (« scaler ») le parent d'un rigidbody. Il est cependant permis de redimensionner un gameObject qui n'a pas d'enfant et qui est rigidbody.

Propriétés de Rigidbody

Mass	La masse de l'objet (unité arbitraire). Il est recommandé de ne pas dépasser une différence de 100 unités entre les masses de vos différents *rigidbodies*.
Drag	La résistance de l'air qui affecte le *rigidbody* lors de son déplacement par l'ajout de forces. La valeur « 0 » signifie aucune résistance et *infiniy* arrête le déplacement de l'objet immédiatement.
Angular Drag	La résistance de l'air qui affecte le rigidbody lorsqu'il tourne par l'ajout de torsion (*torque*). La valeur « 0 » signifie aucune résistance et *infiniy* arrête la rotation de l'objet immédiatement.
Use Gravity	Est-ce que l'objet doit être affecté par la gravité
Is Kinematic	Si cette propriété est activée, l'objet ne sera pas contrôlé par l'engin physique et pourra être manipulé à l'aide de sa propriété *Transform*. Peut être utilisé pour animer des plates-formes par exemple.
Constraints	Restreins les *rigibodies* dans leur mouvement :
Freeze Position	Empêche les déplacements dans les axe X, Y et Z du monde.
Freeze Rotation	Empêche la rotation dans les axes les axe X, Y t Z du monde.

FixeUpdate()

Fonction à utiliser à la place du Update() habituel lorsqu'on utilise un rigidbody et qu'on applique une force à répétition. Si la force n'est appliquée qu'une fois, il est possible de le faire dans la fonction Update().

Appliquer des forces à un objet physique : Rigidbody

Rotation d'objet physique selon les axes du monde : AddTorque()

Applique une force angulaire graduelle sur un objet (effet d'accélération en rotation) selon les axes du monde
Pour appliquer une force instantanée, il faut ajouter ***ForceMode.Impulse*** comme dernier paramètre.

GetComponent<Rigidbody>().AddTorque (force : Vector3**, mode :** ForceMode **= ForceMode.Force)**

GetComponent<Rigidbody>().AddTorque (x : float, y : float, z : float , mode : ForceMode = ForceMode.Force)

```
Rigidbody rigidbodyCube; // contiendra la référence au composant rigidbody de l'objet

void Start()
{
   rigidbodyCube = GetComponent<Rigidbody>();     //on assigne à la variable le composant Rigidbody d'un objet 3D
}


//Exemples d'utilisation de AddTorque pour faire tourner un objet
void FixedUpdate()
{
   rigidbodyCube.AddTorque(0, 0, 10);          //applique une force de rotation de 10 dans l'axe de Z du monde
   rigidbodyCube.AddTorque (Vector3.reight*10);       //applique une force de rotation de 10 dans l'axe de y du monde
}
```

Rotation d'objet physique selon les axes locaux de l'objet : AddRelativeTorque()

Applique une force angulaire graduelle sur un objet (effet d'accélération en rotation) selon les axes locaux de l'objet.
Pour appliquer une force instantanée, il faut ajouter ***ForceMode.Impulse*** comme dernier paramètre.

GetComponent<Rigidbody>().AddRelativeTorque (force : Vector3**, mode :** ForceMode **= ForceMode.Force)**

GetComponent<Rigidbody>().AddRelativeTorque (x : float, y : float, z : float , mode : ForceMode = ForceMode.Force)

```
Rigidbody rigidbodyCube;          // contiendra la référence au composant rigidbody de l'objet
float vitesseRotation = 100f;      //multiplicateur de vitesse de rotation

void Start()
{
   rigidbodyCube = GetComponent<Rigidbody>();     //on assigne à la variable le composant Rigidbody de l'objet
}

// Ajoute un torque selon les touches
void FixedUpdate()
{
   var forceRotation = Input.GetAxis("Horizontal") * vitesseRotation ;
   rigidbodyCube.AddRelativeTorque(0, 0, forceRotation );   //applique une force de rotation dans l'axe Z de l'objet
}
```

Déplacement d'objets physiques selon les axes du monde : AddForce()

GetComponent<Rigidbody>().AddForce (force : Vector3**, mode :** ForceMode **= ForceMode.Force)**

GetComponent<Rigidbody>().AddForce (x : float, y : float, z : float , mode : ForceMode = ForceMode.Force)

Applique une force graduelle sur un objet (effet d'accélération) selon les axes du monde.
Pour appliquer une force instantanée, il faut ajouter ForceMode.Impulse comme dernier paramètre.

```
Rigidbody rigidbodyCube; // contiendra la référence au composant rigidbody de l'objet

void Start()
{
   rigidbodyCube = GetComponent<Rigidbody>();    //on assigne à la variable le composant Rigidbody d'un objet 3D
}

//fonction qui s'exécute à chaque frame de l'engin physique et  qui  rafraîchie l'état des éléments physiques
void FixedUpdate()
{
   //exemples d'utilisation
   rigidbodyCube.AddForce(0, 0, 10);            //applique une force de 10 dans l'axe de Z du monde
   //ou
   rigidbodyCube.AddForce(Vector3.forward*10);      //applique une force de 10 dans l'axe de Z du monde


   var  vitesseVector3 = new Vector3(0,20,10);
   rigidbodyCube.AddForce(vitesseVector3);         //applique une force de 10 dans l'axe de Z du monde
   rigidbodyCube.AddForce(Vector3.up*10);          //applique une force de 10 dans l'axe de Y du monde
   rigidbodyCube.AddForce(Vector3.right*10);         //applique une force de 10 dans l'axe de Z du monde

}
```

Déplacement d'objets physiques selon les axes locaux de l'objet : AddRelativeForce()

GetComponent<Rigidbody>().AddRelativeForce (force : Vector3**, mode :** ForceMode **= ForceMode.Force)**

GetComponent<Rigidbody>().AddRelativeForce (x : float, y : float, z : float , mode : ForceMode = ForceMode.Force)

Applique une force graduelle sur un objet (effet d'accélération) selon les axes locaux de l'objet.
Pour appliquer une force instantanée, il faut ajouter ***ForceMode.Impulse*** comme dernier paramètre.

```
Rigidbody rigidbodyCube;       // contiendra la référence au composant rigidbody de l'objet
float vitesseMonte = 100f;      //multiplicateur de vitesse de monter et de descendre

void Start()
{
   rigidbodyCube = GetComponent<Rigidbody>();      //on assigne à la variable le composant Rigidbody d'un objet 3D
}

// Monte ou descend l'objet selon l'axe vertical, avance de 10 dans l'axe Z continuellement
void FixedUpdate()
{
   var forceMonter = Input.GetAxis("Vertical") * vitesseMonte;
   rigidbodyCube.AddRelativeForce(0, forceMonter, 10);  //applique une force vers le haut et de 10 dans l'axe de z de l'objet

}
```

Modifier les propriétés des objets Rigidbody

```
GetComponent<Rigidbody>().mass = 1000;
GetComponent<Rigidbody>().useGravity = true;
GetComponent<Rigidbody>().drag = 0;                                    //friction de l'aire
GetComponent<Rigidbody>().angularDrag = 1;                             //friction de l'aire pour la rotation
GetComponent<Rigidbody>().isKinematic = false;                         //si true, la physique est désactivée
GetComponent<Rigidbody>().velocity = new Vector3(0,10,0);              //vélocité selon l'axe du monde
print(GetComponent<Rigidbody>().velocity.z);
GetComponent<Rigidbody>().angularVelocity = new Vector3(15,0,0);    // la vitesse de rotation
GetComponent<Rigidbody>().SetMaxAngularVelocity(10);       //fixe la vitesse de rotation maximale permise
```

Pour appliquer ou modifier les contraintes des objets Rigidbody

GetComponent<Rigidbody>().constraints

```
// Contraint toute rotation du rigidbody (X, Y, Z)
GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotation;
ou
rigidbodyCube.constraints = RigidbodyConstraints.FreezeRotation;

// Contraint tout déplacement du rigidbody (X, Y, Z)
GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezePosition;

// Contraint tout déplacement et toute rotation du rigidbody (X, Y, Z)
GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeAll;

// Contraint toute rotation en X et Y du rigidbody
GetComponent<Rigidbody>().constraints = RigidbodyConstraints.FreezeRotationX | RigidbodyConstraints.FreezeRotationY;

// Enlève toutes les contraintes de déplacement et de rotation du rigidbody
GetComponent<Rigidbody>().constraints = RigidbodyConstraints.None;
```