

GameObject et composants

Dans Unity, un **GameObject** est l'entité de base dans une scène. Il peut représenter des personnages, des objets, des lumières, des caméras, et bien plus encore. Cependant, un GameObject en lui-même n'a pas de comportement ou d'apparence spécifique. C'est là que les **composants** entrent en jeu. Il s'agit de modules que vous pouvez ajouter à un GameObject pour lui donner des fonctionnalités spécifiques. Nous avons vu des composants tels que :

- **Transform** : Gère la position, la rotation et l'échelle du GameObject dans l'espace 3D.
- **Rigidbody2D** : Permet au GameObject de réagir aux forces physiques, comme la gravité et les collisions.
- **Collider2D** : Définit la forme de collision du GameObject pour détecter les interactions avec d'autres objets.
- **SpriteRenderer** : Affiche une image 2D (sprite) sur le GameObject.

1.1 Propriété/méthodes du GameObject

Voici quelques autres méthodes utiles que vous pouvez utiliser avec les GameObjects :

- `name` : Permet d'obtenir ou de définir le nom du GameObject. C'est le nom que vous voyez dans le haut de l'Inspector.
- `tag` : Permet d'obtenir ou de définir le tag du GameObject. Les tags sont utilisés pour catégoriser les GameObjects et faciliter leur identification. Ex : le tag "Player" pour le personnage principal et "Enemy" pour tous les ennemis. C'est utile pour les collisions et les interactions.
- `transform` : Permet d'accéder au composant Transform du GameObject, qui gère sa position, rotation et échelle dans la scène.
- `parent` : Permet d'obtenir ou de définir le parent du GameObject dans la hiérarchie des objets. Un GameObject peut être un enfant d'un autre GameObject, ce qui affecte sa position et son comportement.
- `SetActive(bool)` : Active ou désactive le GameObject. Lorsqu'un GameObject est désactivé, il n'est plus mis à jour ni rendu dans la scène.
- `GetComponent<T>()` : Permet d'obtenir une référence à un composant spécifique attaché au GameObject. Par exemple, `GetComponent<Rigidbody2D>()` retourne le composant Rigidbody2D du GameObject.
- `Find(string name)` : Méthode statique qui permet de trouver un GameObject dans la scène par son nom.
- `FindWithTag(string tag)` : Méthode statique qui permet de trouver un GameObject dans la scène par son tag.

Une méthode statique signifie que vous l'appellez directement sur la classe `GameObject` elle-même, plutôt que sur un objet réel dans la scène. Exemple : `GameObject.Find("Player")`.

1.2 Rechercher un GameObject dans la scène

Parfois, vous aurez besoin de trouver un GameObject spécifique dans votre scène par son nom. Unity fournit une méthode pratique pour cela : `GameObject.Find("NomDuGameObject")`. Cette méthode retourne une référence au GameObject cor-

respondant au nom spécifié.

```
GameObject player = GameObject.Find("Player");
if (player != null)
{
    // Le GameObject "Player" a été trouvé, vous pouvez maintenant interagir avec lui
}
else
{
    // Le GameObject "Player" n'a pas été trouvé
}
```

Vous pourriez également utiliser `GameObject.FindWithTag("TagDuGameObject")` pour trouver un `GameObject` par son tag, ce qui est souvent plus efficace si vous avez plusieurs objets avec le même nom. Pour le moment, nous n'avons pas vu les tableaux donc nous rechercherons un seul `GameObject` avec ce tag mais il est possible d'en récupérer plusieurs avec `GameObject.FindGameObjectsWithTag("TagDuGameObject")`, qui retourne un tableau de `GameObjects`. (il y a un "s" à `GameObjects`).

```
GameObject enemy = GameObject.FindWithTag("Enemy");
if (enemy != null)
{
    // Le GameObject avec le tag "Enemy" a été trouvé
}
else
{
    // Aucun GameObject avec le tag "Enemy" n'a été trouvé
}
```

1.3 Activer et désactiver les GameObjects par script

Lorsqu'on désactive un `GameObject`, tous ses composants et ceux de ses enfants sont également désactivés. Cela signifie qu'ils ne seront plus mis à jour ni rendus dans la scène. Pour désactiver un `GameObject`, vous pouvez utiliser la méthode `SetActive(false)`. Pour le réactiver, utilisez `SetActive(true)`.

Souvent, il sera réactivé à partir d'un autre script ou d'un événement dans le jeu. Nous verrons comment faire plus tard.

```
// Désactiver le GameObject
gameObject.SetActive(false);

// Réactiver le GameObject
gameObject.SetActive(true);
```

1.4 Accéder aux composants par script

Pour interagir avec les composants d'un `GameObject` via un script, vous pouvez utiliser la méthode `GetComponent<T>()`, où `T` est le type du composant que vous souhaitez accéder. C'est une bonne pratique de stocker une référence au composant dans une variable pour éviter d'appeler `GetComponent` plusieurs fois, ce qui peut être coûteux en termes de performance. On place souvent cette initialisation dans la méthode `Start()` ou `Awake()` au début du cycle de vie du script.

Ensuite, vous pouvez utiliser cette référence pour accéder aux propriétés et méthodes du composant.

Par exemple, pour accéder au composant `Rigidbody2D` d'un `GameObject`, vous pouvez faire comme suit :

```
Rigidbody2D rb;
SpriteRenderer sr;

void Start()
{
    rb = GetComponent<Rigidbody2D>();
    sr = GetComponent<SpriteRenderer>();
}
```

Une fois que vous avez la référence au composant, vous pouvez l'utiliser pour manipuler le `GameObject`. Par exemple, pour appliquer une force au `Rigidbody2D` ou changer la couleur du `SpriteRenderer` :

```
void Update()
{
    // Appliquer une force vers le haut
    if (Input.GetKeyDown(KeyCode.Space))
    {
        rb.AddForce(new Vector2(0, 300f));
    }

    // Changer la couleur du sprite en rouge
    if (Input.GetKeyDown(KeyCode.R))
    {
        sr.color = Color.red;
    }
}
```

Avec cette approche, vous pouvez facilement interagir avec les composants de vos `GameObjects` et créer des comportements dynamiques dans votre jeu.

1.5 Désactivation des composants

Il est également possible de désactiver individuellement des composants spécifiques d'un `GameObject` sans désactiver le `GameObject` lui-même. Cela peut être utile si vous souhaitez temporairement désactiver certaines fonctionnalités tout en gardant le `GameObject` actif dans la scène. Pour désactiver un composant, vous pouvez simplement définir sa propriété

enabled sur false. Par exemple, pour désactiver un SpriteRenderer, vous pouvez faire comme suit :

```
SpriteRenderer sr;

void Start()
{
    sr = GetComponent<SpriteRenderer>();
}

void DisableSpriteRenderer()
{
    sr.enabled = false;
}
```

De cette façon, le GameObject restera actif, mais le sprite ne sera plus rendu à l'écran. Vous pouvez réactiver le composant en définissant enabled sur true :

```
void EnableSpriteRenderer()
{
    sr.enabled = true;
}
```