Introduction aux animations simples (Tweening)

L'animation est un élément fondamental dans les jeux vidéo, permettant de rendre l'expérience plus dynamique et immersive. L'animation peut être utilisée pour diverses actions, comme le déplacement des objets, les effets visuels, les transitions d'interface, etc. Ce cours se concentrera sur les animations simples avec **Tweening** dans Unity.

---

1. Qu'est-ce que le Tweening ?

Le **Tweening** est une technique d'animation qui consiste à calculer et interpoler des valeurs entre deux états (par exemple, la position d'un objet au début et à la fin d'une animation) pour créer un mouvement fluide. Contrairement à l'animation par image clé (keyframe), où chaque position ou état est défini manuellement, le Tweening permet de créer des transitions plus naturelles sans avoir à définir chaque étape de l'animation.

---

2. Pourquoi Utiliser le Tweening ?

- **Fluidité :** Le Tweening permet des mouvements fluides et cohérents entre les points d’animation sans avoir à intervenir manuellement sur chaque frame.
- **Performance :** Il est plus performant que l’animation image par image, surtout pour des animations simples et répétitives.
- **Facilité d’utilisation :** Il est plus facile et rapide de configurer un Tween qu'une animation traditionnelle.

---

3. Types de Tweening dans Unity

Unity propose plusieurs façons d'implémenter le **Tweening** :

a) Animation par script avec Lerp (Linear Interpolation)

La fonction **Lerp** permet de calculer l'interpolation linéaire entre deux valeurs. Par exemple, pour animer la position d'un objet, vous pouvez utiliser Lerp pour interpoler entre deux positions.

Exemple de code :

```
using UnityEngine;

public class MoveObject : MonoBehaviour
{
    public Vector3 startPos;
    public Vector3 endPos;
    public float duration = 2.0f;
    private float startTime;

    void Start()
    {
        startPos = transform.position;
        endPos = new Vector3(5f, 0f, 0f); // Exemple de des
tination
        startTime = Time.time;
    }

    void Update()
    {
        // Calcule l'intervalle entre le temps actuel et le
temps de départ
        float t = (Time.time - startTime) / duration;
        transform.position = Vector3.Lerp(startPos, endPos,
t);
    }
}
```

Dans cet exemple, l'objet se déplace de la position **startPos** à la position **endPos** en **2 secondes**.

b) Animation avec le package DOTween

DOTween est un package Unity populaire qui facilite le Tweening et offre des fonctionnalités avancées comme les courbes d’animation personnalisées, la

gestion de plusieurs animations simultanées, et bien plus encore.

1. **Installation de DOTween :**
   - Ouvrez Unity et allez dans **Window > Package Manager**.
   - Recherchez **DOTween** dans la section **Unity Registry** et installez-le.
2. **Utilisation de DOTween pour déplacer un objet :**

```
using DG.Tweening;
using UnityEngine;

public class TweeningExample : MonoBehaviour
{
    void Start()
    {
        // Déplacer l'objet sur 3 unités en 2 secondes
        transform.DOMove(new Vector3(5f, 0f, 0f), 2f);
    }
}
```

1. **Animation de la rotation :**

```
using DG.Tweening;
using UnityEngine;

public class RotateObject : MonoBehaviour
{
    void Start()
    {
        // Rotation de l'objet sur l'axe Y en 3 secondes
        transform.DORotate(new Vector3(0f, 180f, 0f), 3f);
    }
}
```

1. **Animation de l’échelle :**

```
using DG.Tweening;
using UnityEngine;
```

```
public class ScaleObject : MonoBehaviour
{
    void Start()
    {
        // Changer l'échelle de l'objet à (2, 2, 2) en 2 se
condes
        transform.DOScale(new Vector3(2f, 2f, 2f), 2f);
    }
}
```

4. Courbes de Tweening et Easing

Le **Tweening** permet de définir des courbes de mouvement pour contrôler la vitesse d’animation tout au long de son cycle. Cela permet de rendre l'animation plus naturelle. Les **Easing Functions** sont des courbes qui permettent de contrôler l'accélération et la décélération.

- **Ease.Linear** : Mouvement constant, sans accélération ou décélération.
- **Ease.InOutQuad** : L'animation commence et termine lentement, avec une accélération au milieu.
- **Ease.OutBounce** : L'animation ralentit et donne un effet de rebond à la fin.

Exemple d’utilisation de DOTween avec Ease :

```
using DG.Tweening;
using UnityEngine;

public class EasingExample : MonoBehaviour
{
    void Start()
    {
        // Déplacer avec un effet d'accélération et décélér
ation
        transform.DOMove(new Vector3(5f, 0f, 0f), 2f).SetEa
se(Ease.InOutQuad);
    }
}
```

5. Animation d'UI avec Tweening

Le Tweening peut également être utilisé pour animer des éléments de l'interface utilisateur (UI) comme les boutons, les images et les textes. Voici quelques exemples d'animations courantes pour l'UI.

a) Animation de la transparence d'un objet UI :

```
using DG.Tweening;
using UnityEngine;
using UnityEngine.UI;

public class FadeUI : MonoBehaviour
{
    public Image image;

    void Start()
    {
        // Animer la transparence de l'image
        image.DOFade(0f, 2f); // De pleine opacité à transp
arente en 2 secondes
    }
}
```

b) Animation d'un bouton UI avec un effet de montée :

```
using DG.Tweening;
using UnityEngine;
using UnityEngine.UI;

public class ButtonAnimation : MonoBehaviour
{
    public Button button;

    void Start()
    {
        // Faire "monter" le bouton en animation
        button.transform.DOLocalMoveY(50f, 0.5f).SetEase(Ea
```

```
se.OutBounce);
    }
}
```

6. Conclusion

Le Tweening est une méthode puissante et flexible pour créer des animations fluides et dynamiques dans Unity. Que vous utilisiez des **scripts simples avec Lerp** ou un **package comme DOTween**, le Tweening vous permet de contrôler facilement les mouvements, la rotation, l'échelle, et même les effets UI dans vos jeux.

Le Tweening est particulièrement utile pour des animations simples qui ne nécessitent pas une interpolation manuelle de chaque image, tout en permettant des effets dynamiques et fluides qui enrichissent l'expérience de jeu.

Résumé des Concepts Clés :

- **Tweening** : Technique d'animation fluide entre deux états.
- **Lerp** : Interpolation linéaire pour animer la position et d'autres valeurs.
- **DOTween** : Outil puissant pour gérer le Tweening dans Unity avec des fonctionnalités avancées.
- **Easing** : Contrôle de l'accélération et de la décélération d'une animation.
- **Animations UI** : Utilisation du Tweening pour animer les éléments d'interface (ex. transparence, position, taille).

Le Tweening est un moyen rapide et efficace d'ajouter des animations simples dans vos projets Unity.