Ajout d'un système de score et affichage dans l'UI

Introduction

Dans de nombreux jeux, un système de score est essentiel pour suivre la performance du joueur. Cela peut inclure un simple compteur de score, mais aussi des informations supplémentaires, comme les vies restantes, les niveaux atteints, etc. Afficher ce score de manière efficace dans l'interface utilisateur (UI) du jeu permet d'améliorer l'interaction et l'engagement du joueur.

Ce cours se concentre sur la création d'un système de score dans Unity, son stockage et son affichage dynamique dans l'UI du jeu.

---

1. Concept du Système de Score

Le **système de score** dans un jeu suit l'évolution des actions du joueur (tuer des ennemis, collecter des objets, atteindre des objectifs). Le score peut être incrémenté ou décrémenté en fonction des événements du jeu.

Objectifs du système de score :

- Garder une trace du score du joueur tout au long de la partie.
- Afficher le score dans l'interface utilisateur de manière claire.
- Sauvegarder le score entre les sessions du jeu.

---

2. Implémentation du Système de Score

2.1 Créer un Script pour le Système de Score

1. Créez un nouveau script C# dans Unity nommé `ScoreManager`.
2. Ce script contiendra une variable pour le score, des méthodes pour incrémenter et récupérer ce score, et une fonction pour sauvegarder ou charger ce score.

Exemple de script ScoreManager :

```
using UnityEngine;
using UnityEngine.UI;  // Nécessaire pour manipuler l'UI

public class ScoreManager : MonoBehaviour
{
    public int score;  // Le score du joueur
    public Text scoreText;  // Référence au texte de l'UI o
ù le score sera affiché

    // Méthode pour incrémenter le score
    public void AddScore(int points)
    {
        score += points;
        UpdateScoreDisplay();  // Met à jour l'affichage du
score
    }

    // Méthode pour mettre à jour l'affichage du score
    private void UpdateScoreDisplay()
    {
        scoreText.text = "Score: " + score.ToString();  //
Affiche le score dans le Text de l'UI
    }

    // Sauvegarde du score avec PlayerPrefs
    public void SaveScore()
    {
        PlayerPrefs.SetInt("PlayerScore", score);
        PlayerPrefs.Save();
    }

    // Chargement du score depuis PlayerPrefs
    public void LoadScore()
    {
        if (PlayerPrefs.HasKey("PlayerScore"))
        {
            score = PlayerPrefs.GetInt("PlayerScore");
            UpdateScoreDisplay();  // Met à jour l'affichag
```

```
e avec le score chargé
        }
    }
}
```

2.2 Ajouter une Référence au Texte de l'UI

Pour afficher le score dans l'interface, vous devez avoir un composant **Text** (ou **TextMeshPro**) dans votre scène. Voici comment procéder :

1. **Créer un Text UI** : Allez dans le menu `GameObject -> UI -> Text` (ou `TextMeshPro` si vous utilisez TextMeshPro).
2. **Positionner le Text** : Déplacez-le à un endroit visible de l'écran (en haut, à droite de préférence pour les scores).
3. **Lier le script à l'UI** : Dans l'inspecteur, ajoutez le script `ScoreManager` à un objet de la scène (comme un GameObject vide). Ensuite, faites glisser le composant **Text** dans le champ `scoreText` du script `ScoreManager` dans l'inspecteur.

---

3. Affichage Dynamique du Score

Le système de score doit être mis à jour en temps réel en fonction des actions du joueur (collecte d'objets, élimination d'ennemis, etc.). Pour ce faire, vous pouvez appeler la méthode `AddScore()` à partir d'autres scripts, par exemple, lorsque le joueur collecte un objet ou touche un ennemi.

Exemple de mise à jour du score à partir d'un autre script :

Supposons que vous ayez un script pour gérer les objets collectés. Chaque fois que le joueur collecte un objet, vous augmentez le score.

```
public class CollectibleItem : MonoBehaviour
{
    public int pointsValue = 10;  // Valeur du score pour c
et objet

    private ScoreManager scoreManager;

    void Start()
    {
```

```
        // Trouver le ScoreManager dans la scène
        scoreManager = FindObjectOfType<ScoreManager>();
    }

    // Lorsqu'un objet est collecté
    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            scoreManager.AddScore(pointsValue);  // Incréme
nte le score
            Destroy(gameObject);  // Détruit l'objet collec
té
        }
    }
}
```

Dans cet exemple, chaque fois que le joueur collecte un objet, la méthode `AddScore()` est appelée et le score est mis à jour.

---

4. Sauvegarde et Chargement du Score

Pour que le score persiste même après la fermeture du jeu, vous pouvez utiliser **PlayerPrefs** pour sauvegarder et charger le score.

Sauvegarder le Score :

Dans votre script `ScoreManager`, la méthode `SaveScore()` sera utilisée pour sauvegarder le score à la fin du jeu ou après chaque session.

```
// À la fin du jeu ou à un moment clé
ScoreManager.Instance.SaveScore();
```

Charger le Score :

Lors du démarrage du jeu ou au début de la scène, vous pouvez charger le score sauvegardé.

```
// Charger le score dès que le jeu commence
```

```
ScoreManager.Instance.LoadScore();
```

Cela garantit que le score est récupéré même après que l'application soit fermée et rouverte.

---

5. Affichage du Score à l'Écran

Dans l'UI, le score sera affiché dynamiquement via le composant `Text` que vous avez associé à votre script. Le script mettra à jour cette valeur chaque fois que le score change.

---

6. Améliorations Potentielles

1. **Affichage du Score Maximale :** Ajoutez une fonctionnalité pour afficher le score maximal atteint au cours de plusieurs sessions de jeu.
2. **Animations et Effets Sonores :** Ajoutez des animations ou des effets sonores lorsque le score change pour améliorer l'expérience utilisateur.
3. **Affichage des Scores en Temps Réel :** Utilisez un script pour afficher un compteur de score qui s'actualise en fonction des actions du joueur.
4. **Affichage du Score sur un Écran de Fin de Partie :** Si vous avez une scène de fin de jeu, vous pouvez afficher le score final du joueur.

Conclusion

L'ajout d'un système de score dans Unity est une fonctionnalité fondamentale pour de nombreux jeux. En combinant la gestion de la logique du score avec un affichage dynamique dans l'UI, vous offrez au joueur une rétroaction en temps réel sur sa progression. La sauvegarde et le chargement du score garantissent également que les joueurs peuvent poursuivre leur jeu là où ils l'ont laissé, augmentant ainsi l'engagement et la satisfaction.