Projet : Améliorer la fluidité et les performances du jeu

L'objectif de cet exercice est de mettre en œuvre des techniques d'optimisation pour améliorer la fluidité et les performances de votre jeu. Vous allez utiliser des outils et des méthodes pour réduire la charge du processeur et du GPU, tout en optimisant l'utilisation de la mémoire et la gestion des objets.

Étapes de l'exercice :

1. Analyse des Performances avec Unity Profiler

Avant d'optimiser, vous devez d'abord analyser la performance de votre jeu à l'aide du **Profiler** d'Unity. Cela vous permettra d'identifier les zones de votre jeu qui causent des ralentissements.

- **Utilisation du Profiler :**
  - Ouvrez le Profiler dans Unity (menu **Window** > **Analysis** > **Profiler**).
  - Exécutez votre jeu et surveillez les paramètres comme l'utilisation du **CPU**, **GPU**, et **RAM**.
  - Identifiez les pics de performance qui correspondent aux zones où votre jeu ralentit.

2. Optimisation des Scripts

Les scripts qui s'exécutent dans la méthode `Update()` peuvent causer des problèmes de performance. Voici quelques actions à prendre pour optimiser les scripts :

- **Réduisez les appels dans `Update()` :**
  - Ne mettez que les calculs les plus nécessaires dans la méthode `Update()`.
  - Utilisez des **coroutines** ou des événements pour exécuter des actions uniquement lorsque cela est nécessaire.
- **Réduisez l'utilisation de `Find()` et d'autres appels coûteux :**

    - Evitez d'utiliser `GameObject.Find()` dans la boucle de jeu. Préférez la référence directe aux objets.

- **Utilisez des pools d'objets pour éviter des allocations répétées :**
    - Créez des pools d'objets pour réutiliser des objets (par exemple, des ennemis ou des projectiles) au lieu de les créer et de les détruire fréquemment.

3. Optimisation Graphique

L'optimisation graphique permet de réduire la charge du GPU et d'améliorer les performances, surtout sur des appareils avec des ressources limitées.

- **Utilisation du Level of Detail (LOD) :**
    - Implémentez des modèles à différentes résolutions pour les objets 3D qui s'éloignent de la caméra. Cela réduit la charge du GPU.
- **Optimisation des shaders :**
    - Utilisez des shaders plus simples et évitez les calculs coûteux dans les shaders (comme les calculs de lumière complexes).
- **Culling :**
    - Implémentez des techniques de culling (comme le **frustum culling**) pour ne rendre que les objets visibles à la caméra.
    - Utilisez l'**occlusion culling** pour ne pas afficher des objets bloqués par d'autres objets.

4. Optimisation de la Mémoire

Réduire l'utilisation de la mémoire est essentiel pour améliorer la fluidité, surtout sur les plateformes mobiles.

- **Optimisation des Textures :**
    - Réduisez la taille des textures utilisées dans votre jeu sans compromettre leur qualité.
    - Utilisez des **textures compressées** comme le format **ETC** pour les plateformes mobiles.
- **Utilisation d'Assets plus légers :**

    - Privilégiez des modèles 3D moins détaillés ou moins complexes si possible.
    - Supprimez les assets inutilisés et vérifiez qu'ils sont bien chargés avant de les utiliser.
- **Gestion de la mémoire dynamique :**
    - Utilisez des techniques comme `Resources.UnloadUnusedAssets()` pour libérer la mémoire après la destruction d'objets non utilisés.

5. Optimisation de l'Audio

Les effets sonores peuvent également impacter les performances, en particulier s'il y a de nombreux sons simultanés.

- **Réduisez le nombre d'effets audio joués simultanément :**
    - Limitez le nombre de sons actifs à tout moment.
    - Utilisez des **Audio Mixers** pour ajuster dynamiquement le volume des sons afin d'éviter la surcharge audio.
- **Compression des fichiers audio :**
    - Compressez les fichiers audio à un format moins lourd (comme **MP3** ou **OGG**) pour économiser de la mémoire.

6. Testez et Comparez les Résultats

Après avoir implémenté ces optimisations, testez à nouveau les performances de votre jeu.

- **Comparez les performances avant et après les optimisations** à l'aide du Profiler.
- Vérifiez la fluidité du jeu et les temps de chargement.
- Testez le jeu sur plusieurs appareils (si possible) pour évaluer les améliorations.

Critères d'évaluation :

- Utilisation efficace du Profiler pour identifier les problèmes de performance.
- Application des techniques d'optimisation adaptées aux scripts, graphismes, mémoire et audio.

- Amélioration mesurable des performances du jeu (framerate stable, réduction des pics de consommation de CPU/GPU, diminution de l'utilisation mémoire).
- Test de la fluidité du jeu avant et après les optimisations.

Bonus (Facultatif) :

- Implémenter des **systèmes de LOD** pour les objets 3D.
- Créer une **gestion dynamique de la qualité graphique** permettant au jeu de s'adapter en fonction des performances du matériel.