Persistance des données avec Unity

Formation dispensée par Gilbert DEMORGNY

La Citation du jour :

"Mal nommer les choses, c’est ajouter au malheur du monde."

- Albert Camus

Et pour commencer :

- Rappel des épisodes précédents
- Et les projets alors ?
- Et aujourd’hui on fait quoi ?

Programme de la journée :

Projet : ChildRoom

1. Avant la persistance viennent les données
2. La persistance oui mais encore
3. Le ScriptableObject
4. Les StreamingAssets
5. Lecture - Ecriture de fichiers

Avant la persistance viennent les données :

Pour rendre des données persistantes, il faut d'abord définir quelles données seront pertinentes et quel seront leur type.

Exemple :

- pour rendre un score persistant, une variable de type Int suffit.
- pour savoir quel joueur a marqué le score, il faudra un Int (pour le score) et un String (pour le nom du joueur).
- pour un système de highscore, il faudra un tableau de Int pour les scores et un tableau de string pour les nom des joueurs, OU un tableau contenant une structure composée d'un int et d'un string pour avoir directement le duo d'information

En résumé, il faut déterminer les informations nécessaires, déterminer les variables associées avec le type de donnée adapté.

La persistance oui mais encore :

Lorsque l'on parle de persistance de donnée, on mélange deux aspects proches mais différents :

- D'un part il y a persistance pendant que l'application fonctionne, exemple conservation d'un score d'un niveau à un autre, d'un nombre de point de vie ou d'un équipement.
- D'autre part, il y a la persistance entre les parties. C'est la sauvegarde. Cela s'applique aux données de jeu des joueurs mais aussi des paramètres de jeu.

On remarquera que les deux peuvent être complémentaires, on peut conserver des données pendant le jeu, puis décider de les sauvegarder pour les charger plus tard.

On distingue donc le système de persistance pendant la partie, pour lequel nous utiliserons des ScriptablesObjects du système de sauvegarde à proprement parler qui lui écrira des informations sur le disque dur.

Le ScriptableObject :

Le ScriptableObject dans unity est l'équivalent du préfab mais pour les données. La fonction de base d'un scriptable object est de contenir des informations qui seront utilisées ailleurs.

Exemple : vous avez besoin de savoir si le joueur à déjà ouvert une porte spécifique, plusieurs solutions :

- Vous avez fait un système d'évent, et tous les gens concernés vont recevoir l'information et donc vont la stocker (Mauvaise idée !!!! une information d'état ne doit être disponible qu'à un endroit, jamais à plusieurs, si une mise à jour ne fait pas vous ne le saurez jamais !)
- Une de vos classe dispose de l'information et toutes les autres classes ont une variable qui leur permet d'accéder à la classe en question (fonctionnel mais un peu lourd)
- La classe qui doit gérer le changement d'état de la porte contient un scriptableobject avec l'information et mets à jour la donnée dans le scriptable object. Toutes les autres classe qui doivent avoir l'information ont accès au scriptableobject et peuvent la lire directement (Proche du précédent mais moins gourmand en ressources).

On voit ici que le ScriptableObject contient l'information et la partage avec toutes les classes qui y ont accès.
On notera qu'il concerve les données UNIQUEMENT pendant que l'application fonctionne, les données sont perdues en cas d'arrêt.

Le ScriptableObject :

Exercice 1 :

Dans le projet ChildRoom en VR, créez un scriptableobject contenant des informations d'état pour élément interactable de la pièce. Le mettre en place dans toutes les classes qui en ont besoin avec la mise à jour des données.

Pour rappel :

- la porte du placard
- le tiroir
- la balle
- l'interrupteur
- le mur

Second rappel : on définit d'abord les données et leur type avant de créer un scriptable object.

Le ScriptableObject :

Exercice 2 :

Une fois le scriptable object mis en place, créez un système de rechargement de scène et faites en sorte que la scène reprenne la même configuration que la scène que vous venez de quitter.

La difficulté vient du fait que votre scène ne charge pas les données du ScriptableObject au chargement de la scène donc il faut corriger cela.

Les StreamingAssets :

Un projet Unity est installé sous forme de package. Aucun élément du projet n’est accessible directement sur le device.

Le répertoire StreamingAssets (Attention à la syntaxe, unity n’acceptera que cette écriture spécifique) est défini comme un dossier de dialogue avec le reste du device. On peut donc y écrire et y modifier des fichiers.

La documentation unity indique quels sont les chemins d'accès selon le device.

Nous utiliserons ce dossier pour écrire nos fichiers de sauvegarde. Il est impératif de le créer avant de chercher à l’utiliser.

Lecture - Écriture dans des fichiers :

Pour effectuer une sauvegarde nous allons utiliser deux choses :

- La class JSONUtility d'Unity, qui permet de générer une représentation JSON des variables public d'un objet OU de générer un objet à partir d'une représentation JSON
- La class File de la bibliotheque System.IO du C#, qui permet la gestion des fichiers

L'idée est de transformer un objet (notre scriptableobject) en JSON puis de l'écrire dans un fichier.
Pour recharger les données, nous lierons le fichier en question puis nous transformerons les données JSON en scriptableobject.

Lecture - Écriture dans des fichiers :

Ecriture d'un fichier :

```
public bool SaveDataToJson(string dataFormated, string fichier)
{
    _Path = Path.Combine(Application.streamingAssetsPath,fichier);
    if(!File.Exists(_Path))
    {
        File.Create(_Path);
    }
    File.WriteAllText(_Path, dataFormated);

    return File.ReadAllText(_Path) == dataFormated;
}
```

Lecture - Écriture dans des fichiers :

Lecture d'un fichier :

```
public string LoadFromJson(string fichier)
{
    _Path = Path.Combine(Application.streamingAssetsPath,fichier);
    if (File.Exists(_Path))
    {
        return File.ReadAllText(_Path);
    } else
    {
        return null;
    }
}
```