Fiche mémo **PlayerPrefs**

Intro : La classe **PlayerPrefs** de **Unity** permet **de sauvegarder sur le disque de l'utilisateur des données**. Ces données sont enregistrées et conservées même après la fermeture du jeu ou l'extinction de l'ordinateur de l'utilisateur.

Il est possible de sauvegarder (Set) des données et de charger (Get) des données de type **int**, **float** ou **string**.

Appel de la fonction	Exemple de retour	Commentaire / explication
PlayerPrefs.GetInt(''vie'') ;	3	Charge la donnée stockée sur la clé 'vie'. 3 est un exemple fictif de valeur enregistrée
PlayerPrefs.GetInt(''vie'', 3) ;	2	Il s'agit de la même fonction. La valeur retournée ici est 2 (fictif). Le '3' spécifié permet d'indiquer une valeur par défaut si aucune valeur n'est trouvée sur le disque.
PlayerPrefs.SetInt(''vie'', 2) ;		Permet de sauvegarder la valeur 2 sur la clé 'vie'. Si une autre valeur existe déjà sur cette clé, elle est écrasée.
PlayerPrefs.GetFloat(''vitesse'') ;	42,79	Permet de charger un float (valeur à virgule).
PlayerPrefs.GetFloat(''vitesse'', 100.0f) ;	100,0	Permet de charger un float et de donner une valeur par défaut si rien n'est trouvé. Ici, la valeur par défaut est utilisée car rien n'a été trouvé sur le disque.
PlayerPrefs.SetFloat(''vitesse'', 2.2f) ;		On stocke la valeur 2,2 dans la clé vitesse.
PlayerPrefs.GetString(''test'') ;	''exemple''	On charge ce qui est stocké dans la clé 'test'. Il s'agit de la valeur 'exemple' qui a été chargée.
PlayerPrefs.GetString(''test'', ''toto'') ;	''exemple''	On charge la valeur comme dans l'exemple précédent. Si jamais aucune valeur n'est trouvée alors c'est la valeur par défaut ''toto'' qui serait retourné.
PlayerPrefs.SetString(''test'', ''titi'') ;		Ici, on stocke la valeur 'titi' dans la clé 'test'.

Aller plus loin : Il est possible de sauvegarder beaucoup de choses avec les PlayerPrefs. Vous n'avez accès qu'à 3 types primitifs (int, float et string) mais vous pouvez stocker ce que vous voulez ! Par exemple pour stocker / charger un **Vector2** (Idem pour **Vector3**) vous pourriez créer les fonctions suivantes :

```
public void SavePlayerPosition()
{
    // Sauvegarde de la position du joueur (axes X et Y)
    PlayerPrefs.SetFloat("posX", this.gameObject.transform.position.x);
    PlayerPrefs.SetFloat("posY", this.gameObject.transform.position.y);
}

public void LoadPlayerPosition()
{
    // Chargement de la position X,Y sauvegardée
    float savedXpos = PlayerPrefs.GetFloat("posX", 0);
    float savedYpos = PlayerPrefs.GetFloat("posY", 0);
    // On positionne le joueur avec un Vector2
    this.gameObject.transform.position = new Vector2(savedXpos, savedYpos);
}
```

Vous pouvez également sauvegarder des chaînes complexes (JSON, CSV, …) avec SetString puis les charger et enfin des découper (parser) pour récupérer un ensemble de données (vie, mana, xp, or, force, niveau…).