Introduction aux Coroutines et à la fonction "yield WaitForSeconds" dans Unity

Le modèle 3D utilisé dans ce tutoriel : Naval Cannon (https://sketchfab.com/3d-models/naval-cannon-3407164113c2487e9391aa8d2b3902c2)

Bonjour à tous!

Il s'agit d'un didacticiel d'introduction à l'utilisation des coroutines dans Unity ; c'est-à-dire : sur les fonctions qui peuvent être exécutées simultanément avec le cycle d'exécution du jeu principal. Je traiterai le sujet avec un exemple pratique, qui rendra le concept clair et, en attendant, j'en profiterai pour utiliser également la fonction WaitForSeconds, qui fait attendre l'exécution d'une opération pendant quelques secondes.

Le tutoriel est destiné à ceux qui ont déjà une connaissance de base de la programmation C# dans Unity ; il a été enregistré avec la version 2020 de Unity ; cependant, les coroutines (qui sont disponibles dans de nombreux langages de programmation) sont également disponibles dans les versions précédentes de Unity et, bien sûr, elles seront également disponibles dans les versions ultérieures.

Dans la scène que j'utilise dans cet exemple, il y a un modèle 3D d'un canon naval.

Nous voulons que le canon tire le Prefab d'un boulet de canon (une simple sphère noire, en fait) automatiquement toutes les 3 secondes, jusqu'à ce qu'un événement externe se produise qui interrompt cette routine (dans ce scénario, l'intervention externe sera l'utilisateur appuyant sur la touche S ("Stop", "Arrêter")).

Tout d'abord, je crée un script pour le canon, l'appelant "autoshoot".

Dans la classe principale, je définis trois variables :

```
public GameObject spawnPrefab
```

à cette variable publique, je vais lier le préfabriqué Cannonball dans l'éditeur Unity

```
Vector3 spawnPosition

Quaternion spawnRotation
```

J'ai besoin des dernières variables pour spécifier le point de départ de la balle par rapport au canon ; J'ai calculé ce point et l'orientation initiale, j'initialise donc ces deux valeurs au sein de la fonction Start:

```
spawnPosition = transform.position + new Vector3(-1.2f, 1.8f, 0f)

spawnRotation = Quaternion.Euler(-10f, -90f, 0f)
```

D'après la description du problème, il est clair que nous devons implémenter une boucle qui se répète indéfiniment (à moins qu'un événement externe ne la bloque) et qu'elle doit inclure une instruction de délai, afin de la faire attendre 3 secondes avant de passer au suivant itération.

L'instruction qui nous permet d'attendre quelques secondes est WaitForSeconds, qui prend comme argument le nombre de secondes à attendre.

A ce stade, il est clair que la boucle principale à exécuter est donnée par ce bloc WHILE (que j'écris, pour le moment, dans un endroit vide du script):

```
while(true) {

GameObject cannonBall = Instantiate(spawnPrefab, spawnPosition, spawnRotation, null);

yield return new WaitForSeconds(3);

}
```

[Le boulet de canon Prefab est livré avec un script qui, au démarrage, fournit une première poussée vers l'avant à la boule et met l'objet à s'autodétruire après 3 secondes, afin de ne pas remplir l'univers virtuel et la mémoire de boulets de canon]

Où devons-nous placer ce bloc WHILE ?

Nous ne pouvons pas le mettre dans Start, soit à cause d'une erreur de programmation (nous ne pouvons pas y mettre le rendement WaitForSeconds), soit parce que le programme serait en fait bloqué dans Start (NE PAS essayer de le mettre dans Start sans WaitForSeconds, pour essayer la boucle While ainsi défini : l'unité gèlerait) ; de même, nous ne pouvons pas le mettre en Update...

Le problème n'est pas résolu si on met WaitForSeconds dans Update, sans la boucle WHILE, car Update est toujours appelé à chaque frame du jeu : WaitForSeconds ne le retarderait pas.

Il faut donc insérer ce bloc de code à l'intérieur d'une fonction qui retourne notamment le type IEnumerator ; nous appelons cette fonction shoot.

Pour démarrer cette fonction à l'intérieur d'une Coroutine, nous devons utiliser l'instruction StartCoroutine, qui doit avoir le nom de la fonction en paramètre, entre guillemets ; puisque le canon doit commencer à tirer immédiatement, nous écrivons :

```
StartCoroutine("shoot");
```

dans Start, immédiatement après l'initialisation de spawnPosition et spawnRotation.

Enregistrons le script, allons dans l'éditeur Unity et essayons le script défini jusqu'à présent.

Revenons au script pour insérer deux instructions, afin de pouvoir arrêter (touche S) ou redémarrer (touche R) la Coroutine:

```
if(Input.GetKeyDown(KeyCode.S)) StopCoroutine("shoot");

if(Input.GetKeyDown(KeyCode.R)) StartCoroutine("shoot");
```

Enregistrons le script, allons dans l'éditeur Unity et essayons le script défini jusqu'à présent.

Voilà, c'est tout pour ce petit tuto ! J'espère qu'il vous a été utile ! À bientôt!