Einführung in Coroutines und die Funktion "yield WaitForSeconds" in Unity

Das in diesem Tutorial verwendete 3D-Modell: Naval Cannon (https://sketchfab.com/3d-models/naval-cannon-3407164113c2487e9391aa8d2b3902c2)

Hallo zusammen!

Dies ist ein einführendes Tutorial über die Verwendung von **Coroutines** in **Unity**; das heißt: über die Funktionen, die parallel zum Hauptausführungszyklus des Spiels ausgeführt werden können. Ich werde das Thema anhand eines praktischen Beispiels behandeln, um das Konzept zu verdeutlichen und in der Zwischenzeit auch die Funktion **WaitForSeconds** nutzen, die einige Sekunden auf die Ausführung einer Operation warten lässt.

Das Tutorial richtet sich an diejenigen, die bereits über Grundkenntnisse der **C#**-Programmierung in **Unity** verfügen; es wurde mit der Version 2020 von **Unity** aufgezeichnet; die **Coroutines** (die in vielen Programmiersprachen verfügbar sind) sind jedoch auch in früheren Versionen von **Unity** verfügbar und werden natürlich auch in späteren Versionen verfügbar sein.

In der Szene, die ich in diesem Beispiel verwende, gibt es ein 3D-Modell einer Schiffskanone.

Wir möchten, dass die Kanone die **Prefab** einer Kanonenkugel (eigentlich eine einfache schwarze Kugel) automatisch alle 3 Sekunden abfeuert, bis ein externes Ereignis eintritt, das diese Routine unterbricht (in diesem Szenario wird der externe Eingriff der Benutzer sein, der die Taste **S** ("**Stop**") drückt).

Als erstes erstelle ich ein Skript für die Kanone und nenne es "**autoshoot**".

Innerhalb der Hauptklasse definiere ich drei Variablen:

```
public GameObject spawnPrefab
```

mit dieser öffentlichen Variable verknüpfe ich die Kanonenkugel-**Prefab** im **Unity-Editor**

```
Vector3 spawnPosition

Quaternion spawnRotation
```

Ich benötige die letzten Variablen, um den Startpunkt der Kugel in Bezug auf die Kanone zu bestimmen; ich habe diesen Punkt und die anfängliche Ausrichtung berechnet, also initialisiere ich diese beiden Werte innerhalb der **Start**-Funktion:

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

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

Aus der Problembeschreibung geht hervor, dass wir eine Schleife implementieren müssen, die sich unendlich oft wiederholt (es sei denn, ein externes Ereignis blockiert sie), und dass sie eine Verzögerungsanweisung enthalten muss, damit sie 3 Sekunden wartet, bevor sie zur nächsten Iteration übergeht.

Die Anweisung, die uns erlaubt, einige Sekunden zu warten, ist **WaitForSeconds**, die als Argument die Anzahl der zu wartenden Sekunden erhält.

Zu diesem Zeitpunkt ist klar, dass die auszuführende Hauptschleife durch diesen **WHILE**-Block gegeben ist (den ich im Moment an eine leere Stelle im Skript schreibe):

```
while(true) {

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

yield return new WaitForSeconds(3);

}
```

*[Die Kanonenkugel-**Prefab** wird mit einem Skript geliefert, das der Kugel beim Start einen anfänglichen Vorwärtsschub gibt und das Objekt so einstellt, dass es sich nach 3 Sekunden selbst zerstört, um das virtuelle Universum und den Speicher nicht mit Kanonenkugeln zu füllen.]*

Wo sollen wir diesen **WHILE**-Block platzieren?

Wir können ihn nicht in **Start** platzieren, entweder aufgrund eines Programmierfehlers (wir können dort kein **Yield-WaitForSeconds** einfügen), oder weil das Programm tatsächlich in **Start** stecken bleiben würde (versuchen Sie NICHT, ihn in **Start** ohne **WaitForSeconds** zu platzieren, um die so definierte **While**-Schleife auszuprobieren: **Unity** würde einfrieren); ebenso können wir es nicht in **Update** setzen...

Das Problem wird nicht gelöst, wenn wir **WaitForSeconds** in **Update** einfügen, ohne die **While**-Schleife, denn **Update** wird immer noch bei jedem Frame des Spiels aufgerufen: **WaitForSeconds** würde es nicht verzögern.

Wir müssen daher diesen Codeblock in eine Funktion einfügen, die insbesondere den Typ **IEnumerator** zurückgibt; wir nennen diese Funktion **shoot**.

Um diese Funktion innerhalb einer **Coroutine** zu starten, müssen wir die Anweisung **StartCoroutine** verwenden, die den Funktionsnamen als Parameter zwischen Anführungszeichen haben muss; da die Kanone sofort zu feuern

beginnen muss, schreiben wir:

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

innerhalb von **Start**, unmittelbar nach der Initialisierung von **spawnPosition** und **spawnRotation**.

Wir speichern das Skript, gehen in den **Unity-Editor** und probieren das bis hierhin definierte Skript aus.

Gehen wir zurück zum Skript, um zwei Anweisungen einzufügen, damit wir die **Coroutine** anhalten (Taste **S**) oder neu starten (Taste **R**) können:

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

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

Speichern wir das Skript, gehen wir zum **Unity-Editor** und probieren wir das bis hierhin definierte Skript aus.

Nun, das war's mit diesem kurzen Tutorial! Ich hoffe, es war hilfreich für euch! Wir sehen uns bald wieder!