EINSTIEG

In anderen Spielen sehen wir manchmal, wie der Spieler nicht immer am Anfang des Levels starten muss, wenn er stirbt, sondern er die Möglichkeit hat, sogenannte Checkpoints freizuschalten, wenn er bestimmte Orte im Spiel erreicht. Dies wollen wir bei uns auch einbauen. Zuerst brauchen wir wieder einen Marker dafür. Hierfür ziehen wir uns einen weiteren Stein in das Level und platzieren ihn ungefähr auf der Hälfte des Levels auf einer Plattform. Wir wollen, dass das Spiel merkt, sobald man einen solchen Marker passiert und dann einen anderen Checkpoint spawned. Hierfür benötigt der Stein, wie auch beim Level Übergang, einen Collider. Deshalb fügen wir hier eine *Box Collider 2D* Komponente hinzu. Und diese machen wir auch wieder zu einem Trigger, damit der Spieler hindurchlaufen kann. Dann ändern wir noch den Tag dieses Objekts auf *Respawn*. Nun erstellen wir ein neues Skript. Dieses Skript ist aber nicht für die Checkpoint Markierung, sondern für den Player. Wir wollen, dass der Spieler selber dafür zuständig ist, sich um die Checkpoints zu kümmern. Deshalb erstellen wir unter *Skripts -> Player* das *PlayerCheckpoints* Skript. Das neue Skript hängen wir dann sofort an den Player an. In diesem neuen Skript benötigen wir zuerst einmal eine neue Variable in der Klasse, in der die Position des letzten Checkpoints gespeichert ist, damit der Unity weiß, wo er den Spieler respawnen soll, sobald er stirbt:

```
Private Vector3 currentCheckpointPosition;
```

Jetzt benötigen wir wieder die Methode, die ausgeführt wird, sobald man einen Trigger berührt. Und auch hier wollen wir zuerst überprüfen, ob es auch das Objekt mit dem Tag *Respawn* ist, das den Trigger berührt:

```
Private void OnTriggerEnter2D(Collider2D collision)
{
      if(collision.tag == „Respawn“)
      {

      }
}
```

Wir wollen, dass jedes Mal, wenn ein Checkpoint berührt wird, die Position dieses Checkpoints als letzte Checkpoint Position gespeichert wird, damit der Spieler auch wirklich beim letzten Checkpoint spawned. Dafür schreiben wir in die *If-Abfrage* das Folgende:

```
currentCheckpointPosition = collision.transform.position;
```

Jetzt stellt sich die Frage, was mit dem Spielcharakter passieren soll, wenn noch kein Checkpoint erreicht wurde. Also ganz am Anfang des Levels. Hier können wir einfach sagen, dassn wenn noch kein Checkpoint erreicht wurde, der Spieler auf der Startposition

des Frosches gespawned werden soll. Dies tun wir in der *Start()* Funktion der Klasse. Wie wir uns erinnern, wird diese aufgerufen, wenn der Spieler erstellt wird. Und das geschieht jedes Mal, wenn ein neues Level geladen wird.

```
private void Start()
{
      currentChekpointPosition = transform.position;
}
```

Wir müssen den Wert, den wir jetzt in der Variable haben, noch in den Player schreiben. Und dafür erstellen wir eine neue Funktion, die wir *Respawn* nennen.

```
public void Respawn()
{
      transform.position = currentCheckpointPosition;
}
```

Da diese Funktion *public* ist, können wir sie von anderen Skripten aus aufrufen. Wir müssen diese Funktion jetzt jedes Mal ausführen, wenn der Spieler stirbt. Was mit dem Spieler passiert, sobald er ins Wasser fällt, entscheiden wir ja im *Water* Skript. Deshalb gehen wir dort mal hin. Zuerst brauchen wir hier jedoch eine Referenz auf unser *PlayerCheckpoints* Skript. Hierfür erstellen wir ganz oben in der Klasse eine neue Variable:

```
Public PlayerCheckpoints playerCheckpoints;
```

Im Unity Editor ist beim *Water* Skript jetzt ein neues Feld mit dem Namen *Player Checkpoints* aufgetaucht. Da unser *PlayerCheckpoints* Skript auf dem Spieler liegt, müssen wir das *Player*-Objekt in dieses Feld ziehen. Im *Water* Skript löschen wir dann die Zeile, in der If-Abfrage, in der wir die komplette Szene neu laden und schreiben stattdessen das rein:

```
playerCheckpoints.Respawn();
```

Nun wird nicht jedes Mal das Level neu geladen, sondern nur noch die Position des Spielers auf den letzten Checkpoint gesetzt. Im Spiel sehen wir jetzt, dass der Frosch tatsächlich am Checkpoint erscheint, sobald er diesen berührt hat. Da wir beim Wasser und dem Spieler nicht mit *Prefabs* gearbeitet haben, heißt das, dass die anderen Level unsere Änderungen nicht automatisch übernommen haben. Deshalb müssen wir jetzt noch durch die anderen Level gehen und die Änderungen am Spieler und dem Wasser dort auch durchführen, damit es dort auch funktioniert.

AUFGABE

Baue die Funktion ein, mehrere Checkpoints benutzen zu können. So soll der Spieler, sobald er einen Checkpoint erreicht, zukünftig von diesem starten. Hierfür soll der gleiche Stein benutzt werden wie der, der für das Ziel benutzt wurde