EINSTIEG

In diesem Kapitel wollen wir dem Spiel sagen, wann es welche Animation für den Frosch abspielen soll. Hierfür benötigen wir ein neues Fenster. Und zwar das **Animator** Fenster. Dieses ist wie auch das **Animation** Fenster aus dem letzten Kapitel im *Window* dropdown Menü zu finden. Hierfür muss *Window -> Animation -> Animator gedrückt werden.*

Im neuen Fenster sehen wir schon, dass da unsere zwei Animationen *walk* und *idle* drinnen sind. Und wir sehen auch, dass *walk* Orange hervorgehoben ist. Das ist der Clip der ganz am Anfang abgespielt wird, sobald das Spiel läuft. Natürlich können wir den auch ändern. Um das zu tun, müssen wir *idle* mit einem Rechtsklick anwählen und dort dann *Set as Layer Default* Status drücken. Wenn wir jetzt wieder das Spiel abspielen, sehen wir, dass der Frosch nicht mehr die ganze Zeit läuft, sondern jetzt die *idle* Animation abspielt.

Um jetzt Logik einbauen zu können, benutzen wir sogenannte *Transitions*. Diese erstellen wir, indem wir auf eine der Animationen im *Animator* Fenster einen Rechtsklick tätigen und dort *Make Transition* auswählen. Dann können wir von dort aus eine Transition ziehen, indem wir auf eine andere Animation klicken. Jetzt fügen wir zwei Transitions hinzu, die von walk zu idle gehen und umgekehrt. Eine weitere Sache, die wir benötigen, sind Parameter. Diese ähneln Variablen im Skript. Und die können wir hinzufügen, indem wir statt *Layers* im **Animator** Fenster *Parameters* anklicken. Wir können einen neuen Parameter hinzufügen, indem wir auf das Plus drücken. Dann können auswählen, von welchem Typ der Parameter sein soll. In unserem Fall benötigen wir einen Bool Parameter. Diesen nennen wir *isWalking*.

Wenn wir jetzt die Transition von idle zu walk auswählen, erscheinen viele Einstellungsmöglichkeiten im **Inspector**. Zunächst müssen wir hier die *Has Exit Time* ausschalten und die *Transition Duration* auf 0 setzen. Ganz unten hat man die Möglichkeit, Konditionen hinzuzufügen. Wenn diese gegeben sind, wird die Transition ausgelöst und die aktive Animation ändert sich. Wir fügen die Kondition, wenn *isWalking* ist *true* hinzu. Das gleiche machen wir jetzt mal umgekehrt in die andere Richtung. Wir wählen die Transition von walk zu idle, ändern die Einstellungen genauso wie bei der anderen Transition, mit dem Unterschied, dass wir am Ende *isWalking* ist *false* als Kondition angeben. Nun wird jedes Mal, wenn *isWalking* wahr ist, die *walk* Animation abgespielt, und wenn *isWalking* Falsch ist, die Idle Animation.

Nun müssen wir den Parameter noch so beeinflussen, dass *isWalking* wahr ist immer, wenn sich der Frosch bewegt. Dies tun wir über den Code. Hierfür gehen wir zurück in das *PlayerMovement* Skript. Wir erstellen eine neue Variable vom Typ Animator und nennen sie *animator*:

```
public Animator animator;
```

Nun können wir vom Unity Fenster aus dieser Variable einen Wert zuordnen. Um das zu machen, gehen wir zum Spieler und gehen dort zur *PlayerMovement* Komponenten. Dort ist ein neues Feld mit dem Namen Animator hinzugekommen. Jetzt ziehen wir das *PlayerSprite* Objekt in das neue Feld. Dieser Schritt wird häufig vergessen. Wenn also etwas nicht läuft, wie es sollte, sollte man mal seine Referenzen überprüfen.

Nun gehen wir wieder zurück in das Skript. Wie wir zuvor herausgefunden haben, hält die Variable *horizontal* den Wert -1, 0 oder 1, je nachdem, ob wir den Pfeil nach rechts oder links drücken, oder keine der beiden Richtungen. Dies können wir auch nochmal überprüfen, indem wir den Wert in die **Console** schreiben.

```
Debug.Log(horizontal);
```

Nun benötigen wir einen Gleichheitsoperator. Davon gibt es zwei, == und *!*=.

- == brauchen wir, wenn wir überprüfen wollen, ob etwas gleich ist.
- *!*= ist dafür da, um zu prüfen, ob etwas ungleich ist.

In unserem Fall benötigen wir den ungleich Operator. Wir wollen nämlich herausfinden, ob *horizontal* ungleich 0 ist. Wenn horizontal 1 oder -1 ist, wissen wir, dass sich der Frosch gerade bewegt. In der Update Funktion fügen wir nun unter der Zeile, in der wir den Input auslesen, das Folgende hinzu:

```
bool isWalking = horizontal != 0;
```

Diese Variable enthält jetzt einen Wert, der uns sagt, ob der Frosch sich gerade bewegt. Diesen schreiben wir mit der folgenden Zeile in den Parameter:

```
animator.SetBool(„isWalking", isWalking);
```

Der String beschreibt, in welchen Parameter es hineingeschrieben werden soll. Der Wert danach gibt an, was hineingeschrieben werden soll.

Wie sich der Parameter ändert, können wir jetzt auch im Animator Fenster sehen. Und der Frosch läuft nun auch erst, wenn er sich wirklich bewegt.

AUFGABE

Füge die Animation Clips aus dem letzten Kapitel in einen *Animator Controller ein* und benutze ihn so, dass der Frosch die Lauf-Animation abspielt, wenn er sich über den Boden bewegt und auch beachtet wird, in welche Richtung er sich bewegt.