



Einstieg

Für den nächsten Schritt wurde bereits ein Skript geschrieben, um es uns ein wenig einfacher zu machen. Um dieses zu importieren, müssen wir es, wie wir das mit den Grafiken gemacht haben, in das Projekt ziehen. Und zwar ziehen wir das Skript mit dem Namen *CharacterController2D* neben unser *PlayerMovement* Skript in den Player Ordner. Das neue Skript verbinden wir wieder mit dem Player Objekt, wie wir es mit dem *PlayerMovement* Skript auch gemacht haben. Im Inspector sehen wir auch, dass das Skript, welches wir eben hinzugefügt haben, mehr Einstellungen hat als unser *PlayerMovement* Skript. Zu diesen Einstellungen kommen wir aber später. Nun können wir wieder zurück in das *PlayerMovement* Skript gehen. Um jetzt von dort auf die Funktion unseres anderen Skriptes zugreifen zu können, müssen wir uns zuerst einmal mit Variablen vertraut machen. Variablen kann man sich als Box vorstellen in die man Text, Zahlen oder ein anderes Objekt rein machen kann. Wir wollen jetzt eine solche Variable für unser anderes Skript anlegen. Um das zu machen, schreiben wir das folgende über unsere *Start()* Funktion:

```
public CharacterController2D characterController2D;
```

Public sagt uns, dass es von außerhalb des Skriptes benutzt werden kann. Da das Skript, welches wir hinein speichern wollen, *CharacterController2D* heißt, geben wir das der Variable als Typ mit und zum Schluss sagen wir noch unter welchen Namen wir die Variable benutzen wollen. Das wäre *characterController2D*. Um das eigentliche Skript jetzt mit der Variable zu verbinden, müssen wir zurück in das Unity Fenster gehen. Hier ist im **Inspector** unter dem *PlayerMovement* Skript ein neues Feld mit dem Namen *Character Controller* hinzugekommen. Dort wird nun aus dem Inspector das andere Skript hineingezogen. Jetzt ist es möglich vom von dort aus das *Character Controller 2D* Skript zu referenzieren. Im Skript *Player Movement* schreiben wir in der *Update()* Funktion das folgende:

```
characterController2D.Move();
```

Move() ist eine Funktion, die im anderen Skript definiert wurde. Wenn man nun über *Move()* hovert sieht man, dass man ihr eine Variable vom Typ Float mitgeben muss. Das trifft sich ganz gut, da wir von der vorherigen Funktion wissen, dass sie uns eine Zahl zurückgibt. Mit dem folgendem Code können wir den Wert, den wir dort zurückbekommen in eine Variable speichern und diese benutzen um es in die *Move()* Funktion zu schreiben.



```
Float horizontal = Input.GetAxisRaw(„Horizontal“);  
charackterController2D.Move(horizontal);
```

Nun haben wir aber noch ein Problem. Die *Update()* Funktion hat die Eigenschaft, dass sie einmal pro Frame ausgeführt wird. Bei verschiedenen starken Computern kann es sein, dass sie unterschiedlich hohe Frames pro Sekunde haben. Wenn die *Move()* Funktion den Charakter nun pro Ausführung um einen Meter bewegt, würde das bedeuten, dass der Spieler auf einem schnelleren PC z.B. 60 Meter in einer Sekunde zurücklegt und auf einem langsameren Computer kommt er nur 30 Meter weit, da nur die Hälfte der Frames laufen. Das ist natürlich nicht gut, wenn sich so der Spieler einen Vorteil verschaffen kann. Um dieses Problem zu lösen, können wir eine Funktion mit dem Namen *FixedUpdate()* benutzen. Um das zu tun, fügen wir das folgende hinzu:

```
Private void FixedUpdate()  
{  
  
}
```

Diese Funktion funktioniert ähnlich wie die *Update()* Funktion insofern, dass sie einmal pro Frame ausgeführt wird. Anders als die vorherige Funktion orientiert sich diese an Physics Frames welche immer eine feste Anzahl pro Sekunde haben. Nun verschieben wir die *Move()* Funktion in diese hinein und stoßen auf das nächste Problem. Die *horizontal* Variable scheint nicht zu existieren. Das liegt daran, dass wir sie einer anderen Funktion erstellt haben. Diese Variable kann man nur in der Funktion benutzen, in der sie erstellt wurde. Um das zu lösen, ziehen wir die Variable aus der Funktion in die Klasse. Dafür fügen wir das folgende unter der *characterController2D* variable hinzu:

```
Float horizontal;
```

Und löschen den Typ Float vor der Variable, in der Update Funktion, damit er nicht versucht eine neue Variable zu erstellen, sondern die nimmt die in der Klasse bereits vorhanden ist und dort den Wert hineinschreibt. Nun hat auch *FixedUpdate()* Zugriff auf die Variable.

Wenn wir jetzt zurück zum auf das Unity Fenster gehen und auf den Play Knopf drücken können wir sehen, dass wir den Frosch mit den Pfeiltasten bewegen können. Allerdings fällt auf die Seite, um sobald er sich anfängt zu bewegen. Wir sehen, wenn wir genauer hinschauen, dass der Frosch einen kapselförmigen *Collider* hat. Das bedeutet, dass er sich wie eine Kapselform verhält. Deshalb fällt er auch bei der kleinsten Bewegung um, da eine Kapsel unten abgerundet ist. Um das zu verhindern, können wir im **Inspector** für den Player in der *RigidBody 2D*



Komponenten einen Hacken bei *Freeze Rotation* setzen. Damit verhindern wir die Rotation des Spielers und somit, dass er umfallen kann.

Jetzt wollen wir noch die Geschwindigkeit des Spielers einstellen können. Um das zu tun, gehen wir wieder zurück in *PlayerMovement* Skript. Hier fügen wir eine neue Variable in der Klasse hinzu:

```
Public float speed = 1f;
```

Jedes mal, wenn man eine Float Variable erstellen will, muss man ein f hinzufügen um dem Skript klarzumachen das es sich um eine Float handelt. Diesen Wert multiplizieren wir jetzt mit der Wert in der *Move()* Funktion, um die Geschwindigkeit zu beeinflussen.

```
charackterController2D.Move(horizontal * speed);
```

Nun können wir im Unity Fenster, im **Inspector** die Geschwindigkeit des Spielers ändern indem die *Speed* Variable anpassen.



Aufgabe

Befolge die oben stehenden Schritt für Schritt Anleitung, um ein Skript zu erstellen mit dem der Frosch in der Lage ist, sich nach links und rechts zu bewegen.