



Einstieg

Warnung: Dieses Kapitel enthält wichtige Schritte in Text-Form welche nicht im Video aufgeführt sind.

In diesem Kapitel geben wir dem Spieler die Option zu springen. Hierfür gehen wir in das *PlayerMovement* Skript. Wir brauchen das Folgende in der *Update()* Funktion:

```
if(Input.GetButtonDown(„Jump“))  
{  
  
}
```

Dies ist eine if-Abfrage. Wenn die Aussage in der runden Klammer *wahr* ist, wird der Teil in den geschweiften Klammern ausgeführt. Wenn es *falsch* ist, wird der Teil übersprungen. Jetzt erstellen wir eine weitere Variable in der Klasse. Diese ist vom Typ Bool und die nennen wir *jump*. Der Datentyp Bool speichert, ob etwas *True* (Wahr) oder *False* (Falsch) ist.

Bool jump;

Diese Variable setzen wir nun in der If-Abfrage auf *true*.

```
if(Input.GetButtonDown(„Jump“))  
{  
    jump = true;  
}
```

Nun weiß das Skript, dass die Sprung-Taste gedrückt wurde. Das müssen wir jetzt auch der *Move()* Methode mitgeben, indem wir es mit in die Klammern schreiben.

*charackterController2D.Move(horizontal * speed, **jump**);*

Jetzt müssen wir die *jump* Variable aber noch zurücksetzt, da der Spieler sonst nur noch Springen würde. Um das zu tun, setzen wir die Variable wieder auf Falsch, nachdem wir sie für die *Move()* Funktion benutzt haben.

*charackterController2D.Move(horizontal * speed, jump);*
jump = false;

Wenn wir jetzt zurück zum Spiel gehen sehen wir, dass der Spieler nun in der Lage ist zu Springen. Allerdings sinken wir jedes Mal ein wenig in den Boden ein, wenn wir wieder aufkommen. Um das zu reparieren, passen wir die Kollisionen ein wenig an. Dazu gehen wir im **Inspector**, während der Spieler angewählt ist, auf



die *Rigidbody 2D* Komponente. Dort wählen wir unter dem Punkt *Collision Detection* statt *Discrete*, *Continuous* aus. Dies wird verhindern, dass der Spieler, jedes Mal wenn er springt, in den Boden einsinkt.

Jetzt ist der Sprung aber noch ein wenig langsam. Um das zu ändern, können wir die Werte ein wenig anpassen. Zuerst einmal können wir die *Gravity Scale in der Rigidbody 2D Komponenten*, über den **Inspector**, auf 3 erhöhen. Wir sehen, dass der Spieler jetzt schneller fällt, aber auch nicht mehr so hoch springen kann, weil wir die Schwerkraft erhöht haben. Um ihn wieder höher springen zu lassen, können wir die *Jump Force* erhöhen. Diese finden wir in der *Character Controller 2D* Komponenten weiter unten im **Inspector**. Wenn wir den Wert hier auf 750 ändern, fühlt sich der Sprung gleich viel besser an. Mit diesen Werten kann aber gerne herumprobiert werden, bis man selber zufrieden ist.

Vorsicht: Alle Werte, die man ändert solange das Spiel läuft, werden zurückgesetzt sobald man das Spiel anhält. Diese Werte müssen dann nochmal eingetragen werden.

Gerade gibt es noch kein Limit an Sprüngen, die man machen kann. Das heißt, dass man so oft Springen kann wie man will, auch wenn man bereits in der Luft ist. Das wollen wir natürlich nicht, da der Spieler so ja einfach durch das Level fliegen kann. Um das zu beheben, implementieren wir jetzt noch einen *Ground Check* der überprüft, ob der Spieler gerade den Boden berührt. Hierfür gibt es bereits zwei Felder in der *Character Controller 2D* Komponenten mit den Namen *What is Ground* und *Ground Check*.

Zuerst müssen wir Unity klarmachen was alles der Boden sein soll. Hierfür wählen wir in der **Hierarchy** mal das Objekt *Ground* an. Rechts oben im **Inspector** gibt es ein Dropdown Menü mit dem Namen Layer. In diesem Menü klicken wir *Add Layer ...* und fügen *Ground* hinzu. Nachdem wir das getan haben können wir den Layer nun jetzt für alle Elemente, die als Boden agieren sollen auswählen. Jetzt müssen wir der *Character Controller 2D* Komponenten sagen wie die Layer heißt welche den Boden entspricht. Dafür wählen wir für *What Is Ground*, *Ground* aus. Für den nächsten Schritt müssen wir das Player Objekt in der Hierarchy öffnen und dort ein neues Objekt erstellen. Um das zu tun müssen wir einen Rechtsklick tätigen und dort *Create Empty* drücken. Den nennen wir jetzt zu *GroundCheck* um. Nun müssen wir im **Inspector** einen *Circle Collider 2D* hinzufügen, indem wir auf *Add Component* drücken und dort nach ihm suchen. Diesen Collider müssen wir nun so anpassen, dass er zu den Füßen des Frosches passt. Dafür können wir, in der Komponenten, den Radius ändern und dann in der **Scene View** die Position anpassen. Zuletzt müssen wir den Collider noch zu einem Trigger machen, indem wir die Option *Is Trigger* im **Inspector** aktivieren. Nun wird er nicht mehr direkt mit



anderen Objekten in der Spielwelt kollidieren, sondern nur ein Signal schicken, wenn er sich mit einem anderen Collider überschneidet. Jetzt müssen wir dieses neu erstellte Objekt von der **Hierarchy** in den Slot *Ground Check* der *Character Controller 2D* Komponenten, auf dem Player, ziehen.

Wenn wir jetzt zurück in das Unity Fenster gehen können wir sehen, dass wir nur noch springen können, wenn wir den Boden berühren.

Es kommt aber ein weiteres Problem auf. Wenn wir an die Seite einer Plattform springen und in sie hinein laufen, sind wir in der Lage an ihr dranzuhängen. Um das zu verhindern, hat Unity eine hilfreiche Komponente. Um diese hinzuzufügen, gehen wir zum *Plattform* Objekt und fügen dort die *Plattform Effector 2D* Komponente hinzu. Jetzt müssen wir dem *Box Collider 2D*, auf dem selbigen Objekt, noch sagen, dass ein Effector hinzugefügt wurde, indem wir dort *Used By Effector* aktivieren.

Nun ist es nicht mehr möglich, sich an der Seite der Plattform festzuhalten. Eine andere Funktion, die der Effector mit sich bringt, ist die Möglichkeit sie von unten durchlässig zu machen. Das bedeutet, dass man die Möglichkeit hat von unter der Plattform auf diese draufzuspringen indem man hindurchspringt. Das kann man in der *Plattform Effector 2D* Komponenten einstellen.



Aufgabe

In diesem Kapitel fügen wir die Funktion hinzu den Frosch springen zu lassen. Dafür muss anhand des Tutorials das *PlayerMovement* Skript und die Plattform angepasst werden.