Einstieg

Jetzt fehlt uns nur noch eine Animation, und zwar die Sprung-Animation. Wenn wir uns die Frames (*Assets -> Art -> Animations -> Frog -> jumping*) unserer Sprung-Animation anschauen, sehen wir, dass diese einen festen Ablauf haben. Der Frosch springt hoch, bleibt ein wenig in der Luft und kommt dann runter. Hier stoßen wir allerdings auf ein Problem. Es kann nämlich sein, dass der Sprung unterschiedlich lang sein kann. Wenn der Frosch irgendwo runterspringt fällt er ziemlich lange und wenn er von unten hoch auf eine Plattform springt, fällt er fast überhaupt nicht. Dies müssen wir jetzt in die Sprung-Animation einbauen.
Um das zu erreichen, teilen wir die Animation in zwei Teile. Absprung inklusive Mittelteil und Landung, damit wir diese separat voneinander abspielen können.

Wir erstellen die Absprung-Animation in dem wir das Gleiche tun wie bei der Idle und Lauf Animation. Wir gehen in das **Animation** Fenster (nicht vergessen das *PlayerSprite* Objekt in der **Hierarchy** auszuwählen) und erstellen dort einen neuen Clip, mit dem Namen *jump*, im *AnimationClips* Ordner. Für den Absprung ziehen wir nun alle Sprung Frames von 0 bis 16 in den Clip. Dann dürfen wir nicht vergessen die Geschwindigkeit anzupassen, indem wir die Länge des Clips beeinflussen. Ungefähr 17 Frames lang sollte gut aussehen. Wir müssen jetzt aber noch eine andere Sache ändern. Hierfür gehen wir zur *jump* Animation im **Project** Fenster (*Assets -> Art -> AnimationClips -> jump*). Wenn wir sie ausgewählt haben, sehen wir im **Inspector** die Option *Loop Time* aktiviert ist. Das bedeutet, dass sich die Animation, sobald wir sie das erste Mal ausgelöst haben, immer wiederholen wird. Im Falle des Laufens und dem Idling war das gut. Hier wollen wir dieses Verhalten allerdings nicht. Der Absprung soll nur einmal abgespielt werden, wenn wir Springen. Also nehmen wir hier das Häkchen raus.

Um die neue Animation einzubinden, gehen wir zum **Animator** und dort sehen wir schon, dass da die *jump* Animation erschienen ist. Hier benötigen wir neue Transitions von *walk* zu *jump* und von *idle* zu *jump*. Und zwar jeweils in beide Richtungen. Wenn wir das getan haben, fügen wir auch einen neuen Parameter hinzu. Diesen nennen wir *isGrounded*. Bei allen neuen Transitions muss die Exit Time ausgeschaltet und die Transmission Duration auf 0 gesetzt werden. Die Transitions brauchen außerdem die folgenden Conditionen:

<u>idle -> jump:</u>
isGrounded = false

jump -> idle:
isGrounded = true
isWalking = false

walk -> jump:
isGrounded = false

jump -> walk:
isGrounded = true
isWalking = true

Jetzt müssen wir wieder zurück in das *PlayerMovement* Skript und von dort den *isGrounded* Parameter beeinflussen, so wie wir das schon mit dem *isWalking* Parameter getan haben. Das *characterController2D* Skript hat eine hilfreiche Variable, die *grounded* heißt. Diese können wir vom *PlayerMovement* Skript aus auslesen. Um den Wert auszulesen, müssen wir das Folgende schreiben:

characterController2D.grounded

Diese Zeile setzen wir in der *Update()* Funktion direkt unter die Zeile, in der wir den *isWalking* Parameter setzen. Außerdem können es zuerst einmal in der Konsole ausgeben lassen, indem wir die Log-Funktion benutzen:

Debug.Log(characterController2D.grounded);

Wenn jetzt das Spiel läuft sehen wir, dass die Konsole *True* ausgibt, wenn wir uns gerade auf dem Boden befinden und *False* zeigt, sobald wir springen. Um diesen Wert nun in den Parameter zu schreiben, statt in der Konsole auszugeben, ersetzen wir die Zeile mit dem Folgenden:

Animator.SetBool(„isGrounded“, characterController2D.grounded);

Jetzt Springt der Frosch auch schon jedes Mal wenn wir springen. Und nun müssen wir nur noch die Lande Animation hinzufügen. Hierfür erstellen wir eine weitere Animation mit dem Namen *landing*. Hier ziehen wir jetzt die Sprung-Frames von 17 bis 21 rein und passen sie so an, sodass die Geschwindigkeit stimmt. Auch hier müssen wir die *Loop Time* aus machen, damit die Animation nicht wiederholt wird.

Nun müssen wir diese Animation nur richtig einbauen. Hierfür müssen wir nur noch Änderungen am Animator vornehmen. Wir wollen das die lande Animation jedes Mal nach der *jump* Animation ausgeführt wird. Um das zu erzielen löschen wir die Transition *jump -> walk* und *jump -> idle*. Jetzt machen wir eine neue Transition von jump zu landing. Hier deaktivieren wir wieder die *Exit Time* und setzen die *Transition Duration* auf 0. Die Kondition für diese Transition ist

isGrounded = *True*. Von landing machen wir zwei Transitions einmal zu walk und einmal zu idle. Beide haben diesmal aber eine *Exit Time*. Nämlich eine *Exit Time* von 1. Das bedeutet, dass die *landing* Animation einmal komplett abgespielt wird bevor es weiter zu nächsten Animation geht. Die *Transition Duration* ist wieder 0. *Landing -> idle* hat die Kondition *isWalking* = *False* und *landing -> walk* hat die Kondition *isWalking* = *True*.

Nun sehen wir im Spiel, wie die Landung flüssiger ist als zuvor. Der Übergang verläuft gerade aber ein wenig zu langsam. Wir haben zuvor gelernt, wie wir die Geschwindigkeit im Animationsclip ändern, es gibt aber auch einen Weg, um sie zu beeinflussen. Hierfür gehen wir zum Animator und klicken dort auf die *landing* Animation. Im **Inspector** finden wir dann einen Speed Wert. Je höher dieser ist, desto schneller läuft die Animation ab. Wir setzen ihn mal auf zwei und sehen sofort, wie die Landung natürlicher aussieht als vorher.

Aufgabe

Erstelle jetzt eine Sprung-Animation inklusive Clip und Implementation in den *Animator Controller.* Der Frosch soll am Ende die Sprung Animation auch wirklich auch abspielen wenn die Sprung-Taste gedrückt wurde.