„EXPONENTIELLE FUNKTIONEN FÜR NICHT-LINEARE VERBINDUNGEN“

SOFTWARE: CROSS-PLATTFORM / SOFTIMAGE XSI
RIGGING TUTORIAL
LEVEL: EXPERTE

Written by Christoph Schinko, 2007

1. EINLEITUNG

Parameter mittels Expressions zu verbinden ist simpel und zugleich höchst effektiv: keine zusätzlichen Objekte, f-curves oder Constraints. Als pure Mathematik hinter den Kulissen sind Expressions äußerst robust und eröffnen dem 3D Artist ungeahnte Möglichkeiten.

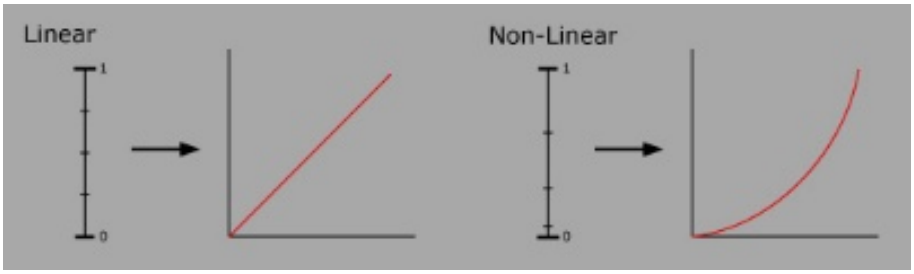

Bild 1: Lineare vs. nicht-lineare Verbindung

Oft kommt einem jedoch die lineare Natur jener Verbindung in die Quere. Besonders beim Rigging-Prozeß, wo man mit Bone-Rotationen und Shape-Interpolationen zu tun hat, wünscht man sich oft eine organischere Verbindung – einen nicht-linearen Link. Etwas daß, als Graph dargestellt, eher wie eine exponentielle oder quadratische Funktion ausschaut, anstatt einer geraden Linie [Bild 1].

Dann würden Bones nicht mehr so plötzlich anfangen zu rotieren, und Shapes könnten endlich harmonisch mit Bone-Rotationen zusammen arbeiten. Leider kann der dafür vorgesehenen XSI Funktion „Link deform with orientation“ in komplexeren Rigs nicht vertraut werden – der generierte Link ist nämlich nachträglich nicht mehr justierbar.

2. EINE LINEARE VERBINDUNG

Unser Beispiel zum Tag: das Hosenbein eines Charakters. Wenn der Fuß auf 25° angehoben wird, soll das Hosenbein anfangen sich an den Schuh schmiegen. Da dies auf Grund der Topologie der Hose nicht per point weighting erreicht werden konnte, wurde ein Korrektur-Shape modelliert und soll nun mit der Rotation des Fusses verbunden werden. [Bild 2]

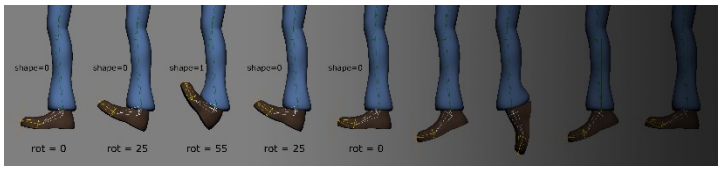

Bild 2: Das Bein des Charakters & die verschiedenen Stadien des Korrektur-Shapes

Das Korrektur-Shape soll sich erst ab einer Fuß-Rotation von 25° bemerkbar machen und sich graduell bis zu seinem Maximum bei 55° steigern.

Der herkömmliche Weg dazu ist denkbar einfach, man legt eine Expression auf den weighting Parameter des Korrektur-Shapes im Animation Mixer und regelt diesen über erwähnte Fuß-Rotation. Da wir es hier mit einer Rotations-Bandbreite von 25°-55° zu tun haben, müssen wir diese erst auf die für Shape Animation verwertbare Bandbreite (0-1) bringen. Dazu subtrahiert man 25 von der Rotation, dividiert das Ergebniss durch 30.

shape weight = (rot – 25) /30

Da die Hose sich erst ab 25° deformieren soll, müssen wir eine derartige Begrenzung in die Expression einfügen.

cond(rot < 25, 0, “OurExpression”)

Oder für XSI verständlicher:

cond(footBone.kine.local.rotz < 25, 0, (footBone.kine.local.rotz – 25) /30)

Die eben erarbeitete Lösung funktioniert zwar, es ist aber wie befürchtet eine direkt proportionale Verbindung für diesen Fall nicht optimal. Bei einer

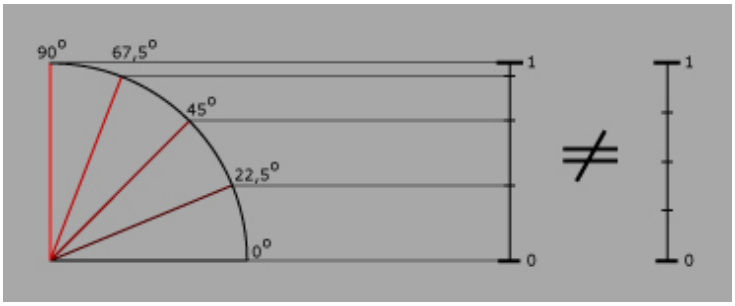

Bild 3: Warum Rotationen und lineare Verbindungen nicht zusammen passen

Rotation von ca. 35° beginnt die Hosen-Geometrie den Schuh zu durchdringen und die Bewegung schaut generell unnatürlich aus. [Bild 3]

Was wir also brauchen ist eine Möglichkeit den eben erstellten Link zu verändern, flexibel zu machen. Man müsste ihn biegen können, sodaß man erstens eine organische Bewegungskurve erhält, und zweitens die Geometrie den Schuh nicht einfach durchdringt.
In diesem Falle: wenn die Fuß-Rotation 25° erreicht, muß die Hose langsam anfangen das Korrektur-Shape einzublenden, und dann immer schneller auf das Maximum bei 55° zusteuern.

Hinweis: Natürlich kann man den eigentlichen Rotationswert des Fuß-Bones als Multiplikator in der Expression verwenden und so nicht-Linearität der Funktion erreichen (Ergebnis * Fußrotation und so fort), das ist jedoch eine ewige Fitzelei und kann später kaum noch editiert werden. Weiters nimmt es, wenn man es für einen Charakter mehrere Male machen muss, und man es jedesmal mit neuen Rotationsbereichen und -

richtungen zu tun hat, sehr viel Zeit in Anspruch.

Es wäre daher viel besser, eine von der Fußrotation unabhängige Expression zu verwenden, und diese mit einer nicht-linearen mathematischen Funktion zu erweitern, die uns die gewünschte Flexibilität verleiht.

Wenn das möglich ist, haben wir eine verstellbare Kurve (nicht als f-curve editierbar), die komplett ohne keys oder zeit-abhängigen Parametern auskommt – nur eine jener simplen und effektiven Expressions, die hinter den Kulissen ihren Dienst tun wird, egal was wir dem Charakter oder seiner Rig antun.

3. EINE NICHT-LINEARE VERBINDUNG

Okay! Lasst uns zuerst einmal einen Blick auf exponentielle Graphen werfen, die scheinen unseren Wunschvorstellungen ziemlich genau zu entsprechen: am Anfang nur ein leichtes Ansteigen, daß gegen Ende immer schneller wird. [Bild 4]

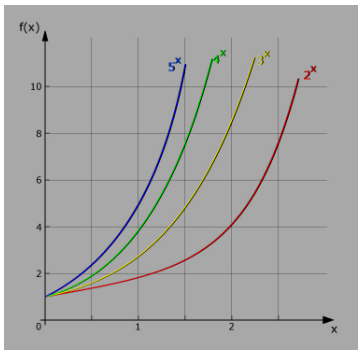

Bild 4: Exponentielle Funktionen

Wählen wir zunächst einfach mal eine der oben gezeigten Funktionen, so würde zB. 3^x sicher ein schönes Ergebnis für unser Beispiel liefern. Die Kurve ist bei x=2 bereits sehr steil, sodaß wir nur den Bereich x=0 bis x=2 in unserer Expression verwenden würden. Der Rest wird einfach abgeschnitten, bzw nicht beachtet, indem wir den Wertebereich wie vorher neu definieren und von 25-55 auf 0–2 bringen.

Beispiel:
Bone Rotation:
25° – 55°
Wir brauchen:
0 – 2

Ähnlich wie vorher:
x = (rot – 25) / 15

Daraus ergibt sich folgende Funktion:

$f(x) = 3^x = 3^{(rot - 25) / 15}$

Was uns dabei sofort auffällt, ist, daß der Graph bei einem f(x)-Wert von 1 anfängt, also nicht durch den Nullpunkt geht, was auch bei allen anderen exponentiellen Funktionen, und natürlich jeder Zahl mit 0 als Exponent der Fall ist.

Wir ziehen also einfach 1 vom Ergebnis ab, und erhalten folgende Funktion: [Bild 5]

$f(x) = 3^x - 1 = 3^{(rot - 25) / 15} - 1$

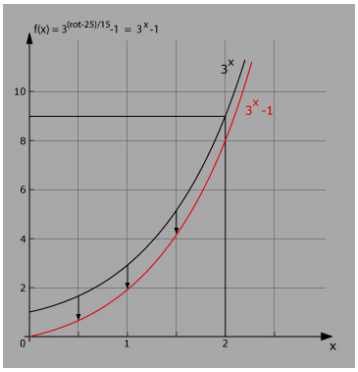

Bild 5: 3^x Funktion durch 0,0

Die Kurve fängt jetzt bei 0 an (wie unser Shape), und endet bei 8, was wir durch eine weitere Division leicht in den Griff bekommen, und somit unser angestrebtes Spektrum von 0-1 erreichen; optimal zum Einsatz von Shape-Blending und unzähligen anderen Rigging-Aufgaben!

So weit, so gut, aber was wenn wir schließlich doch eine exponentielle Funktion mit einer **anderen Basis** verwenden wollen? Vielleicht 5^x? Wie kann man generell alle exponentiellen Funktionen normalisieren und sie in unsere gewählte Bandbreite von x=0 bis x=2 bändigen?

Jede Funktion müsste durch verschiedene Faktoren dividiert warden um eine 0-1 Bandbreite zu gewährleisten, siehe folgende Beispiele [Bild 6]:

Aus dem obenstehenden Bild läßt sich erkennen, daß jede exponentielle Funktion, durch das Quadrat ihrer Basis dividiert, am Ende der gewählten Bandbreite (x=2) immer ein Wert von 1 ergibt.

→ Wir können jetzt die Funktionsbasis beliebig wählen, und erhalten somit: a^x

Fügen wir das in unsere vorherige Funktion ($3^x - 1$) ein, so ergibt sich:

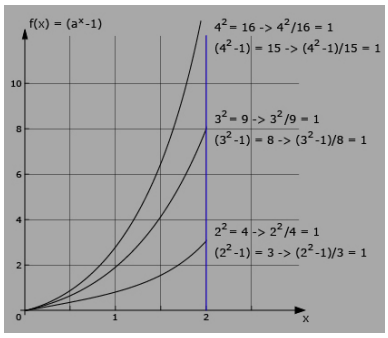

Bild 6: Normalisierung aller Funktionen durch Division

$f(x) = (a^x - 1) / (a^2-1)$

Als Ergebnis erhalten wir eine exponentielle Funktion, deren Basis wir frei wählen können, und deren Ergebnis immer durch die selben Start- und Endpunkte geht (0,0 und 2,1) - an diesen also wie festgenagelt ist - während wir die dazwischenliegende Kurve beliebig verbiegen können und somit eine breite Fläche von Kurven zur Verfügung haben. [Bild 7]

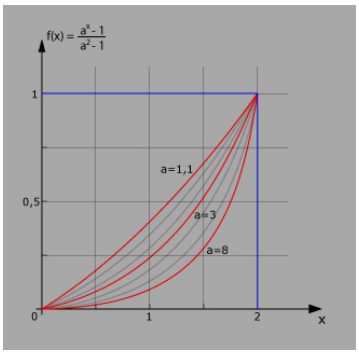

Bild 7: Alle Funktionen passieren die selben Start- und Endpunkte

Nun würde sich das gelinkte Shape zunächst nur langsam bemerkbar machen, und schließlich immer stärker sichtbar werden, bis zum definierten Maximalwert. Die Basis „a" wird über ein Null-Objekt oder custom parameter gesteuert, oder kann einfach hard-gecoded werden, je nach Wunsch.

Die Expression für unser Shape Weight:

*pow(BaseNull, (footBone – 25) / 15) - 1 / (BaseNull * BaseNull– 1)*

or

*pow(BaseNull.kine.local.posy, (footBone.kine.local.rotz – 25) / 15) - 1 / (BaseNull.kine.local.posy * BaseNull.kine.local.posy – 1))*

Schaut so aus als hätten wir oben gestelltes Problem gelöst – wenn wir aber noch ein bißchen weiter-

#probieren, werden wir erkennen, daß wir mit nur geringen Erweiterungen ein viel umfaßenderes Tool erstellen können.

4. ERWEITERTES NICHT-LINEARES VERHALTEN

Invertieren wir unsere Kurve für einen langsamen **Abstieg**, so nehmen wir die negativen Ergebnisse der vorigen Funktion (nach unten spiegeln) und verschieben sie mit +1 wieder nach oben. [Bild 8]

*f(x) = **(-1)** * previousExpression **+ 1***

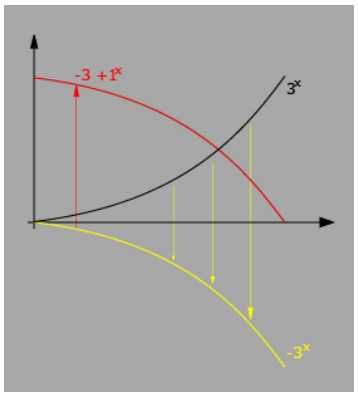

Bild 8: Ein neuer Kurven-Typ

Durch Einführung eines neuen Parameters (switch) könnten wir zwischen den zwei Kurven-Typen hin und her schalten. Ist „switch" auf 1, so bekommen wir die ursprüngliche Kurve, bei -1 erhalten wir die nach unten fallende Schwestern-Kurve.

Die Expression müsste so angelegt werden:

*switch * (previousExpression) – (switch – 1) / 2*

oder, etwas verwirrender:

*switch.kine.local.posy * (pow(BaseNull.kine.local.posy, (footBone.kine.local.rotz – 25) / 15) - 1 / (BaseNull.kine.local.posy * BaseNull.kine.local.posy – 1)) – (switch.kine.local.posy – 1) /2*

In dieser Setup definiert der "switch" Parameter den Typus der Kurve, während der Basis-Parameter den Grad der Krümmung einstellt. Jener Basis-Parameter muß allerdings >1 sein (sonst ergibt sich eine Konstante).

Zum Schluß könnten wir noch zwei weitere Kurven-Typen in Betracht ziehen – Kurven mit **anfänglich steilem** Anstieg/Abfall, und folgender **langsamer** Annäherung an das Endergebnis. [Bild 9]

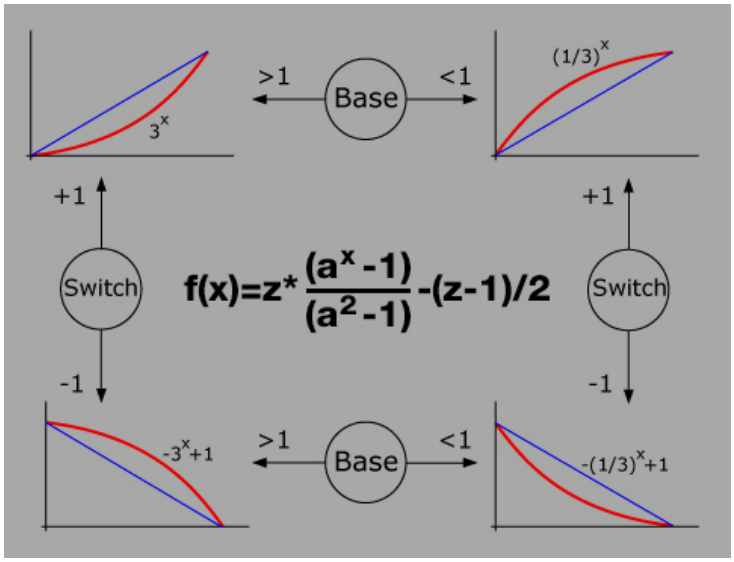

Bild 10: Die finale Formel und alle vier möglichen Kurven-Typen

Dazu müssten wir eigentlich nur 1 durch unsere Funktion dividieren, also anstatt 3^x einfach $1/3^x$ oder noch besser: 0.33^x!!

Das Schöne dabei ist, daß wir an unserer erarbeiteten Funktion nichts ändern oder hinzufügen müssen – der Basis Parameter muß lediglich >0 und <1 sein, anstatt 2 oder 3!

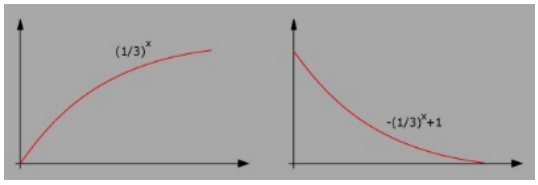

Bild 9: Zwei weitere Kurven-Arten

Wir können also mit den bereits vorhandenen zwei Parametern die Expression dazu benutzen um alle 4 Kurventypen zu generieren, und jeweils deren Krümmung, fast bis hin zur Linearität. [Bild 10]

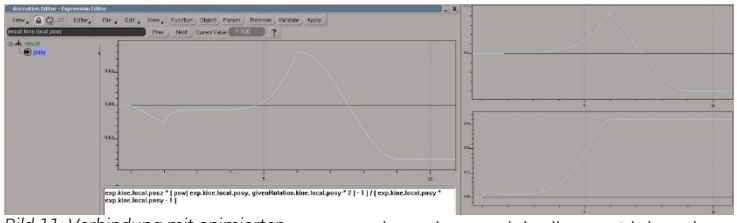

Bild 11: Verbindung mit animierten Parametern

5. ABSCHLIESSENDES

Die zwei Parameter können entweder in der Szene gelassen werden, um die Funktionen auch weiterhin beliebig fein-tunen zu können, oder sie können schließlich auch hart-gecoded werden, je nach Situation und was man für besser erachtet.

Obwohl es natürlich auch andere Wege gibt, direkte Verlinkungen organischer zu gestalten, so sind die meisten Lösungen entweder kompliziert einzustellen, oder sie machen die Rig instabil und fehleranfällig, besonders wenn man auf mehreren Rotationsachsen arbeitet.

Die hier vorgestellte Lösung ist einfach und sauber, und kann vor allem grafisch & direkt im Viewport(!) eingestellt werden.

Hinweis: Das Ergebnis der Expression kann wie immer mit *View>ShowGraph* im Expression Editor als Graph sichtbar gemacht werden, und, noch ein Denkanstoß: unsere Parameter könnten sogar animiert werden, um noch komplexere nicht-lineare Links mit unserer Expression zu erreichen! [Bild 11]

Ich hoffe dieses Tutorial war hilfreich und hat dazu inspiriert, ein bißchen mit Expressions und exponentiellen Funktionen zu experimentieren. Bei Fragen und Feedback stehe ich immer gerne zur Verfügung - Viel Spaß!

Verfaßt von Christoph Schinko, 2007
office[@]christoph-schinko.com
www.christoph-schinko.com