2015
3
DIGITAL PRODUCTION

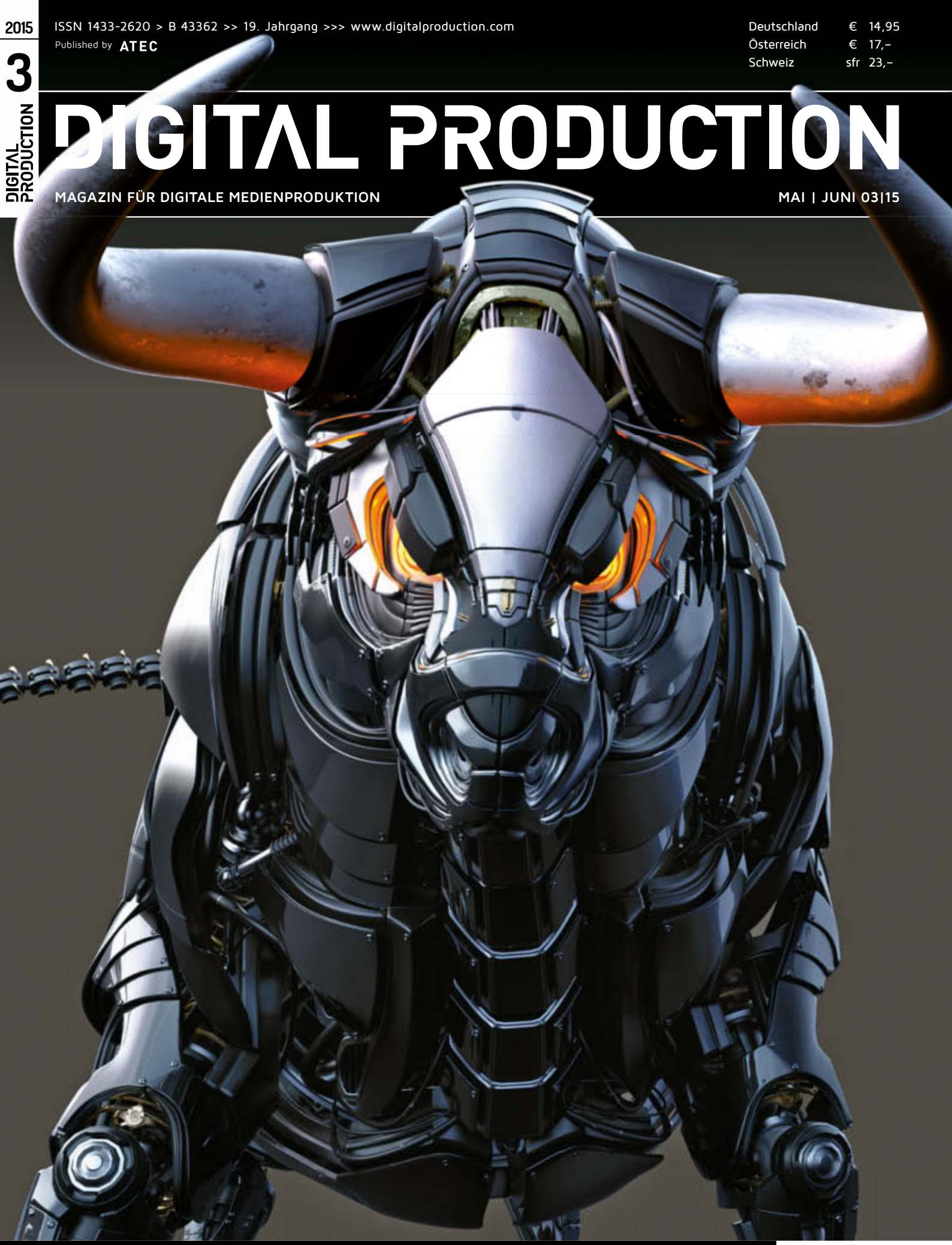
ISSN 1433-2620 > B 43362 >> 19. Jahrgang >>> www.digitalproduction.com
Published by ATEC

Deutschland € 14,95
Österreich € 17,–
Schweiz sfr 23,–

DIGITAL PRODUCTION

MAGAZIN FÜR DIGITALE MEDIENPRODUKTION

MAI | JUNI 03|15



VFX & Social Media

Vom Feed zum Showreel:
Was macht Sinn im Netz?

Software satt!

ZBrush 4R7, Shooter Suite,
Nuke, Power Reducer und mehr

Studioflüstern

FuseFX, Rise, Elefant
Studios und BigHugFX

4 194336 214951 03

Turtle-Modus des MoSpline-Objekts

Abbildung 1: Befehle zum Drehen der Turtle um ihre drei Achsen

Das MoSpline-Objekt ist im MoGraph-Menü von Cinema 4D verborgen und kommt oft bei der Erzeugung von abstrakten Spline-Formen und Textverfremdungen zum Einsatz. Ein spezieller Turtle-Modus kann jedoch auch zur parametrischen Erzeugung von Formen über Skripte und Zeichenfolgen verwendet werden. Die folgenden Passagen aus dem aktuellen Kompendium zu Cinema 4D, das im Schwerpunkt Animationstechniken und -funktionen in Cinema 4D 16 behandelt, geben eine Hilfestellung für die Bedienung dieses Systems. Nachfolgende Abschnitte im Buch vertiefen die Arbeit mit diesem Turtle-Modus und zeigen ausführlich die Animation der generierten Formen oder die Erzeugung von Polygonen.

von Arndt von Koenigsmarck

Dieser etwas lustig anmutende Modusname ist dem Zeichensystem zu verdanken, das nun benutzt werden kann. Dabei lässt sich eine Art Zeichenstift, Turtle genannt, über Kommandos steuern. Dies mag zuerst etwas umständlich wirken und ist sicherlich auch mit Lernaufwand verbunden, eröffnet jedoch ganz neue Welten, wie bereits das Umschalten in diesen Modus eindrucksvoll beweist. Das MoSpline zeichnet uns einen stilisierten Baum, der über Einstellungen auf der Werte-Dialogseite auch noch interaktiv gesteuert werden kann. Dazu gleich noch mehr.

Der Kern des Systems bleibt der Turtle-Dialogbereich, wo wir unter anderem einen Startstring und Regeln für das Zeichnen vorgeben können. Dazu müssen wir aber erst die Kommandos kennen, die uns hier zur Verfügung stehen.

Beginnen wir mit den grundsätzlichen Befehlen zum Verschieben unserer zeichnenden Schildkröte und zum Abbiegen nach links oder rechts. Das Kürzel F malt uns eine gerade Linie in Blickrichtung der Schildkröte. Diese blickt anfangs immer in Z-Richtung des MoSpline-Objekts. Wie weit wir mit diesem F-Befehl kommen, können Sie über den Wert für die Standardlänge in der Werte-Kategorie einstellen. Dort finden Sie auch eine Standarddicke, die zum Zeichnen verwendet wird. Diese Dicke kommt erst dann zum Einsatz, wenn wir den MoSpline mit einem Sweep-Objekt verwenden, um Geometrie zu erzeugen oder aber wenn Sie die Anzeigemodi „Doppellinie" oder „Volle Form" in den Objekt-Einstellungen des MoSplines verwenden. Alternativ hierzu können auch konkrete Werte mit dem Befehl verbunden werden. Das sähe dann so aus: F(a,b,c), wobei a für Länge der Linie, b für die Dicke dieser Linie und c für die gewünschte Anzahl an Unterteilungen auf dieser Linie stehen.

Zum Abbiegen oder Drehen der Schildkröte können Sie die Zeichen + oder - verwenden. Auch diese können in der Schreibweise +(Winkel) oder -(Winkel) direkt mit Gradwinkeln benutzt werden. Das Pluszeichen dreht die Turtle im Uhrzeigersinn, das Minuszeichen gegen den Uhrzeigersinn. Verwenden Sie + und - ohne eine Winkelangabe dahinter, so wird der Standardwinkel aus den Werte-Einstellungen herangezogen. Neben den Drehungen um die y-Achse der Turtle können Sie mit den Kürzeln & und ^ um die x-Achse, und mit den Kürzeln \ oder / um die z-Asche drehen lassen. Auch hier lassen sich auf Wunsch konkrete Winkel direkt in einer Klammer dahinter angeben. Die Abbildung 1 fasst illustrativ noch einmal die verschiedenen Kürzel zum Drehen der Turtle zusammen.

Für noch mehr Kontrolle können Sie auch eigene Benutzerdaten definieren und deren Namen direkt als Variablen verwenden. So könnten Sie zum Beispiel den Begriff H_Winkel anlegen und dann schreiben F+(H_Winkel)F. Die Turtle fährt eine Standardlänge in z-Richtung, biegt um ihren H-Winkel nach rechts ab und fährt wieder eine Standardlänge nach vorne. Den Umgang mit Benutzerdaten können Sie im Kompendium ausführlich im XPresso-Kapitel nachlesen. Ebenfalls praktisch ist, dass auch die Parameter aus dem Werte-Bereich über Kürzel in die Befehle eingebracht werden können. So stehen _move zum Beispiel für die Standardlänge und _rotate für den Standardwinkel. Schauen Sie mal in der nächsten Abbildung 2, was wir allein schon mit diesem Wissen anstellen können.

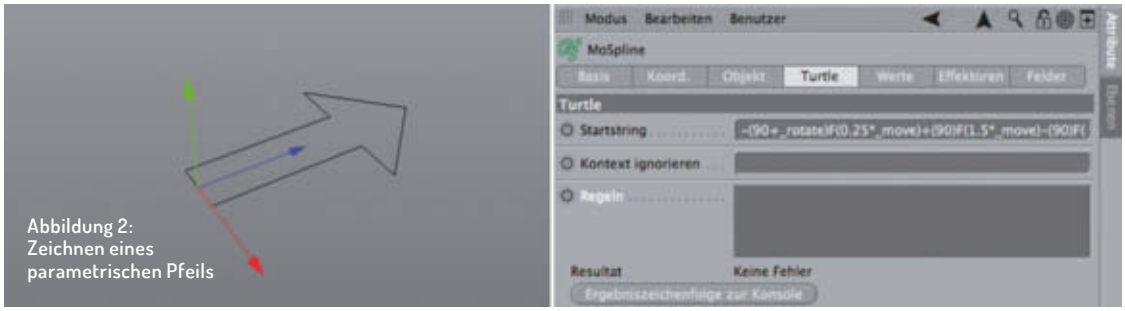


Abbildung 2: Zeichnen eines parametrischen Pfeils

Wie Sie bereits in Abbildung 2 erkennen können, habe ich dort mit den uns bislang bekannten Befehlen einen Pfeil zeichnen lassen. Die Größe und Richtung dieses Pfeils kann über Standardlänge und Standardwinkel in den Werte-Einstellungen gesteuert werden. Diese Befehle habe ich dafür als Startstring verwendet:

```
-(90+_rotate)F(0.25*_move)+(90)F(1.5*_
move)-(90)F((sin(45)-0.25)*_move)+(135)
F+(90)F+(135)F((sin(45)-0.25)*_move)-(90)
F(1.5*_move)+(90)F(0.25*_move)
```

Natürlich wirkt dies auf den ersten Blick kryptisch, aber Schritt für Schritt betrachtet ist es recht simpel. Mit -(90+_rotate) drehen wir uns zuerst um 90 Grad nach rechts, korrigieren dies aber zugleich wieder durch die Auswertung des Standardwinkels. Danach spazieren wir um ein Viertel der Standardlänge in Blickrichtung nach vorn. Nun drehen wir uns um 90 Grad nach links, spazieren um 150 Prozent der Standardlänge nach vorn und so weiter.

Wie Sie sicher bemerkt haben, sind dies eigentlich immer die gleichen Befehle. Stellt sich also die Frage, wie solche Skripte vereinfacht werden können. In solchen Fällen können wir im Bereich „Regeln" Variablen definieren, so wie Sie es bereits zum Beispiel aus C.O.F.F.E.E. gewohnt sind. So könnten wir dies als Regeln definieren:

```
A=F(0.25*_move)
B=F(1.5*_move)
C=F((sin(45)-0.25)*_move)
```

Nun könnten wir für den Startstring schreiben:

```
-(90+_rotate)A+(90)B-(90)C+(135)F+(90)
F+(135)C-(90)B+(90)A
```

Ich hoffe, dieses kleine Beispiel macht Ihnen den Nutzen bereits deutlich. Hier hört der Funktionsumfang aber keineswegs auf, denn es können auch Variablen neu definiert und somit rekursive Skripte erstellt werden. Rekursiv bedeutet, dass die gleiche Abfolge an Befehlen mehrfach durchlaufen werden kann. Auf diese Weise können bereits einfache Befehlsfolgen komplexe Strukturen malen. Zudem müssen wir nicht unbedingt in einem Strich durchzeichnen lassen. Wenn Befehle in eckigen Klammern stehen, wird dafür eine neue Turtle erzeugt. Nach diesen Befehlen wird dort weiter gemalt, wo die Turtle vor den eckigen Klammern aufgehört hatte. Dies eignet sich zum Beispiel hervorragend zum Zeichnen von Zweigen und Ästen.

Lassen Sie uns einmal dies hier probieren:

```
A=FT[+(70-60*_index/_total)B][-(70-60*_
index/_total)B]A
B=F&B
```

Tragen Sie diese Zeilen als Regeln ein und verwenden Sie schlicht den Buchstaben A als Startstring. Wenn Sie nun noch mit den Werten für Wachstum, Standardwinkel und Tropismus arbeiten, können Sie mit wenigen Handgriffen Strukturen erzeugen, die an ein Palmblatt oder den Ast einer Tanne erinnern. Wie ist die möglich? Schauen wir uns die Zeilen genauer an.

Der Startstring sorgt dafür, dass am Anfang unsere Variable A ein Mal ausgeführt wird. Dort verschiebt der Befehl F zuerst die Turtle um eine Standardlänge nach vorn. Der Buchstabe T steht für Tropismus und ist direkt mit dem gleichnamigen Parameter der Werte-Seite verbunden. Damit wird eine Art Erdanziehung simuliert, die beispielsweise einen Ast nach unten zieht. Unsere gezeichnete Linie wird sich also dadurch automatisch nach unten krümmen, wenn Tropismus erhöht wird. Es folgen eckige Klammern. Das, was jeweils in einem Klammerpaar steht, wird durch eine separate Turtle gezeichnet. Die Befehle in der ersten und der zweiten eckigen Klammer sind identisch, nur das Vorzeichen ist negiert. Diese Klammern zeichnen uns die seitlichen Äste, links und rechts am Zweig. Über +(70) wird die Turtle zuerst im Uhrzeigersinn um ihre y-Achse gedreht. Von diesem Winkel ziehen wir jedoch „60*_index/_total" wieder ab. Diese beiden Begriffe haben mit dem Wachstum zu tun, das Sie über Wachstum in den Werte-Einstellungen kontrollieren. Je größer diese Wert ist, desto länger wird unser Ast und desto mehr Zweige sind an seinen Seiten zu finden. Im Prinzip legt dieser Wert nur fest, wie oft der Befehl A ausgeführt werden soll. Der Begriff _index steht für die Nummer des gerade ausgeführten Befehls und _total steht für die Gesamtzahl aller Befehle. Da die Gesamtzahl der Befehle direkt mit dem Wachstum zusammenhängt, erhalten wir über diesen Ausdruck eine Stellschraube, um den seitlichen Winkel der Äste zum Ende des Asts immer kleiner werden zu lassen. Der Ausdruck _index/_total ergibt also für die Zweige am Ende des Asts höhere Werte als für die zuerst erzeugten Äste an der Wurzel des Asts.

Nach dieser Drehung der neu erstellen Turtle folgt der Befehl B, hinter dem F&B als Befehlsfolge steckt. F& kennen wir bereits. Die Turtle marschiert nach vorn und dreht sich anschließend um den Standardwinkel nach unten. Das letzte Zeichen ist wieder der Buchstabe B. Dies ist das Geheimnis der Rekursion, denn über diesen Buchstaben ruft sich praktisch diese Befehlsfolge F& immer wieder neu auf. Wen wir also F&F&F&F& usw. immer weiter wiederholen, ergibt sich dadurch eine Linie, die sich abschnittsweise immer weiter krümmt. Theoretisch würde dieser Befehl somit nie zum Ende kommen. Hier greift jedoch der Wachstum-Wert, denn dieser regelt, wie oft solche Befehle wiederholt werden. Somit ist die erste eckige Klammer fertig definiert und somit ein nach rechts abzweigender Ast definiert. Es folgt eine zweite eckige Klammer mit gleichem Inhalt. Diesmal wird anfangs jedoch der Befehl „-" verwendet, die Turtle also auf die andere Seite des Asts gedreht. Am Schluss folgt auch hier erneut der Buchstabe A. Die gesamte Zeile wird durch die Erhöhung des Wachstum-Werts immer wieder aufgerufen und erzeugt somit ein Zweigpaar nach dem anderen. Schon erstaunlich, was mit so ein paar Befehlen schon möglich ist, oder? Die Abbildung 3 zeigt Ihnen Impressionen unseres aktuellen MoSplines mit dem beschriebenen Skript.

Um einen Überblick darüber zu bekommen, wie die komplette Formel tatsächlich aussieht, können Sie jederzeit die Schaltfläche für „Ergebniszeichenfolge zur Konsole" unter dem Regeln-Feld anklicken. In Abhängigkeit des eingestell-

Abbildung 3: Mögliche Formen des MoSplines mit dem beschriebenen Skript

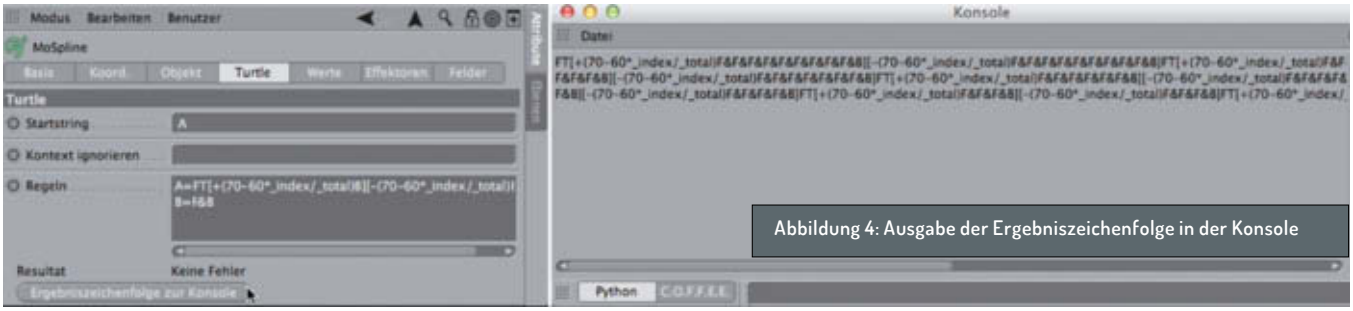


Abbildung 4: Ausgabe der Ergebniszeichenfolge in der Konsole

Abbildung 5: Inkrementales Anpassen von Rotationen und Längen

Abbildung 6: Wirkung des Tropismus

ten Wachstum-Werts können Sie dann in der Konsole eine Befehlskette finden, bei der alle Rekursionen aufgelöst wurden. Dies kann mit hohen Wachstum-Werten schnell ein ganz schöner Wust werden. Die Konsole finden Sie im Skript-Menü von Cinema 4D. Die folgende Abbildung 4 stellt diese Ausgabemöglichkeit dar.

Inkrementale Veränderungen

Momentan haben wir bereits einige Stellschrauben in unseren Regeln untergebracht, um schrittweise Veränderungen an den Strukturen zu zeigen. So liegen die seitlichen Zweige an der Spitze des Asts stärker an als an der Astwurzel. Derartige Veränderungen lassen sich aber auch auf anderem Wege mit dem Wachstum-Wert verknüpfen.

Die einfachsten Optionen sind dabei das Multiplizieren respektive das Dividieren mit jeder Rekursion. Auf diese Weise könnten wir die Strichlänge mit der Wachstumsphase steigern oder Winkel mit dem Wachstum des Zweigs verringern. Wie Sie sicher wissen, könnten wir alle diese Manipulationen allein durch Multiplikationen erzeugen. Eine Division zum Beispiel durch 2 entspricht schließlich einer Multiplikation mit 0.5. Dennoch mag es praktisch sein, für das Multiplizieren und das Dividieren separate Befehle und Kürzel zu haben. Wir schauen uns diese Befehle in einer Auflistung kurz an und werden dann unserer Ast-Generierung damit etwas tunen.

Befehle für die Multiplikation

- "(Wert) - Multipliziert eine Verschiebung mit dem Wert pro Wachstumsgeneration
- ;(Wert) - Multipliziert eine Drehung mit dem Wert pro Wachstumsgeneration
- !(Wert) - Multipliziert eine Skalierung mit dem Wert pro Wachstumsgeneration

Befehle für die Division

- _(Wert) Dividiert eine Verschiebung durch den Wert pro Wachstumsgeneration
- @(Wert) Dividiert eine Drehung durch den Wert pro Wachstumsgeneration
- ?(Wert) Dividiert eine Skalierung durch den Wert pro Wachstumsgeneration

Wenn wir also möchten, dass die zuerst gezeichneten Zweige am Ende nicht so lang werden, fügen wir einfach einen Multiplikator hinzu, der geringer als 1 ist. Das könnte dann so aussehen:

```
B="(0.6)F&B
```

Hiermit reduzieren wir die Verlängerung der Äste auf 60 Prozent pro Wachstumsgeneration. Wenn jetzt noch die Biegung der Zweige nach unten mit der Länge der Äste ansteigen soll, ergänzen wir dies:

```
B="(0.6)F;(1.1))&B
```

Damit wird dann die Drehung am Ende der Zweige jeweils um 10 Prozent pro Wachstum-Wert stärker, was dann sogar zum Einrollen der längeren Zweige führen kann, wenn man es mit dem Standardwinkel übertreibt. Die Abbildung 5 zeigt Ihnen Eindrücke der zuletzt vorgenommenen Veränderungen.

Wie Sie vielleicht schon bemerkt haben, ist das Wachstum keineswegs auf den Wert 10 beschränkt. Sie können über direkte Werteingaben auch weit höhere Vorgaben machen. Entsprechend wird sich unser Ast oder Palmblatt immer weiter ausbreiten. Um nun diese Animation noch etwas ansprechender zu gestalten, verlangsamen wir das Wachstum der seitlichen Zweige. Wir geben dazu einfach eine Bedingung vor, die zuerst erfüllt werden muss, bevor die seitlichen Zweige gezeichnet werden. Das könnte dann so aussehen:

```
B:(_growth>8)="(0.6)F;(1.1))&B
```

Hier hat sich also die Schreibweise für die Variable etwas geändert. Durch den Doppelpunkt muss erst die Bedingung in der runden Klammer erfüllt werden, damit B die Zeichenkette nach dem Gleichheitszeichen erhält. Hier wurde festgelegt, dass zuerst ein Wachstum-Wert von über 8 erreicht werden muss, bis die seitlichen Zweige sprießen dürfen. Dies verändert auch die Gesamtform etwas. Das Blatt respektive der Ast bleibt dadurch am Ansatz etwas schmaler.

Zwei weitere Dinge möchte ich noch ergänzen. Ich würde gern den Tropismus über die Länge, also mit zunehmendem Wachstum, immer stärker werden lassen und der Ast soll noch eine Verlängerung erhalten, bevor die seitlichen Zweige ansetzen. Beides ist wieder schnell erledigt. Hier der neue Startstring, der durch die Ergänzung von F(3) die dreifache Standardlänge vorweg zeichnet, bevor mit dem Buchstaben A die Regeln greifen:

```
Startstring: F(3)A
Regeln: A=FT(1.5)[+(70-60*_index/_total)B]
[-(70-60*_index/_total)B]A
```

Am Anfang der Regel für A erkennen Sie, dass ich T durch T(1.5) ersetzt habe. Damit wird der Tropismus, also praktisch die Erdanziehung, die auf das Blatt oder den Ast wirkt, mit jedem Iterationsschritt verstärkt. Probieren Sie einfach mal T(5) aus, um die Wirkung noch eindrucksvoller begutachten zu können. Der Ast krümmt sich dann immer früher in Richtung Boden, so wie es in der Bildfolge von Abbildung 6 zu sehen ist. **› ei**

Der Autor ist zertifizierter Maxon Instructor und bietet Coachings und individuelle Schulungen zu Cinema 4D an. Zudem ist er seit über zehn Jahren als Dienstleister für 3D-Visualisierungen und Plug-in-Entwicklungen tätig und hat diverse internationale Fachbücher veröffentlicht.