2017
2
ISSN 1433-2620 > B 43362 >> 21. Jahrgang >>> www.digitalproduction.com
Published by ATEC

Deutschland € 17,70
Österreich € 19,–
Schweiz sfr 23,–

DIGITAL PRODUCTION

DIGITAL PRODUCTION

MAGAZIN FÜR DIGITALE MEDIENPRODUKTION

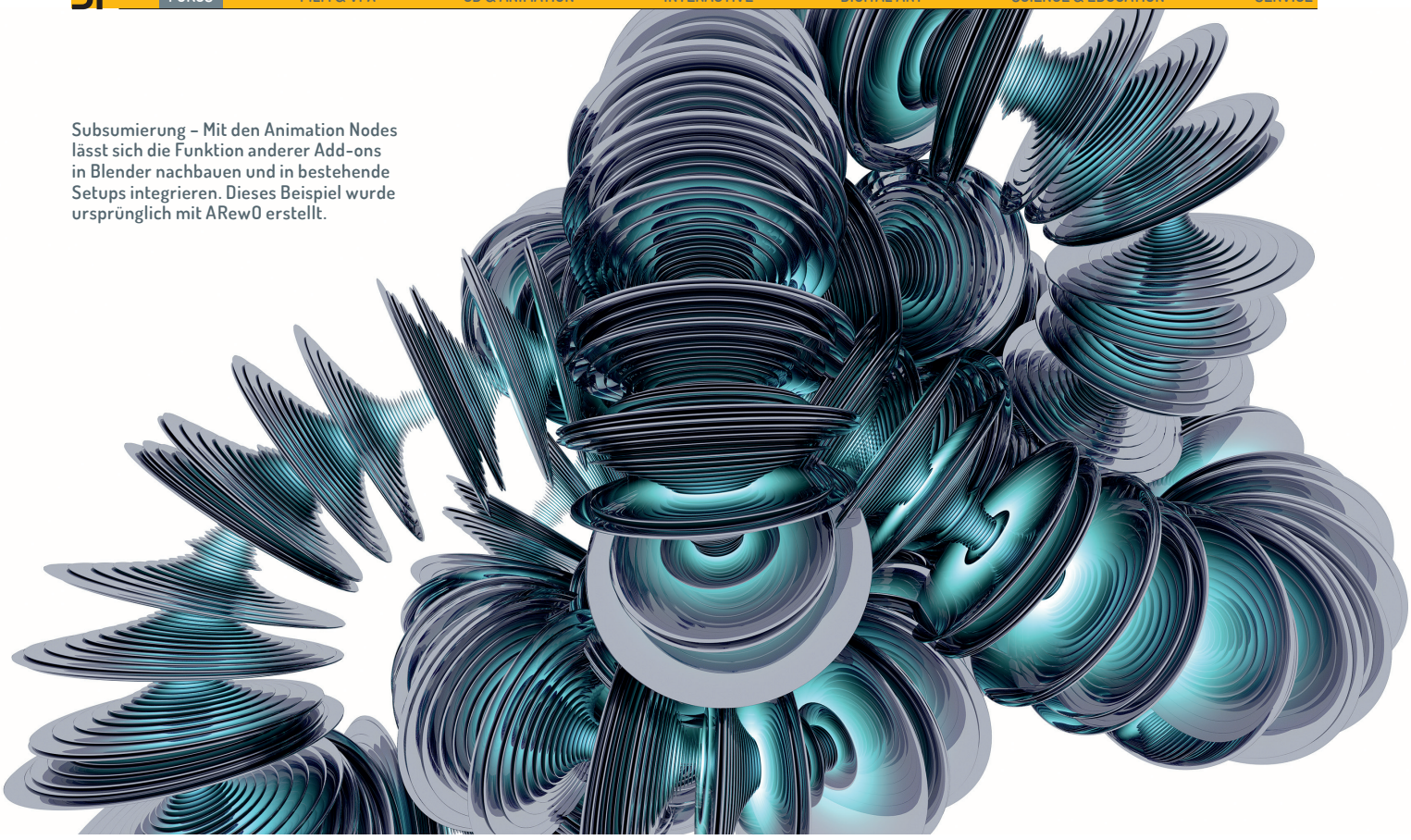
MÄRZ | APRIL 02:2017

VR & AR
Wie nutzt man VR & AR jenseits vom Hype?

Motion Graphics
Prozedurale Grafiken, Audio-Cues & neue Tools

Tools & Tricks!
Avid MC, Lightworks, Houdini, Blender & mehr

4 194336 217709 02 >



Subsumierung – Mit den Animation Nodes lässt sich die Funktion anderer Add-ons in Blender nachbauen und in bestehende Setups integrieren. Dieses Beispiel wurde ursprünglich mit ARew0 erstellt.

Animation Nodes Reloaded

Das Add-on „Animation Nodes" hat sich in der Motion-Graphics-Fraktion inzwischen so weit etabliert, dass es aufgrund der hohen Nachfrage von Renderfarmen unterstützt wird. Währenddessen schreiten die Arbeiten an Version 2.0 voran, die neben deutlich besserer Performance auch neue, weniger technische Workflows bieten wird. Grund genug also für DP, mal wieder einen Blick darauf zu werfen, den Workshop aus Ausgabe 04:2016 fortzuführen und einen Ausblick auf die Zukunft der Animation Nodes zu werfen.

von Gottfried Hofmann

Die Animation Nodes in Blender erfüllen zuallererst den Zweck, Animationen prozedural zu erzeugen. Eher nebenbei sind Hilfsmittel für prozedurale Modellierung vorhanden. Eine große Anzahl der Nodes ist dafür gedacht, Attribute und Einstellungen in Blender auszulesen und zu schreiben. Damit sind die Animation Nodes auch ein mächtiger Ersatz für Driver. Zu guter Letzt fügen die Animation Nodes neue Funktionen hinzu und vereinfachen manche Aufgaben in Blender ganz extrem. Einen Einblick haben wir bereits im Workshop in Ausgabe **DP 04:2016** gegeben, wo die Text-Werkzeuge der Animation Nodes vorgestellt wurden. Mit deren Hilfe wurde die individuelle Animation von Buchstaben ein Kinderspiel. Den Workshop können Sie kostenlos unter **bit.ly/animationnodes** herunterladen. Dort finden Sie auch die .blend-Dateien für diesen Workshop.

Clones

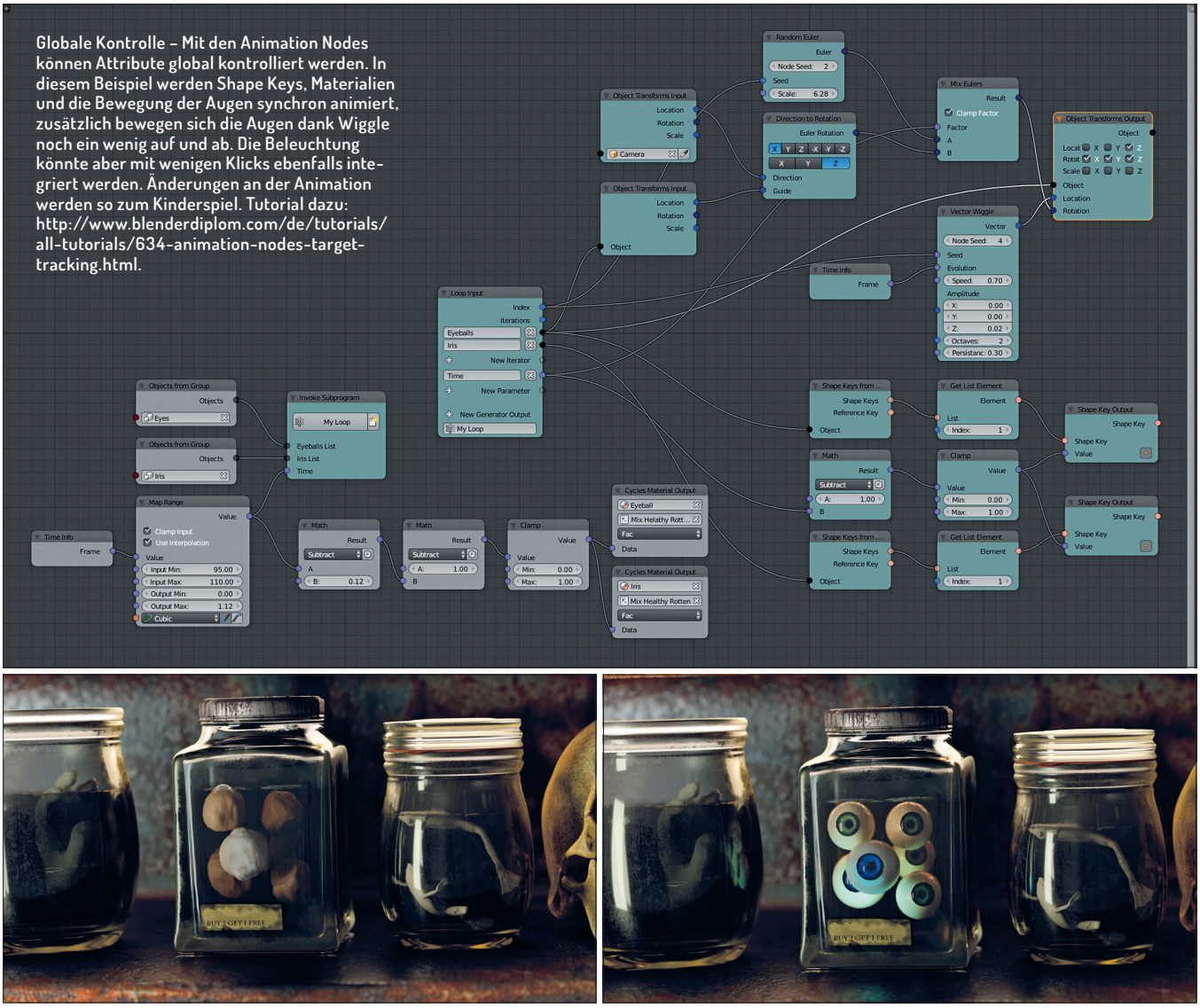
In dieser Ausgabe wollen wir ein weiteres Basiswerkzeug für Motion Graphics mittels Animation Nodes implementieren: Ein System zum Klonen von Objekten. Gegenüber dem Array Modifier in Blender wird das Ergebnis einige Vorteile bieten. Die kopierten Objekte werden Instanzen des Ursprungsobjekts bilden. Dadurch können diese effizient mit Cycles gerendert werden und verbrauchen fast keinen Speicher. Die Eigenschaften der Instanzen können zudem mit einer Art Delta bearbeitet werden, wie z.B. ein zufälliger Offset oder eine andere Farbe. Da Animation Nodes sehr flexibel sind, kann der fertige Aufbau auch mit dem Setup zur individuellen Animation von Buchstaben aus der **DP 04:2016** kombiniert werden.

Coin Stack

Als konkretes Beispiel für die Anwendung eines Array Modifiers mit Offset ziehen wir einen Stapel Münzen heran, der anhand eines Eingabewertes wachsen und schrumpfen kann. Konkret arbeiten wir mit Version 1.6 der Animation Nodes, die direkt von Github bezogen werden kann (**https://github.com/JacquesLucke/animation_nodes/releases**).

Wir starten mit einem Zylinder, den wir im Edit Mode auf 5% seiner Z-Höhe skaliert haben (S Z 0,05), und einem frischen Animation Nodes Tree. Auto Execution lassen wir eingeschaltet, das Häkchen bei „Always" entfernen wir aber, damit die Animation Nodes nicht die komplette Zeit über einen CPU-Kern zu 100% auslasten. Stattdessen setzen wir Häkchen bei den drei übrigen Triggern, damit wir ein Update sehen, sobald wir etwas verändern.

Um das Klonen von Objekten kümmert sich der Node „Object Instancer". Fügen Sie diesen hinzu (Object > Instancer) und wählen Sie den Zylinder bzw. die Münze im Objekt-Drop-down aus. Jetzt können Sie mittels „Instances" eine beliebige Anzahl an Duplikaten erzeugen, diese sind aber alle am gleichen Ort wie das ursprüngliche Objekt. Sie können diesen Sachverhalt überprüfen, indem Sie im Viewport mehrfach versuchen, die Zylinder anzuwählen. Sie werden feststellen, dass sich jetzt mehrere Objekte an derselben Stelle befinden. Diese tragen Namen wie „instance_gbfos_0". Sie können die Instanzen auch verschieben. Sie sehen dann



die Parent-Linie von den Instanzen zu einem unsichtbaren Container-Objekt, welches das Elternteil aller von den Animation Nodes erzeugten Objekte ist und die Bezeichnung „Animation Nodes Object Container" trägt. Dadurch bleibt nicht nur der Outliner aufgeräumt, Sie können damit auch wenn nötig die gesamte Szene von den durch die Animation Nodes erzeugten Objekten reinigen.

Manuelles Verschieben ist aber nicht das, wofür die Animation Nodes entwickelt wurden. Um die Münzen platzieren zu können benötigen wir eine Schleife bzw. einen Loop, der über die Liste der Objekte iteriert. Dass die Ausgabe der Object-Instancer-Node eine Liste ist, erkennt man daran, dass der Ausgabe-Socket ein wenig durchsichtig ist, was für alle Listen in den Animation Nodes gilt.

Iterator

Einen Iterator über eine Liste erzeugt man in den Animation Nodes in zwei Schritten: Zuerst erstellt man ein sogenanntes „Subprogram" und wählt dort „Loop" aus. Dann erzeugt man einen Node, mit dem die so erzeugte Funktion aufgerufen werden kann. Fügen Sie ein Loop-Subprogram hinzu (Subprograms > Loop). Dem neu erzeugten Node kann man ganz unten einen Namen geben, z.B. „Placer". Das sollten Sie auch gleich machen, damit Sie später anhand des Namens die Funktion aufrufen können.

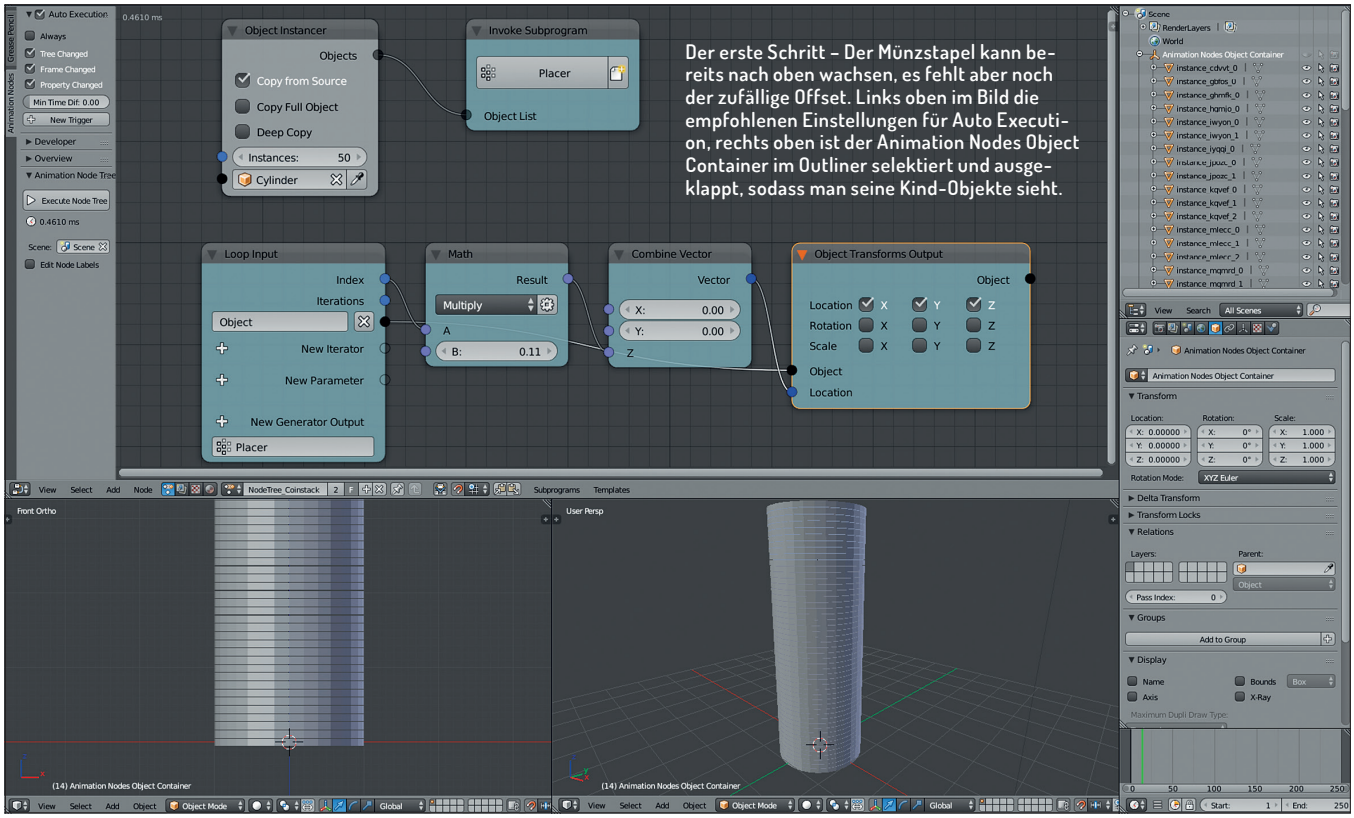
Um die Position der Instanzen zu verändern, wird ein „Object Transforms Output"-Node benötigt (Object > Transforms Output). Fügen Sie diesen hinzu und verbinden Sie den Eingang bei „Object" mit dem Ausgang „New Iterator" des „Loop Input"-Nodes. Der „Object Transforms Output"-Node hat nun seine Farbe an die des „Loop Input"-Nodes angepasst. Durch die automatische Farbkodierung können Sie leicht erkennen, welche Nodes zu welchen Funktionen bzw. Unterprogrammen gehören.

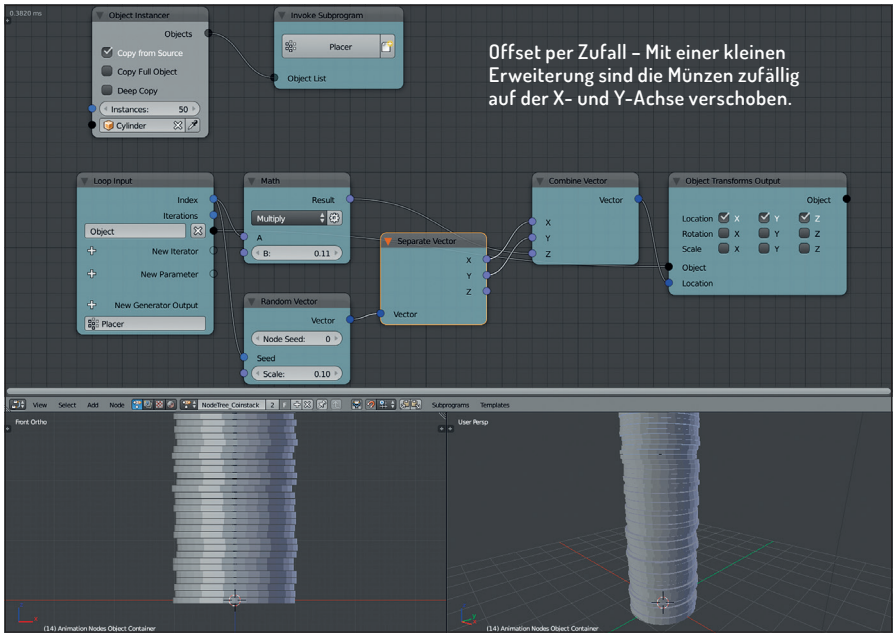
Noch ist aber nichts im 3D-View passiert. Das hat zwei Gründe: Zum einen wird das erzeugte Unterprogramm noch nicht aufgerufen, zum anderen ändert der „Object Transforms Output"-Node noch nichts. Um das Unterprogramm aufzurufen, fügen Sie einen „Invoke Subprogram"-Node hinzu (Subprograms > Invoke Subprogram). Sie können das Unterprogramm entweder direkt in der Liste auswählen oder zunächst einen allgemeinen „Invoke Subprogram"-Node hinzufügen und dann die Funktion in dem Node mittels „Choose" auswählen. Wählen Sie in beiden Fällen die eben erzeugte Schleife.

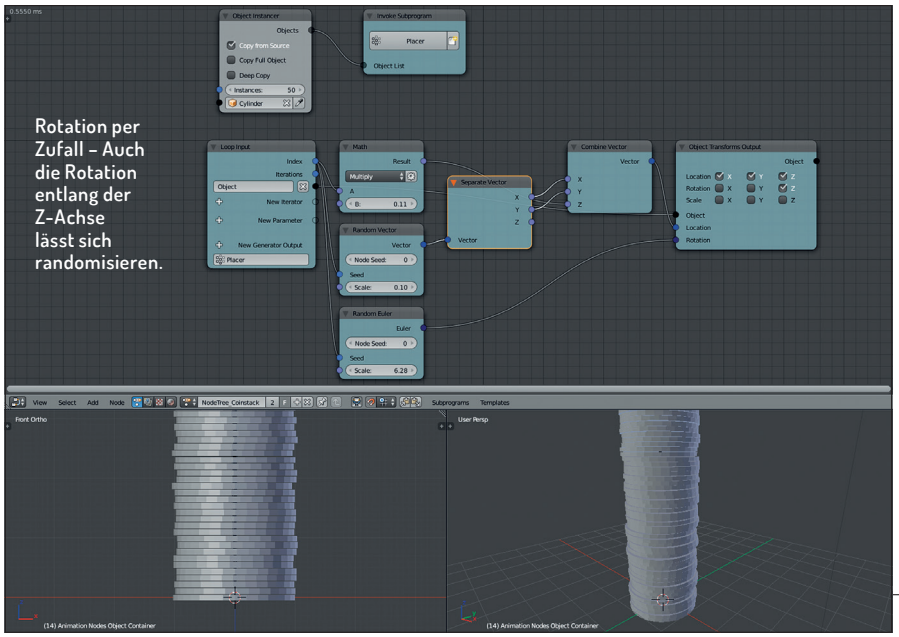
Der „Invoke Subprogram"-Node hat jetzt einen neuen Eingang erhalten und die gleiche Farbe wie die Nodes, die die Schleife definieren. Verbinden Sie den Ausgang des „Object Instancer"-Nodes mit dem Eingang. Jetzt iteriert die Schleife über die instanziierten Objekte, ohne diese zu verändern.

Um die Position der Klone zu modofizieren, muss zunächst bei dem „Object Transforms Output"-Node eingeschaltet werden, dass diese die Position verändert. Wählen Sie die drei Häkchen bei „Location" an und ein neuer Eingangs-Socket erscheint. Es handelt sich dabei um einen Vektor, der die gewünschte Position der Klone annimmt.

Jetzt bleibt noch die Frage, woher diese Position genommen werden soll. Der „Index"-Ausgang des „Loop Input"-Nodes bietet sich dafür an, denn er gibt für jedes







Objekt eine andere Zahl aus (0 für das erste, 1 für das zweite usw.). Verbinden Sie „Index" mit „Location". Es wird automatisch ein Node zwischengeschaltet, da der Index ein Integer ist, die Location hingegen ein Vektor. Der zwischengeschaltete Node „Combine Vector" erzeugt einen Vektor aus den drei Eingaben. Jetzt verteilen sich die Münzen entlang der X-Achse, denn der Index wird automatisch mit dem obersten Socket, sprich dem Wert für die X-Achse, verbunden. Verbinden Sie den Index stattdessen mit dem Eingang für die Z-Achse, damit sich die Münzen nach oben stapeln.

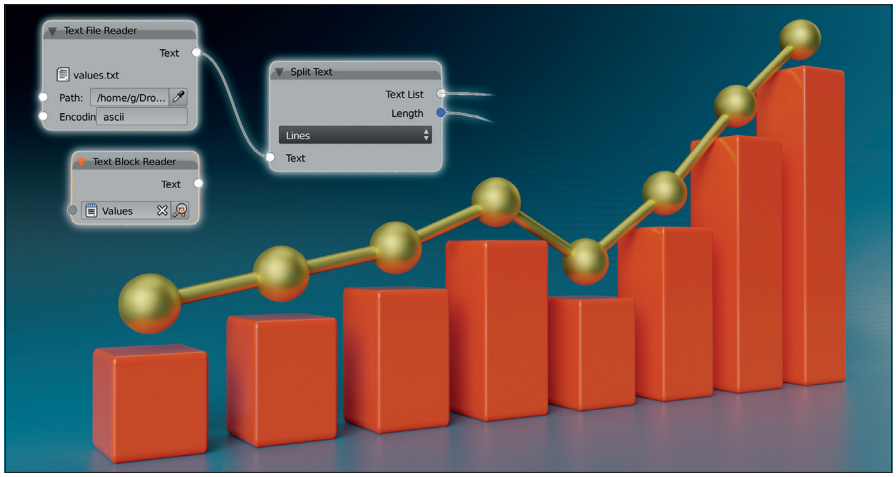
Noch finden sich große Lücken zwischen den Münzen. Für einen richtigen Stapel muss der Abstand verkleinert werden. Das erreichen Sie mit einem Mathematik-Node (Number > Math). Setzen Sie diesen zwischen „Index" und „Combine Vector". Er ist bereits auf „Multiply" eingestellt. Wenn Sie die untere Zahl ändern, verändert sich der Abstand entsprechend. Setzen Sie den Abstand z.B. auf 0,11, damit die Münzen fast aufeinander liegen (siehe obere Abbildung).

Sie haben jetzt einen Stapel Münzen, dessen Höhe Sie über die „Instances" des „Object Instancer"-Nodes beeinflussen können. Was jetzt noch fehlt ist eine Option, mit der die Münzen leicht versetzt gestapelt werden können. Diese Erweiterung lässt sich mit wenig Aufwand realisieren.

Offset

Der Offset soll zufällig in X- und Y-Richtung gehen. Dafür eignet sich ein randomisierter Vektor, denn der besteht quasi aus drei zufäl-

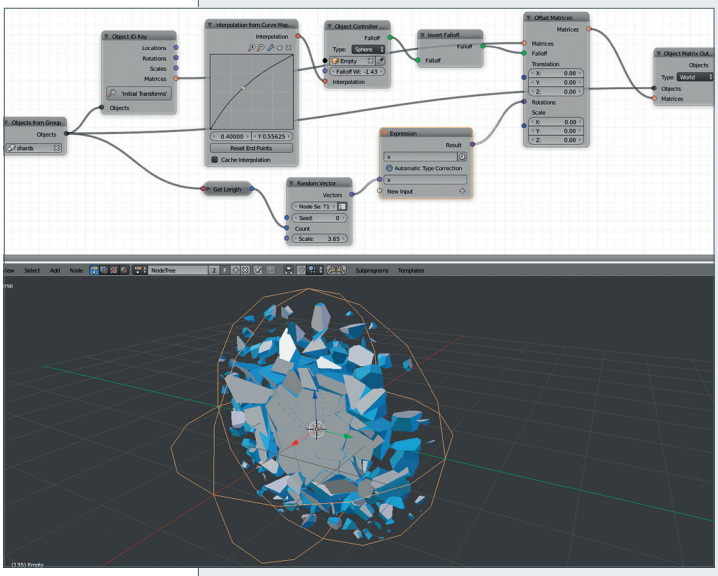
ligen Zahlen, von denen wir zwei brauchen. Fügen Sie einen „Random Vector"-Node hinzu (Vector > Randomize) und verbinden Sie den Eingang von „Seed" mit dem „Index" des „Loop Input"-Nodes. Jetzt erhält jede Münze einen eigenen, zufälligen Vektor mit maximaler Länge wie unter „Scale" angegeben. Dank der automatischen Konvertierung bei Animation Nodes können Sie den „Vector"-Ausgang direkt mit dem „X"-Eingang des „Combine Vector"-Nodes verbinden. Wie durch Zauberhand erscheint ein neuer Node dazwischen: „Separate Vector". Noch sind lediglich die beiden X-Komponenten verbunden. Verbinden Sie zudem noch die Y-Komponenten, so verschieben sich die Münzen zufällig auf dem Stapel, allerdings noch viel zu extrem. Reduzieren Sie die „Scale" des „Random Vector"-Nodes auf z.B. 0,10, damit die Münzen lediglich ein wenig verschoben sind (siehe mittlere Abbildung links).



Einlesen von Werten aus Datei – Ein weiteres Feature der Animation Nodes, das es sich zu evaluieren lohnt: Die Inhalte von Textdateien oder Textblöcken in Blender können eingelesen werden. Die Möglichkeiten zum Parsen sind allerdings noch sehr low-level. Man kann sich aber z.B. einen eigenen CSV-Reader zusammenklicken. Das von Kmeirlaen und Rohan Dalvi inspirierte Beispiel nutzt ein reguläres Säulendiagramm, eine Erweiterung des Münzenstapels ist aber ebenfalls denkbar.

Rotation

Die Rotation der Münzen kann ebenfalls zufällig variieren. Das sollte aber nur entlang der Z-Achse geschehen. Setzen Sie im „Object Transforms Output"-Node daher bei Rotation nur das dritte Häkchen. Wenn Sie den Node ein wenig breiter ziehen, werden sie feststellen, dass neben den Boxen jeweils die Achse angegeben ist, in diesem Fall die Z-Achse. Es erscheint ein neuer Eingang mit drei Werten. Da Sie aber nur das Häkchen bei Z gesetzt haben, wird nur der dritte Kanal einen Einfluss haben.



Der Eingangs-Socket ist dunkelblau. Das bedeutet, dass er eine Rotation erwartet. Im Nodes-Menü finden Sie unter „Rotation" den Eintrag „Random Euler". Fügen Sie diesen hinzu und verbinden Sie „Seed" wieder mit dem „Index"-Ausgang des „Loop Input"-Nodes. Da Sie die Rotation im „Objects Transforms Output"-Node auf die Z-Achse beschränkt haben, können Sie den „Euler"-Ausgang direkt mit dem „Rotation"-Eingang verbinden. Jetzt werden alle Münzen zufällig entlang der Z-Achse rotiert. Der Wert „Scale" des „Random Euler"-Nodes nimmt Radianten als Eingabe. Wenn Sie dort „2*pi" eingeben, erhalten Sie die Entsprechung für eine maximale Rotation von 360 Grad in Radianten, denn das Eingabefeld funktioniert wie ein Taschenrechner. Das ist praktisch bei allen Eingabefeldern für Zahlen in Blender der Fall.

Ausblick auf Version 2.0 der Animation Nodes

Die nächste Version der Animation Nodes (Version 2.0) wird zwei große Neuerungen bereitstellen. Zum einen wird sie nicht mehr ausschließlich in Python, sondern an Performance-kritischen Stellen auch in Cython implementiert sein. Das bedeutet, dass man das Add-on nicht mehr einfach so herunterladen und installieren kann. In Zukunft wird man auch auf das Betriebssystem achten müssen. Sprich, es wird eine Version der Animation Nodes für Windows, eine für Linux und eine für Mac OS geben. Im Gegenzug wird man mit einem deutlichen Zuwachs an Performance belohnt. Weiterhin wird ein Feature Einzug halten, das bei einigen Aufgaben die Anzahl der benötigten Loops reduzieren und damit die Arbeit erleichtern wird: Falloffs. Manche Nodes werden eine Art Einflussbereich erhalten, der z.B. über Empties kontrolliert werden kann. Das Erleichtert die Erstellung gängiger Motion Graphics wie einfliegende Buchstaben oder die Bewegung einer großen Anzahl von Objekten relativ zu einem Controller-Element. Preview-Builds finden sich auf **Graphicall.org** als Animation Nodes Version 1.7. In diesem Beispiel kontrolliert ein Empty die Skalierung und Rotation hunderter Einzelteile, die per Cell Fracture aus einem Objekt erstellt wurden. Damit kann eine Animation erstellt werden, wie sich ein Objekt aus zahllosen Scherben wieder zusammensetzt (oder umgekehrt). Dabei wird keine einzige Schleife benötigt, da die Nodes für Zufallswerte jetzt auch Listen ausgeben können (der Node für zufällige Rotationen leider noch nicht, weshalb in diesem Beispiel ein Vektor-Node mit dazugeschaltetem Konverter eingesetzt wird). Die neu hinzugekommenen Offset-Nodes iterieren automatisch über die Listen anhand eines Falloffs, der sehr frei definiert werden kann. Hier erfolgt das über ein Objekt, das das Zentrum des Falloffs angibt. Wie genau er abfällt, wird über eine Interpolationskurve beschrieben. Vielen Dank an Jacques Lucke, den Entwickler der Animation Nodes, für das Beispiel.

Fazit

Einen Array Modifier mit Offset – das wünschen sich viele für Blender. Mit den Animation Nodes wird dieser Wunsch Wirklichkeit und lässt sich noch um zusätzliche Features erweitern. Der Artikel zeigt aber mal wieder nur einen winzigen Teil der Mächtigkeit des Add-ons auf - die Spitze der Spitze des Eisbergs sozusagen. **› ei**

Gottfried Hofmann ist Diplom-Informatiker. Er arbeitet als Freelancer in den Bereichen Visualisierung und VFX sowie als Trainer und Consultant für die freie 3D-Software Blender. Als freischaffender Autor schreibt er für Fach- und Computerzeitschriften. Er hat zahlreiche Blender-Tutorials verfasst, u.a. für CG Tuts+ und CG Cookie. Weiterhin betreibt er die Webseite **www.BlenderDiplom.com**, auf der Blender-Tutorials in deutscher und englischer Sprache zur Verfügung stehen.