2019
4
ISSN 1433-2620 > B 43362 >> 23. Jahrgang >>> www.digitalproduction.com
Publiziert von DETAIL Business Information GmbH

Deutschland	€ 17,90
Österreich	€ 19,–
Schweiz	sfr 23,–

DIGITAL PRODUCTION

DIGITAL PRODUCTION

MAGAZIN FÜR DIGITALE MEDIENPRODUKTION

JULI | AUGUST 04:2019

Hardware
Keys, Screens, Storage und mehr im Fokus

Praxis
ZBrush, X-Particles, PBR, OSL, Tyflow, Flame & Resolve

und vieles mehr!
Renderman, Open Timeline IO, Weta, Open Movies ...

4 194336 217907 04

tyFlow, ein neues Partikelsystem für 3ds Max!

Ein neuer Stern geht am 3ds-Max-Himmel auf: Tyson Ibele, der mir früher schon mit interessanten 3D-Projekten auffiel, hat in den letzten Monaten (oder gar Jahren) angefangen, ein neues Partikel-Simulationssystem zu schreiben, und füttert seit einiger Zeit die CG-Gemeinde immer wieder mit verschiedenen schrägen Videos zum momentanen Entwicklungsstand. **von Mike Kuhn**

Seit dem 1. April ist die Public Beta draußen, die jede Person, die sich dafür interessiert, kostenlos nutzen darf. Natürlich erwartet man an diesem Tag hauptsächlich Scherzmeldungen. Selbst die Redaktion der Digital Production schickte die Leser ihrer Website mit einer Meldung ins Bockshorn, welche sehr willkommen gewesen wäre, wäre sie denn kein Aprilscherz gewesen (**bit.ly/sad_but_true_but_sad**). Jedoch stellte sich die Meldung über Tyson Ibelles tyFlow zur Freude aller als wahr heraus.

Jeder, der PFlow liebte, als es herauskam, und bitter enttäuscht wurde, wie schäbig Autodesk es behandelte, wurde von erneutem Enthusiasmus gepackt und machte sich daran, dieses neue Spielzeug zu testen.

Interface und primäre Features

Das Interface ähnelt dem des alten Particle Flow sehr deutlich. Jedoch handelt es sich hier nicht einfach nur um ein Upgrade von Particle Flow. Stattdessen ist tyFlow ein eigenes Tool, das von Grund auf komplett neu erstellt wurde. Die starke Ähnlichkeit zu Particle Flow basiert lediglich auf der Effektivität einer solchen Node-basierten Benutzerumgebung.

tyFlow ist so ausgelegt, dass jede Funktion möglichst alle CPUs nutzt. Der Particle Bind Solver, welcher für die Simulation von Stoffen, weichen Objekten und granularen Systemen genutzt wird, sowie der Particle Physics Operator können für die notwendigen Berechnungen sogar schon jetzt auf die GPU zugreifen. Natürlich empfiehlt es sich in diesem Fall, eine Grafikkarte mit genügend RAM im System eingebaut zu haben. Falls die GPU jedoch nicht genügend RAM zur Verfügung hat, um alle notwendigen Bindings zu speichern, erscheint eine Warnmeldung im 3ds Max Lister und es wird stattdessen die CPU genutzt.

tyFlow vs. Particle Flow

Auch wenn tyFlow auf den ersten Blick Particle Flow sehr ähnlich sieht, gibt es

Auch der tyFlow Editor nutzt die intuitive Nodes-Umgebung.

einige Unterschiede in der Handhabung. „So scheint tyFlow weniger Operatoren zu haben, wenn man sich die bisherige Liste anschaut", schrieb ich, als ich vor wenigen Wochen mit diesem Artikel anfing. Jedoch kommen mittlerweile mit vielen der häufigen Beta-Updates neue interessante Operatoren hinzu. Bei tyFlow werden oft mehrere Funktionen in einen Operator gepackt, wo es bei Particle Flow meistens viele separate Operatoren für die jeweiligen speziellen Aufgaben gibt. Anstelle unterschiedlicher Nodes für die Form des Partikels haben wir in tyFlow nur einen Shape Node. Na ja, eigentlich sind es zwei, wenn man den Birth Shape Node mitzählt. Jedoch hat auch dieser wieder viele mächtige Werkzeuge in einem Operator vereint.

Birth Events und Event Nodes

Zum Thema Birth Events, also den verschiedenen Möglichkeiten der Erstellung der Partikel, sieht es schon wieder ganz anders aus. Dafür gibt es bisher ganze neun verschiedene Nodes. Jeder einzelne davon hat seine spezifischen Eigenschaften. tyFlow ist also sehr effizient aufgebaut: Reduzierungen, wo es sich lohnt, und Vielfalt, wo es gerne gesehen ist.

Eine mögliche Falle, in die viele am Anfang tapsen, ist das Fehlen des aufwendig erstellten tyFlow-Projekts im Rendering. Anders als Particle Flow, wo prinzipiell alles renderfähig ist, benötigt man in tyFlow am Ende des Events einen Mesh Node, der alles erst in ein renderfähiges Mesh verwandelt. Dadurch kommt tyFlow prinzipiell mit allen Renderern klar. Es scheint aber eine Liebschaft mit V-Ray zu haben. So werden zum Beispiel unter anderem auch V-Ray Instances unterstützt. Das ist vor allem dann eine praktische Sache, wenn man Bewegungsunschärfe im Rendering haben möchte.

Im alten Particle Flow fand man alle genutzten Operatoren auch in der Explorer-Liste der eigentlichen Szene verstreut. Nicht so in tyFlow! Hier ist jeder Flow für sich mit all seinen Operatoren innerhalb seines Editors beherbergt. Wenn man also mehrere tyFlow-Objekte in der Szene hat, was machbar und oft auch ratsam ist, wird der 3ds-Max-Explorer nicht mit unzähligen Operatoren zugeschüttet, von denen man nicht genau weiß, wozu sie gehören, falls man nicht alles sauber benannt hat.

Weitere Funktionen

Wirklich auf jede einzelne mögliche Funktion einzugehen, würde wahrscheinlich diese komplette Ausgabe der Digital Production ausfüllen. Somit schauen wir uns nur ein paar unterschiedliche Workflows an. Zum Erlernen von tyFlow bietet es sich auch an, die Beispielszenen zu studieren, welche auch von der Download-Seite von tyFlow angeboten werden. Alternativ dazu gibt es, gerade mal einen Monat nach dem Erscheinen der ersten Beta-Version, inzwischen eine Unmenge an Video-Tutorials zu finden. So viele Tutorials in solch kurzer Zeit zeigt noch einmal deutlich, wie viel Hype und Begeisterung tyFlow in der Community generiert. Selbst bekannte Größen aus den Special-Effects-Bereichen wie Allan McKay haben schon mehrere Video-Tutorials zu unterschiedlichen Anwendungsmöglichkeiten von tyFlow erstellt. Aber genug des allgemeinen Geplänkels. Lasst uns mal ein paar Möglichkeiten testen.

Praxis!

Neben dem Verteilen von Konfetti in der Luft eignen sich Partikelsysteme auch sehr gut, um Gegenstände zu zerbrechen. Das Setup ist sehr einfach: Wir benötigen nur eine Fensterscheibe in einem Rahmen und einen Backstein, um den eine freundliche Aufforderung des Anbieters gewickelt ist. Jetzt sollten wir dieses Konstrukt noch in ein vernünftiges Simulationsobjekt verwandeln.

Wahrscheinlich würde man es in der Geschwindigkeit sowieso nicht sehen, und auch die meisten Physiker werden bestätigen, dass die umwickelte Nachricht keine Fenster zerbrechen kann. Aber ich möchte gerne, dass kleinere Bruchstücke später auch von dem Papier der Nachricht beeinflusst werden. Also habe ich ein neues Objekt erstellt, das die allgemeine Form des Backsteins inklusive der Nachricht umhüllt und so wenige Polygone wie möglich hat. Der Stein und die Nachricht wurden daran geknüpft und dann wird nur dieses Hüllenobjekt animiert, das ab jetzt auch als gemeinschaftliches Kollisionsobjekt dient.

Durch das Unterteilen in einzelne und kleinere Segmente wird das Skelett der verformbaren Kugeln erstellt.

Sim it!

Für die Simulation wird ein tyFlow-Objekt in die Szene gesetzt, das in den Standard-Primitives im Command Panel zu finden ist. In dessen Eigenschaften-Rollout öffnen wir den tyFlow-Editor. In diesem Beispiel bietet es sich an, mit einem Birth Shape Node zu beginnen, dem dann die Fensterscheibe als Referenzobjekt hinzugefügt wird. In diesen Event wurde automatisch ein Display Node eingefügt. Durch diesen Node sehen wir unser Partikelsystem als einfache Pixel, als kleine oder große Punkte, als Ticks oder Sprites, als Bounding Box oder auch als die wirkliche Geometrie. Das ist auch die Auswahl, welche wir einstellen werden. Da jetzt auch das Partikelobjekt in der Szene zu sehen ist, kann die originale Geometrie der Fensterscheibe ausgeblendet werden.

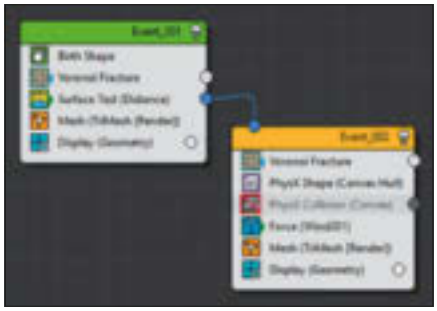
Früher mussten solche zerbrechlichen Objekte immer erst vorher in die gewünschten Einzelteile zerbrochen werden. Im tyFlow-Editor füge ich einfach als Nächstes einen Voronoi Fracture Node in diesem Event hinzu. Darin kann ich definieren, in wie viele Fragmente ich die Geometrie unterteilen möchte, und habe noch unzählige weitere Einstellungsmöglichkeiten, auf die wir in diesem kurzen Artikel gar nicht alle eingehen können. In einem späteren Beispiel werde ich jedoch noch ein paar davon nutzen. Im jetzigen Beispiel habe ich 160 Points für die Erstellung der Bruchstücke eingestellt und den Rest belassen, wie er war.

Surface Test Operator

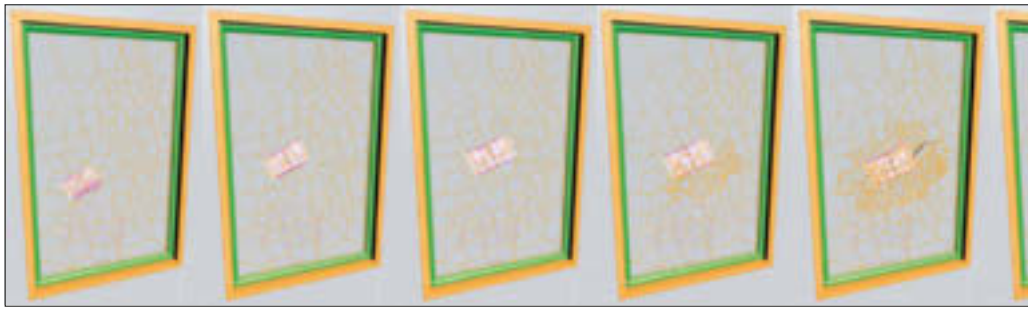
Anstelle eines Collision Operators nutzen wir hier jetzt mal den Surface Test Operator und definieren dort das vorhin erstellte Kollisionsobjekt als Referenz. Der Surface Test Operator leitet ein Ja oder ein Nein an das nächste Event. Das basiert auf dem Auftreffen aus einer Distanz zur Oberfläche sowie den ausgesandten Raycasts. Diese Casts beziehen sich auf in der nächsten Nähe liegende Texturen, Material-IDs, die Normalen oder auch die ankommenden Geschwindigkeit des in den Settings definierten Objekts. In unserem Fall ist die „Distance To Surface" die praktischste Lösung – darin habe ich in meinem Beispiel eine Distanz kleiner als 10 cm eingestellt, weil sich hier ja sowieso alles sehr schnell bewegt. Bei stabilerem Glas könnten wir auch den Geschwindigkeitstest nutzen, um schwächer geworfenen Nachrichten weniger Durchschlagskraft zu geben.

Voronoi

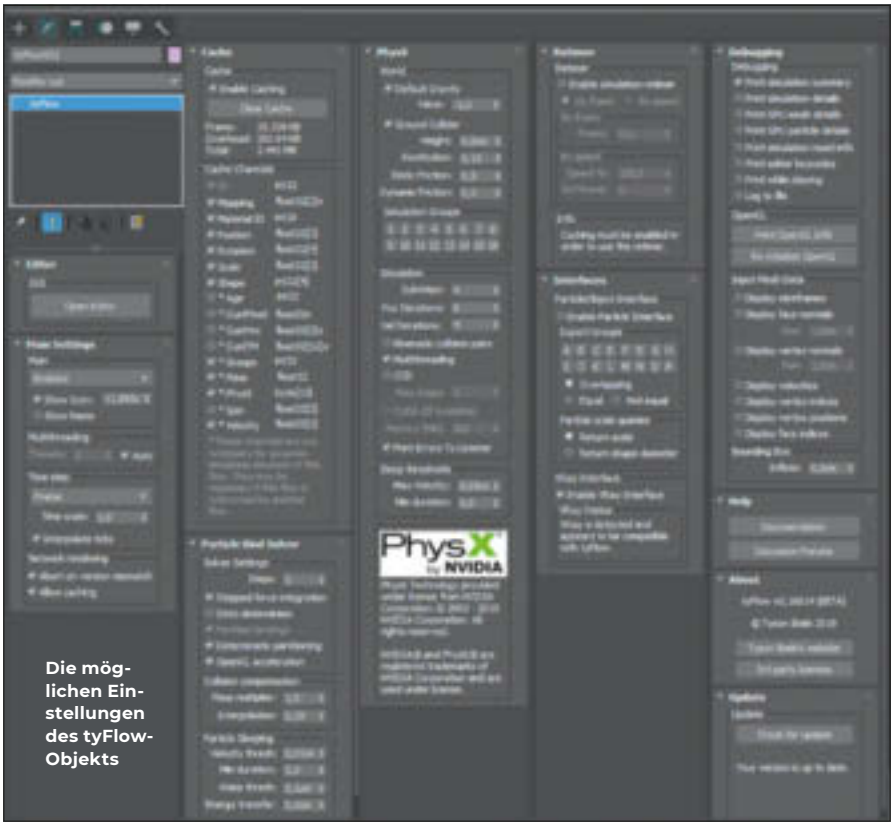
In einem neuen Event wird nun vorgelegt, was die Partikel, die das vorherige Event gerade durch das Bestehen ihres Tests verlassen haben, jetzt zu tun haben. Dazu schieben wir einen neuen Voronoi Fracture



Mit nur wenigen Nodes ist eine Simulation erstellt.



Durch die Annäherung der Nachricht wird die Scheibe weiter zerteilt und gibt letztendlich den Weg frei für die freundliche Aufforderung an die Kunden.



Die möglichen Einstellungen des tyFlow-Objekts

Operator in die Editor-Oberfläche, um daraus ein neues Event zu erstellen. An dieses verbinde ich den Ausgang aus dem Surface Test Node. Somit werden die Fragmente in unmittelbar zu zerbrechender Entfernung weiter zerbrochen und sind bereit, ihre neuen Aufgaben in diesem Event auszuführen.

Ein sehr angenehmes Feature ist die Nutzung der Tab-Taste, um ein Suchfenster zu öffnen, in dem man schnell die gewünschten Operatoren finden und in den Flow einfügen kann. Um dem jetzigen Partikelzustand physikalische Eigenschaften zuzuweisen, ziehen wir einen PhysX Shape Operator in das Event und belassen ihn bei Convex Hull, da Box oder Sphere zu ungenau wären und weil uns die Option, die Kollisionsaußenkanten des originalen Meshs zu nutzen, für die ersten Tests vielleicht zu viel Schnelligkeit rauben könnte. Nachdem alle Tests abgeschlossen wurden, kann man für das finale Rendering immer noch auf diese präzisere, jedoch weniger ressourcenorientierte Alternative zugreifen. Wir arbeiten ja in einer digitalen Umgebung, und da ist ein wenig Tricksen ja auch erlaubt.

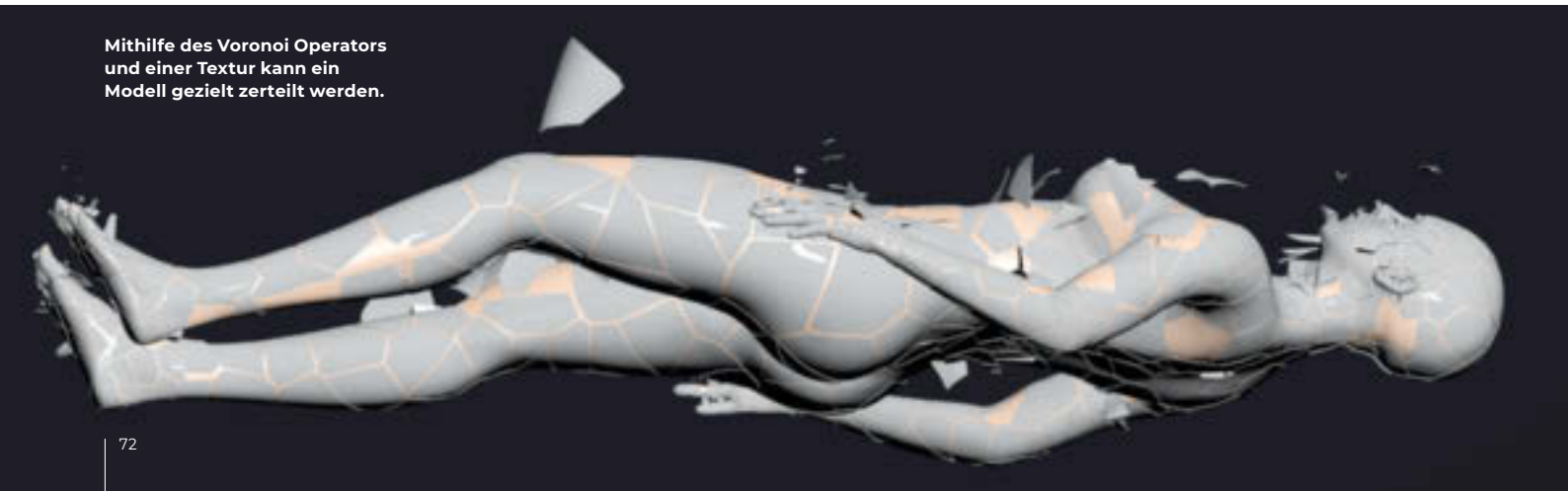
Also erstellen wir ein ganz normales 3ds-Max-eigenes sphärisches Wind-Objekt hinter der Scheibe und schieben einen Force Operator in das Event, dem dieser Wind zugeführt wird. So geben wir den herumfliegenden Glassplittern ein wenig mehr Wums für die Situation. Wie schon vorhin erwähnt, müssen wir den Flow mit einem Mesh Operator abschließen, um unsere Arbeit überhaupt erst renderbar zu machen.

Im Rollout des tyFlow-Objekts selbst gibt es jede Menge an Einstellungen, für die sich eigentlich ein eigenständiges Tutorial lohnen würde. Hier kann auch die prinzipielle Schwerkraft definiert werden, die jedoch von manchen PhysX-Operatoren überschrieben werden kann. Auch findet man hier die automatische Bodenplatte als Kollisionsobjekt.

Solver

Wichtig sind die Solver-Settings. Diese geben vor, wie genau die Berechnungen durchgeführt werden. Mehr Steps ergeben eine genauere Berechnung der Situation, resultieren natürlich aber auch in längeren Berechnungszeiten. Bei meinem Projekt für das Aufmacher-Bild habe ich anfangs einfach den Voronoi Operator eingesetzt und ihn auf nur 1.000 Bruchstücke eingestellt, um meinen Rechner nicht zu arg zu stressen. An sich sah das auch relativ gut aus, aber manche Bruchstücke bestanden aus dem kompletten vorderen Fußbereich und sahen dann halt aus wie ein halber Handschuh für Füße. So etwas kann den gewünschten Effekt dann schon stören.

Glücklicherweise bietet der Voronoi Operator eine ganze Handvoll weiterer Optionen. Neben der einfachen Unterteilung durch Punkte in einer Wolke kann man sie durch Partikel verteilen, gleichmäßig über die Oberfläche streuen oder auch die Zerteilung durch eine Texturinformation definieren. Somit probierte ich den schnellsten und leichtesten Weg und erstellte mithilfe des in 3ds Max eingebauten Surface Map Bakers eine Density Map. Die hatte jetzt auf die Schnelle natürlich nicht die Qualität, die ich in Substance Pain-



Mithilfe des Voronoi Operators und einer Textur kann ein Modell gezielt zerteilt werden.

ter oder Designer, XNormals, Marmoset oder anderen Tools erreichen könnte. Trotzdem bewirkte sie genau das, was ich benötigte. Um den Bauch herum entstanden größere Stücke, während um die Zehen herum viele kleine Bruchstücke generiert wurden.

In dem hinzugefügten Force Operator stellen wir die Schwerkraft auf einen Wert, der die Partikel sanft nach oben schweben lässt, und nutzen den darin integrierten Wind, um ein paar Turbulenzen in die Simulation zu bringen. In einem PhysX Collision Node wird die Person als Kollisionsobjekt hinzugefügt, damit die sich ablösenden Partikel daran entlang und vorbei wandern können.

Um mal etwas anderes zu machen und weil Ostern gerade um die Ecke ist, zeige ich hier die Simulation eines Maschinenpistolenangriffs auf kugelsichere Eier.

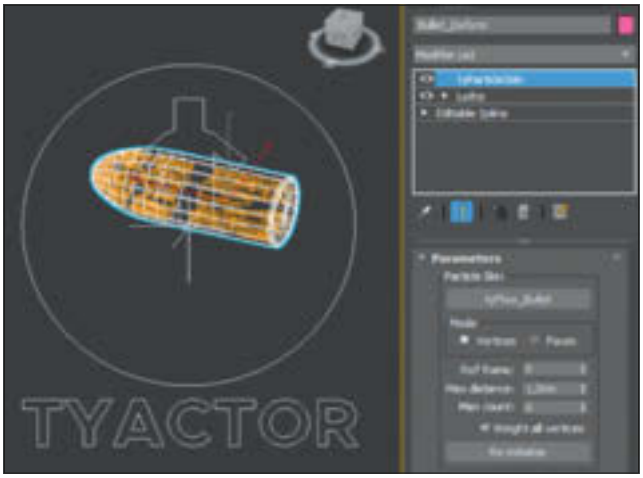
Weiterlesen auf eigene Gefahr

Um etwas in der Szene zu sehen, brauchen wir erst mal richtig große Eier. Zur Erstellung dieser kann ich endlich mal die Egg Grund Shape in 3ds Max nutzen – wahrscheinlich wurde diese Funktion genau deswegen eingebaut. Das resultierende Lathe-Objekt wird dupliziert, die Grund-Shape leicht angepasst, und schon haben wir auch die Eierbecher, in denen unsere Eier ruhen werden. Hier halte ich ein wenig Abstand zum Eierbecher, um dem System eine Chance zu geben, die Kollisionen zu erkennen, sollte ich mich später entscheiden, die Eier doch noch mit einer Achillesferse zu versehen. Hinzu fügen wir noch eine Tischplatte und eine Flasche, die natürlich auch fix mit dem Lathe Workflow erstellt ist. Der Aufbau der Simulation ist in manchen Bereichen ähnlich wie manche der vorherigen Simulationen, und so werden wir hier das Erlernte mit einbeziehen. Für die sich verformenden Kugeln benötigen wir kleine Stücke, welche die Simulation durchmachen und die dann genutzt werden, um das eigentliche Objekt zu verformen. Dafür erstellen wir das erste tyFlow-Objekt und öffnen dessen Editor.

Birth & Bullet

Hier besteht der Flow aus einer Birth Shape, welche wir mit der Geometrie der Kugel füttern, und einem Voronoi Fracture Node. Darin zerteile ich die Kugel in ungefähr 42 Teile und sorge dafür, dass der jeweilige Pivot immer im Zentrum der Bruchstücke liegt. Allerdings sollten die Kugeln vorne mehr verformt werden, während der hintere Bereich eigentlich größtenteils seine Form bewahren sollte. Also nutze ich hier wieder die Technik vom Workflow für das Titelbild und erstelle eine Gradient Map, um den hinteren Bereich der Kugel kaum bis gar nicht zu zerteilen.

Jetzt haben wir die Splitter, die später die Verformungsmöglichkeiten der Kugeln definieren, so verteilt, dass vorne wesentlich mehr Variationen entstehen können. Da die jetzt entstandenen Fragmente noch aneinander liegen und somit in ihrer Funktion als zukünftige Bones wenig Bewegungsspielraum bieten, werden wir diese noch ein wenig schrumpfen. Mit einem Scale Node reduzieren wir die Bruchstücke auf eine Größe von circa 60%, und schon haben wir unser Rig für die Kugeln erstellt. Da die Kugel jetzt als Referenz für die Fragmente gilt, müssen wir diese duplizieren, um dann mit dieser unabhängigen Kopie die weiteren Simulationen durchzuführen. Auf diese Kopie legen wir den Modifikator tyParticleSkin und verweisen darin auf den tyFlow, welcher für die Zerteilung der Originalkugel zuständig ist. Um dieses geriggte Konstrukt in einer weiteren Simulation zu nutzen, muss ein tyActor erstellt werden, der wiederum die geskinnte Kugel beherbergt.



Mithilfe des tyActor-Helferobjekts können geriggte oder animierte Objekte einem tyFlow zugefügt werden.

Jetzt können wir ein neues tyFlow-Objekt erstellen, in dem wir die eigentliche Animation vornehmen. In dessen Editor kommt ein einfacher Birth Operator, um ein paar Partikel zu erstellen, ein Position Icon Operator oder ein Position Object Operator, um festzulegen, wo diese Partikel erstellt werden, und ein Actor Node, der dann die Form der Partikel definiert. Aus dem jetzt mit dem tyActor gefüllten Actor Node geht es weiter in ein neues Event, in dem erst einmal eine Gruppenzugehörigkeit festgelegt wird und die physikalischen Eigenschaften zugewiesen werden. Da diese Einzelteile, welche momentan als Bones für die Kugel dienen, jetzt einfach zerfallen würden, müssen sie mit einem PhysX Bind Node untereinander festgehalten werden. In diesem Node werden die Verbindungen untereinander gerade so festgezurrt, dass sie in Form bleiben, sich aber auch bei einem Aufprall verformen können. Als Kollisionsobjekte werden dann die Eier, die Eierbecher und die Tischplatte definiert.

PhysX

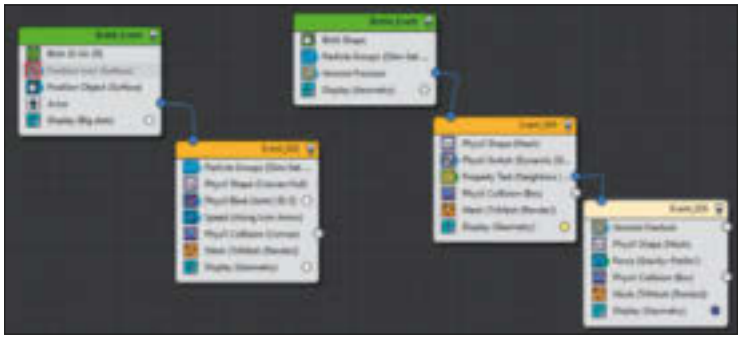
Für den Flow der Flasche kommt wieder der schon bekannte Flow mit einer Birth Shape und einem Voronoi Fracture zum Einsatz. Zusätzlich wird dieser Flow einer anderen Gruppe zugewiesen. Nach dem Zerteilen der Flasche bekommen auch diese Bruchteile ihre physikalischen Eigenschaften, werden aber mit einem PhysX Switch Node erst mal auf die Wartebank gesetzt. So wird die Flasche nicht einfach durch die Schwerkraft zerfallen, sondern erst ab einer gewissen Krafteinwirkung oder einem anderen, von uns zu definierenden Event. In diesem Fall ist es ein Property-Test, der nach Nachbarn aus einer bestimmten Gruppe in unmittelbarer Annäherung schaut. Da die Kugeln zu dieser zu suchenden Gruppe gehören, sendet deren Annäherung den Flow zum nächsten Event, in dem wir die getroffenen Teilstücke noch einmal zerbrechen und auf ihre weitere physikalische Reise schicken.

Ähnlich wie das alte Particle Flow kann auch tyFlow die Simulation als Cache-Datei auf die Festplatte schreiben. So können wir jetzt auch im Bullet-Time-Effekt um unsere kugelsicheren Eier herumkreisen.

Wie schaut es mit tyFlow als Scatter-System aus?

Oft möchte man einfach nur Objekte auf einer Oberfläche verteilen. Dazu gibt es verschiedene Methoden. Das Compound-Objekt Scatter und die alten Partikelsysteme sind ein paar davon. In tyFlow erstelle ich einen einfachen Flow mit einer Birth Shape, in die alle meine vorher modellierten Gebäude kommen, und einen Spawn Node, der


In tyFlow ist es ein Leichtes, die Gesetze der Natur umzukehren.



Der Flow für das Zusammenspiel der Kugeln, der Flasche und den Eiern aus Kryptonit

die Gebäude zum nächsten Event schickt. In diesem Event werden die Gebäude dann auf einer Landschaft verteilt. In diesem Position Object Operator kann auch eine Textur benutzt werden, um Bereiche freizulassen. Allen Gebäuden habe ich vorher ein Multi-Material verpasst und kann jetzt mithilfe eines Material ID Nodes diese IDs zufällig zuweisen lassen. So bekommen wir automatisch eine schöne bunte Stadt. Jetzt haben wir eine Stadt mit leicht chaotisch positionierten Gebäuden. Die Liste der verfügbaren Operatoren bietet hier noch einen weiteren interessanten Kandidaten, den Rasterize Node. Damit können Objekte in einem definierbaren Raster verteilt werden. Fügen wir diesen Operator jedoch einfach unter der Position Surface hinzu, fallen wieder alle Gebäude auf die Null-Ebene zurück. Dem können wir aber abhelfen, indem wir davor noch einen Object Bind Surface Node setzen, die Landschaftsoberfläche als an den Node zu bindende Oberfläche definieren und ihn auf Lock to Surface stellen.

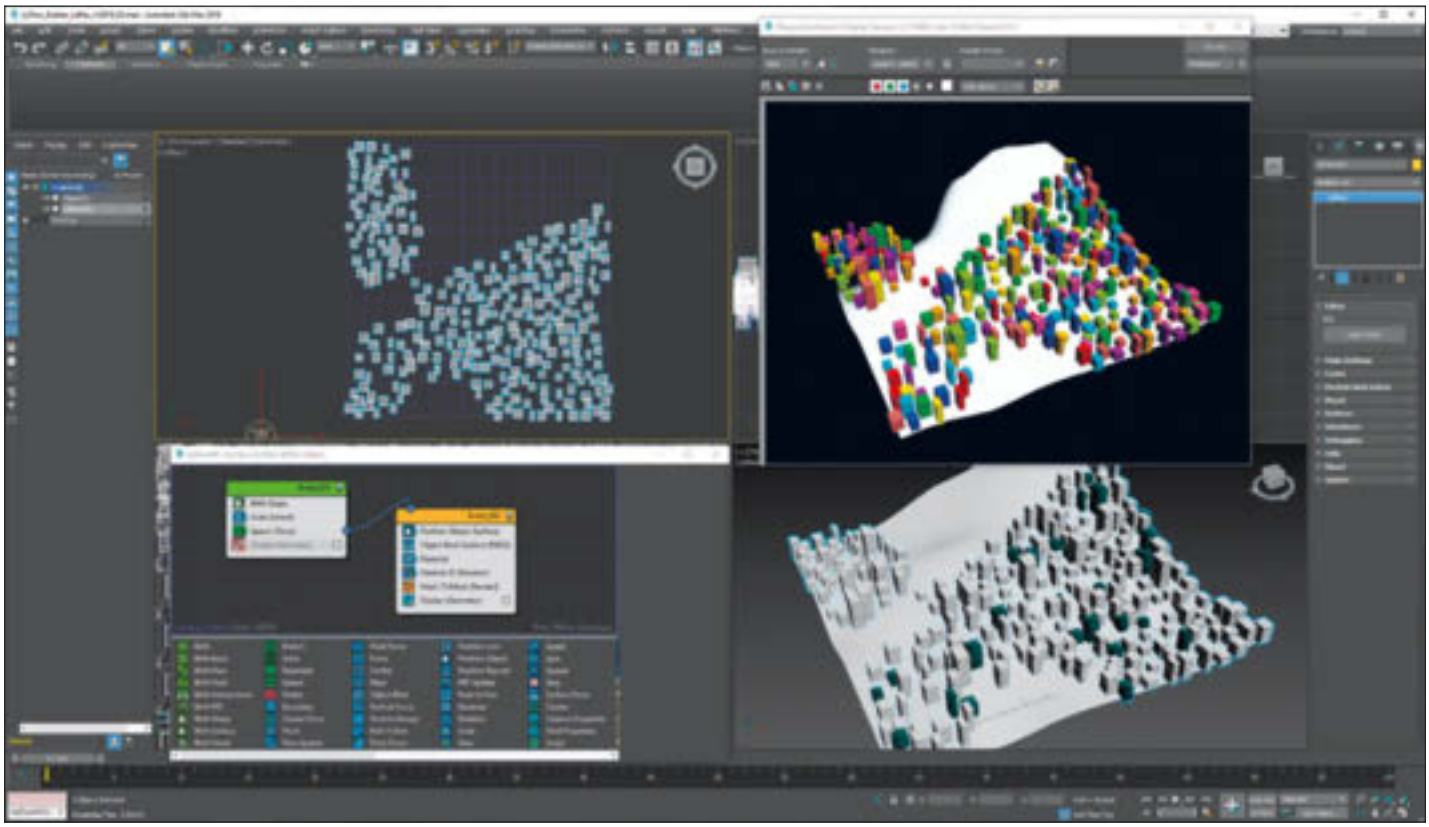
Jetzt haben wir die prinzipielle Verteilung, welche wir durch die Bitmap festgelegt haben, in der sich die Gebäude aber schön geordnet nebeneinander auf der Oberfläche verteilen. Weil ich auch V-Ray-Objekte verwenden kann, benötigt eine riesige Stadt dadurch kaum Speicherplatz in der Szene.

Particle vs. tyFlow

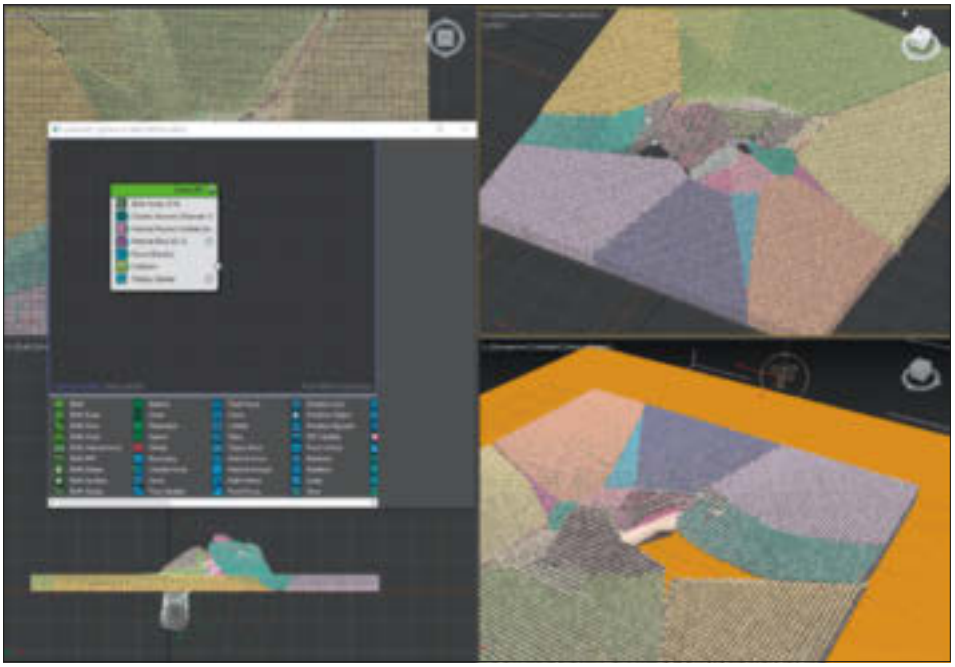
Wie viele der Bausteine von 3ds Max fing auch Particle Flow seine erste Existenz als ein Zusatzpaket eines externen Programmierers an. Es bestand aus verschiedenen Paketen, die man als Plug-in für 3ds Max hinzukaufen konnte. In 3ds Max 2009 wurde im Rahmen der damaligen Subscription Extension zuerst das Basis-Paket von Particle Flow integriert und ein paar Versionen später auch die Zusatzpakete 2 und 3. Eine der damit hinzugefügten Funktionen war der Data Flow Operator, mit dem sich über eine Nodes-Umgebung programmierte Funktionsoperatoren erstellen ließen.

Bei tyFlow kann man aus dem FAQ entnehmen, dass Tyson Ibele auf solch einen Data Operator verzichtet und stattdessen lieber einen stabilen Script Operator hinzugefügt hat. Jetzt mag Scripting nichts für jedermann sein, jedoch kann man in naher Zukunft bestimmt, wie auch beim normalen Max-Script, verschiedene Scripts finden, welche man frei herunterladen oder auch kaufen kann.

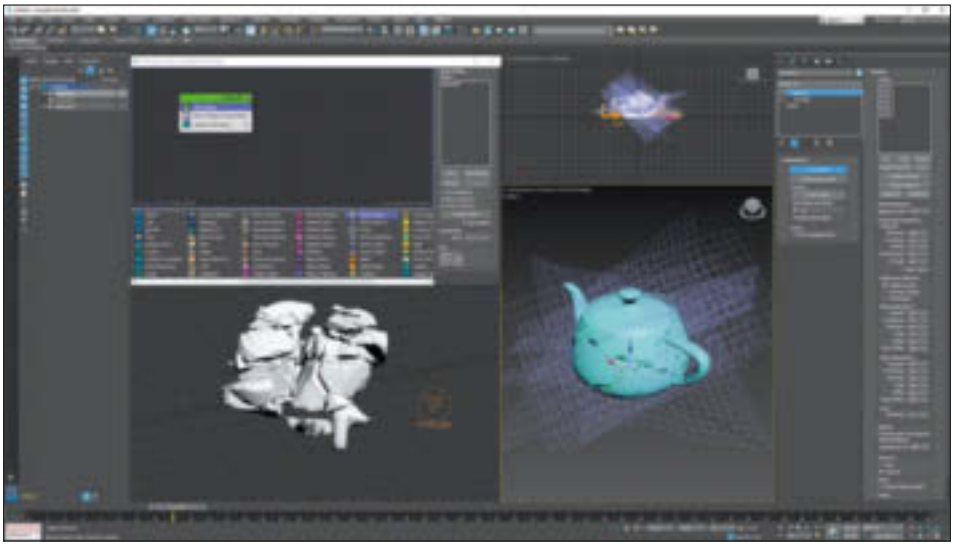
Jetzt sind wir zwar durch ein paar wenige Beispiele durchgegangen, haben aber gerade mal an der äußersten Hülle von tyFlow gekratzt. Alleine in den tyFlow-Operatoren liegen weitere Schätze wie ein granularer Solver für die Simulation von körnigen Systemen wie Sand oder Ähnlichem oder auch der Cloth Solvern und der Fluid Solvern. Neben



Natürlich lässt sich ein Partikelsystem auch zum Verteilen von Objekten nutzen.



Neben weiteren Operatoren für Stoff oder Flüssigkeiten gibt es auch einen für granulare Systeme.



Mit dem tyCarve-Modifikator können eigene Schnitte gezeichnet werden, um Objekte zu zerteilen.

Links die verwendeten Objekte. In der Mitte das Resultat mit tyBoolean. Rechts die Ergebnisse von 3ds Max Boolean Compound Object.

Mit dem tyBoolean-Modifikator kann eine Boolesche Verbindung mit einem anderen Objekt eingegangen werden. Aber man kann ja mehrere Modifikatoren auf ein Objekt legen.

dem tyParticleSkin-Modifikator kommen noch weitere, teils experimentelle Modifikatoren mit in diesem Paket. Der tyCarve-Modifikator erlaubt das freie Zeichnen von Linien über ein Objekt, welche dann die Schnitte definieren, mit denen das Modell zerteilt wird. Dafür kann ich mich frei im Viewport drehen und zeichne diese Schnitte immer parallel zu meiner jeweiligen Ansicht. Dieses zerteilte Objekt kann dann auch direkt in einem Flow genutzt werden.

Der tyBoolean-Modifikator ist ein experimenteller Modifikator, um Constructive-Solid-Geometry-Operationen an Meshobjekten vorzunehmen. In einfacheren Worten ausgedrückt, ermöglicht er das Zusammenfügen oder Subtrahieren unterschiedlicher Geometrien. Der Modifikator an sich ist sehr einfach. Er erlaubt eine Funktion mit einem Objekt. Ich kann also Objekte zusammenfügen, ein Objekt von einem anderen subtrahieren oder auch die Schnittmenge aus beiden Objekten erstellen. Hier habe ich einfache Formen erstellt und diese mal mit dem tyBoolean-Modifikator zusammengefügt und einmal mit der neuesten Version vom 3ds-Max-eigenen Boolean Compound Object. Dann habe ich diese Grundobjekte mehrfach dupliziert und vor der Boolean-Operation die Anzahl der Polygone erhöht.

Fazit

Es gibt in der Beta-Phase noch ein paar Ecken, an denen geschliffen werden muss, und es steht bislang auch noch kein Preis oder Release-Datum fest. Aber eigentlich will ich dieses neue Tool haben, sobald es den Markt betritt.

Es ist mir schon fast egal, ob ich es in einer meiner baldigen Produktionen nutzen würde. Allerdings bietet dieses schöne, neue Spielzeug so viele Möglichkeiten, dass es eine ganze Menge an Einsatzgebieten abdecken kann.

Für die nähere Zukunft würde ich mir eine Tutorial-Serie zu tyFlow in der Digital Production wünschen, welche unterschiedliche Möglichkeiten von tyFlow behandelt. Sehr gerne von unterschiedlichen Autoren, um den Lesern auch den Vorteil zu bieten, unterschiedliche Herangehensweisen kennenzulernen – in diesem Sinne fliegt der tyFlow-Staffel-Stab weiter an Rainer Duda!

› ei

Mike Kuhn hat die Prüfung zum 3ds Max Certified Trainer abgelegt, ist Autor und Entwickler mit mehr als 20 Jahren Erfahrung in Industrie-Visualisierung und verschiedensten Workflows. **www.in3.de**