2014
5
DIGITAL PRODUCTION

ISSN 1433-2620 > B 43362 >> 18. Jahrgang >>> www.digitalproduction.com
Published by ATEC

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

DIGITAL PRODUCTION

MAGAZIN FÜR DIGITALE MEDIENPRODUKTION

AUGUST 05|14

The Mill

Menagerien und Multiexporter vom UK-Vorzeigestudio

Destruction & Regen

Im Houdini-Workshop zum Wettergott werden

LED-Einstieg

Die Erleuchtung oder nur ein Glühwürmchen?

4 194336 214951 05

Particle Meshing mit OpenVDB. Links die Partikelsimulation, rechts das fertig gemeshte Ergebnis. Dazwischen wurden zahlreiche Volume-Filter von OpenVDB eingesetzt, um aus den recht spärlich verteilten Partikeln ein detailliertes Wasser-Mesh mit dünnen Strukturen zu erzeugen.

Was ist OpenVDB und was bringt's?

Derzeit wird die VFX-Welt mit zahlreichen Akronymen von Open-Source-Projekten der großen Hollywood-Studios überhäuft. Ein Beispiel ist OpenVDB von Dreamworks. Es steckt in Houdini und ein paar anderen Programmen und hat irgendetwas mit Volumen zu tun. Aber was genau kann man erwarten, wenn ein Progamm OpenVDB unterstützt?

von Gottfried Hofmann

Digital Production klärt auf und hat sich mit dem Chefentwickler Ken Museth auf der FMX 2014 in Stuttgart über die Technik und die Vor- und Nachteile des Open-Source-Ansatzes unterhalten.

Eine fast leere Schachtel und die Unendlichkeit

Rauch- und Feuersimulationen nutzen normalerweise eine sogenannte Domain. Darin finden die Berechnungen statt und dort werden später die Volumen-Daten in Voxeln – das sind dreidimensionale Pixel – gespeichert. Man muss sich das wie eine Schachtel vorstellen, in der die Simulation respektive später deren Ergebnis steckt.

Dabei gibt es aber zwei Probleme: Zum einen ist die Domain endlich. Manche Programme begegnen diesem Problem, indem die Domain bei Bedarf größer und kleiner gemacht wird. Das andere Problem besteht darin, dass in den meisten Fällen weite Bereiche der Domain leer bleiben.

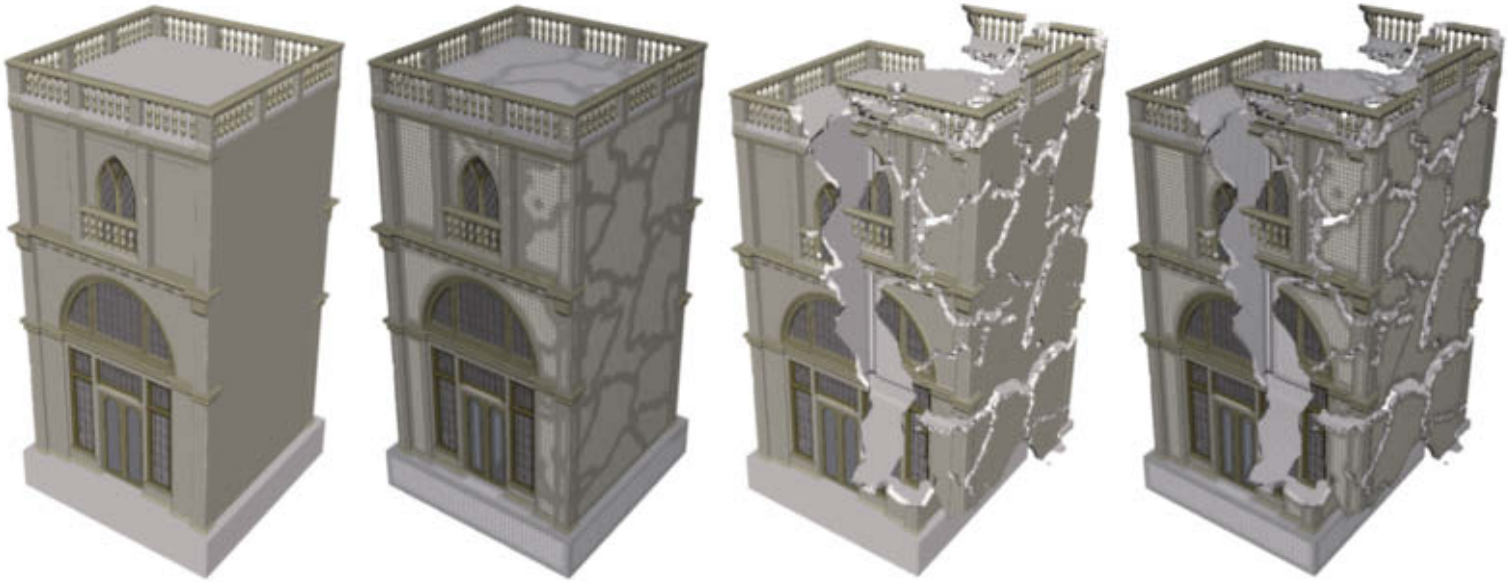
Eine runde Rauchsäule wird eine eckige Domain im besten Fall zu 75 Prozent ausfüllen. Weht ein laues Lüftchen und die Rauchsäule ist dadurch ein wenig gebogen, ist weniger als die Hälfte der Domain ausgefüllt, und je nach Biegung der Rauchsäule reduziert der Effekt den „Füllstand" schnell auf unter 10 Prozent. Oder nehmen wir als Beispiel eine runde Explosion in einer würfelförmigen Domain. Der Optimalfall wären hier 52 Prozent, aber das ist ja optimal und nicht real. Normalerweise wären auch in diesem Anwendungsfall weite Bereiche der Domain leer. Sprich: In der Schachtel ist eigentlich fast nichts drin – aber warum ist das ein Problem?

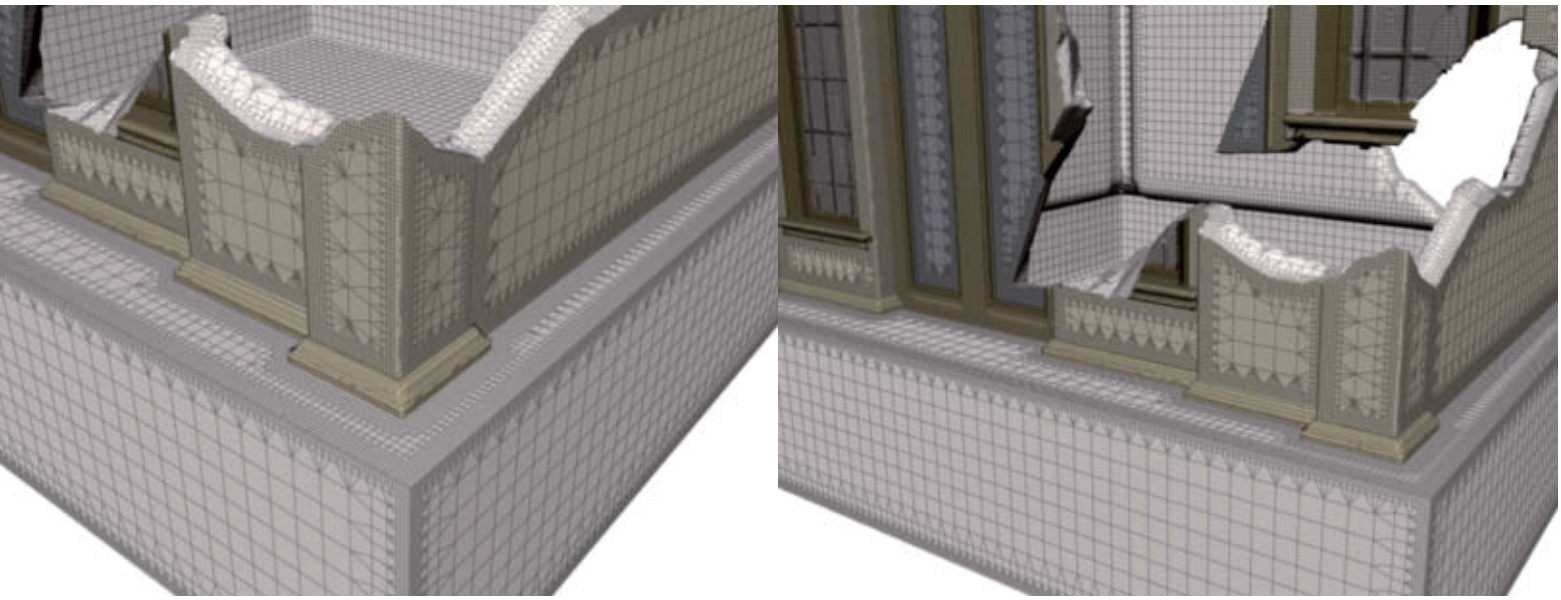
Langer Weg, umsonst gegangen

Erstens ist das ein Speicherproblem. Auch wenn die Daten auf der Festplatte komprimiert gespeichert werden – im Arbeitsspeicher sind sie nicht mehr komprimiert und jede Zelle der Domain braucht Platz, auch die leeren Zellen. Ein 2D-Vergleich würde ein Bild mit fast ausschließlich schwarzen Pixeln und ein paar spärlich verteilten Objekten zeigen. Wenn man es unkomprimiert speichert, braucht dieses Bild genauso viel Platz wie ein detailliertes Foto mit der gleichen Auflösung.

Es gibt aber noch einen weitaus gravierenderen Nachteil, wenn man mit spärlich besetzten Voxel-Daten auf die traditionelle Art arbeitet, nämlich beim Rendern. Moderne Path Tracer arbeiten mit Strahlen, die in eine Szene geschickt werden. Trifft ein Strahl

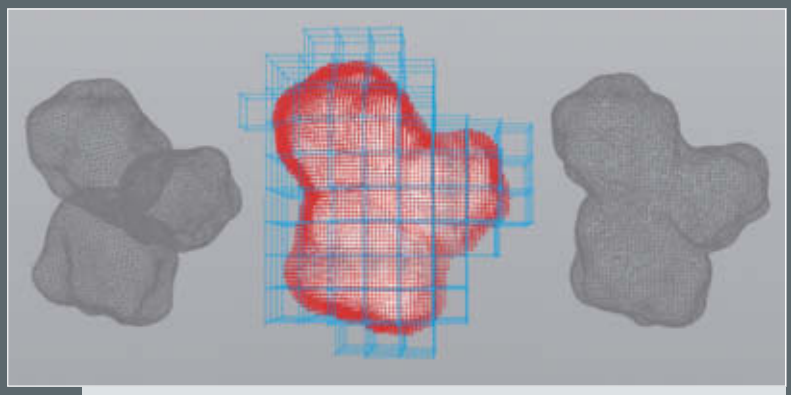
Mesh-Zerstörung. OpenVDB bringt Werkzeuge für volumetrische Zerstörung von Meshes mit. Die Bruchkanten erhalten eine detaillierte Geometrie.



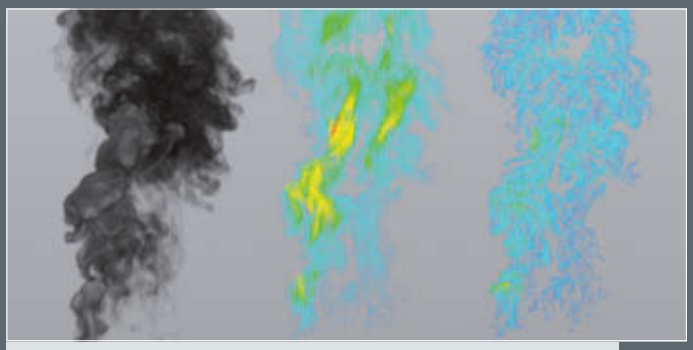


auf eine Oberfläche, wird eine Berechnung durchgeführt. Bei Volumen ist das nicht so einfach. Hier trifft der Strahl auf eine Domain und erfährt, dass alles, was folgt, Volumen ist, bis die Domain zu Ende ist.

Also dringt der Strahl in das Volumen ein. Nachdem er eine bestimmte Wegstrecke zurückgelegt hat, wird vereinfacht gesagt nachgesehen, ob an der Stelle zum Beispiel Rauch ist. In einem solchen Fall wird berechnet, ob der Strahl abgelenkt oder auf irgendeine Weise verändert wird oder einfach nur weiter eindringt. Gesetzt den Fall, dass ein Strahl eine Domain durchschritten hat, ohne dabei je auf Rauch gestoßen zu



Mesh-Reparatur. Links ein Mesh mit Löchern, mehreren Eigen-Überschneidungen und anderen Problemen. OpenVDB erzeugt daraus ein Volumen (Mitte). Danach kann es gemesht werden, um eine saubere Geometrie zu erhalten.



Metadaten. OpenVDB kann mehr speichern als einfach nur Volumen-Daten. In der Mitte die Geschwindigkeit, mit der sich der Rauch ausbreitet, und rechts die Wirbelstromstärke, jeweils aus einem zweidimensionalen Ausschnitt. Links die Dichte des Auftriebs, hellere Bereiche drängen stärker nach oben.

sein, dann ist er den ganzen Weg umsonst gegangen. Bis dahin hat er aber möglicherweise Hunderte Lookup-Operationen durchgeführt. Verschwendete Rechenzeit. Und die Person vor dem Bildschirm ärgert sich vermutlich, weil das Rendern so lange dauert.

OpenVDB schlägt beide Fliegen mit einer Klappe, indem eine Datenstruktur genutzt wird, die man sich vereinfacht als Baum aus Micro-Domains vorstellen kann. Dieser kann beliebig groß werden und hinterlässt einen deutlich kleineren Fußabdruck im Speicher. In Bereichen, in denen kein Rauch ist, kommen Raytracing-Strahlen gar nicht erst auf Idee, dass dort überhaupt welcher sein könnte. Es verwundert also nicht, dass ausgerechnet die Path Tracing Engine Arnold zum Rendern von Volumen auf OpenVDB zurückgreift. Kurz: OpenVDB ermöglicht größere Simulationen bei höherer Auflösung, die schneller gerendert werden können.

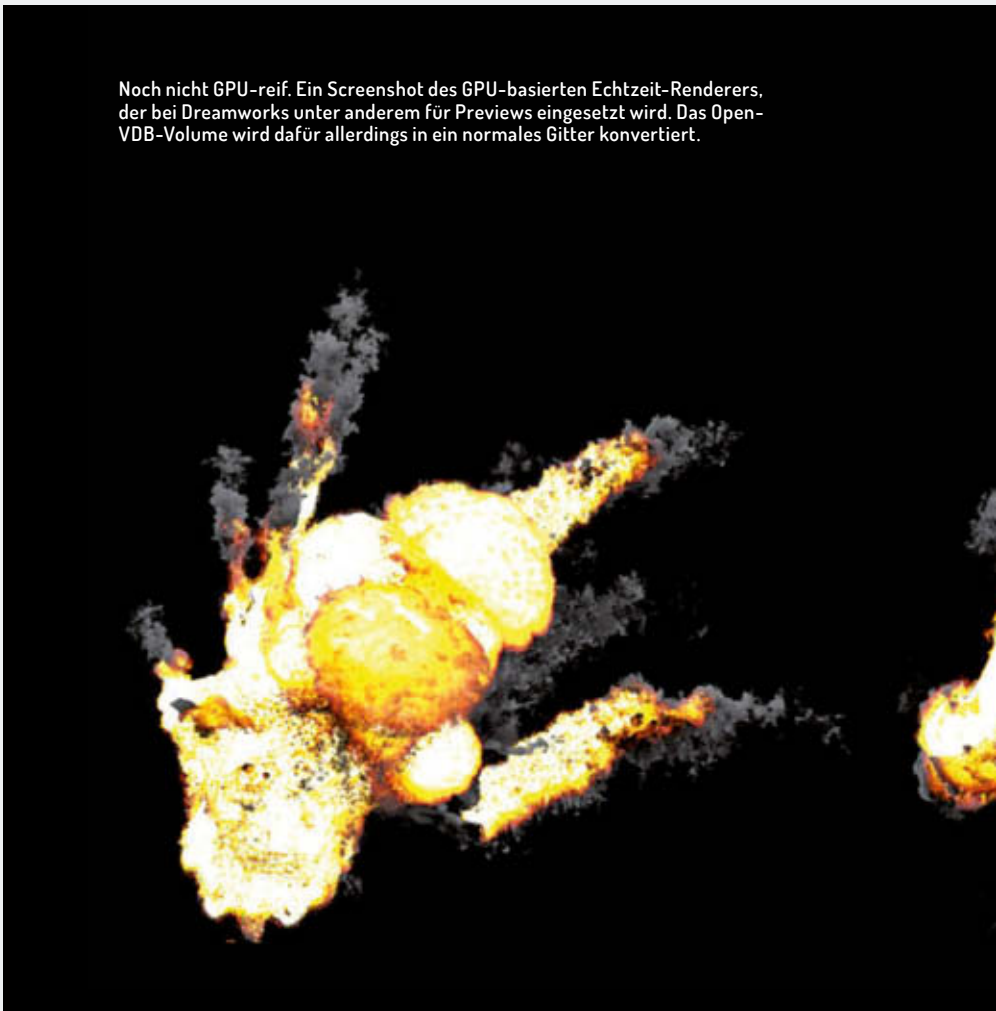
Compositing-Toolbox für Volumen-Daten

OpenVDB ist aber nicht nur eine effiziente Datenstruktur, sondern bringt auch eine Reihe von Werkzeugen mit, um diese Daten zu bearbeiten. Beispiele sind hier der berühmte Gaußsche Weichzeichner, Dilade und Erode, Boolsche Operationen und diverse Operationen zum Mischen von Volumen. Einfach gesagt: OpenVDB erlaubt Compositing mit Volumen-Daten. Mit seinem Node-basierten Bedienkonzept war Houdini natürlich prädestiniert dafür, OpenVDB von Anfang an zu integrieren. Ein Mesher und Konvertierwerkzeuge sind auch mit von der Partie. Der Nutzer bekommt dadurch zum Beispiel die Möglichkeit an die Hand, eine partikelbasierte Flüssigkeitssimulation in ein VDB-Set zu konvertieren, diese dann mit den diversen Compositing-Werkzeugen zu bearbeiten und zu guter Letzt mit einem Mesh zu überziehen. Das erlaubt eine große Freiheit, das Aussehen der Flüssigkeit zu beeinflussen, ohne immer wieder neu simulieren zu müssen. Auch Meshes können mit OpenVDB zu Volumen konvertiert werden. Das Ergebnis kann dann zum Beispiel für volumetrisches Fracturing verwendet und danach zurückkonvertiert werden. Das ist aber noch nicht alles. OpenVDB liefert zum Beispiel auch Werkzeuge zum Sculpten von Volumen, was sich beispielsweise für das Erstellen von Wolkenformationen nutzen lässt.

In welchen Tools steckt eigentlich OpenVDB?

Insgesamt ist das Projekt sehr groß. Durch die Open-Source-Natur steht es jedem Hersteller frei, es in seine eigenen Programme zu integrieren, doch das erfordert einiges an Arbeit.

Perfekt eingebunden ist OpenVDB nur in Houdini. Arnold nutzt es zum schnellen Rendern von Volumen-Effekten via Plug-in, auch Pixars Renderman kann mit OpenVDB umgehen. Teile von OpenVDB finden sich in Realflow wieder. Für Maya hat Dreamworks einige Nodes für die Arbeit mit OpenVDB publiziert und in Blender wird seit Mitte letzten Jahres an der Integration von OpenVDB gearbeitet. Die Ergebnisse sind aber noch nicht in den Hauptzweig eingeflossen.



Noch nicht GPU-reif. Ein Screenshot des GPU-basierten Echtzeit-Renderers, der bei Dreamworks unter anderem für Previews eingesetzt wird. Das OpenVDB-Volume wird dafür allerdings in ein normales Gitter konvertiert.

Interview mit Ken Museth von Dreamworks Animation

Auf der FMX 2014 haben wir die Gunst der Stunde genutzt und uns mit Ken Museth unterhalten, dem Chef der R&D-Abteilung für VFX bei Dreamworks Animation und Erfinder von OpenVDB.

DP: Hallo Ken, derzeit arbeitest du bei Dreamworks Animation. Was genau machst du dort, an welchen Projekten arbeitest du und wie sieht deine Beteiligung an OpenVDB aus?

Ken Museth: Ich arbeite seit 2009 bei Dreamworks Animation in der Gegend um Los Angeles. Ich bin der Leiter der Abteilung Forschung und Entwicklung für VFX und leitender Ingenieur. Ich leite ein Team von Forschern, die Werkzeuge für die Erstellung von Animationen entwickeln. Ich bin aber auch selbst ein Entwickler. Wie ich an VDB beteiligt bin? Nun, ich habe VDB im Jahr 2009 erfunden und die Ergebnisse publiziert. Seither wurde es Stück für Stück verbessert und 2011 wurde es zur Siggraph, der weltgrößten Konferenz für Computergrafik, veröffentlicht. Meine Gruppe arbeitet hauptsächlich daran, OpenVDB instand zu halten, zu verbessern und in regelmäßigen Abständen neue Versionen herauszubringen.

DP: Aus der Sicht eines normalen Nutzers – was kann man erwarten, wenn eine Applikation OpenVDB unterstützt?

Ken Museth: Lass uns das im Kontext der Art von Volumen betrachten, die seit vielen Jahren genutzt werden, auch in Software von Drittanbietern. Das sind normalerweise sogenannte „dichte Gitter" respektive „Dense Grids". Dabei hat der Nutzer eine Art Schachtel und die Effekte sind darauf beschränkt. Außerhalb der Schachtel finden keine Simulation und kein Rendering statt. Die Größe der Schachtel hat starken Einfluss auf die Menge an Speicher, die der Computer dafür braucht. Je mehr man die Auflösung der Schachtel vergrößert, desto mehr Voxel finden sich darin. Voxel bezeichnen Pixel mit drei statt zwei Dimensionen. Das Problem ist also, dass diese Dense Grids eine große Menge an Speicher benötigen. Was VDB erlaubt, ist, diese Schachtel wegzuwerfen. Es erlaubt dir, Informationen auf eine Art und Weise zu speichern, die sehr kompakt ist. Für den Endbenutzer bedeutet dies höhere Auflösung und mehr Details bei Simulationen wie Rauch, Feuer oder Wasser. Für viele volumetrische Anwendungen, besonders wenn sie dünn besetzt sind, ist VDB eine sehr gute Option. Seit 2012 ist OpenVDB in Houdini integriert. Wer Version 12.5 oder höher von Houdini einsetzt, verwendet intern OpenVDB. Die meisten Effekte, die in der VFX-Abteilung erstellt werden, nutzen auf die eine oder andere Weise Volumen. Wir haben typische volumetrische Effekte wie animierten Staub, Rauch, Wasser, Feuer und sogar die Zerstörung von Objekten. Viele Effekte, die von außen gar nicht danach aussehen, können Volumen beinhalten. Dazu zählen Rigid Body Dynamics und wie schon erwähnt die Zerstörung von Objekten. Volumen lassen sich dazu nutzen, um zu verhindern, dass sich Objekte überschneiden. Dazu zählen Techniken zur volumetrischen Kollisionserkennung und -auflösung. Ein anderes Beispiel sind Rendering-Applikationen, die ebenfalls an vielen Stellen Volumen einsetzen. Die VFX-Abteilung benötigt also Volumen in vielen Bereichen.

DP: OpenVDB wird von zahlreichen Software-Herstellern unterstützt …

Ken Museth: Arnold hat OpenVDB-Unterstützung über ein Plug-in hinzugefügt, es kann auch in Pixars Renderman eingesetzt werden. Realflow verwendet intern OpenVDB. Es wird von vielen Programmen einge-

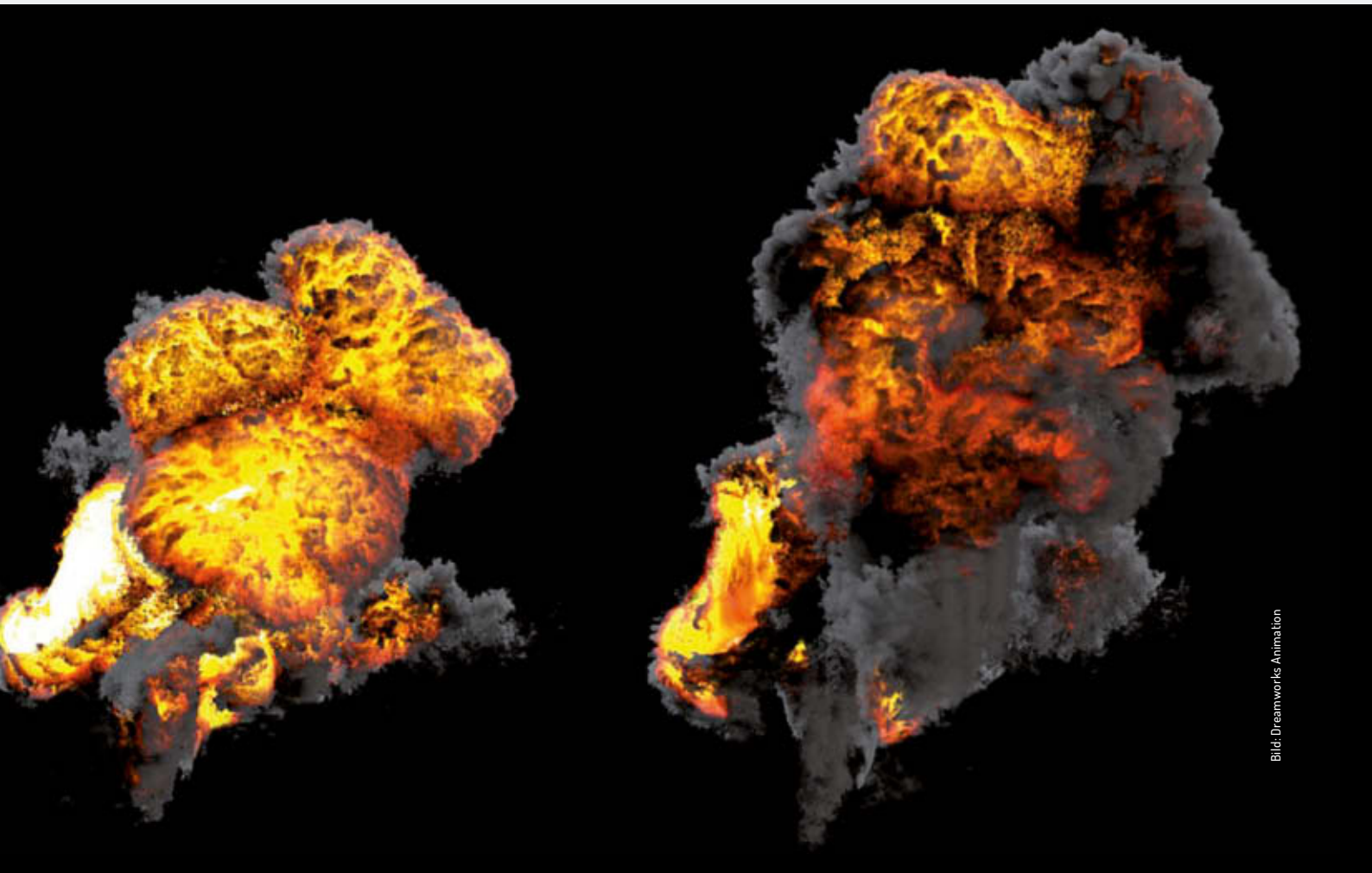


Bild: Dreamworks Animation

setzt. Das war einer der Hauptgründe, warum Dreamsworks VDB als Open Source herausbrachte: damit Fremdfirmen es für uns in ihre Produkte integrieren. Das gibt uns mehr Zeit für andere Entwicklungen. Man glaubt gar nicht, wie viel Zeit wir normalerweise damit verbringen, Werkzeuge in unsere Pipeline zu integrieren. Dazu gehören die Entwicklung von Benutzeroberflächen, spezielle Skripte und andere Arten von Support-Arbeiten. Diese Aufgaben benötigen sehr viel Zeit. Indem andere Firmen diese Arbeit machen, können wir andere Entwicklungsaufgaben übernehmen, wie zum Beispiel OpenVDB voranbringen. Dadurch, dass SideFX OpenVDB in Houdini integriert hat und nicht wir, ist bei uns zum Beispiel sehr viel Zeit frei geworden, die wir sogleich für die weitere Entwicklung genutzt haben.

DP: Das ist einer der Vorteile, ein Produkt als Open Source herauszubringen. Gibt es auch Nachteile?
Ken Museth: Einer der Nachteile ist, dass es äußerst zeitintensiv ist. Wir verbringen viel Zeit damit, die Softwarequellen zu verwalten, Build-Systeme zu pflegen, Dokumentation zu schreiben et cetera. Es gibt auch einiges an Druck aus der Open-Source-Gemeinschaft. Wir bekommen häufig Anfragen nach neuen Funktionen. Eine häufig gestellte Frage ist zum Beispiel: „Warum unterstützt ihr kein Windows?" Solche Anfragen muss man immer damit vergleichen, was Dreamworks eigentlich will. Wir setzen überhaupt kein Windows ein, sondern nutzen Linux-Maschinen. Also haben wir auch kein Interesse daran, Windows als Plattform zu unterstützen. In diesem Fall hatten wir aber Glück, da SideFX Windows unterstützt und sie sich darum kümmern. Mit diesem Druck umgehen zu müssen sowie die viele Zeit, die dafür draufgeht, sind für uns zwei der größten Nachteile an einem Open-Source-Modell. Aber die werden von den Vorteilen mehr als ausgeglichen. Der größte ist, wie bereits erwähnt, dass OpenVDB an allerlei Stellen eingesetzt wird, wir uns aber nicht um die Integration in all diese Werkzeuge kümmern müssen und unsere Zeit für andere Entwicklungsaufgaben verwenden können. Ein anderer Vorteil liegt in der Natur von Open Source an sich. Es ist einfach ein sehr guter Weg, um mit anderen Filmstudios und Leuten aus dem akademischen Bereich zusammenzuarbeiten. Wir haben da ein Stück Software ohne Probleme und Hürden mit geistigem Eigentum, das jeder nutzen und an dem jeder mitarbeiten kann. Manchmal nehmen Forschungsgruppen an Universitäten OpenVDB als Grundlage für ihre Forschungsarbeiten und manche ihrer Ideen und Resultate können zurück in OpenVBD fließen.

DP: Der Ingenieur eines anderen Open-Source-Projekts aus dem VFX-Umfeld sagte, dass Open Source nicht wirklich billiger sei als herkömmliche Inhouse-Entwicklung, aber auch nicht teurer. Was ist dein Bauchgefühl?
Ken Museth: Da würde ich ihm zustimmen, es ist nicht teurer. Aber viel wichtiger: Es hebt die Qualität! Der Code wird so einfach viel besser, als wenn man nur für sich selbst entwickelt. Du teilst ihn mit der gesamten Welt, jeder kann ihn sich ansehen. Das ist ein riesiger Ansporn, die Qualität sowohl des Quellcodes als auch der Dokumentation so gut wie nur möglich zu halten. Und das zahlt sich später aus, denn es ist viel einfacher, ein Projekt instand zu halten, das gut dokumentiert ist und eine saubere API hat.

> »Code und Dokumentation werden besser, wenn man sie mit der Welt teilt.«
>
> Ken Museth
> Chef-Ingenieur bei DreamWorks Animation, Erfinder von OpenVDB

DP: Das ist genau das, was uns Larry Gritz gesagt hat, als er mit uns über seine Erfahrungen mit der Öffnung des Open-Shading-Language-Projekts gesprochen hat. Er hat auch noch angemerkt, dass die Veröffentlichung als Open Source die Entwickler vor ein Publikum stelle. Ist das eher gut oder eher schlecht, wenn man deswegen viele E-Mails bekommt?
Ken Museth: Genau so sieht es aus! Aber um ehrlich zu sein: Als wir das Projekt angefangen haben, wussten wir gar nicht, worauf wir uns da einlassen. Es war eine Art Überraschung für uns, aber eine wirklich gute. Wir haben Entwickler bei Dreamworks, die haben sich unseren Code angesehen und meinten, das wäre das sauberste Stück Software, das sie im gesamten Studio je zu Gesicht bekommen hätten. Und das ist genau das, was wir erreichen wollten.

DP: Bekommt ihr ansonsten auch viel Lob oder eher Kritik?
Ken Museth: Ich denke, eines der größten Probleme mit OpenVDB ist, dass es sehr viel „Template Metaprogramming" nutzt. Es ist gut dokumentiert, aber wenn man diese Art des Programmierens nicht gewohnt ist, wird die Einarbeitung in den Quellcode ziemlich schwierig. Und wenn man dann noch bedenkt, dass VDB an sich, also die Datenstruktur, auch ein wenig kompliziert ist, verwundert es nicht, dass wir nur von recht wenigen Leuten Beiträge zum Kern der Bibliothek bekommen haben. Ich glaube, das liegt daran, dass die Interna einfach ein wenig einschüchternd sein können. Es ist nicht einfach, anzufangen und mit der Datenstruktur zu arbeiten. Die meisten Beiträge, die wir bekommen haben, sind Werkzeuge, die auf der Datenstruktur aufsetzen, was natürlich auch sehr nützlich ist.

DP: War es schwierig, OpenVDB als Open Source zu veröffentlichen?
Ken Museth: Mit Sicherheit! OpenVDB war das erste und bisher einzige Open-Source-Projekt von Dreamworks Animation. Es hat fast zwei Jahre gedauert, um die Erlaubnis zu erhalten. Die meisten Leute haben uns sehr in dieser Zeit unterstützt, sodass wir mit diesem Projekt zu keinem Zeitpunkt an die Wand gefahren sind. Aber die Rechtsabteilung hat sehr lange gebraucht, bis sie ihr Okay gab. Die Unterstützung durch Drittanbieter war das wichtigste Verkaufsargument. Die Tatsache, dass wir von SideFX unterstützt wurden, hat auch sehr geholfen.

DP: SideFX war von Anfang an dabei?
Ken Museth: Ja, wir haben eng mit Entwicklern von SideFX zusammengearbeitet. Sie haben sich den Code angesehen und Änderungsvorschläge unterbreitet.

DP: Welche Software von Drittanbietern sollte deiner Meinung nach OpenVDB einsetzen?
Ken Museth: Wir würden OpenVDB zum Beispiel gerne in Maya und Blender sehen. Das sind die beiden wichtigsten Drittanbieterprojekte, in denen wir uns OpenVDB wünschen. Wir haben eine grundsätzliche Unterstützung für Maya, da haben wir sogar ein paar Nodes veröffentlicht. Aber bei uns wird Maya nur wenig für volumetrische Effekte eingesetzt, darum wünschen wir uns, dass jemand aus der Open-Source-Gemeinschaft die Integration in Maya übernimmt.

DP: Plant ihr GPU-Unterstützung?
Ken Museth: Das ist eine sehr gute Idee. Wir haben das noch nicht und es ist auch nicht so ganz klar, wie wir das erreichen können. OpenVDB ist stark für normale Prozessoren optimiert und Grafikkarten haben eine komplett andere Architektur. Es ist keine leichte Aufgabe, aber es wäre auf jeden Fall ein sehr interessanter Weg, den wir da gehen würden. Wir stehen bereits in Kontakt mit Nvidia. Dort wurde uns zurückgemeldet, dass sie erste Erfolge mit der Portierung auf Cuda haben. Ich vermute aber, dass wir einige Änderungen durchführen müssten. Wir haben inhouse einen GPU-basierten Renderer, der grundsätzlich mit OpenVDB umgehen kann. Aber der wandelt es einfach nur in eine herkömmliche Gitterstruktur um, bevor die Daten in den Speicher der Grafikkarte geladen werden. Die kurze Antwort ist: GPU-Unterstützung ist etwas, was wir uns näher ansehen wollen.

DP: Welche anderen Pläne habt ihr? Vielleicht eine Datenstruktur zum Speichern von Partikeln?

Ken Museth: Ja, Partikeldaten sind ein großes Problem, an dem wir derzeit arbeiten. Während der Arbeiten am kommenden „Drachenzähmen leicht gemacht 2" standen wir häufiger vor dem Problem, riesige Mengen an Partikeldaten in VDBs zu konvertieren. Wir reden hier von Bereichen von bis zu 300 Millionen Partikeln. Da haben wir gemerkt, dass existierende Werkzeuge nicht ausreichen. Also haben wir angefangen nach Wegen zu suchen, die Partikelinformationen in denselben Baumstrukturen zu speichern, in denen normalerweise die VDB-Daten abgelegt werden. Damit konnten wir einige Erfolge vorweisen. Wir hoffen, dieses Jahr im Sommer zur Siggraph OpenVDB 3.0 vorstellen zu können. Das Speichern von Partikeldaten wäre ein Teil davon. Es handelt sich auch um ein interessantes Projekt, da es das erste Mal ist, dass wir eine Kooperation mit einem anderen Studio eingehen. Ein Forscher von Double Negative wird mit uns daran arbeiten. Wir denken aber auch an Level of Detail (LoD). Wir suchen nach Wegen, unterschiedliche Auflösungen der Volumendaten zu speichern, damit ein Renderer anhand des Kameraabstands leicht die richtige Auflösung aussuchen kann. Wir wollen auch Werkzeuge zur Modellierung von Wolken haben und arbeiten an Out-of-Core-Loading. Dann wird nicht das gesamte Volumen beim Rendern geladen. Vielmehr soll OpenVDB das dynamische Nachladen managen, wenn Strahlen von der Kamera unterschiedliche Bereiche in der Szene treffen, und nur diese Bereiche laden.

DP: Bist du der Ansicht, dass Rendern auf der GPU in Hinblick auf die Speichervoraussetzungen überhaupt brauchbar sein wird?

Ken Museth: Die Volumen, mit denen wir hantieren, haben normalerweise kaum mehr als ein paar Gigabytes. Solange die Grafikkarte so viel RAM hat, sollte es möglich sein.

DP: Und wie sieht es mit verteiltem Rendern über das Netzwerk aus?

Ken Museth: Darüber haben wir nachgedacht, aber das ist nichts, was wir bei Dreamworks normalerweise machen. Wir haben natürlich eine Renderfarm, aber jeder Node bekommt eine vollständige Kopie der Szene. Ich denke, OpenVDB könnte diese Form des Renderns gut handhaben, da es ein „Tile Grid" ist. Die Blattknoten könnten auf viele verschiedene Computer verteilt werden. Ich habe gehört, dass es ein Studio gibt, das diese Idee tatsächlich verfolgt, und zwar für Flüssigkeitssimulationen. Es hat bestätigt, dass es funktioniert.

DP: Gibt es weitere Open-Source-Kandidaten bei Dreamworks?

Ken Museth: Wir haben damals über einige Projekte gesprochen, aber ich denke, da ist nichts weiter in der Pipeline. Wir hatten uns auch überlegt, vor OpenVDB ein anderes Projekt als Testballon zu veröffentlichen. Momentan weiß ich aber von keinem Open-Source-Projekt, das ansteht. Ich wäre aber nicht überrascht, wenn doch eines herauskommt, wenn man bedenkt, was für einen Erfolg wir mit OpenVDB hatten. **› ei**

Gottfried Hofmann hat an der FAU Erlangen-Nürnberg Informatik studiert. Er arbeitet als Freelancer im VFX-Bereich sowie als Trainer für die freie 3D-Software Blender. Als freischaffender Autor schreibt er für Fach- und Computerzeitschriften. Er hat zahlreiche Blender-Tutorials verfasst, unter anderem 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.