2014
1
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

JANUAR | FEBRUAR 01|14

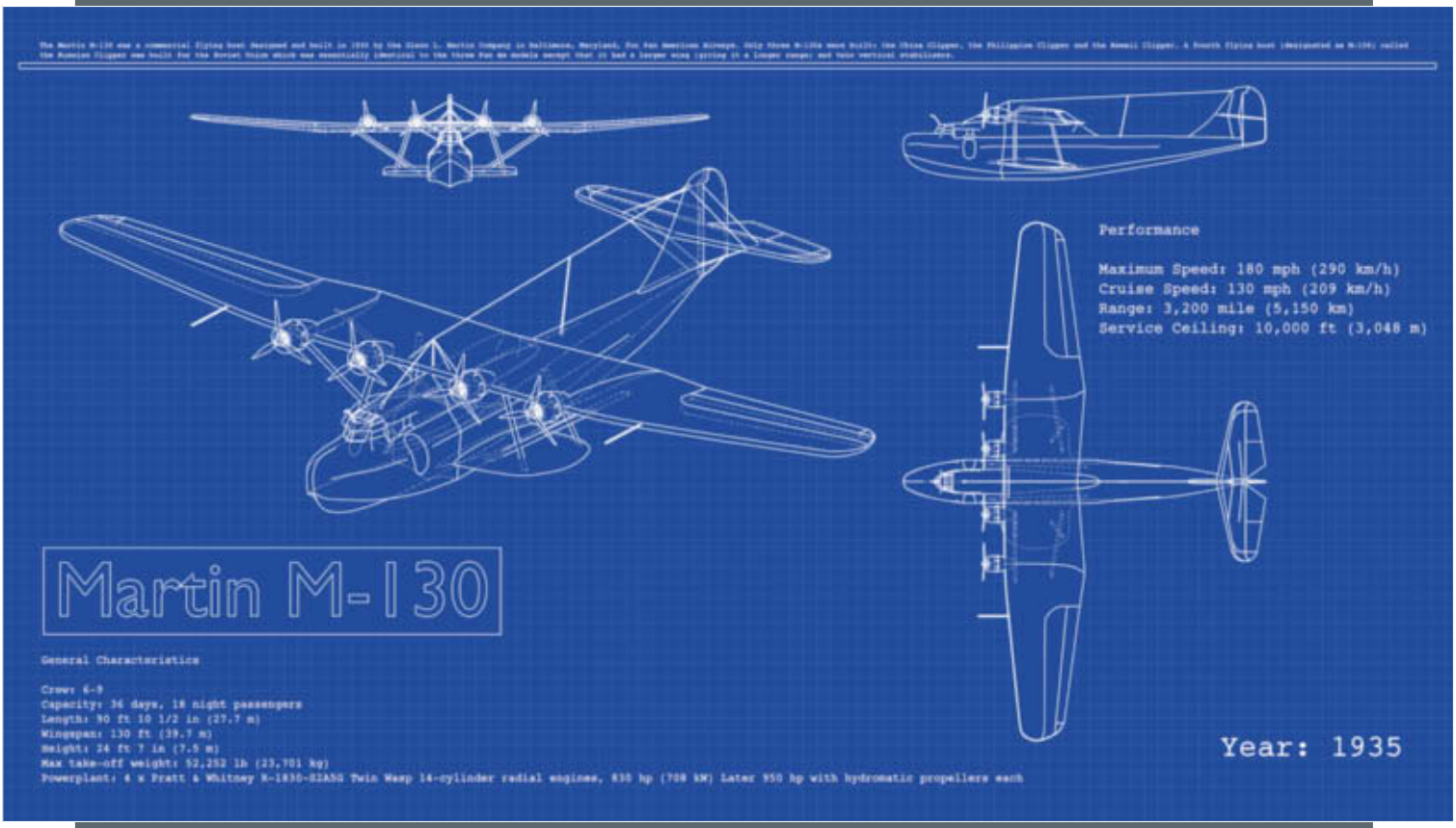
Motion Capture
Von der Kinect bis zum Königsweg bei Weta Digital

Desolation of Smaug
Peter Jackson gönnt dem Kleinen keine Ruhe

Farbecht
Kalibrierung im Test, von günstig bis großartig

4 194336 214951 01

Freestyle & Toon Shader rendern flott. Bei diesem Freestyle-Beispiel wird zuerst der Text gerendert, später werden die Freestyle-Linien von dem 3D-Modell hinzugefügt.



Freestyle

NPR Rendering in Blender

Für den Bereich fotorealistisches Rendering bringt Blender seit längerer Zeit die Pathtracing Engine Cycles mit. Für den nicht fotorealistischen (NPR) Bereich, wie beispielsweise Cartoons, musste bisher auf den zweiten mitgelieferten Renderer, Blender Internal, ausgewichen werden. Dieser ist aber inzwischen schon arg in die Jahre gekommen und bot für den Zeichnungsstil nur begrenzte Möglichkeiten. Seit Version 2.67 ist in Blender das Render-Add-on Freestyle integriert, das ausschließlich Linien rendert und dabei eine große Bandbreite an Konfigurationsmöglichkeiten besitzt.

von Gottfried Hofmann

Freestyle wird als Zusatz für den internen Renderer angeboten, kann aber dank des Compositors in Blender auch mit Cycles und anderen Render Engines kombiniert werden.

The Story of (Free-)Style

Die Entwicklung von Freestyle geht zurück auf eine wissenschaftliche Veröffentlichung im Jahr 2004. Ein Jahr später wurde ein Open-Source-Projekt ins Leben gerufen, um die in dem Paper beschriebenen Methoden in einer eigenständigen Render Engine zu implementieren. Im Jahr 2007 schlugen dann zwei Entwickler auf der Mailing-Liste von Blender vor, Freestyle in die 3D-Suite zu integrieren. Das Projekt wurde im Jahr 2008 im Rahmen des „Google Summer of Code" gestartet und wird seither in einer separaten Blender-Version, einem sogenannten Entwicklungszweig oder „Branch", geführt.

Während der Entwicklung ging die normale Entwicklung von Blender natürlich weiter und so kam es dazu, dass die Freestyle-Entwickler ihr Werkzeug zweimal mehr oder minder komplett überarbeiten mussten. Das erste Mal nach der Veröffentlichung von Blender 2.50, das zweite Mal nach der Integration von „BMesh", welches das in die Jahre gekommene Mesh-System in Blender komplett durch moderne Technologie ersetzte.

Im Jahr 2013, neun Jahre nach der Publikation des Papers und fünf Jahre nach Beginn der Integrationsarbeiten, wurde Freestyle endlich in den Hauptzweig von Blender aufgenommen und steht seit Version 2.67 jedem Blender-Nutzer zur Verfügung.

Freestyle, was ist das eigentlich?

Salopp gesagt, ist Freestyle ein Werkzeug, um Linien auf Basis der Kanten und Konturen von 3D-Objekten zu zeichnen. Das passiert, nachdem das eigentliche Rendering beendet ist und kann damit als Post-Processing-Effekt betrachtet werden. Sprich, wenn man ein cartoon-ähnliches Rendering wünscht, dann braucht man neben Freestyle noch einen Toon-Shader.

Seit Version 2.68 bringen sowohl der Blender-interne Renderer als auch Cycles einen solchen mit. Wenn man nur eine Strichzeichnung benötigt, dann kann man den ersten Schritt weglassen und ausschließlich Freestyle-Linien rendern.

Derzeit werden die Linien als Bitmaps ausgegeben, für die Zukunft ist aber auch ein Export als Vektorgrafik im SVG-Format geplant.

Anwendungsbeispiele

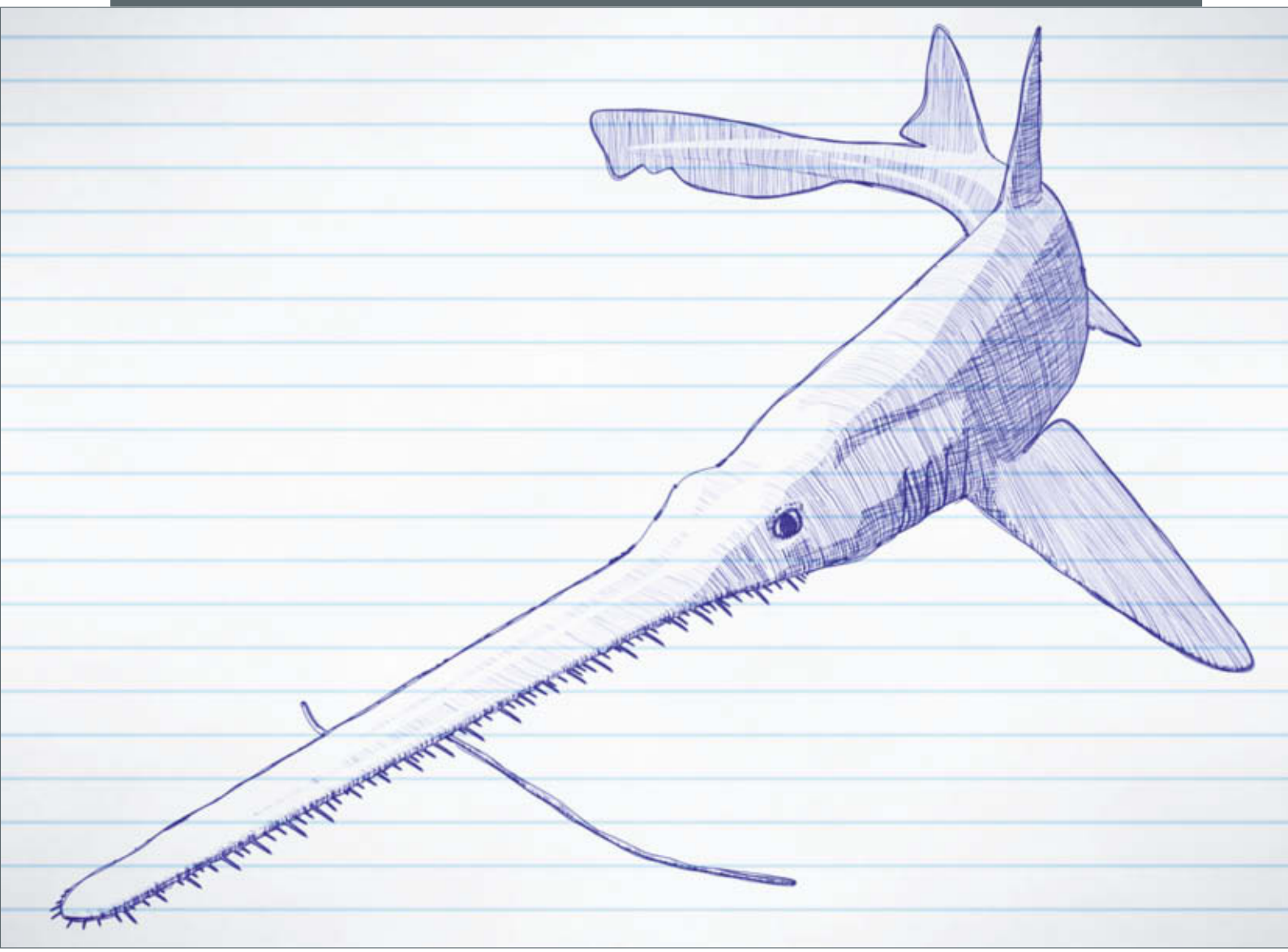
Bei den Suzanne Awards, dem Kurzfilmprogramm der jährlichen Blender-Konferenz in Amsterdam, waren schon im Jahr 2012 zahlreiche Einreichungen zu sehen, die mithilfe von Freestyle erstellt wurden. Das lag wohl unter anderem daran, dass das Rendering von Freestyle in Verbindung mit Toon-Shadern und eventuellem Postprocessing im Compositor ausgesprochen schnell vonstatten geht. Die möglichen Anwendungen sind jedenfalls vielfältig, einige Beispiele sollen hier vorgestellt werden.

Am naheliegendsten ist es natürlich, zusammen mit einem Toon-Shader Freestyle für digitalen Zeichentrick einzusetzen. Hier werden einige Möglichkeiten geboten, verschiedene Stile zu erzeugen. Die Linien können zum Beispiel so gerendert werden, als wären sie mit einem typischen Kalligrafie-Federkiel gezeichnet; hierbei ändert sich die Dicke abhängig vom Winkel. Oder als hätte die Hand des Zeichners leicht gezittert, ein Stil, der vor allem durch die späten Peanuts-Comics große Beliebtheit erlangte.

Für Zeichnungen stehen einige Werkzeuge bereit. So lassen sich mehrere Linien leicht versetzt mit variierender Dicke und Deckkraft erzeugen, als hätte der Zeichner mehrere Striche gezogen. Auch lässt sich einstellen, dass die Linien jeweils etwas länger als die eigentliche Kante gezeichnet werden, was ebenfalls der Glaubwürdigkeit als Zeichnung zugute kommt. Selbst die Kästen, mit denen Zeichner vor dem eigentlichen Werk die Proportionen abstimmen, lassen sich mit Freestyle erzeugen.

Eine weitere Anwendungsmöglichkeit ist, pseudo-technische Zeichnungen aus 3D-Modellen zu erzeugen. Diese lassen sich zum Beispiel für Präsentationen nutzen. Das 3D-

Die schraffierten Oberflächen bei diesem Beispiel von Charblaze sind durch Texturen erzeugt, Freestyle ist für die Kantenlinien und Konturen verantwortlich.



Interview mit Tamito Kajiyama

Digital Production sprach mit Tamito Kajiyama, dem Freestyle-Entwickler, der die Integration in Blender bisher am weitesten begleitet hat. Seit 2009 ist er der Hauptverantwortliche für das Projekt.

DP: **Hallo Tamito, erzähl uns doch ein wenig über dich und deine Rolle beim Projekt „Freestyle-Integration".**
Tamito Kajiyama: Ich bin ein Amateur-Softwareentwickler und engagiere mich schon seit Jahren im Bereich Open Source und freie Software. Meine Programmiererfahrungen liegen vor allem im Bereich der Programmiersprache Python und ihrer Anwendung. Beispiele sind ein japanischer Unicode En- und Dekodierer für Python, diverse Add-on-Skripte für ein 2D-Vektor-Malprogramm und interaktive virtuelle Figuren für den Desktop. Blender nutze ich noch nicht so lange, ich habe damit im Jahr 2006 angefangen. Seit 2008 arbeite ich an der Integration von Freestyle in Blender. Anfangs zusammen mit Maxime Curioni, der das Projekt im gleichen Jahr im Rahmen des Google Summer of Code (GsoC) angestoßen hatte. Seit 2009 bin ich Leiter des Projekts.

DP: **Wie würdest du Freestyle beschreiben?**
Tamito Kajiyama: Hier würde ich gerne meine Beschreibung aus den Release-Notes für Blender 2.67 zitieren: „Freestyle ist eine neue, nicht photorealistische (NPR) Render Engine, integriert in Blender. Ursprünglich wurde sie als Standalone-Programm in einem akademischen Forschungsprojekt entwickelt. Freestyle stellt dem Blender-Nutzer ein neues Tool bereit zum Erstellen von 2D-Linienzeichnungen aus einer gegebenen 3D-Szene, die mit Blender erstellt wurde. Die erzeugten Linien können mit einer Vielzahl von Optionen stilisiert werden wie Farbe, Transparenz, Liniendicke und Geometrie der Linien. Die dadurch erzeugte Zeichnung kann mit anderen Render-Komponenten, wie zum Beispiel Render-Passes vom Blender-internen Renderer oder Cycles, über Render-Layer und den Compositor kombiniert werden.

Beispiele für die Nutzung von Freestyle sind Cartoon-Rendering, Architekturvisualisierung, technische Zeichnungen, Blaupausen und computergenerierte Skizzen. Freestyle erweitert die Render-Möglichkeiten von Blender auf der Basis von geometriebasierten Visualisierungslösungen, zugeschnitten auf 2D-artige Computergrafik."

DP: **Wann hast du zum ersten Mal von Freestyle gehört und warum hast du beschlossen, an der Integration in Blender mitzuwirken?**
Tamito Kajiyama: Wenn ich mich recht entsinne, habe ich von Freestyle durch einen Nachrichtenartikel über Blender-bezogene Projekte im Rahmen des GsoC-2008-Programms erfahren. Maximes Vorschlag, Freestyle zu integrieren, war eines der akzeptierten Projekte. Seit ich mit Blender meine ersten Erfahrungen gesammelt habe, war Cartoon-Rendering meine Lieblingsanwendung von 3D-Kunst. Ich hatte bis dahin wenig Erfolg damit, in Blender cartoonähnliche Effekte zu erzielen. Die von Freestyle beworbenen Möglichkeiten waren für mich daher sehr attraktiv.

Ich begann mit dem Standalone-Freestyle-Programm zu arbeiten und war schlicht beeindruckt von den Ergebnissen. Anfang September kam dann das erste Windows-Build des Freestyle-Zweigs von Blender heraus. Ich war einer der vielen Blender-Nutzer, die auf die Arbeit von Maxime schon seit Monaten warteten, also habe ich mich gleich darauf gestürzt. Die Resultate der ersten Rendering-Tests von Freestyle in Blender waren einfach großartig. Währenddessen fand ich ein paar kleine technische Probleme, die ich selbst beheben konnte. Seither arbeitete ich mit Maxime, um die Integration von Freestyle in Blender zu verbessern. Für mich war es eine glasklare Entscheidung, ein wenig Verantwortung zu übernehmen, sowohl als Mitglied des Teams um den Freestyle-Zweig als auch als Blender-Commiter. Denn für mich war es ein Vergnügen, direkt zu den Verbesserungen beitragen zu können.

DP: **Warum hat die Integration so lange gedauert und was waren die größten Hürden, die du überwinden musstest?**
Tamito Kajiyama: Das Ganze hat fünf Jahre gedauert, weil das Projekt nur wenige Mitglieder mit wenig Zeit hatte. Nachdem ich es im Jahr 2009 übernommen hatte, war ich der einzige aktive Entwickler, der an Freestyle in Blender arbeitete. Ich habe sowohl Familie als auch Arbeit und konnte daher nicht viel Zeit in Freestyle stecken. Ich wusste, dass die Erwartungen der Nutzer sehr hoch waren, aber ich habe mir absichtlich keinen Stress gemacht, da das Projekt für mich eine Freizeitbeschäftigung war. Neben den beschränkten Ressourcen gab es noch zahlreiche technische Schwierigkeiten im Freestyle-Branch, die geraume Zeit und Entwicklungsaufwand in Anspruch nahmen. Die größten Hürden waren Stabilität und die Interaktion zwischen Mensch und Maschine.

Am Anfang des Projekts war das Linien-Rendern mit Freestyle sehr instabil. Das Programm stürzte selbst bei einfachsten Szenen ab. Zum Beispiel konnte Freestyle keine Szene rendern, wenn sich Objekte hinter der Kamera befanden. Solche Szenen resultierten in den meisten Fällen in einem Programmabsturz. Das war ein äußerst kritisches Problem, besonders für Animationen. Außerdem gab es viele Fälle, in denen das Resultat nicht den Erwartungen entsprach, und hässliche Artefakte. Ein häufig auftretendes Problemen waren zufällige Linien, die nur in einem kleinen Teil der Bilder einer Animation auftauchten. Diese Probleme von Instabilität und Artefakten waren die Hauptpriorität des Integrationsprojekts und die Stabilität sowohl des Programms als auch der Ausgabe hat sich seither deutlich gebessert.

Man muss darauf hinweisen, dass die ursprüngliche Instabilität der Freestyle Render Engine eine natürliche Folge ihres Ursprungs im akademischen Umfeld war und nicht Schuld der Autoren. Wie in vielen anderen Bereichen von Wissenschaft und Technik geht es in der Computergrafik darum, so viele Publikationen wie nur möglich anzusammeln, in Form von technischen Papern und Büchern. In diesem Kontext sind die dazugehörigen Softwareprodukte häufig in vielen Bereichen unfertig. Vollständig funktionsfähige Programme sind eben meist nicht notwendig für die Experimente, die eine Publikation begleiten. Ich vermute, der Hauptanteil von akademisch entstandenem Code wird nach einer Publikation irgendwo gespeichert und der Öffentlichkeit nie zugänglich gemacht. Vermutlich, weil Wissenschaftler nicht genug Ressourcen haben, um ihren Code fertig zu schreiben und zu veröffentlichen. In dieser Hinsicht ist Freestyle ein Glücksfall, denn die ursprünglichen Autoren hatten beschlossen, ihre Forschungsergebnisse als freie Software zu veröffentlichen. Ich hatte die seltene Gelegenheit, Freestyle von einem akademischen Produkt in ein robustes System für Computergrafik zu verwandeln, das bei einem weiten Feld von echten Anwendungen hilfreich sein kann. Das war (und ist immer noch) eine große Herausforderung des Freestyle-Projekts.

Eine weitere Komponente des Projekts, die mehr Zeit benötigte als erwartet, war die Entwicklung einer grafischen Benutzeroberfläche

(GUI), die für Künstler einfach zu bedienen ist. Freestyle ist so aufgebaut, dass der Nutzer über Python vieles programmieren kann. Zu Beginn des Integrationsprojekts wurde von den Nutzern erwartet, dass sie ihre eigenen Stil-Module über Python erstellten. Diese Voraussetzung hat den Nutzerkreis natürlich ganz schön eingeschränkt. Blender-Nutzer sind ja nicht von Natur aus Python-Progammierer.

Ich habe immer wieder Anfragen nach einem künstlerfreundlichen GUI erhalten, mit dem man die Parameter für die Linienstile interaktiv bearbeiten kann. Ursprünglich sollte das implementiert werden, nachdem Freestyle im Hauptentwicklungszweig von Blender gelandet war. Aber der Bedarf nach einem GUI für Freestyle war einfach sehr groß. Also wurde beschlossen, zuerst ein GUI zu erstellen und dann das Projekt in Blender aufzunehmen. Das GUI musste von Grund auf erstellt werden. Mein Designziel war, die Optionen für das Erstellen von Linienstilen so flexibel wie nur möglich zu gestalten. Am besten wäre gewesen, wenn das GUI erlaubt hätte, all das zu tun, was über Python möglich war. Dabei wurde auch ein Node-basiertes Interface diskutiert, aber das wäre in Anbetracht des Aufwands und der Entwicklungszeit nicht machbar gewesen. Stattdessen wurde der Weg gewählt, dass ein festgelegter Satz von Linienstil-Optionen sorgfältig ausgewählt und als Teil der Benutzeroberfläche von Blender implementiert wurde. Das jetzige GUI ist das Ergebnis von vielen Designversuchen und Updates über einen großen Zeitraum. Ich hoffe, damit können Blender-Nutzer die Möglichkeiten von Freestyle so weit es geht ausreizen.

DP: **Schon lange bevor Freestyle im Hauptzweig von Blender gelandet ist, wurden damit beachtliche Werke erstellt. Hast du während dieser Zeit von den Nutzern gutes Feedback erhalten?**
Tamito Kajiyama: Ja, von den Early Adopters des Freestyle-Zweigs habe ich eine große Menge an Feedback erhalten. Dieses Feedback von den Nutzern war essenziell für mich, um die Freestyle-Integration zu Ende zu bringen. Es gab viele Formen von Nutzer-Feedback: Bug Reports, Tests von Fehlerbeseitigungen und neuen Features, manche Nutzer erstellten regelmäßig Builds des Freestyle-Zweigs, andere teilten ihre mit Freestyle erzielten Resultate und manche schrieben technische Dokumentationen. Grundsätzlich bin ich aus eigenem Antrieb motivert und arbeite an Freestyle, weil ich das Programm selbst nutzen möchte. Aber ich finde es auch toll, wenn ich sehe, wie andere von meiner Arbeit profitieren. Dieses Gefühl ist sowohl belohnend als auch animierend. Im Falle von Freestyle war das ständige Feedback der Nutzer eine große Hilfe, um einen langen Weg zu Ende zu gehen.

DP: **Über Monate, wenn nicht gar Jahre, haben Blender-Nutzer darum gebeten, wenn nicht sogar gebettelt, dass Freestyle doch endlich in der offiziellen Blender-Version landen solle. War das motivierend oder nervend?**
Tamito Kajiyama: Die Rückmeldungen der Freestyle-Nutzer waren absolut motivierend, denn auf der Nutzerseite war immer etwas Neues zu entdecken. Mit der Zeit dürften manche Nutzer das Interesse verloren haben, aber es gibt immer Neuzugänge mit frischem Interesse an neuen Anwendungen.

Eine derart dynamische und aktive Nutzerbasis zu haben, war ein großes Plus für das Freestyle-Integrationsprojekt. Natürlich gab es Anfragen von Nutzern, die nicht so leicht zu beantworten waren und über die ich lange nachdenken musste. Aber das gehört dazu und grundsätzlich habe ich sie gerne bearbeitet.

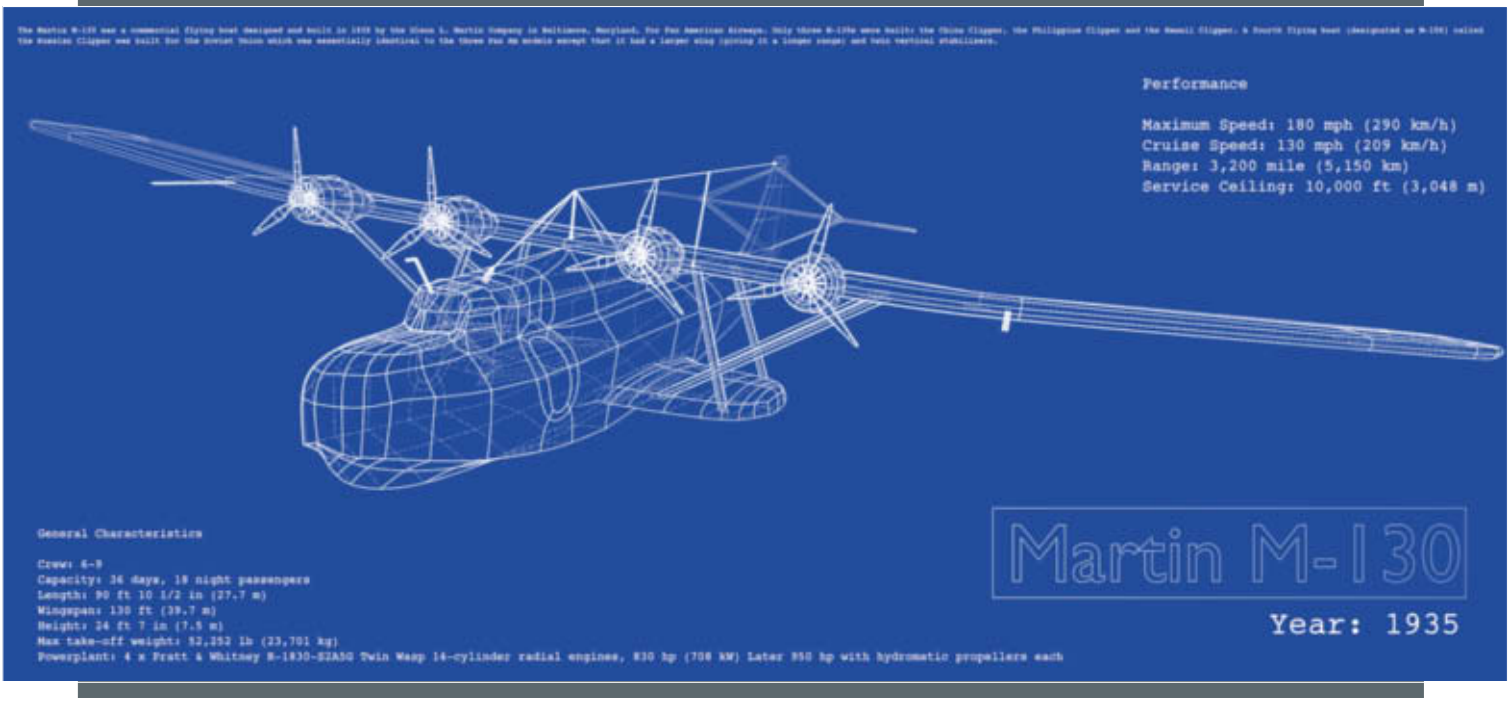
DP: **Suchst du aktiv nach Freestyle-Kunstwerken und was war deiner Meinung nach bisher die genialste Form der Nutzung?**
Tamito Kajiyama: Schon seit die Integrationsarbeiten angefangen haben, suche ich aktiv nach Anwendungen von Freestyle durch die Blender-Nutzer. Ich mag Freestyle-Kunst einfach und sehe Positives, egal wie die Gesamtqualität ist. Es ist für mich also nicht einfach, die beste Anwendung auszuwählen. Ich habe hier ein paar Beispiele gesammelt:

Ryuichi Oosawa hätte mit diesem Bild beinahe den Splash-Screen-Contest von Blender 2.67 gewonnen.


Overweight Boss: Dieses Freestyle-Bild von McLelun wurde von Tamito Kajiyama als Splash-Screen für die Version 2.67 von Blender ausgewählt.


Anime mato.sus304 hat dieses Anime-Bild mit Freestyle erstellt. Es ist eines der Demo-Bilder für Freestyle aus den Release-Notes von Blender 2.67. Lizenz: CC-BY-SA

Für Wireframe Render lässt sich Freestyle besonders gut einsetzen. Hier ein stark stilisierter Drahtgitter-Render mit verdeckter Geometrie (gestrichelte Linien) und Fadeout. Die beiden Features schlagen sich allerdings auch deutlich auf die Renderzeit nieder.



Modell zweimal dupliziert und rotiert, ergibt Ansichten von allen drei Achsen, ein weiteres Duplikat zweimal rotiert, eine Isometrie. Da Freestyle auch verdeckte Kanten rendern kann, lässt sich so das gesamte Modell abbilden, zum Beispiel mit den verdeckten Bereichen als gestrichelte Linien. Freestyle geht hier sogar so weit, dass man unterschiedliche Überdeckungstiefen auswählen kann.

So ist es zum Beispiel möglich, dass Kanten, die von zwei Flächen überdeckt sind, anders dargestellt werden als Kanten, die nur von einer Fläche überdeckt sind. Ein Gefühl für die Tiefe eines Modells lässt sich herstellen, indem die Kanten – je nach Entfernung von der Kamera – unterschiedliche Farben, Dicke oder Transparenz erhalten, natürlich mit feinen Gradienten.

Ansehnliche Wireframes

Dank Freestyle lässt sich auch mit wenig Aufwand ein Wireframe-Render des 3D-Modells erstellen. Hierfür müssen lediglich alle Kanten des Objekts als Freestyle-Kanten markiert werden – und schon ist der Wireframe-Render fertig. Dieser kann natürlich auch über den Original-Render gelegt werden und von den oben beschriebenen Features, wie der Darstellung von verdeckten Kanten und Fadeout in die Tiefe, profitieren.

Ein letztes praktisches Anwendungsbeispiel ist das Erstellen eines Logos. Dabei kann zuerst eine Version in 3D mit Beleuchtung, Materialien et cetera erstellt werden. Später kann dann noch die 2D-Version mit Freestyle ausgerendert werden. Normalerweise würde man erst die 2D-Version in einem Vektorgrafik-Programm erstellen. Letztgenanntes Beispiel ist ein Fall, der wohl am meisten vom geplanten SVG-Export profitieren würde.

Wie funktioniert Freestyle praktisch?

Wer sein Rendering mit Freestyle-Linien aufpeppen will, sollte sich zuallererst darüber im Klaren sein, dass Freestyle derzeit nur für den internen Renderer zur Verfügung steht.

Wer für das Rendering Cycles verwenden will, sollte eine Kopie der Szene erstellen und dort auf den Blender-internen Renderer wechseln. In den Render-Einstellungen des Properties-Panels findet sich dann unter dem „Post-Processing"-Tab ein neues Tab mit der Bezeichnung „Freestyle". Dort findet sich neben einem Häkchen zum Aktivieren von Freestyle nur eine Option für die Dicke der Linien. Den Rest der Optionen findet man bei den Einstellungen für die Render-Layer. Bei der „Line Thickness" kann man zwischen absoluter und relativer Dicke wählen. Absolut bedeutet, dass die Linien immer die gleiche Dicke haben, unabhängig von der Größe des Renderings.

Das bedeutet natürlich auch: Wenn man zum Beispiel für ein Poster zunächst Preview-Renderings mit geringer Auflösung macht, erscheinen in der finalen Version in voller Auflösung die Linien plötzlich sehr dünn. Um das Problem schnell ausgleichen zu können, findet sich daher bei der Absolut-Version noch ein Multiplikator. Wenn man zum Beispiel die Preview-Renderings bei 25 Prozent durchgeführt hat, so kann man für die finale Version dort einfach den Wert 4.0 eingeben. Die Dicken der unterschiedlichen Linienstile werden dann mit diesem Wert multipliziert.

Im Tab für die Render-Layer befinden sich dann die eigentlichen Einstellungen für die Freestyle-Linien. Standardmäßig ist der Parameter-Editier-Modus eingestellt. Hier erstellt man seine Linienstile, wie sonst von Blender gewohnt, über diverse Knöpfe und Werte. Für fortgeschrittene Nutzer gibt es außerdem noch den „Python Scripting Mode". Dieser erlaubt größere Freiheiten und zudem den einfachen Austausch von Freestyle-Stilen über Python-Dateien. Diese Übersicht beschränkt sich aber auf den Modus zum Editieren von Parametern.

Linien und ihr Stil

Jeder Freestyle-Layer besteht aus einem Set von Linien. Dieses bestimmt, welche Kanten gerendert werden. Zum Beispiel könnte man nur die Kontur des Objekts zeichnen oder sämtliche Kanten, die man vorher als Freestyle-Kante im Editier-Modus markiert hat. Oder nur die verdeckten Kanten. Jedem Linien-Set ist ein Stil zugeordnet. Dieser beschreibt letztlich, wie die Linien aussehen sollen. Einfluss nehmen kann man dabei auf mehrere Parameter. Zum einen auf den Strich selbst – soll der Strich eine durchgehende Linie sein oder periodisch unterteilt werden? Sollen mehrere Kanten zu einem Strich zusammengefasst oder vielleicht doch jede Kante einzeln gezeichnet werden? Außerdem können Farbe, Transparenz und Dicke beein-

flusst werden. Diese drei Parameter können jeweils über den Modifier verändert werden. Möglich ist beispielsweise, die Dicke entlang des Strichs variieren zu lassen, zum Beispiel könnte sie immer dünner werden. Andere verfügbare Optionen sind die Distanz zur Kamera, Entfernung von einem gegebenen Objekt (zum Beispiel ein Empty) oder durch das Material, welches das Objekt an dieser Kante hat.

Anders hingegen verhält es sich mit dem letzten Parameter, der die Geometrie der Linien selbst beeinflusst. Hier finden sich vor allem verschiedene Arten von Rauschen und mathematische Funktionen. Eine mit zittriger Hand erstellte Zeichnung könnte man zum Beispiel über eine Sinus-Verzerrung zusammen mit ein wenig Perlin-Rauschen nachbilden. Bei den Einstellungen für die Geometrie findet sich aber noch viel mehr. So kann jeder Strich in beide Richtungen ein wenig verlängert werden, oder man kann die Bounding-Boxes der einzelnen Elemente nachzeichnen lassen. Die Aufteilung in Liniensets und -stile mag zwar auf den ersten Blick etwas kompliziert und verwirrend erscheinen, ist aber streng logisch aufgebaut. Will man zum Beispiel verdeckte Kanten anders darstellen als sichtbare, so benötigt man zwei Sets: eines für die nicht verdeckten Kanten, das man zum Beispiel mit einer durchgehenden Linie darstellen könnte, und eines für die verdeckten Kanten. Letzteres könnte zum Beispiel leicht transparent und gestrichelt dargestellt werden.

Kombination mit anderen Render Engines

Die Nutzung des Blender-internen Renderers ist vielleicht nicht jedermanns Sache. Zum Glück gibt es eine einfache Möglichkeit, Freestyle mit anderen von Blender unterstützten Render Engines zu nutzen, zum Beispiel mit Cycles. Dabei ist von Vorteil, dass Freestyle ein Post-Processing-Effekt ist. Der Trick ist Folgender: Man macht eine Kopie der Szene (zum Beispiel mit verlinkten Objekten, damit sich Änderungen auf beide Szenen auswirken) und wählt bei der Kopie der Szene den Blender-internen Renderer. Dort aktiviert man Freestyle und schaltet bei den Includes für die Render-Layer alles aus bis auf Freestyle. Damit erhält man in der zweiten Szene ausschließlich die Freestyle-Linien. Über den Compositor lassen sich diese dann mit dem Rendering der anderen Engine kombinieren.

Fazit

Mit Freestyle hat die NPR-Community um Blender ein sehr mächtiges Werkzeug für zahlreiche Anwendungsfälle an die Hand bekommen.

Es gab Befürchtungen, dass dieser Zweig mit der Einführung von Cycles vernachlässigt werden würde, was sich aber nicht als richtig herausgestellt hat. Mit dem Projekt BEER (Blender Extended Expressive Rendering) sollen zudem die Toon-Shading-Optionen des internen Renderers deutlich erweitert werden.

In der Kombination mit Freestyle könnte sich Blender zum Geheimtipp für die Produktion von Zeichentrick mittels 3D mausern.

› 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, u. a. für CG Tuts+ und CG Cookie. Weiterhin betreibt er die Webseite www.Blender-Diplom.com, auf der Blender-Tutorials in deutscher und englischer Sprache zur Verfügung stehen.

Bei einer Auflösung von 5.120 x 2.880 Pixeln benötigte dieser Render von Frederik Steinmetz inklusive Compositing rund 50 Sekunden.