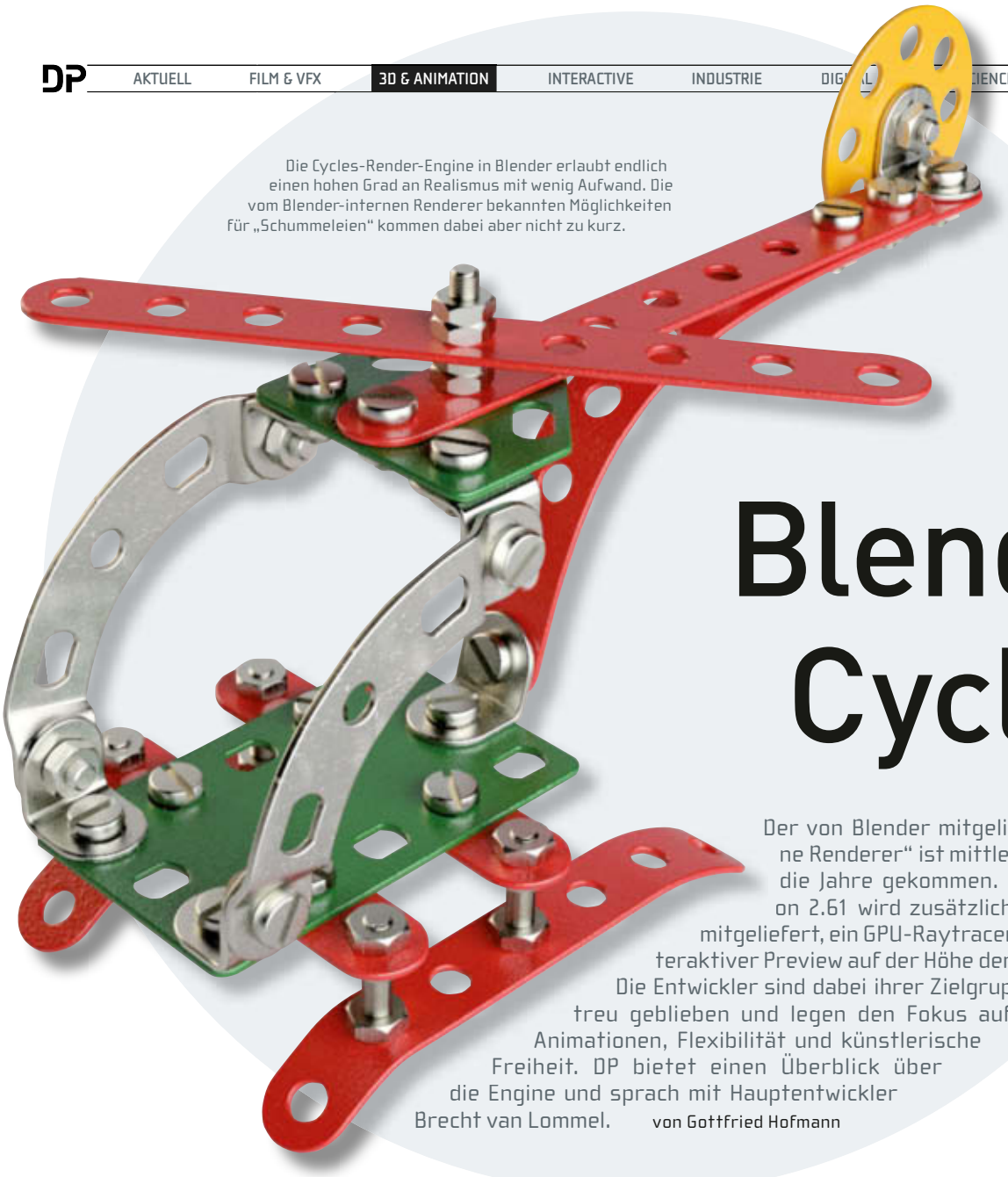


Die Cycles-Render-Engine in Blender erlaubt endlich einen hohen Grad an Realismus mit wenig Aufwand. Die vom Blender-internen Renderer bekannten Möglichkeiten für „Schummeleien" kommen dabei aber nicht zu kurz.

Blender Cycles

Der von Blender mitgelieferte „interne Renderer" ist mittlerweile arg in die Jahre gekommen. Seit Version 2.61 wird zusätzlich „Cycles" mitgeliefert, ein GPU-Raytracer mit interaktiver Preview auf der Höhe der Zeit. Die Entwickler sind dabei ihrer Zielgruppe treu geblieben und legen den Fokus auf Animationen, Flexibilität und künstlerische Freiheit. DP bietet einen Überblick über die Engine und sprach mit Hauptentwickler Brecht van Lommel.

von Gottfried Hofmann

Für den einen ein Fluch, für den anderen ein Segen – die interne Render-Engine von Blender. Gute Perfomance bei unrealistischen Settings, eine große Auswahl an Möglichkeiten, das Ergebnis zu beeinflussen, und eine sehr gute Anbindung ans Compositing prädestinieren die Engine für Animationen und Motion Graphics.

Für Realismus muss man aber einiges an Arbeit investieren und grundlegende Dinge wie Global Illumination und Kaustiken fehlen völlig. Wer das benötigt musste bisher zu einem externen Renderer greifen wie Luxrender, Yafaray, Octane oder V-Ray.

Viele Nutzer wünschten sich diese Features aber Blender-intern. Ihnen wurde in Version 2.61, die als verfrühtes Weihnachtsgeschenk am 14. Dezember 2011 erschienen ist, nun Gehör verschafft. Aber anstatt den internen Renderer immer weiter aufzubohren hat man sich dazu entschieden, eine Engine von Grund auf neu zu schreiben und dabei auch den Workflow rundzuerneuern.

Cycles

Herausgekommen ist Cycles, ein Raytracer mit interaktiver Preview, der sich auf dem Hauptprozessor genauso wohl fühlt wie auf der Grafikkarte mit OpenCL oder Cuda. Während man Änderungen vornimmt, werden diese in der Live-Preview automatisch übernommen, was Test-Renderings größtenteils überflüssig macht. Shader- und Textur-Workflow sind nun komplett node-basiert mit einer zusätzlichen Baum-Ansicht für schnelle Korrekturen oder einfache Materialien. Im Prinzip arbeitet die Engine realistisch respektive physikalisch korrekt. Man hat als Nutzer aber große Freiheiten, wenn es darum geht, auf den Realismus Einfluss zu nehmen.

Hintergrund

Ähnliche wie der von Sony Pictures Imageworks eingesetzte „Arnold"-Renderer ist Cycles ein reiner Pathtracer. Filme wie „Wolkig mit Aussicht auf Fleischbällchen" zeigen, dass diese Technik inzwischen durchaus für Animationen eingesetzt werden kann. Der fertige Render dauert damit zwar etwas läger, die Interaktivität beim Editieren erlaubt aber wiederum deutliche Zeitersparnisse bei der Arbeit.

Wenn man sich die zugrunde liegende Technik näher ansieht, leuchtet schnell ein, wo man bei Cycles schummeln kann, um entweder Zeit zu sparen oder künstlerische Freiheit zu erlangen. Die Kamera „verschießt" dabei Strahlen, die auf verschiedene Objekte treffen und dabei ihren Weg und ihren Charakter verändern können. Aus den ge-

Gottfried Hofmann studiert Informatik an der Friedrich Alexander Universität Erlangen-Nürnberg. Zur Finanzierung seines Studiums arbeitet er als Freelancer im VFX-Bereich, wofür er hauptsächlich Blender einsetzt. In seiner Freizeit erstellt er Tutorials, die er auf seiner Webseite **www.BlenderDiplom.com** kostenlos zur Verfügung stellt.

wonnenen Informationen errechnet die Kamera einen Farbwert für jeden Pixel. Lässt man dem Spiel lange genug freien Lauf, erhält man am Ende ein physikalisch korrektes Resultat. Wenn man jedoch eingreift, lässt sich Renderzeit sparen oder künstlerische Freiheit erlangen.

[© Mike Pan / Blender Foundation 2011. Lizenz: CC-BY 3.0 http://www.blender.org/development/release-logs/blender-261/blender-261-demo-files/]

Ein einfacher Shader in Cycles – Über Nodes lassen sich Shader in Cycles nach Belieben mischen. Das Bild oben ist die Live-Preview bei moderaten 512 Samples.

Zeit sparen und vieles ändern

Am einfachsten lässt sich Zeit sparen, indem man die Anzahl der „Bounces" reduziert. Ein Wert von null schaltet ganz automatisch die Global Illumination aus. Außerdem kann man für jedes Objekt festlegen, ob es für bestimmte Strahlen sichtbar sein soll, hier genannt „Ray Visibility". Nutzer von Cinema 4D werden sich dabei gleich zu Hause fühlen, denn die Funktion ähnelt stark dem dortigen „Render Tag". So lassen sich Objekte mit einem Klick aus Spiegelungen entfernen oder für die Kamera unsichtbar machen (was besonders bei Licht emittierenden Objekten praktisch ist).

Es bieten sich aber noch deutlich mehr Möglichkeiten. Denn über die „Light Path"-Node lassen sich auch Shader mittels der Strahl-Informationen beeinflussen. Dadurch ist es zum Beispiel möglich, Schatten oder Kaustiken zu verstärken, abzuschwächen oder einzufärben. Gleiches gilt für Reflexionen, die nochmals feiner über Materialeigenschaften ansprechbar sind. In der aktuellen Entwicklungsversion sind zudem schon die ersten Optionen fürs Compositing integriert, darunter rudimentäre Render-Layer sowie der Holdout-Shader, mit dem Objekte als transparente Bereiche im Render erscheinen.

Fazit

Mit Cycles schließen die Render-Bordmittel von Blender endlich zur Konkurrenz auf. Gleichzeitig wird der Workflow flexibilisiert und die zahlreichen Möglichkeiten zum Schummeln schließen eine Lücke, die von den externen Engines bisher offen gelassen wurde und die vormals der Blender-interne Renderer ausfüllen musste. Freelancer, kleine Studios und VFX-Interessierte sollten die Entwicklungen im Auge behalten.

Ausblick

Die Render-Engine Cycles befindet sich momentan noch in einer frühen Entwicklungsphase. Für die nähere Zukunft geplant (und in der Version 2.62, die bei Erscheinen dieser Ausgabe der DP aktuell sein wird) möglicherweise schon integriert: zusätzliche Manipulationsmöglichkeiten bei den Shadern, um beispielsweise einen Toon-Look zu erreichen, Optimierungen der Geschwindigkeit und Render-Layer sowie -Passes. Für die nähere Zukunft sind Subsurface Scattering, Motion Blur und Volumetrics im Visier.

Displacement und Depth of Field – Die Strukturen auf diesen Christbaumkugeln sind durch Displacement entstanden.

[© Olivier Amrein 2011. Lizenz: CC-BY 3.0 www.blendswap.com/3D-models/misc-objects/christmas-balls/]

(© William Maanders, Lizenz: CC-BY 3.0)

Brecht van Lommel ist der Hauptverantwortliche für die Entwicklung der Cycles-Render-Engine.

Ich habe meine ideale Render-Engine geschrieben

Die DP im Gespräch mit Brecht van Lommel über die neue Render-Engine für Blender: Cycles. Brecht ist der Hauptverantwortliche für die Entwicklung.

DP: **Wie ist es dazu gekommen, dass du die Entwicklung einer komplett neuen Render-Engine für Blender angestoßen hast?**
Brecht van Lommel: Ich hatte schon lange an dem bisherigen Blender-internen Renderer gearbeitet. Wir haben ihn in vielen Projekten wie den Open Movies eingesetzt. Leider hat er viele Limitierungen. Seine Entwicklung geht zurück auf die späten 80er Jahre und nach und nach wurde immer mehr hinzugefügt. Der Kern ist nicht sehr modular und auf direkte Beleuchtung und Rasterisierung ausgelegt. Neues hinzuzufügen wurde immer schwieriger. Ich versuchte, den Kern aufzubohren und zu erneuern, aber das funktionierte nicht sehr gut. Irgendwann hatte ich viel freie Zeit und da habe ich angefangen, meine eigene ideale Render-Engine zu schreiben. Ich habe dabei versucht, Erfahrungen aus den Open Movies mit einfließen zu lassen. Nach einem halben Jahr habe ich die Sache Ton Roosendaal, dem Vorsitzenden der Blender Foundation, gezeigt und war interessiert. Jetzt arbeite ich wieder für die Blender Foundation, die Hälfte der Zeit ausschließlich an der Cycles-Engine als neuer Renderer für Blender.

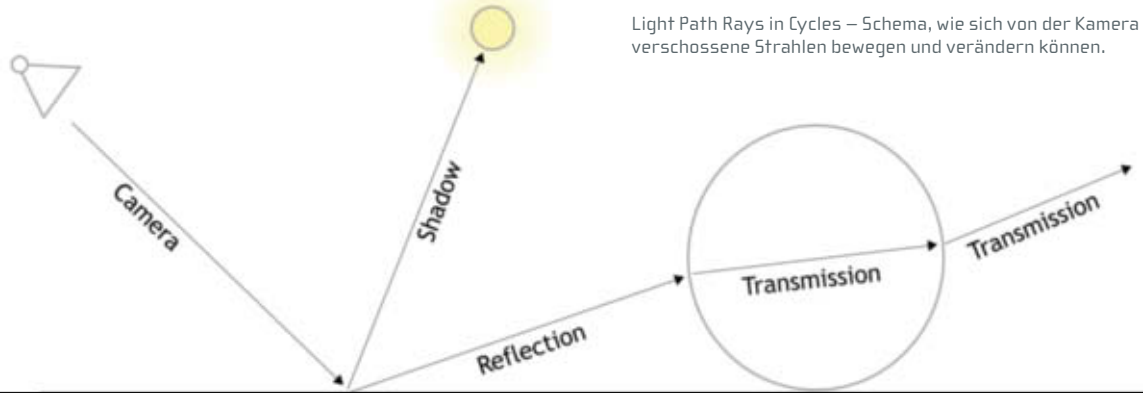
DP: **Was ist die Zielgruppe von Cycles?**
Brecht van Lommel: Die Zielgruppe ist in etwa die gleiche wie die von Blender selbst – einzelne Künstler und kleine Studios. Cycles ist nicht als Render Engine für Hollywood konzipiert. Sie soll einfach zu benutzen sein und schnell Ergebnisse liefern mit möglichst wenig technischen Parametern, in die man sich erst einlernen müsste. Weiterhin habe ich großen Wert auf Interaktivität gelegt. In Cycles hat man eine interaktive Preview, die sich updatet während man arbeitet. Das ist sehr wichtig. In den Open Movies gab es viele komplexe Szenen, bei denen Minuten vergehen konnten, bis man nach dem Drücken auf „Render" die ersten Pixel sah. Feintuning von Parametern ist in einer solchen Situation natürlich äußerst schwierig. Darum habe ich Cycles als interaktiven Raytracer konzipiert. Er aktualisiert während man Parameter ändert und man sieht dadurch die Ergebnisse so schnell wie möglich.

DP: **Du hast Cycles nicht absolut physikalisch korrekt aufgebaut, sondern erlaubst zahlreiche Schummeleien?**
Brecht van Lommel: Wir versuchen, zahlreiche Schummeleien zu erlauben, die in diversen physikalischen Render Engines nicht möglich sind. Grundsätzlich sieht die Sache so aus: Wenn man Animationen rendern möchte und sich eine höhere Geschwindigkeit wünscht, dann muss es möglich sein, diverse Dinge ausschalten zu können. Global Illumination, Kaustiken und Feineinstellungen bei den Bounces sind allgemeine Beispiele. Wir haben aber noch viel mehr im Gepäck, um Sachen zwar nicht physikalisch korrekt, aber brauchbar für Animationen darstellen zu können.

DP: **Animationen werden mit Cycles im nächsten Open Movie genug gerendert werden. Freust du dich schon auf die kommenden Aufgaben?**
Brecht van Lommel: Das ist in den nächsten vier Monaten, da bin ich schon ein wenig ängstlich. Wir werden sehen wie es funktioniert, zur Not haben wir noch den Blender-internen Renderer in der Hinterhand. Aber Cycles wird ausgiebigen Tests im Bereich VFX unterzogen werden. Render Layer und Render Passes sollen mit realen Aufnahmen komponiert werden. Dazu kommt Umgebungslicht von realen Aufnahmen. Das wird der Fokus der nächsten Monate: Cycles für VFX-Arbeiten fit machen.

DP: **Das Shader-System wurde maßgeblich von der Open Shading Language (OSL) beeinflusst, sie wird aber nicht direkt eingesetzt?**
Brecht van Lommel: Wir haben momentan zwei Backends. Eines unterstützt OSL, lässt sich aber nicht direkt nutzen. Das andere ist auf CPU und GPU implementiert. Direkte OSL-Unterstützung in Blender ist aber geplant.

DP: **Dann wird man auch eigene oder fremde OSL-Shader in Blender nutzen können?**



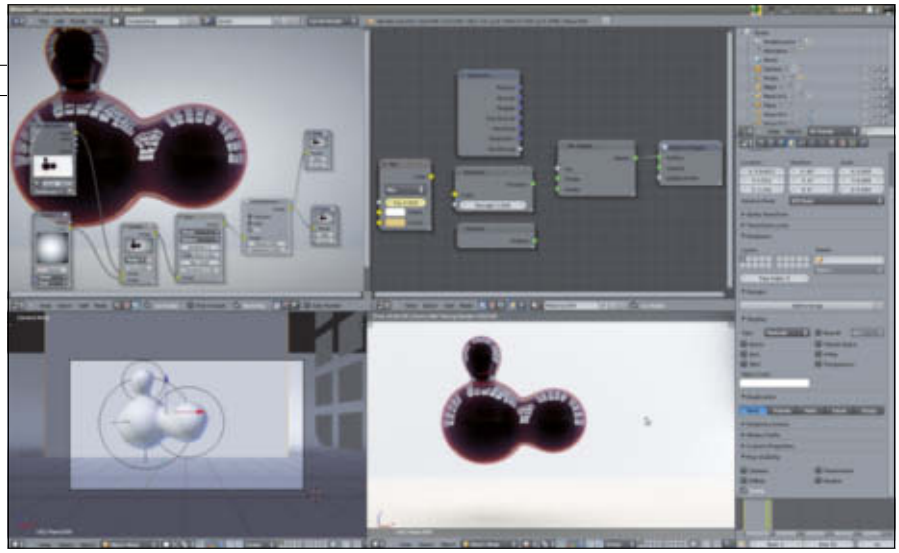
Light Path Rays in Cycles – Schema, wie sich von der Kamera verschossene Strahlen bewegen und verändern können.

(© Brecht van Lommel 2011)

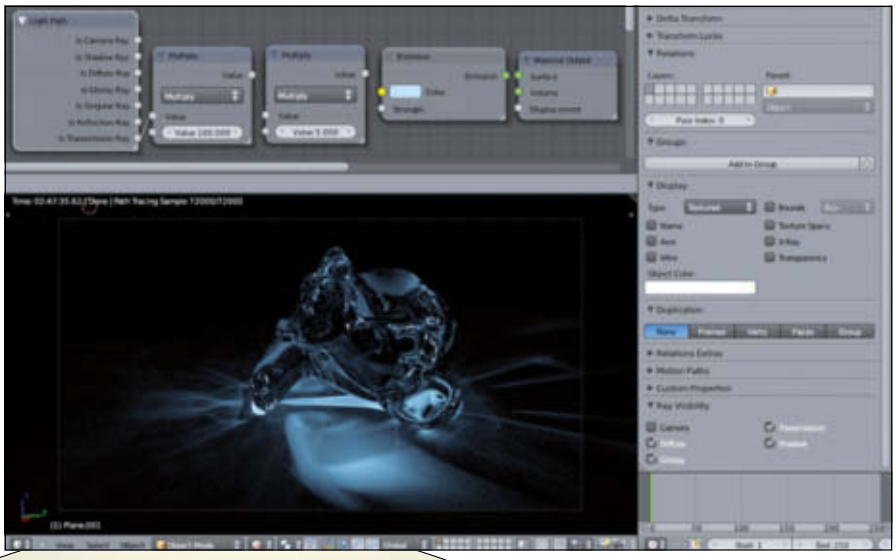
Brecht van Lommel: Das Hauptproblem ist, OSL auf der GPU zum Laufen zu bekommen. Das ist fast so viel Arbeit, wie bisher in Cycles steckt. Das ist keine einfache Aufgabe, wir brauchen mehr Leute, die uns dabei helfen.

DP: **Fürs Rendern auf der Grafikkarte unterstützt Cycles sowohl OpenCL als auch Cuda. Ist das nicht viel Extra-Aufwand?**
Brecht van Lommel: Im Grunde sind sich die Sprachen sehr ähnlich. OpenCL hat lediglich ein paar mehr Einschränkungen als Cuda. Sobald man sich auf OpenCL beschränkt, ist es sehr einfach, parallel für Cuda und die CPU zu entwickeln. Die reine CPU-Version ist wichtig, da es Leute ohne passende Grafikkarten gibt (man denke an Renderfarmen). Oder man hat eine Szene, die viel Speicher benötigt. OpenCL ist auch sehr wichtig, denn es wird sowohl von AMD und Nvidia unterstützt. Cuda hat die beste Performance, es macht also Sinn, alle drei zu unterstützen. Dann hat jeder Nutzer die Möglichkeit, seine Hardware perfekt auszureizen.

DP: **Was das Interface angeht, setzt du sowohl auf Nodes als auch auf Layer?**
Brecht van Lommel: Wir haben uns für Nodes entschieden, weil man damit viele coole Tricks und Kombinationen erstellen kann, die mit Layern nicht möglich wären. Allerdings tendieren Nodes dazu, schwierig zu handhaben zu sein. Für Szenen mit vielen Materialien werden wir Presets anbieten – Node-Gruppen mit Voreinstellungen für Plastik oder Metall. Die wird man in einer einfachen Baumstruktur übereinanderlegen können. Damit sind schnell Resultate möglich und man hat immer noch die Möglichkeit, all die komplexen und verrückten Dinge über Nodes zu realisieren. **› ei**



Schummeleien: Ein einfaches Beispiel, wie sich in Cycles auf den Realismus Einfluss nehmen lässt. Hinter dem Blob befindet sich eine Plane, die orangefarbenes Licht emittiert. Dieses ist nur in den Reflexionen auf dem Blob sichtbar und verleiht ihm eine leichte Kante. Damit die Rückwand nicht auch angestrahlt wird, wurde zudem die Rückseite der Plane „ausgeschaltet". Die Spiegelungen der Hohlkehle im Blob wurden ebenfalls entfernt. Links oben sieht man den fertigen Render mit einem einfachen Compositing.



Schummeleien Teil 2 – Dieser Glas-Affenkopf wird ausschließlich indirekt über seine eigenen Kaustiken einer unsichtbaren Lichtquelle beleuchtet.

Der Frost in dieser Szene wurde über Partikel realisiert, die Cycles dank Instancing effizient rendern kann.