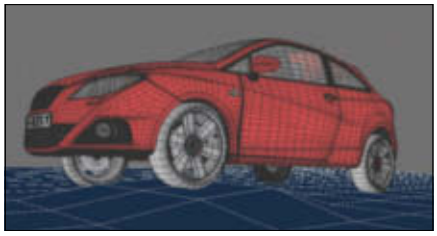
Realistische Fahrdynamiksimulation

Sebastian Krieg, 3D-Freelancer und Gründer des Animationsstudios „3d-bewegungswerkstatt“, hat im Zeitraum von nur einem Jahr eine Physik-Engine zur Simulation realer Fahrphysik in der 3D-Umgebung von Cinema 4D entwickelt. Im folgenden Artikel geht er nicht nur auf technische Details ein, sondern auch darauf, wie es zur Idee kam und welche umfangreichen Voraussetzungen eine solche Entwicklung erfordern.

von Sebastian Krieg

Wie kommt man überhaupt auf die wahnwitzige Idee, komplett alleine eine Physik-Engine zu entwickeln? Zur Beantwortung dieser Frage muss Sebastian Krieg ein wenig in der Geschichte zurückgehen: „Alles fing damit an, dass während meines 3D-Studiums, einer meiner Professoren zu uns in den Saal kam und erzählte, dass die Firma ESG in Fürstenfeldbruck einen Studenten sucht, der Erfahrung mit Fahrzeugvisualisierungen und 3D-Animationen hat“, erinnert sich Krieg.

Da Krieg bereits während des laufenden Semesters mehrere detaillierte 3D-Fahrzeuge für einen Kurzfilm modelliert hatte, fiel die Wahl glücklicherweise auf ihn. „Die ESG ist ein großer Konzern, der neue Technologien für Land-, See- und Luftfahrzeuge entwickelt. Für diesen Konzern sollte ich im Bereich Automotive mehrere Fahrzeuganimationen erstellen, welche die Funktionsweise neuer Fahrerassistenzsysteme zum Ziel hatten“, erzählt der 3D-Artist zu seiner damaligen Aufgabenstellung. Grundsätzlich erst mal ein 3D-Job wie viele andere. Doch



Einzelradaufhängung Die Räder des Fahrzeugs reagieren auf die Unebenheiten des Untergrunds

das, was sich daraus entwickeln würde, hätte er nicht erwartet. Die Animationen waren sehr langwierig und die Bewegungen nicht immer zufriedenstellend. Gerade Fahrzeugbewegungen sind nicht immer leicht per Keyframe-Animation zu erstellen.

Nach hunderten von gesetzten Keys, immer wieder neue Ideen, zahlreichen Änderungen, wieder einen großen Teil der Arbeit erneut zu beginnen, das war bei der Vielzahl der Sequenzen oft ziemlich ernüchternd. „Nach mehreren Wochen waren die Animationen endlich fertig – ebenso wie ich“, offenbart Krieg mit einem Schmunzeln.

Seine Visualisierungen kamen gut an, und so sollten weitere Filme folgen. Wobei er zu diesem Zeitpunkt nicht allzu begeistert über diesen Umstand war, wieder wochenlang Keyframes setzen zu müssen. Deswegen schlug er vor, zuvor ein paar Tools zur Arbeitserleichterung zu programmieren. Dies war die Geburtsstunde der „Physik-Engine“.

Die Vorbereitungen

Die Grundvoraussetzungen für Krieg waren gut. Er hatte Zeit, bekam Geld, hatte für kurze Fragen Fahrzeug-Ingenieure im Rücken und die Zusage der Leitung, dass die Entwicklung rechtlich seine Entwicklung bleibe und die Firma nur einfache Nutzungsrechte für die Tools beansprucht. Ein wichtiger Punkt in Bezug auf seine Motivation für das Projekt.

Zu Beginn überlegte er, auf welche Art er die Tools entwickeln wollte. Ein Plug-in schien ihm anfangs die beste Möglichkeit, er entschied sich aber nach reiflicher Überlegung um. Die Tools sollten mit Maxons nodebasierter Entwicklungsumgebung XPresso und einigen integrierten C.O.F.F.E.E.-Nodes erstellt werden. Dafür gab es gute Gründe: „XPresso-basierte Entwicklungen sind seit Version 10 von C4D absolut versionsunabhängig und müssen nicht, wie häufig bei Plug-ins notwendig, neu kompiliert werden, erklärt Krieg. „Dazu kommt, dass XPresso plattformunabhängig ist und nicht extra wie ein Plug-in installiert werden muss“, rundet er seine Entscheidung ab. Zwar können Plug-ins im Editor schneller arbeiten, doch ist dies für das später gerenderte Ergebnis unerheblich.

Die Radrotation

Das erste Problem, das es zu lösen galt, war die Rotation der Räder. Dieses Problem hört sich einfach an, wird aber häufig unterschätzt. Die Rotation der Räder sollte in Abhängigkeit zum Tempo des Fahrzeugs und zur zurückgelegten Strecke synchron laufen, egal ob vorwärts oder rückwärts. Zudem sollte die Rotation kontinuierlich aussehen.

Nach einer Recherche im Internet wurde er fündig. Die physikalische Formel für die Berechnung einer Radrotation in Abhängigkeit zur zurückgelegten Strecke. Die Lösung war einfach – zu einfach.

Zwar tat die in XPresso umgesetzte Formel genau das, was von ihr erwartet wurde, doch leider hatte Krieg das Phänomen übersehen, das es auch beim Film gibt. Da ein Film ebenso wie eine standardmäßige Animation

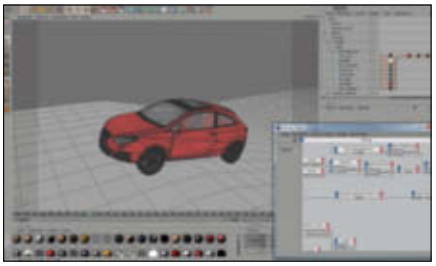
mit 25 Bildern pro Sekunde abläuft, kommt es mit zunehmender Rollgeschwindigkeit des Rades zu einem optischen Zurückdrehen des selbigen. Diesen Effekt kannte man schon, allerdings kam es des Öfteren während einer Animation zu der Situation, dass bei einer bestimmten Geschwindigkeit das Rad komplett zu stehen oder zu springen schien. Dies passiert genau dann, wenn sich nach einem Frame die Position der Felgen eines Rades wieder an derselben Stelle befindet wie im vorherigen Frame, obwohl sich das Rad gedreht hat. Dieser Zustand war inakzeptabel und erforderte eine andere Vorgehensweise.

„In der abgewandelten Form löste ich das Problem durch Ermittlung zweier aufeinanderfolgender Rotationswinkel. Ergibt sich die Situation, dass in zwei aufeinanderfolgenden Frames dieselbe Position der Felge vorhanden ist, so wird auf den aktuellen Winkel ein winziger Winkelanteil hinzuaddiert, wodurch gewährleistet wird, dass sich die Räder immer vorwärts drehen", führt er aus.

Die Untergrund-/Radkollision

Ein weiteres Problem bei der Animation eines Fahrzeugs brachte die Kollision des Rades mit dem Untergrund mit. Hier sollte es zu keinen Durchdringungen kommen. „Besonders nervenaufreibend werden Keyframe-Animationen ja, wenn sich das Fahrzeug über einen sehr unebenen Boden bewegt. Hier muss man fast Frame für Frame für jedes Rad neue Keys setzen", sagt Krieg.

Um diese Arbeit zu umgehen, suchte er nach geeigneten Mitteln, die ihm bei der Lösung des Problems behilflich sein könnten.



Fertiges Fahrzeug Bei dem gezeigten Modell sind die XPresso-Tags bereits integriert, wie im Objektmanager zu erkennen ist

Nach längeren Recherchen fand er unter den XPresso-Nodes ein Node namens Ray-Kollision. Für dessen Funktion wird einmal, ausgehend vom globalen Positionsvektor eines jeden Rades, zum Y-Anteil (Höhe) ein sehr hoher Wert addiert und einmal ein sehr hoher Wert subtrahiert. Dadurch erhält man einen sehr hohen positiven Wert und einen sehr hohen negativen Wert. Zwischen diesen beiden exakt übereinanderliegenden Punkten erzeugt das Ray-Kollision-Node einen virtuellen Strahl, und errechnet damit die Auftreffposition des Strahls auf den Untergrund.

An diese Position setzt man nun sein jeweiliges Rad. Da allerdings der Mittelpunkt des Rades auf den Auftreffpunkt gesetzt wird, steckt es somit zur Hälfte unter der Erde. Um das Rad nun exakt auf der Oberfläche entlanglaufen zu lassen, addiert man zur Y-Position des Rades noch einmal dessen Radius hinzu. Zwar hat ein Polygon-Rad nicht mehr die Eigenschaft „Radius", allerdings kann man sich mit der halben Y-Höhe der das Rad umgebenen Bounding-Box behelfen. Auf diese Art und Weise laufen alle Räder immer korrekt auf dem Untergrund.

Der Hebemechanismus

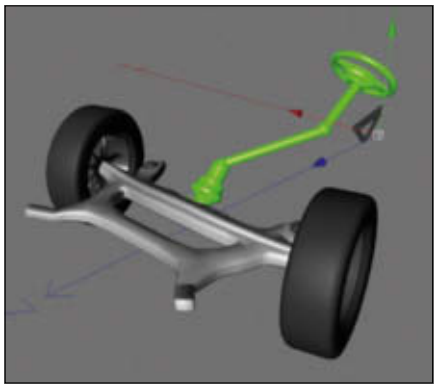
Das Problem der Räder auf dem Untergrund war gelöst. Allerdings bleibt die Karosserie starr auf einer Höhe. Als Lösung wählte der 3D-Experte ein Modell für den vereinfachten Hebemechanismus der Karosserie, da der physikalisch korrekte Ansatz diesen Artikel sprengen würde. Dieser vereinfachte Mechanismus erwies sich als sehr realistisch.

Der Hebemechanismus steuert nur den Schwerpunkt der Karosserie. Egal in welchem Winkel sich die Karosserie befindet, entweder liegen alle Räder auf einer Höhe, oder zwei Reifen standen im gleichen Verhältnis oberhalb des Schwerpunktes wie die beiden Reifen unterhalb.

Daraus ergab sich die Schlussfolgerung, dass sich die Höhe des Karosserie-Schwerpunktes aus der Summe der einzelnen Radhöhen geteilt durch vier ergibt. Allerdings muss noch ein manuell einstellbarer Offset zur Höhe hinzuaddiert werden, da der Schwerpunkt der Karosserie im Normalzustand nicht genau auf Höhe der Reifenmittelpunkte liegt, sondern etwas erhöht. Schon mit diesem Modell lässt sich der Hebemechanismus sehr gut abbilden.

Der automatische Lenkeinschlag

„Die Berechnung des automatisch ausgeführten Lenkeinschlags ist mathematisch

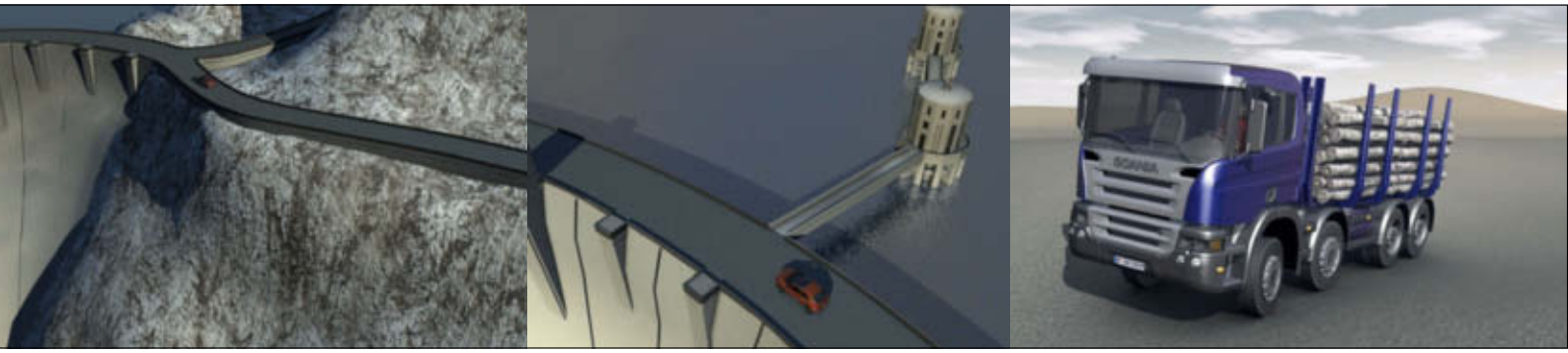


Lenkung Auch das Lenkrad reagiert auf den Achseinschlag der Räder

gesehen eher schwere Kost, da hierfür auf trigonometrische Funktionen wie den Tangens und den Arkustangens zurückgegriffen werden muss", umschreibt Sebastian Krieg die nächste Problemstellung auf dem Weg zur realistischen Physik-Engine.

Gegeben war ein Fahrzeug, das sich auf einem Spline entlang bewegt und Änderungen sowohl in der Z-Richtung als auch in der X-Richtung zeigt. „Die Lösung für dieses Problem schien schwer. Ich wusste nicht genau, wie ich an diese Sache herangehen sollte. Glücklicherweise kam mir ein Diplomand der ESG zur Hilfe, der die Problemstellung aus seiner Diplomarbeit im Bereich Fahrzeugtechnik kannte", fügt Krieg an. Sie setzten sich gemeinsam an die Lösung des Problems. Nach einer Weile erhielten sie für den Lenkwinkeleinschlag eine Lösungsformel, welche die Änderung des Winkeleinschlags pro Frame berechnet.

Hierbei werden Berechnungen aus den aktuellen und den vorherigen globalen X- und Z-Koordinaten des Fahrzeugs, die Länge zwischen den Mittelpunkten der Vorder-

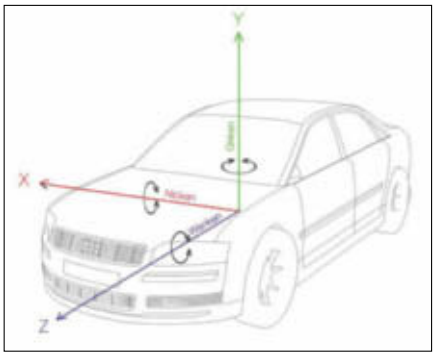


und Hinterachse sowie die Länge zwischen Schwerpunkt und Mittelpunkt der Hinterachse benötigt.

Das Nick-/Wankmodell

Waren die bisherigen Probleme für den motivierten Entwickler noch gut zu erarbeiten, so stieß er beim Nick-/Wankmodell sowohl im mathematischen, physikalischen, als auch im fahrzeugtechnischen Verständnis an seine Grenzen. Er musste über seine bisherigen Fähigkeiten hinauswachsen. Die Karosserie eines Fahrzeugs wird bei der Fahrt über eine beliebige Strecke verschiedensten Kräften ausgesetzt. So treten zum Beispiel bei Beschleunigungs- beziehungsweise Bremssituationen Nickbewegungen auf. Bei Kurvenfahrten, Bodenwellen sowie ähnlichen Einflüssen treten weiterhin Wankbewegungen auf. Diese Bewegungen galt es in mathematische Formeln zu betten.

Um die Abläufe zu diesem Mechanismus zu verstehen, waren lange Diskussionen mit unterschiedlichen Ingenieuren und Diplomanden der ESG notwendig. Zwar lag die meiste Arbeit bei ihm, doch einige Tipps zu den einzelnen Verfahren halfen ihm auf die Sprünge. „Die mehrmonatige Entwicklung dieses Modells nahm die meiste Zeit in Anspruch", erinnert sich Krieg. „Es brachte mich fast zur Aufgabe des Projektes." Das Herzstück der XPresso-Schaltung des Nick-/Wankmodells ist eine mit den anderen Modell-Berechnungen verflochtenes C.O.F.F.E.E.-Node. Dieser Node besteht aus zehn Eingängen und zwei Ausgängen. In die Eingänge laufen einerseits die Werte für die Längs- und Querkräfte und andererseits die aktuelle Höhe der einzelnen Räder sowie die Geschwindigkeit, mit der deren Höhenänderungen im Vergleich



Nick-/Wankmodell Hier sind deutlich die jeweiligen Achsen und ihre Funktion zu sehen

zum vorherigen Frame erreicht wurde. An den Ausgängen erhält man den aus den Eingangswerten und der Fliehkraft errechneten Wankwinkel und den aus den Eingangswerten und der Längskraft errechneten Neigungswinkel der Karosserie.

„Das Ganze war soweit halb so schlimm. Kern der Physik-Engine ist das Innere des C.O.F.F.E.E.-Nodes. In ihm wird aus einer Kombination von Systemmatrix, dem vorherigen Zustandsvektor, der Eingangsmatrix und dem Eingangsvektor der aktuelle Zustandsvektor", erklärt er.

Hierzu werden dutzende von Matrizen und Vektoren miteinander verknüpft und berechnet. Dazu kommen noch weitere Berechnungen, welche die Kernformeln ergänzen. Als Ergebnis erhält man an den Node-Ausgängen seine gewünschten Winkel für die Nick- und Wankbewegung der Karosserie. Zur Kontrolle der Formeln überprüft er die Ergebnisse mit Hilfe von Matlab, eine Software zur Lösung mathematischer Probleme und zur Darstellung der Ergebnisse.

Grundlagen und Methoden

„Wer bestrebt ist, selbst einmal eine Physik-Engine zu entwickeln, sollte nicht die dazu notwendigen Grundlagen und Vorkenntnisse unterschätzen. Allzu schnell gelangt man an die Grenzen seines aktuellen Wissens", sagt Krieg aus seiner eigenen Erfahrung.

Die fortgeschrittenen Erfahrungen im Umgang mit den in Cinema 4D enthaltenen XPresso-Nodes und C.O.F.F.E.E. kann man da schon beinahe vernachlässigen. „Durch mein Studium der Wirtschaftsinformatik und fünf Jahre Berufserfahrung als Anwendungsentwickler war dieser Wissensbereich für mich glücklicherweise recht leicht zu erlernen," sagt er weiter und fügt an: „Worauf es viel mehr ankommt ist das spezifische Wissen, das die Grundlagen einer Physik-Engine ausmacht. Hierfür benötigt man nicht unerhebliche mathematische Grundkenntnisse, die Basis für die gesamte Arbeit sind. Ohne sie kann ein Entwickler die Arbeit gar nicht beginnen, da sie auch die Basis für die weiteren erforderlichen Kenntnisse bilden."

Den Hauptanteil macht lineare Algebra aus, welche sich mit Vektorräumen und den linearen Abbildungen zwischen ihnen beschäftigt. Dazu gehören auch lineare Gleichungssysteme und Matrizen. Vektoren und Matrizen sind in einem solchen Projekt für die Berechnung von Zustandsänderungen unerlässlich. Sie werden in der Engine beispielsweise für das Nick-, Wank- und Hebemodell verwendet. In diesen Betrachtungen geht es zum großen Teil um Änderungen der Richtung, welche mit Vektoren und Matrizen

Erwerb ausgestatteter Fahrzeuge

Die Physik-Engine an sich ist momentan noch nicht kommerziell erhältlich. Allerdings bestehen zwei Möglichkeiten, an mit der Physik-Engine ausgerüstete Fahrzeuge zu kommen. Zum einen lassen sich ausgerüstete Fahrzeuge über die Firma DOSCH DESIGN erwerben. Diese bietet unterschiedliche Fahrzeugpakete mit bis zu 20 Fahrzeugen an, von normalen Autos und Jeeps bis hin zu schweren, vierachsigen Lkws. Diese qualitativ hochwertigen Fahrzeugmodelle sind direkt in eigenen Szenen einsetzbar.

Die zweite Möglichkeit ausgerüstete Fahrzeuge zu bekommen ist, dass Interessenten ihre eigenen Fahrzeugmodelle an die 3d-bewegungswerkstatt schicken, welche dort gegen eine Aufwandspauschale korrekt mit der Physik-Engine ausgestattet und dann zurückgeschickt werden.

Kontakt: Sebastian Krieg, 3d-bewegungswerkstatt
E-Mail: skrieg@3d-bewegungswerkstatt.de
Web: www.3d-bewegungswerkstatt.de

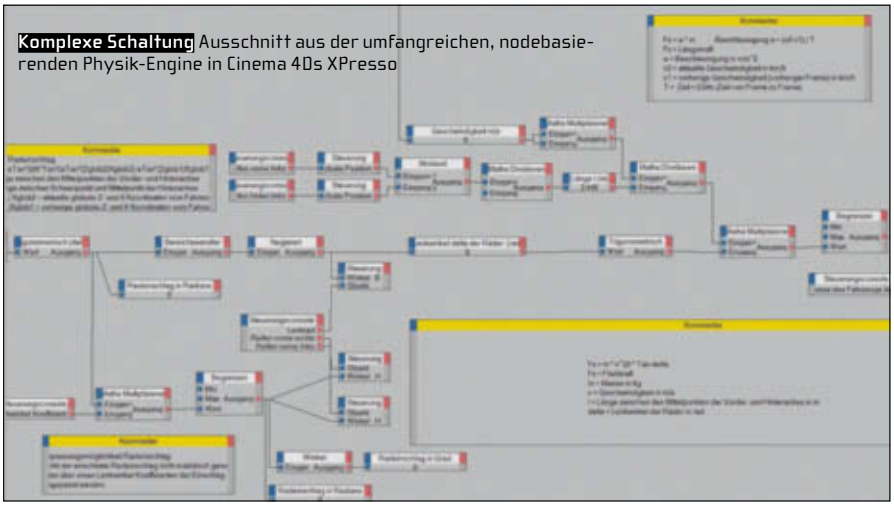
berechnet werden können. Dazu kommt die Differentialrechnung zum Einsatz, die eng mit der Integralrechnung verwandt ist. Zentrale Bedeutung hat sie bei der Berechnung lokaler Veränderungen von Funktionen. So ist beispielsweise mit Berechnung der Tangentensteigung für die Fahrbahn der Lenkeinschlag der Räder zu ermitteln.

Die Integralrechnung ist neben der Differenzialrechnung der wichtigste Teil der Analysis. Mit ihr lassen sich Berechnungen durchführen, die auf Werteunterschiede zwischen Zeitpunkten beruhen. Die Werte zwischen diesen Zeitpunkten werden mit der Integralrechnung angenähert. Dieses Verfahren findet sich in den physikalischen Berechnungen der Engine wieder. So muss die zurückgelegte Strecke bei zunehmender oder abnehmender Beschleunigung zwischen den fest vorgegebenen Zeitpunkten per Integralrechnung näherungsweise berechnet werden. Neben den mathematischen sind auch physikalische Vorkenntnisse unerlässlich. Dieses Gebiet ist nicht ganz so klar abgesteckt, wie die benötigte Mathematik. Für die physikalischen Vorgänge werden Modelle und Gleichungen aus unterschiedlichen Bereichen benötigt.

So braucht man beispielsweise Gleichungen für Beschleunigung und Gravitation, die auf ein Fahrzeug einwirken, sowie die Gesetzmäßigkeiten für die Berechnung der auf ein Fahrzeug einwirkenden Kräfte wie die Quer- und Längsbeschleunigung. Hierfür müssen die passenden Modelle zusammengesucht werden und dann auf Grundlage der Mathematik ausgewertet werden. Letztlich kommen noch die fahrzeugtechnischen Grundlagen hinzu.

„Mathematik und Physik sind für die Entwicklung einer Physik-Engine ohne Wert, wenn dem Entwickler die Kenntnisse über Fahrzeugtechnik und Fahrzeugverhalten fehlen", bringt Krieg es auf den Punkt. „Hier geht es beispielsweise um den Einfluss von Radstand, Masse und Federung auf das Fahrverhalten, um Kennwerte für die Raddämpfung, den maximalen Radeinschlag oder die Bodenhaftung. Ebenso findet sich hier das Einspurmodell."

Für diese Modelle müssen Kennzahlen und Gesetzmäßigkeiten recherchiert und auf Grundlage der mathematischen und physikalischen Vorkenntnisse berechnet werden. Nur bei Kenntnis all dieser verschiedenen Grundvoraussetzungen kann die Realisation einer Physik-Engine gelingen. Fehlt eines dieser Gebiete, ist solch ein Vorhaben nicht mehr zu realisieren.

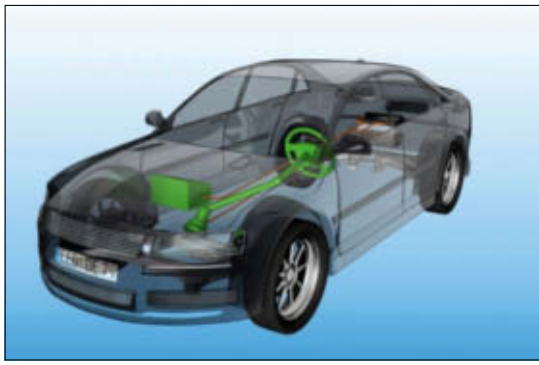


Komplexe Schaltung Ausschnitt aus der umfangreichen, nodebasierenden Physik-Engine in Cinema 4Ds XPresso

Vorzüge der Physik-Engine

Warum sollte ein Kunde Interesse an einer Physik-Engine haben, wenn es sogar schon ein paar auf dem Markt gibt? Die Frage beantwortet Sebastian Krieg zum Teil so: „Zum einen unterscheidet sich die Umsetzung von anderen Engines. Während alle auf dem Markt verfügbaren Engines auf einem zu installierenden Plug-in basieren, handelt es sich hier um eine frei verwendbare Schaltung, die ohne Installation auf jedem Rechner ausgeführt werden kann. Der besondere Vorteil liegt darin begründet, dass man bei einem Versionswechsel von Cinema 4D oder beim Wechsel des Betriebssystems den Code nicht erneut kompilieren muss, so wie es bei Plug-ins üblicherweise der Fall ist. Wenn Anbieter von Plug-ins eines Tages die Betreuung aufgeben, kann dieses in neuen Versionen von Cinema 4D nicht mehr verwendet werden. Diese Engine dagegen kann von Version 10 bis weit in die Zukunft verwendet werden."

Ein weiterer Vorteil ist die absolut einfache Handhabung der mit der Physik-Engine ausgestatteten Fahrzeuge. „Um eine physikalisch korrekte Fahrt etwa durch die Straßen einer Architekturvisualisierung auszuführen, muss der Anwender lediglich das ausgerüstete Fahrzeug in die Szene per Copy-and-Paste einfügen und den gewünschten Streckenspline sowie das Untergrundobjekt per Drag-and-Drop vom Objektmanager in zwei Textboxen ziehen", umschreibt es Krieg. Der benötigte Fahr-Spline muss nur in der Oberansicht passend zur Straße liegen, wobei dessen Höhe irrelevant ist. Per Druck auf den Abspiel-Button fährt das Auto vom Anfang des Splines bis zu seinem Ende mit allen dazugehörigen Bewegungen, ohne auch nur einen einzigen Keyframe gesetzt zu haben. Nur wenn man Stopps und Geschwindigkeitsänderungen durchführen möchte, sind ein paar wenige Keys nötig. Auch Szenen mit beliebig vielen Fahrzeugen sind kein Problem, da jedes Fahrzeug seine eigene Steuerung hat.



Der letzte Vorteil liegt in den sehr realistischen Bewegungen der Karosserie und der Räder. „Da fast sämtliche Bewegungen auf Grundlage von realen, physikalischen Gesetzen berechnet werden, sind diese per manueller Key-Animation nicht nachzustellen", bringt es der 3D-Experte auf den Punkt und fügt an. „Der Aufwand für komplexe Fahranimationen reduziert sich auf wenige Minuten Einrichtungszeit. Und sollte man einen Teil der Fahranimation doch per Hand animieren wollen, so lässt sich die Fahrphysik kontrolliert ein- und ausschalten."

Zielgruppen

Zielgruppe für diese Physik-Engine sind alle Berufssparten, bei denen es auf physikalisch korrekte Fahranimationen ankommt und bei denen mit geringem Aufwand 3D-Szenen mit Leben gefüllt werden sollen: Etwa Architekten, die ihre Architekturszenen mit Fahranimationen erweitern wollen, oder Werbefilme, wo oft realistische Fahrzeug-Animationen benötigt werden. Die Engine ist so variabel einsetzbar, dass sie all diese Bereiche abdeckt. › jb

Sebastian Krieg ist 3D-Freelancer und Gründer des Animationsstudios 3d-bewegungswerkstatt. Er absolvierte erfolgreich seine Studiengänge als Diplom-Wirtschaftsinformatiker und Bachelor of Arts in der Fachrichtung Computeranimation. Neben seiner Tätigkeit als Freelancer gibt er Seminare und Workshops für Cinema 4D, doziert an Hochschulen und hält zahlreiche Vorträge über XPresso-Programmierung.