Hoffmann, J. E., “3D-Computer Animation - Wenn die Bilder laufen lernen”, DOS International (www.pc-magazin.de), 05/1997, 292-296.

Jan Eric Hoffmann

Grafik und Animation auf dem heimischen Rechner werden immer erschwinglicher. So sind moderne Videokarten fast ausschließlich mit 3D-Beschleunigern ausgestattet. Dieser Beitrag zeigt, wie Sie Ihre Grafik mit mathematischen Verfahren zum Leben erwecken können.

3D-Computer-Animation

Wenn die Bilder laufen lernen

Bewegte Bilder bestimmen die Medienwelt. Während bis vor wenigen Jahren noch handgezeichnete Cartoons („Tiny Toons“) allgegenwärtig waren, sind es heute Computer-Tricks, die uns mit einer Flut von bunten Bildern überschwemmen. Nicht nur die Werbung hat längst die Chancen moderner Präsentation begriffen, auch das Kino bringt rechnerunterstützt immer spektakulärere Filme hervor – man denke an Terminator II oder Independence Day.

Sogar der heimische PC bleibt von dieser Bilderflut nicht verschont. Nachdem heute kaum noch Grafikkarten ohne Videobeschleunigung, speziell für Windows, auf den Markt kommen, wird es demnächst keine Grafikkarte mehr ohne „3D-Beschleunigung“ geben. Vielleicht sind Sie ja sogar schon Besitzer eines Video-Beschleunigers, der Ihren Spielen „Dampf macht“.

Interessieren Sie sich nun auch noch fürs Programmieren, so steht der Entwicklung eigener 3D-Computer-Animationen nichts mehr im Wege. Freilich, es ist leichter gesagt als getan, denn das Programmieren eigener Animationen erforderte bis vor kurzem gute Assembler-Kenntnisse, da die Hochsprachen-Routinen nicht schnell genug waren, um komplexe 3D-Animationen darstellen zu können. Die 3D-Beschleuniger haben in dieser Beziehung allerhand geändert. Alle Funktionen zur Darstellung von dreidimensionalen Objekten, die früher in handoptimierte Assemblerroutinen gepackt wurden, um die nötige Geschwindigkeit zu erlangen, sind heute in die Videokarte beziehungsweise ein API des Betriebssystems eingebaut.

Keyframing

Eine weit verbreitete und von jedem Animationsprogramm

zur Verfügung gestellte Animationstechnik ist das „Keyframing“. Bei diesem Verfahren gibt das Programm den Status der Animation an verschiedenen Stützstellen (Keys) vor. Der Computer berechnet dann eigenständig mit entsprechenden mathematischen Verfahren (Interpolations- oder Approximationsverfahren) den Zustand zwischen den einzelnen „Keys“. Im Vergleich zum Herstellen eines Zeichentrickfilms ist es also nicht nötig, für jedes zu berechnende Bild den Animationszustand explizit anzugeben. Der Vorgang der Zwischenwertberechnung wird auch mit „inbetweening“ bezeichnet.

Um jedoch realistische Animationen zu erzeugen, wie zum Beispiel die Bewegung eines Menschen, ist es nötig, Animationen mit sehr vielen „Keys“ zu erzeugen. Ein Wert von fünf bis 15 Keys pro Sekunde ist durchaus üblich. Eine so hohe Anzahl von Stützstellen ist erforderlich, da bei den meisten Interpolationsverfahren kein direkter Zusammenhang zwischen den zu interpolierenden Werten – wie etwa einer menschlichen Bewegung und dem Interpolationsverfahren – besteht.

Physikalische Gesetze

Ein anderer Ansatz, der diese Probleme umgeht, ist „physically based animation“. Dieses Verfahren versucht, die Animation auf Grundlage von physikalischen Gesetzen zu berechnen. Wie Sie sich vorstellen können, sind die hierfür notwendigen Berechnungen sehr aufwendig. Insbesondere entstehen Probleme dadurch, daß es sich meist nicht um Massepunkte, sondern um Körper handelt.

Ein Beispiel hierfür ist die „rigid body animation“, eine Animation von starren Körpern. Diese Methode versucht, in einer virtuellen Computer-Welt die Körper möglichst realistisch im Rahmen der physikalischen Gesetze zu simulieren. Dabei müssen die Regeln nicht unbedingt den physikalischen Verhältnissen auf der Erde entsprechen. So ist es zum Beispiel durch Anpassen der Gravitationsbeschleunigung möglich, Verhältnisse wie auf dem Mond zu simulieren.

Bänder und Federn

Ein „Rigid-body“-Animationssystem kann Objekte miteinander verknüpfen, etwa durch Bänder, Federn, Dämpfer und starre Verbindungen. Eine realistische Objektbewegung ergibt sich dann durch die auf die Objekte einwirkenden Kräfte, wie zum Beispiel Schwer-, Beschleunigungs- und Reibungskräfte. Je nach der zu simulierenden Umgebung berücksichtigt das Animationssystem auch äußere Einflüsse, wie beispielsweise den Luftwiderstand. Mit Hilfe der physikalischen Bewegungsgleichungen läßt sich die Bewegung eines Körpers als Bewegung seines Schwerpunkts berechnen. Von den Bewegungsgleichungen aus gesehen, hat man es also nur mit einem Massepunkt zu tun, so daß das Übertragen von bekannten Formeln für Massepunkte aus der Physik weiter keine Probleme bereitet.

Damit es nicht zu einem Schnitt zwischen den Objekten kommt, müssen Kollisionen zwischen den Objekten berücksichtigt werden. Überdurchschnittlich groß ist der Rechenaufwand für Objekte, die nicht aus Polygonen, sondern aus gekrümmten Flächen bestehen, wie zum Beispiel Spline-Patches. Die Lösungen lassen sich meist nicht explizit darstellen und müssen daher angenähert werden.

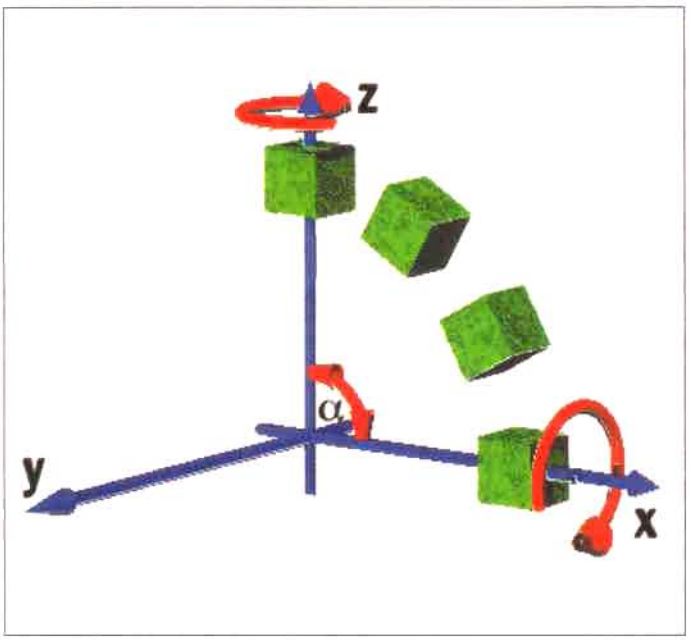

Bild 1. Es gibt mehrere Darstellungen für eine Rotation in Eulerwinkeln

Ein praktischer Ansatz ist es, am Kollisionspunkt Federn einzufügen. Bei entsprechender Wahl der Spannung und der Federkonstanten führt dies zu brauchbaren Resultaten. Probleme entstehen, wenn ein Objekt nicht in einem Punkt mit einem anderem Objekt kollidiert, sondern wenn die gemeinsamen Kollisionspunkte eine Fläche bilden. Dies ist zum Beispiel bei einem Würfel der Fall, der senkrecht auf eine Ebene fällt. Die Federtechnik kann dazu führen, daß der Würfel einen seitlichen Drall erhält, den er eigentlich nicht haben dürfte. Andere Modelle versuchen Naturphänomene, wie zum Beispiel Wind, Wolken oder Wasserwellen zu simulieren.

Bewegungsmodelle

Es gibt spezielle Bewegungsmodelle, die das Animationsprogramm bei der Animation von Tieren und Menschen unterstützen. So exisitieren neben vergleichsweise einfachen Bewegungsmodellen für Würmer und Schlangen auch kombinierte Bewegungs- und Verhaltensmodelle, etwa für Fische.

Sehr komplex ist die Animation von menschlichen Bewegungen. Hier hat es in der letzten Zeit sehr große Erfolge gegeben. So kann das 3D-Studio MAX sogenannte „BiPed-Plug-in-Bewegungen“ von Zweibeinern durch Vorgabe ihrer Fußabdrücke erzeugen. Dies geschieht durch Verwendung von Techniken, wie sie auch die „Rigid-body“-Animation verwendet, wie etwa Kollisionserkennung mit dem Boden und Gleichgewichtsberechnungen aufgrund von angreifenden Kräften sowie biomechanischen Erkenntnissen über menschliche Bewegungsabläufe.

Insbesondere ist hier das „Humanoid“-System zu nennen. Auf der Basis von Kollisionserkennung erlaubt Humanoid die gleichzeitige Animation von mehreren menschlichen Figuren. Das System verfügt über verschiedene Animations-“Generatoren“, die Bewegungen wie zum Beispiel Gehen oder Greifen erzeugen. Neben den Bewegungsgeneratoren stellt Humanoid auch Module für die integrierte Erzeugung und realistische Simulation und Animation von Hautoberflächen zur Verfügung.

Mathematischer Hintergrund

Trotz seiner Probleme führt das „Keyframing“ bei geeigneter Wahl des Interpolationsverfahrens zu beeindruckenden Er-

gebnissen – die Vielzahl von Implementationen sprechen für sich. Und viel wichtiger: Die Technik ist mit vergleichsweise wenig Aufwand zu realisieren.

Für die mathematische Formulierung der Bewegungen sollen die zu bewegenden 3D-Objekte aus Polygonen zusammengesetzt sein. Anstatt die Bewegung jedes einzelnen Punkts darzustellen, ist es günstiger, die Objekt-Animation bezüglich des Objektschwerpunkts zu beschreiben. Daß es sich dabei wirklich um den physikalischen Schwerpunkt handelt, ist nicht wichtig. Wichtig ist nur, daß dieser „Schwerpunkt" der Punkt ist, um den sich das Objekt dreht.

Mathematisch läßt sich Bewegung als Kombination von Translation (Positionsveränderung), Rotation (Drehung), Skalierung und Scherung (Dehnung) darstellen. Für die Animation von Körpern, die nicht ihre physikalischen Abmessungen ändern, reichen Translation und Rotation aus.

Um eine Keyframing-Animation zu erstellen, ist es also notwendig, zu entsprechenden Zeitpunkten Translations- und Rotationsvorgaben zu machen. Ein Programm berechnet Translation und Rotation für jeden Zeitpunkt der Animation einzeln und kombiniert den entsprechenden Bewegungszustand. Sie erhalten so eine Animation, die ihre Vorgabepunkte durchläuft (interpoliert).

Interpolation

Mathematisch handelt es sich hierbei um ein Interpolationsproblem. Für die Translationsinterpolation läßt sich die Interpolationsfunktion als Kurve im Raum veranschaulichen, die die vorgegebenen Positionspunkte interpoliert. Dieses Interpolationsproblem kann durch verschiedene Ansätze gelöst werden.

Nehmen wir an, wir hätten (n+1) Stützstellen (Keys) (t_i,p_i) mit $p_i=(x_i,y_i,z_i)$, wobei die t_i den Zeitpunkt angeben, bei dem unsere Objektbewegung die Position p_i annehmen soll. Als Interpolationsfunktion suchen wir also eine Funktion $\varphi(t)$, die die Bedingnung $\varphi(t_i)=p_i$ (i=0,1..n) erfüllt. Diese Funktion soll uns für jeden Zeitpunkt $t\in[t_0,t_n]$ einen Punkt im Raum liefern, der die Position des von uns animierten Objekts repräsentieren soll.

Die einfachste Lösung besteht darin, alle Punkte durch Geraden zu verbinden. Die Gleichung für das i-te Geradenstück g_i, das die Punkte p_i und p_{i+1} verbindet, entnehmen Sie Formel 1.

Formel 1

$$g_i(t)=p_i+\frac{t-t_i}{t_{i+1}-t_i}(p_{i+1}-p_i),\ t\in[t_i,t_{i+1}]$$

Das Ergebnis wäre jedoch eine sehr unnatürliche Art der Bewegung: sehr „zackig". Sofern die Simulation natürliche und keine roboterähnlichen Bewegungen ausführen soll, ist die sogenannte „lineare Interpolation" für die Interpolation von Translationen ungeeignet.

Der nächste Ansatz ist, anstatt der Geraden die verschiedenen Punkte durch ein Polynom zu verbinden. Ein Polynom ist eine Funktion (Formel 2).

Formel 2

$$f(x)=a_0+a_1x+a_2x^2+\ldots+a_nx^n$$

„n" heißt der „Grad" des Polynoms. Sind (t_i,p_i) die zu interpolierenden Punkte, dann wird ein Polynom gesucht, das für alle (n+1)-Punkte die Bedingung $f(t_i)=p_i$ (i=0,1...n) erfüllt. Um das Interpolationspolynom zu erhalten, ist es notwendig, die Koeffizienten $a_0...a_n$ zu bestimmen.

Ein naheliegender und rein mathematischer Ansatz ist es, ein Gleichungssystem mit den Interpolationsbedingungen aufzustellen. Die benötigten Gleichungen sind in Formel 3 dargestellt.

Formel 3

$$\begin{aligned}
a_0+a_1t_0+a_2t_0^2+K+a_nt_0^n&=p_0\\
a_0+a_1t_1+a_2t_1^2+K+a_nt_1^n&=p_1\\
&M\\
a_0+a_1t_n+a_2t_n^2+K+a_nt_n^n&=p_n
\end{aligned}$$

Es läßt sich zeigen, daß es genau ein Polynom gibt, das diese Bedingung erfüllt. Die Lösung des Gleichungssystems existiert und ist eindeutig. Für (n+1)-Punkte ist dies ein Polynom n-ten Grades.

Bei der Berechnung mit dem Computer ist es jedoch nicht empfehlenswert, die Koeffizienten $a_0...a_n$ durch Lösung des Gleichungssystems zu bestimmen, da dieses meist eine sehr schlechte „Kondition" hat. Das heißt, es ist sehr anfällig gegen Rundungsfehler und kann somit zu numerischen Instabilitäten führen. Besser ist es, das Interpolationspolynom nach den stabilen Verfahren von Lagrange oder Newton zu bestimmen.

Die Interpolation durch Polynome eignet sich leider nur für eine relativ kleine Anzahl von Punkten (<10). Da die Animation von Objekten jedoch teilweise sehr viele Punkte erfordert, um die gewünschte natürliche Bewegung zu erzeugen, ist Interpolation mittels Polynomen für uns ungeeignet. Was wir benötigen, sind Splines.

Splines

Der Begriff „Spline" stammt aus dem Schiffbau und bezeichnet einen dünnen Stab, im deutschen auch als „Straklatte" bezeichnet. Dieses Werkzeug diente zum Bestimmen des Verlaufs der in Längsrichtung verlaufenden Planken. Das Prinzip des „Strakens" haben sich die Mathematiker abgeschaut und definierten einen Spline als die Kurve, die durch vorgegebene Punkte so verläuft, daß die nötige Deformationsenergie minimal ist. Die Grundidee beschreibt die Kurve nicht als Ganzes, sondern stückweise durch Polynome niedrigen Grades. Die gesamte Kurve ergibt sich dann durch die Aneinanderreihung der einzelnen Kurvenstücke.

Mittlerweile gibt es eine Fülle von Spline-Kurven. Wir wollen uns speziell mit der Hermite- oder Ferguson-Darstellung des kubischen Splines beschäftigen. Ein Kurvenstück wird dabei durch ein Polynom dritten Grades (kubische Funktion, deshalb auch „kubischer Spline") repräsentiert (Formel 4).

Formel 4

$$s_i(t)=a_i(t-t_i)^3+b_i(t-t_i)^2+c_i(t-t_i)+d_i,\ t\in[t_i,t_{i+1}]$$

Damit es zu einem sauberen Übergang zwischen den verschiedenen Kurvenstücken kommt, muß zusätzlich zur Position auch noch gefordert werden, daß die ersten Ableitungen (Steigungen im Punkt) interpoliert werden. Dadurch erhalten sie eine einmal stetig differenzierbare Funktion. Das bedeutet, daß sich die Geschwindigkeit des animierten Objekts nicht sprunghaft ändert, sondern daß sich eine Geschwindigkeitsänderung gleichmäßig vollzieht. Üblicherweise wird für andere Arten von Splines (etwa B-Splines) neben der Stetigkeit der ersten Ableitung auch die Stetigkeit der zweiten Ableitung gefordert. Dann verläuft auch die Beschleunigung gleichmäßig, ohne Sprünge. Für unsere Zwecke reicht jedoch die Stetigkeit der ersten Ableitung aus. Die Gleichung für unser Interpolationspolynom s_i zeigt Formel 5.

Formel 5

$$s_i(t)=3a(t-t_i)^2+2b(t-t_i)+c$$

Für die zwei aufeinander folgenden Punkte p_i,p_{i+1} und ihre ersten Ableitungen (Tangenten) p'_i,p'_{i+1} gilt es also, die Bedingungen von Formel 6 zu erfüllen. $\Delta t_i=t_{i+1}-t_i$ sei dabei die Länge des Zeitintervalls zwischen der i-ten und (i+1)-ten Stützstelle:

Formel 6

$$\begin{aligned} s_i(t_i) &= d_i &&= p_i \\ s'_i(t_i) &= c_i &&= p'_i \\ s_{i+1}(t_{i+1}) &= a_i\Delta t_i^{\,3} + b_i\Delta t_i^{\,3} + c_i\Delta t_i + d_i &&= p_{i+1} \\ s'_{i+1}(t_{i+1}) &= 3a_i\Delta t_i^{\,2} + 2b_i\Delta t_i + c_i &&= p'_{i+1} \end{aligned}$$

Nach ein wenig Rechnerei erhält man die fehlenden Koeffizienten a_i und b_i des Polynoms (Formel 7).

Formel 7

$$a_i = \frac{2(p_i - p_{i+1})}{\Delta t_i^{\,3}} + \frac{2(p'_i - p'_{i+1})}{\Delta t_i^{\,2}}$$

$$b_i = \frac{2(p_{i+1} - p_i)}{\Delta t_i^{\,3}} + \frac{2(p'_i - p'_{i+1})}{\Delta t_i^{\,2}}$$

Jetzt werden nur noch die ersten Ableitungen benötigt. Diese werden einfach als Mittelwert aus den beiden anliegenden Sekanten angenommen (Formel 8).

Formel 8

$$p'_i = \frac{1}{2}\left((p_i - p_{i-1}) + (p_{i+1} - p_i)\right) = \frac{1}{2}(p_{i+1} - p_{i-1})$$

Entsprechend gilt Formel 9.

Formel 9

$$p'_{i+1} = \frac{1}{2}(p_{i+2} - p_i)$$

Hermite-Splines, bei denen die Tangenten auf die obige Weise berechnet werden, bezeichnet man auch als „Catmul Rom"-Splines. Nach dem Einsetzen der Koeffizienten a_i,b_i,c_i,d_i ergibt sich für das Kurvenstück s_i die gesamte Formel 10.

Formel 10

$$s_i(t) = p_i\left(2\left(\frac{t-t_i}{\Delta t_i}\right)^3 - 3\left(\frac{t-t_i}{\Delta t_i}\right)^2\right) + p'_i\Delta t_i\left(\left(\frac{t-t_i}{\Delta t_i}\right)^3 - 2\left(\frac{t-t_i}{\Delta t_i}\right)^2 + \left(\frac{t-t_i}{\Delta t_i}\right)\right)$$
$$+p_{i+1}\left(-2\left(\frac{t-t_i}{\Delta t_i}\right)^3 + 3\left(\frac{t-t_i}{\Delta t_i}\right)^2\right) + p'_{i+1}\Delta t_i\left(\left(\frac{t-t_i}{\Delta t_i}\right)^2 - \left(\frac{t-t_i}{\Delta t_i}\right)^2\right)$$

Kommen wir nun zum schwierigsten Teil bei der Interpolation von Bewegungen, der Interpolation der Rotation. Üblicherweise werden Rotationen durch sogenannte „Eulerwinkel" beschrieben, die eine beliebige Rotation durch die hintereinander ablaufende Ausführung von drei Rotationen um entsprechende orthogonale (senkrecht aufeinander stehende) Raumachsen beschreiben. Im einfachsten und üblichen Fall entsprechen diese Raumachsen der X-, Y- und Z-Achse. Für die Interpolation von Rotationen sind Eulerwinkel ungeeignet, da sie keine eindeutige Parametrisierung der Rotation darstellen. Das bedeutet, es gibt mehrere Darstellungen für eine Rotation in Eulerwinkeln. Ein Beispiel zeigt Bild 1.

Führen Sie erst eine beliebige Drehung um die X-Achse durch und danach eine Drehung um 90 Grad um die Y-Achse. Anstatt um die X-Achse zu drehen, kann man also nach der Y-Drehung um 90 Grad einfach die Z-Achse entsprechend um den negativen Winkel drehen. Dadurch, daß die Y-Drehung die X- und Z- Achse aufeinander dreht, wird es unmöglich, eine Drehung um die X-Achse auszuführen. Dies führt zum Verlust eines Freiheitsgrads. Diesen Effekt bezeichnet man auch als „glimbal lock". Mathematisch kommt der „glimbal lock" durch eine Singularität in der Eulerwinkel-Parametrisierung zustande.

Als Problem für die Interpolation von Rotationen macht sich insbesondere bemerkbar, daß es mehr als einen Weg zwischen zwei Rotationen geben kann. Für das Erstellen einer Animation ist es aber enorm wichtig, im voraus zu wissen, welche Zwischenwerte generiert werden sollen. Hierfür ist es im allgemeinen notwendig, daß der kürzeste Weg zwischen zwei Rotationen gewählt wird (Bild 2).

Stellen Sie sich ein Objekt vor, dessen Mittelpunkt sich irgendwo auf der Z-Achse befindet. Als Endzustand der Rotation nehmen wir das an der X,Y-Ebene gespiegelte Objekt. Als Zwischenwerte kommen hier sowohl die einfache Rotation um die X-Achse sowie die kombinierte Rotation um Z- und Y-Achse in Frage. Für die Rotationsinterpolation von Objekten sind Eulerwinkel daher ungeeignet. Die Probleme der Eulerwinkel entstehen, weil sie keine geeignete Parametrisierung für die Beschreibung von Rotationen sind. Dies liegt insbesondere daran, daß der Vektorraum der Rotationen kein einfacher dreidimensionaler Vektorraum ist, sondern eine geschlossene, gekrümmte, dreidimensionale Mannigfaltigkeit.

Eine andere Möglichkeit, Rotationen darzustellen, ist, eine Rotationsachse und einen Rotationswinkel anzugeben. Dies wollen wir jedoch nicht weiterverfolgen, da es eine viel besser geeignete Darstellungung von Rotationen gibt: die Quaternionen. Es ist sehr einfach, Quaternionen in die Darstellung von Rotationsachse/-winkel $\vec{u}$ und entsprechend die Präsentation von Rotationsachse/-winkel in Quaternionen umzurechnen.

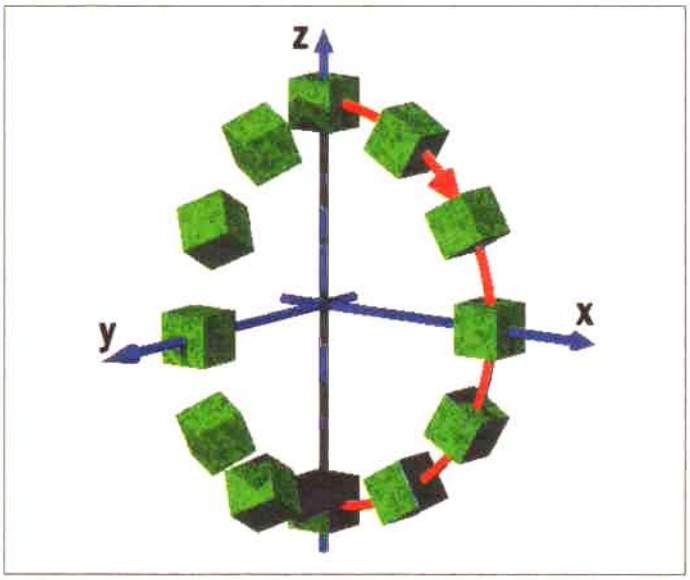

Bild 2. Für eine Animation wird der kürzeste Weg zwischen zwei Rotationen gewählt

Quaternionen

Für die Beschreibung von Rotationen in der Computer-Grafik

und -Animation haben sich die „Einheits-Quaternionen" für die Darstellung von Rotationen bewährt. Quaternionen sind hyperkomplexe Zahlen. Eine komplexe Zahl ist definiert als a+bi mit $i^2=-1$.

Komplexe Zahlen spielen eine wichtige Rolle in der höheren Analysis. Sie ermöglichen unter anderem das Ziehen von negativen Wurzeln. Quaternionen sind eine Erweiterung der komplexen Zahlen. Statt einer imaginären Einheit besitzen sie drei. Sie wurden von Sir William Hamilton in die Mathematik eingeführt. Hamilton hatte über Jahre hin versucht, die komplexen Zahlen zu erweitern und eine sinnvolle Multiplikation auf Tripeln (dreidimensionale Vektoren) zu definieren. Am 16. Oktober 1843 – er war auf dem Weg zur Royal Irish Academy auf der Broome Bridge in Dublin – erkannte er, daß eine Multiplikation zwar auf Tripeln nicht möglich ist, dafür aber auf komplexen Zahlen mit drei imaginären Einheiten mit folgenden Eigenschaften:

$i^2=j^2=k^2=-1$, $ij=k$, $ji=k$

Hamilton ritzte seinen Geistesblitz mit einem Messer in die Broome Bridge und verewigte ihn so. Die Zahl

$q=xi+yi+zk+w$

nannte er eine „Quaternion". Üblich ist es auch, den Imaginärteil als Vektor aufzufassen:

$$q=\left(w,\vec{v}\right)=w+v_x i+v_y j+v_z k$$

w bezeichnet man auch als den Realteil von q.

Die Multipikation von zwei Quaternionen $q_1=\left(w_1,\vec{v}_1\right)$ und $q_2=\left(w_2,\vec{v}_2\right)$ läßt sich dann sehr kompakt wie in Formel 11

Formel 11

$$q_1\cdot q_2=\left(w_1w_2-\left\langle\vec{v}_1,\vec{v}_2\right\rangle,\ w_1\vec{v}_2+w_2\vec{v}_1+\vec{v}_1\times\vec{v}_2\right)$$

angeben. Dabei ist $\left\langle\vec{v}_1,\vec{v}_2\right\rangle$ das Vektor-Skalarprodukt zwischen den Vektoren $\vec{v}_1$ und $\vec{v}_2$. Zu beachten ist bei der Quaternionenmultiplikation, daß sie nicht kommutativ ist. Das bedeutet, es ist nicht egal, in welcher Reihenfolge die Multiplikationen durchgeführt werden. Es gilt daher im allgemeinen $q_1\cdot q_2\neq q_2\cdot q_1$.

Zu jeder Quaternion $q=\left(w,\vec{v}\right)$ existiert ein konjugiertes Quaternion $\bar{q}=\left(w,-\vec{v}\right)$.

Dieses ermöglicht die einfache Berechnung des Skalarprodukts zweier Quaternionen (Formel 12)

Formel 12

$$\left\langle q_1,q_2\right\rangle=\frac{1}{2}\left(q_1\cdot\bar{q}_2+q_2\cdot\bar{q}_1\right)=w_1w_2+x_1x_2+y_1y_2+z_1z_2$$

und der euklidischen Norm eines Quaternions (Formel 13).

Formel 13

$$\|q\|=\sqrt{\left\langle q,q\right\rangle}=\sqrt{q\cdot\bar{q}}=\sqrt{w^2+x^2+y^2+z^2}$$

Besonders wichtig für die Anwendung ist, daß Quaternionen normtreu unter der Multiplikation sind. Dies sichert unter anderem, daß hintereinander ausgeführte Drehungen als Multiplikation von Quaternionen aufgefaßt werden können:

$$\|q_1\cdot q_2\|=\|q_1\|\cdot\|q_2\|$$

Eine Quaternion wird invertiert, indem man es konjugiert und zusätzlich durch seine Länge teilt und es so zu einer Einheits-Quaternion macht:

$$q^{-1}=\frac{\bar{q}}{\|q\|}$$

Einen dreidimensionalen Vektor $\vec{u}$ dreht man mit Quaternionen q, indem man $\vec{u}$ als Quaternion ohne Realteil auffaßt:

$$v_{rot}=q^{-1}\cdot v\cdot q\quad\left(v=\left(0,\vec{u}\right)\right)$$

v_{rot} ist wieder eine Quaternion, bei dem der Realteil Null ist. Der Grund dafür liegt darin, daß Rotationen orthogonale Abbildungen sind und somit Skalar- und Kreuzprodukt erhalten.

Formel 14

$$R=\begin{pmatrix}1-2v_y^2-2v_z^2 & 2v_xv_y-2wv_z & 2v_xv_z+2wv_y\\ 2v_xv_y+2wv_z & 1-2v_x^2-2v_z^2 & 2v_yv_z-2wv_x\\ 2v_xv_z-2wv_y & 2v_yv_z+2wv_x & 1-2v_x^2-2v_y\end{pmatrix}$$

Rotationsmatrix

Die Rotationsmatrix zu einer Einheits-Quaternion $q=\left(w,\vec{v}\right)$, $\|q\|=1$ erhält man durch Formel 14.

Am einfachsten läßt sich ein Quaternion aus der Darstellung von Rotationsachse/-winkel gewinnen. Für eine Rotation von Θ um die Achse $\vec{n}$ lautet die entsprechende Quaternion:

$$q=\left(\cos\left(\frac{\Theta}{2}\right),\ \frac{\vec{n}}{\|\vec{n}\|}\cdot\sin\left(\frac{\Theta}{2}\right)\right)$$

Wir sind nun in der Lage, Rotationen mit Quaternionen darzustellen. Was uns noch fehlt, ist die Interpolation zwischen einzelnen Quaternionen. Da es sich hier um einen gekrümmten Raum handelt, ist das auch nicht so einfach wie für die oben besprochene Interpolation im R^3.

Die Einheits-Quaternionen bilden im Vektorraum der Quaternionen eine Untergruppe bezüglich der Multiplikation, da die Multiplikation normtreu ist. Das bedeutet: Multipliziert man zwei Einheits-Quaternionen, erhält man eine Quaternion, die wieder die Länge eins hat. Da die Untergruppe der Einheits-Quaternionen genau die Quaternionen enthält, die die Länge eins haben, entspricht sie der Oberfläche einer vierdimensionalen Kugel. Eine lineare Interpolation zwischen zwei Quaternionen kann daher erklärt werden als der kürzeste Weg auf der vierdimensionalen Kugeloberfläche. Diese lineare Interpolation für Quaternionen auf der Kugeloberfläche wird als SLERP (für Spherical Linear intERPolation) bezeichnet (Formel 15):

Den notwendigen Winkel Θ erhält man aus dem Skalarprodukt von q_1 und q_2:

$$\cos\Theta=\left\langle q_1,q_2\right\rangle$$

Um professionelle Ergebnisse bei der Interpolation von Rotationen zu erhalten, reicht diese einfache lineare Interpolation von Quaternionen natürlich nicht aus. Zum Erzeugen von Spline-Quaternionen-Kurven ist jedoch das Exponentieren und Logarithmieren von Quaternionen nötig. Glücklicherweise fallen die Unstetigkeitsstellen, die bei der linearen Quaternioneninterpolation entstehen, nicht so ins Auge, wie es bei der Positionsinterpolation der Fall ist. Die lineare Interpolation von Quaternionen führt daher bereits zu beeindruckenden Ergebnissen.

In der nächsten Ausgabe des PC Magazins DOS setzen wir die gewonnenen Erkenntnisse in reale Programmierung um. Die Beispielprogramme zeigen den portablen Einsatz von Grafikroutinen und versetzen Ihre Programme in die Lage, schnelle Animationen im „state of the art" zu zeigen. *wr*

Formel 15

$$q=\mathrm{Slerp}\left(q_1,q_2,t\right)=\frac{\sin\left((1-t)\cdot\Theta\right)}{\sin\Theta}q_1+\frac{\sin\left(t\cdot\Theta\right)}{\sin\Theta}q_2$$