Visualisierung der Hypersphäre

Softwarepraktikum im WS 2002/03 bei
Dr. Michael Winckler und Dr. Susanne Krömker

Maximilian Albert
Anhalter42@gmx.de

Heidelberg, den 14. April 2003

1 Einleitung

Der vorliegende Bericht dokumentiert die Arbeit an meinem Softwarepraktikum im Wintersemester 2002/03 an der Universität Heidelberg, in dessen Rahmen ein Programm zur Visualisierung der vierdimensionalen Kugel erstellt werden sollte. Angeregt wurde ich dazu durch einen uralten Artikel in einer 1988er Ausgabe von *Spektrum der Wissenschaft*. Dieser enthielt eine kurze Referenz zu den Arbeiten von Thomas Banchoff[1] sowie einige Bilder eines von ihm erstellten Trickfilmes, der Animationen der ins Dreidimensionale projizierten Hyperkugel zeigte. Da mich diese Bilder schon damals faszinierten, beschloss ich, dieses Thema bei meinem Softwarepraktikum aufzugreifen.

In diesem Bericht soll zunächst der mathematische Hintergrund erläutert werden. Anschließend wird kurz auf das Benutzerinterface des Visualisierungsprogramms eingegangen, und zuletzt folgt eine eingehendere Beschreibung der verwendeten C++ Klassen und Algorithmen.

2 Notation und Sprachgebrauch

Weil in diesem Artikel häufig genug von Kugeln in unterschiedlichen Dimensionen die Rede sein wird und ich zur Vereinfachung der Sprechweise einige "inoffizielle" Begriffe verwende, ist es angebracht, vorab die Terminologie zu klären.

Unsere Untersuchungen spielen sich hauptsächlich im $\mathbb{R}^3$ und $\mathbb{R}^4$ ab; es wird angenommen, daß der Leser mit dem $\mathbb{R}^n$ und den Grundlagen der Linearen Algebra im $\mathbb{R}^n$ vertraut ist. Das zentrale von uns betrachtetet Objekt ist die vierdimensionale Einheitskugel $S^3 = \{x = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4 \mid |x| = 1\}$ die wir oft auch einfach "Hyperkugel" nennen. In Veranschaulichungen und im Zusammenhang mit der Hopf-Abbildung taucht auch die dreidimensionale Einheitskugel $S^2 = \{x \in \mathbb{R}^3 \mid |x| = 1\}$ auf, die wir schlicht als "Kugel" oder auch "die gewöhnliche Kugel" bezeichnen (man beachte, daß der Index in S^3 und S^2 jeweils

[1] Professor an der Brown University, Providence; vgl. auch [1]

um eins geringer ist als die Dimension des die Kugeln enthaltenden Raumes; dies liegt daran, daß die S^k eine k-dimensionale Mannigfaltigkeit ist und dieses Charakteristikum daher zur Bezeichnung verwendet wird). Streng genommen müßte man in der Bezeichnung zwischen "Kugel" (=Vollkugel) und "Sphäre" (Kugeloberfläche) unterscheiden. In Anbetracht der Tatsache, daß im von uns betrachteten Zusammenhang keine gefüllten Kugeln vorkommen, werden wir diese beiden Terme allerdings synonym für die Kugeloberfläche gebrauchen.

Eine zentrale Rolle spielt ebenfalls die stereographische Projektion (sie wird in Abschnitt 3.3 erläutert), wobei wir das Adjektiv "stereographische" gelegentlich weglassen, weil wir keine anderen Projektionsarten betrachten werden.

Schließlich treten des öfteren Kreise auf, die innerhalb der S^2 oder S^3 liegen (d.h. Schnittmengen dieser Kugeln mit geeigneten Ebenen). Derartige Kreise sollen "2-sphärische Kreise" bzw. "3-sphärische Kreise" genannt werden. Der Begriff des Hopf-Kreises (ein spezieller 3-sphärischer Kreis) wird in Abschnitt 3.2 definiert. Allgemein sollen beliebige Teilmengen der Hyperkugel in Ermangelung eines kreativeren Namens schlicht "Hypermengen" heißen.

3 Mathematik

3.1 Überblick

Das im Rahmen des Praktikums entwickelte Programm HyperView dient zur Visualisierung der Hyperkugel. Die übliche Verfahrensweise bei der Visualisierung höherdimensionaler Objekte ist die Projektion in den dreidimensionalen Raum. Aus den verschiedenen zur Verfügung stehenden Möglichkeiten kommt in unserem Fall die stereographische Projektion zur Anwendung, die weiter unten ausführlicher erläutert wird (der Vorteil gegenüber anderen Projektionsarten ist der, daß die Hyperkugel *bijektiv* ins Dreidimensionale abgebildet werden kann). Der naive Ansatz, einfach die gesamte Hyperkugel zu projizieren, schlägt jedoch fehl, da das Bild der gesamte $\mathbb{R}^3$ wäre (vgl. Abschnitt 3.3). Wir können daher immer nur Teile der Hyperkugel gleichzeitig abbilden.

Als nächstes stellt sich die Frage, welche Hypermengen dafür am geeignetsten sind. Man stelle sich die analoge dreidimensionale Situation vor: Eine transparente Kugel stehe mit dem Südpol auf der Ebene und im Nordpol sei eine punktförmige Lichtquelle angebracht. Färbt man Punkte auf der Kugeloberfläche schwarz ein, so repräsentiert ihr Schattenbild die Projektion auf die Ebene. Der Schattenwurf der gesamten Kugel ist analog zu oben die komplette Ebene. Man kann die Einfärbung jedoch beispielsweise auf einige Breitenkreise beschränken und erhält dann als Bild unter Projektion eine Familie von konzentrischen Kreisen in der Ebene.

Dieses Beispiel wird auch in dem eingangs erwähnten *Spektrum der Wissenschaft*-Artikel herangezogen mit der Bemerkung, daß Banchoff bei seinen Animationen in vollkommener Analogie die vierdimensionalen "Breitenkreise" der Hyperkugel abbildet. Das ist jedoch falsch und hat mich bei der Einarbeitung in das Thema anfangs ziemlich verwirrt, weil ich mir nicht vorstellen konnte, wie aus simplen Breitenkreisen durch Projektion derart deformierte und verschlungene Tori entstehen können, wie sie auf Banchoffs Bilder zu sehen sind. In der

Tat erkennt man durch kurzes Nachdenken, daß die Projektion eines vierdimensionalen Breitenkreises ins Dreidimensionale eine gewöhnliche Kugel oder gar lediglich ein Kreis ist (je nachdem wie man den a priori im Vierdimensionalen gar nicht definierten Begriff "Breitenkreis" interpretiert), also keine Spur von Tori.

Nach einigem Nachforschen bin ich darauf gestossen, daß Banchoff einen zusätzlichen Trick anwendet, um mehr von der Struktur der Hyperkugel zu beleuchten und die Bilder dadurch interessanter zu machen. Als zu projizierende Teilmengen der S^3 verwendet er nämlich nicht Breitenkreise, sondern kompliziertere Punktmengen, die sich mit Hilfe der in Abschnitt 3.2 beschriebenen Hopf-Abbildung gewinnen lassen. Wie die Gesamtheit aller Breitenkreise bei der Kugel bilden diese Mengen eine Partition der Hyperkugel. Wenn man die Projektionen aller dieser Mengen betrachtet, erhält man demnach ebenfalls eine Veranschaulichung der Hyperkugel, die jedoch, wie wir sehen werden, eine viel reichhaltigere Struktur zu bieten hat und daher weitaus interessantere Bilder liefert.

Im folgenden sollen nun als erstes die Hopf-Abbildung so weit wie für unsere Zwecke nötig und anschließend die stereographische Projektion untersucht werden. Zuletzt wird gezeigt, wie beide bei der Visualisierung der Hyperkugel zusammenspielen.

3.2 Die Hopf-Abbildung

Die Hopf-Abbildung[2] ist eine unter vielen Gesichtspunkten interessante stetige Abbildung von S^3 nach S^2 . Ein sehr schöner elementarer Überblick wird im Artikel [2] von David Lyons gegeben. Weitere Informationen finden sich z. B. in [3]. Wir stellen hier die für uns interessanten Eigenschaften zusammen.
Definiert ist die Hopf-Abbildung durch

$$\begin{array}{cccc} h: & S^3 & \longrightarrow & S^2 \\ & \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} & \longmapsto & \begin{pmatrix} x_1^2 + x_2^2 - x_3^2 - x_4^2 \\ 2(x_1x_4 + x_2x_3) \\ 2(x_2x_4 - x_1x_3) \end{pmatrix} \end{array}$$

Man mache sich klar, daß das Bild eines Punktes aus S^3 tatsächlich Betrag 1 hat, also in S^2 liegt (der Betrag des Bildpunktes ist nämlich gerade gleich dem Betrag des Urbildpunktes zum Quadrat, wie unmittelbar nachzuprüfen ist).

Die Hopf-Abbildung hat die wichtigen Eigenschaften, stetig (da sie in jeder Komponente polynomial ist) und surjektiv zu sein (letzteres ist nicht ganz so unmittelbar einsichtig, kann aber z. B. aus der unten angegebenen Formel (1) abgelesen werden – etwa für t=1 erhält man ein Urbild des Punktes P; man rechne dies nach). Gemäß [2] ist die Urbildmenge von $P \in S^2$ ein Kreis im $\mathbb{R}^4$. Dieser ist natürlich ganz in S^3 enthalten, also in unserer Sprechweise ein

[2] Heinz Hopf, 1884-1939

3-sphärischer Kreis. Genauer gilt[3] für $P = (p_1, p_2, p_3, p_4)$

$$h^{-1}(P) = \left\{ \frac{1}{\sqrt{2(1+p_1)}} ((1+p_1)i + p_2 j + p_3 k) \cdot (\cos(t) + \sin(t)i) \right\}_{0 \leq t \leq 2\pi} \tag{1}$$

Hierbei fassen wir $\mathbb{R}^4 = \mathbb{H}$ als Raum der Quaternionen auf, zu dem $\{1, i, j, k\}$ eine Basis ist; die Multiplikation in (1) ist die Quaternionenmultiplikation. Eine explizite Parametrisierung in Vektorform erhlt man durch Ausmultiplizieren:

$$h^{-1}(P) = \left\{ \frac{1}{\sqrt{2(1+p_1)}} \cdot \begin{pmatrix} -\sin(t) \cdot (1+p_1) \\ -\cos(t) \cdot (1+p_1) \\ p_2 \cos(t) + p_3 \sin(t) \\ p_3 \cos(t) - p_2 \sin(t) \end{pmatrix} \right\}_{0 \leq t \leq 2\pi} \tag{2}$$

Es ist leicht verifizierbar, da es sich hierbei um einen Kreis handelt, der durch eine gewisse Drehung aus dem komplexen Einheitskreis hervorgegangen ist.

Einen solchen 3-sphärischen Kreis, der von der Form $h^{-1}(P)$ mit $P \in S^2$ ist, nennen wir im folgenden einen Hopf-Kreis (bzgl. P). Interessant ist die Beobachtung, daß je zwei dieser Hopf-Kreise verschlungen sind (dies ist in [2] schön beschrieben und in den von HyperView produzierten Bildern gut zu sehen, woraus diese Bilder einen Großteil ihrer Ästhetik beziehen).

Aus der Surjektivität von h folgt, daß die Hopf-Kreise eine Partition von S^3 bilden, ähnlich wie die Breitenkreise, nur in viel interessanterer Weise. Dies nutzen wir für die Visualisierung aus, indem wir die Hopf-Kreise bezüglich gewissen Punkt*mengen* auf der S^2 betrachten und diese dann in den $\mathbb{R}^3$ projizieren. Da die Hopf-Abbildung eine sehr reichhaltige Struktur beinhaltet, liefert dies recht faszinierende Bilder.

Die Tatsache, daß die Hopf-Kreise tatsächlich Kreise sind, verschafft uns, wie wir noch sehen werden, einen großen Vorteil, weil die stereographische Projektion eines Kreises wieder ein Kreis ist, dessen genau Lage man auch berechnen kann. Dies wiederum macht es möglich, die projizierten Kreise exakt darzustellen, so daß wir nicht auf Approximationen z. B. durch Triangulierung oder ähnliche Techniken zurückgreifen müssen, wodurch die graphische Darstellung sehr kantig würde und stark an Ästhetik verlöre. Außerdem erspart dies eine Menge unnötigen Rechenaufwand.

3.3 Die stereographische Projektion

Bemerkung: Da die stereographische Projektion eine sehr elementare und häufig auftauchende Abbildung ist, ist sie dem Leser vermutlich bereits begegnet ist. Wir werden sie trotzdem kurz rekapitulieren, um anschließend eine Deutung der stereographischen Projektion als Inversion an einer Kugel zu geben. Diese Sichtweise ist für uns von großer praktischer Bedeutung, weil sie in der vorliegenden Situation eine sehr geschickte Handhabung ermöglicht (siehe z. B. den Abschnitt 6.2 über die stereographische Projektion eines Kreises).

[3] vgl. einmal mehr [2]

Sei nun K eine Kugel im $\mathbb{R}^n$ und $Z \in K$. Die zugehörige stereographische Projektion im $\mathbb{R}^n$ mit Zentrum Z ist eine bijektive Abbildung von $K \backslash \{Z\}$ auf einen $(n-1)$-dimensionalen Unterraum U von $\mathbb{R}^n$, der in dem Z diametral gegenüberliegenden Punkt tangential an K anliegt (oder auf einen geeigneten zu diesem parallelen Raum). Das Bild eines Punktes $P \in K \backslash \{Z\}$ ist der eindeutig bestimmte Schnittpunkt der Gerade ZP mit U. Üblicherweise ist $K = S^{n-1}$ die n-dimensionale Einheitskugel, $Z = (0, \ldots, 0, 1)$ und U ist einer der Unterräume $\mathbb{R}^{n-1} \times \{-1\} \subseteq \mathbb{R}^n$ bzw. $\mathbb{R}^{n-1} \times \{0\} \subseteq \mathbb{R}^n$.

Im $\mathbb{R}^3$ kann man sich die stereographische Projektion als Schattenwurf vorstellen (wie bereits in 3.1 angedeutet): Man betrachte die dreidimensionale Einheitskugel und platziere eine punktförmige Lichtquelle im Punkt $Z = (0,0,1)$. Ein von Z ausgehender Lichtstrahl, der die Kugel in einem von Z verschiedenen Punkt P trifft, trifft ebenfalls die Ebene $E = \mathbb{R}^2 \times \{-1\} \subseteq \mathbb{R}^3$ in einem eindeutig bestimmten Punkt. Der Schnittpunkt des Strahles mit dieser Ebene (also der "Schatten" von P) ist das Bild von P unter stereographischer Projektion (siehe Abb. 1).

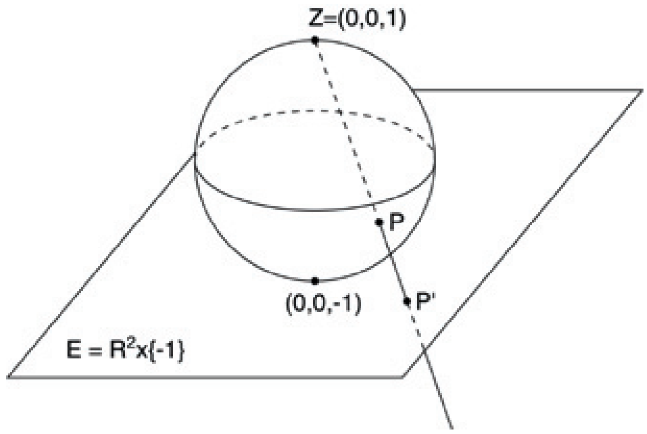

Abbildung 1: Dreidimensionale stereographische Projektion

Häufig wird statt auf die Ebene $\mathbb{R}^2 \times \{-1\}$ auch auf die hierzu parallele Ebene $\mathbb{R}^2 \times \{0\}$ projiziert. Es sei erwähnt, daß dies keinen wesentlichen Unterschied macht. Wie man sich durch ein simples Stahlensatzargument überzeugt, unterscheiden sich die resultierenden Bildmengen nämlich lediglich um eine Streckung mit Faktor $\frac{1}{2}$.

In unserem Fall betrachten wir die vierdimensionale stereographische Projektion π_4. Es ist nicht schwer, sich eine explizite Formel für das Bild eines Punktes $P = (p_1, p_2, p_3, p_4) \in S^3$ zu überlegen (dies ist im wesentlichen wieder eine Anwendung des Strahlensatzes). Es gilt:

$$\begin{array}{llcl} \pi_4: & S^3 \backslash \{Z\} & \longrightarrow & \mathbb{R}^3 \\ & (p_1, p_2, p_3, p_4) & \longmapsto & \left(\frac{p_1}{1-p_4}, \frac{p_2}{1-p_4}, \frac{p_3}{1-p_4} \right) \end{array} \tag{3}$$

Man beachte, daß diese Abbildung wegen $P \neq Z$ und daher $p_4 \neq 1$ wohldefiniert ist; auch ihre Bijektivität ist leicht einzusehen. Je nachdem, wie man die zu projizierende Ebene wählt, kommt wie gesagt ggf. noch ein geeigneter Faktor ins Spiel.

3.4 Inversion an der Kugel

Es lohnt sich, die sterographische Projektion unter einem etwas anderen Blickwinkel zu betrachten und sie als Inversion an einer Kugel zu interpretieren. Der Grund dafür liegt darin, daß die Inversion Kreise auf Kreise abbildet, wovon die Implementierung von HyperView exzessiven Gebrauch macht (vgl. z. B. Abschnitt 6.2).

Gegeben sei eine Kugel K im $\mathbb{R}^n$ mit Mittelpunkt M und Radius r. Die Inversion ι an K bildet jeden von M verschiedenen Punkt P auf einen Punkt Q ab, der durch die beiden folgenden Eigenschaften eindeutig festgelegt ist: Q liegt auf dem von M ausgehenden Halbgeraden MP, und es gilt $\overline{MP} \cdot \overline{MQ} = r^2$. Hieraus ist klar, daß $\iota \circ \iota = \mathrm{id}$ gilt, die Inversion also eine Involution ist, und daß K punktweise fix bleibt. Es ist auch klar, daß der Radius von K nicht wesentlich für die Gestalt der Bildmenge ist, weil eine Änderung des Radius lediglich eine zusätzliche Streckung bewirkt.

Die Inversion hat jedoch auch eine weitaus weniger offensichtliche Eigenschaft, die für unsere Zwecke von Nutzen ist und daher kurz ausgeführt wird. Sie bildet nämlich Kreise auf Kreise ab (wenn man Geraden als Grenzfälle von Kreisen mit unendlich großem Radius ansieht). Genauer gilt, daß die Bilder von Kreisen, die nicht durch M gehen, wieder Kreise sind. Das Bild eines Kreises, der M enthält, ist dagegen eine Gerade. Umgekehrt ist das Bild einer Gerade, die nicht durch M verläuft, ein Kreis, der M enthält. Hierbei ist selbstverständlich immer zu beachten, daß M selbst aufgrund der Definition von ι stets ausgenommen wird, d.h. weder abgebildet wird noch als Bildpunkt vorkommt. Wir wollen die angegebenen Tatsachen hier nicht beweisen. Für eine Behandlung der Inversion am Kreis sei das Buch [4] von Coxeter empfohlen (das auch sonst sehr lesenswert ist).

Die genannten Eigenschaften sind für die Inversion am Kreis wohlbekannt, gelten jedoch auch in höheren Dimensionen. Die Aussage läßt sich sogar dahingehend verallgemeinern, daß etwa bei der drei- und vierdimensionalen Inversion die Bilder von Kugeln selbst wieder Kugeln sind (wobei unter Umständen Ebenen als entartete Kugeln mit unendlich großem Radius auftreten).

Wenden wir die Erkenntnisse über Inversion nun auf die stereographische Projektion π_4 mit Zentrum $Z = (0, 0, 0, 1)$ an, so erkennen wir folgendes: Das Bild eines Punktes P liegt nach Definition der Projektion auf der von Z ausgehenden Halbgeraden ZP. Ferner ist das Bild der Kugel S^3 (die Z enthält) ein dreidimensionaler Unterraum, ohne Einschränkung $\mathbb{R}^3 \times \{0\}$. Folglich können wir π_4 auch als (vierdimensionale) Inversion an der Kugel mit Mittelpunkt Z ansehen. Der Radius der Inversionskugel muß hierbei geeignet gewählt werden. Da wie oben erwähnt eine Veränderung dieses Radius aber keinen wesentlichen Unterschied bewirkt, wollen wir im folgenden annehmen, daß ihr Radius 1 beträgt (zwar ist dann das Bild von S^3 unter Inversion nicht exakt $\mathbb{R} \times \{0\}$, sondern nur ein hierzu paralleler Unterraum, was aber für die folgende Diskussion nicht von Bedeutung ist; den oben erwähnten auftretenden multiplikativen Faktor lassen wir der Einfachheit halber weg, da die resultierende Streckung für die spätere visuelle Darstellung irrelevant ist). Durch diese Interpretation der Projektion als Inversion sehen wir, daß sich die Eigenschaft der Inversion, (dreidimensionale) Kugeln auf Kugeln und Kreise auf Kreise abzubilden, auf die stereographische Projektion überträgt.

Weil wir dieses Ergebnis in Abschnitt 6.2 brauchen werden, wollen wir hier noch explizit angeben, wie man die Bilder von Ebenen bzw. Kugeln bestimmt. Sei zunächst eine Ebene E im $\mathbb{R}^4$ gegeben. Wir wollen ihr Bild unter (vierdimensionaler) Inversion (oder äquivalent unter stereographischer Projektion) an der Kugel mit Mittelpunkt $Z = (0, 0, 0, 1)$ und Radius 1 ermitteln. Diese Inversion bezeichnen wir wie gewohnt mit ι. Für $Z \in E$ ist offensichtlich $\iota(E) = E$. Sei daher $Z \notin E$. In diesem Fall spannen Z und E einen dreidimensionalen Unterraum von $\mathbb{R}^4$ auf. Die Einschränkung von ι auf diesen Unterraum ist eine (dreidimensionale) Inversion, die wir gleichfalls mit ι bezeichnen. Wie bekannt, ist dann $\iota(E)$ eine Kugel K, die durch Z geht (vgl. Abb. 2). Der Z am nächsten liegen-

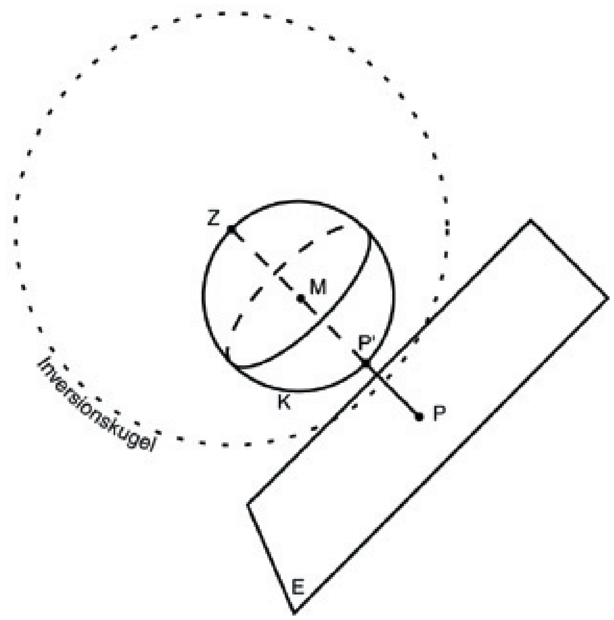

Abbildung 2: Inversion einer Ebene

de Punkt $P \in E$ wird offenbar auf den am weitesten von Z entfernten Punkt $P' = \iota(P)$ auf der Kugel K abgebildet, so daß die Strecke $\overline{Z\,\iota(P)}$ ein Durchmesser von K ist. Nach Definition von ι gilt ferner: $d := \overline{ZP'} = \left(\overline{ZP}\right)^{-1}$. Sei nun n ein normierter Normalenvektor von E. Dann ist $M := Z + \frac{d}{2} \cdot n$ der Mittelpunkt und $\frac{d}{2}$ der Radius von K, womit wir $K = \iota(E)$ vollständig spezifiziert haben. Es sei bemerkt, daß man umgekehrt das Bild einer Kugel, die Z enthält, in ähnlicher Weise bestimmen kann. Man erhält den Normalenvektor der resultierenden Ebene als den eindeutig festgelegten Durchmesser der Kugel, dessen einer Endpunkt mit Z zusammenfällt, und der Abstand des Z am nächsten liegenden Punktes der Ebene ergibt sich analog.

Sei nun eine Kugel K mit Mittelpunkt M und Radius r gegeben, die Z nicht enthält. Wir wissen bereits, daß $K' := \iota(K)$ wieder eine Kugel ist. Allerdings ist $\iota(M)$ nicht der Mittelpunkt von K'. Seien jedoch P und Q die Endpunkte des eindeutigen Durchmessers von K, der parallel zu ZM verläuft (im Fall $Z = M$ sind K und K' konzentrisch, und K' hat den Radius $\frac{1}{r}$). Für $P' = \iota(P)$ und $Q' = \iota(Q)$ ist $P'Q'$ ein Durchmesser von K' (vgl. Abb. 3). Ferner ergibt sich der Mittelpunkt von K' als der Mittelpunkt der Strecke $P'Q'$. Hierdurch ist K' festgelegt.

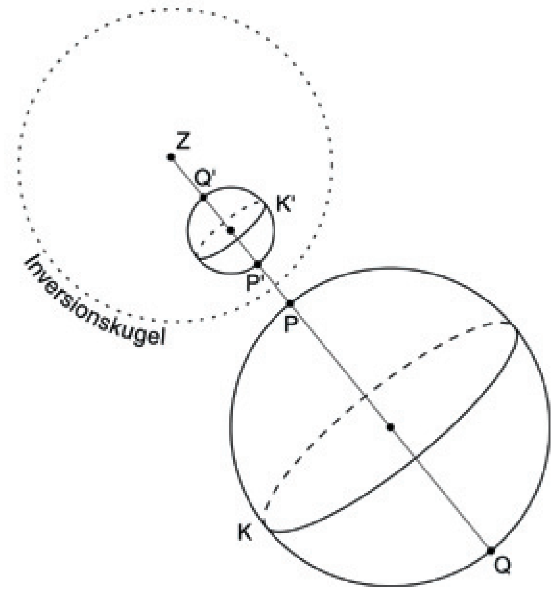

Abbildung 3: Inversion einer Kugel

Die beiden angegebenen Überlegungen finden beispielsweise in der von HyperView benutzten Klasse `Cirle3d` (siehe 5.2.3) Anwendung.

3.5 Zusammenspiel und Anwendung auf HyperView

Nun können wir Hopf-Abbildung und stereographische Projektion zusammenbauen und erhalten einen vollständigen Überblick über das, was bei der Visualisierung der Hyperkugel geschieht. Wir gehen aus von einer geeigneten Teilmenge T der S^2. Bei uns wird dies im folgenden stets ein 2-sphärischer Kreis sein. Dann betrachten wir das Urbild $h^{-1}(T)$ unter der Hopf-Abbildung. Da das Urbild eines einzelnen Punktes unter h ein Kreis ist (nämlich gerade der entsprechende Hopf-Kreis), erhalten wir in diesem Schritt eine Familie von Kreisen im $\mathbb{R}^4$ (es stellt sich heraus, daß $h^{-1}(T)$ für einen Kreis T gerade ein in S^3 eingebetteter Torus ist). Diese Urbildmenge wird nun wiederum in den $\mathbb{R}^3$ projiziert und dort graphisch dargestellt. Es hat ziemlich lange gedauert, bis ich herausfand, daß es sich bei dem Bild eines solchen Torus unter stereographischer Projektion (bzw. gleichwertig unter Inversion) um ein verhältnismäßig gut untersuchtes mathematisches Objekt handelt, das unter dem Namen "Dupinsche Cyclide" bekannt ist. Für diese existiert eine explizite Parametrisierung, was für die Bézier-Darstellung in HyperView recht praktisch ist.

Ich möchte anmerken, daß ich in den Klassenbezeichnungen von HyperView das Wort "Torus" synonym für "Cyclide" verwende. Dies liegt daran, daß ich über weite Strecken der Programmierarbeit noch nicht wußte, daß es ich bei den auftretenden Objekten um Dupinsche Cycliden handelt und sie aufgrund ihres Aussehens immer als "Tori" oder "deformierte Tori" bezeichnete. Daher stammt z. B. der Name der in HyperView verwendeten Klasse `CTorus` ("Circle-Torus" – sie ist für die Darstellung eines solchen deformierten Torus durch eine Familie von Kreisen verantwortlich). Die Bezeichnung ist insofern nicht vollkommen falsch als eine Cycliden ja das Bild eines Torus unter Inversion ist. In der Endversion des Programms werde ich diesen kleinen Fehler vermutlich korrigieren, bis dahin muß der Leser leider mit dieser "historisch" bedingten Ungenauigkeit leben. Mißverständnisse sind allerdings nicht zu befürchten.

Zusammenfassend kann man sagen, daß das Programm HyperView folgendes tut: Als Eingabe erhält es einen 2-sphärischen Kreis T und berechnet zu diesem die zugehörige Cyclide. Hierfür ermittelt es zunächst die Urbildmenge von T unter h und projizert diese anschließend in den $\mathbb{R}^3$. Einzelheiten können in Abschnitt 6 gefunden werden.

Durch exzessives Herumexperimentieren bin ich auch auf einige Zusammenhänge gestoßen, die es ermöglichen, aus der Lage des 2-sphärischen Kreises, von dem man ausgeht, die Lage und Form der resultierenden Dupinschen Cyclide vorherzusagen. Diese Beobachtungen sind allerdings noch nicht vollständig und ich habe sie bisher noch nicht streng bewiesen. Ich vermute jedoch, daß man mit ein bißchen Geschick sogar einen elementargeometrischen Beweis finden könnte. Da die Bézier-Darstellung der Cycliden in HyperView zum Teil auf diesen Erkenntnissen fußt, möchte ich sie an dieser Stelle trotz fehlenden Beweises anführen.

Unser Ausgangspunkt ist ein 2-sphärischer Kreis, festgelegt durch seine Schnittebene mit der S^2. Diese Ebene wiederum spezifizieren wir durch Normalenvektor n und Abstand d vom Ursprung. Für n und d wählen wir abweichend von den sonstigen Gewohnheiten die folgende Darstellung:

$$n = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \cdot \sin(\beta) \\ \sin(\alpha) \cdot \sin(\beta)) \end{pmatrix}, \qquad d = \sin(\gamma)$$

Hierbei können α und β beliebig zwischen 0 und 2π gewählt werden, γ hingegen soll ohne Einschränkung zwischen $-\frac{\pi}{2}$ und $\frac{\pi}{2}$ liegen.

Bemerkung: Die nachfolgenden Ausführungen gelten vorerst nur für den Fall $\beta = 0$. Bis vor kurzem glaubte ich, daß sie allgemein richtig sind, was sich leider als Trugschluß herausstellte. Die Wechselwirkungen für $\beta \neq 0$ sind komplizierter. Ich hoffe, sie demnächst zu entwirren und werde sie dann hier erläutern.

Für $\alpha = 0$ erhalten wir als vom zugehörigen Kreis auf oben beschriebene Weise induzierte Cyclide einen wunderbar symmetrischen Torus, dessen Größe von $\sin(\gamma)$ abhängt. Je näher dieser Wert bei 1 liegt, um so kleiner der Torus. Bei $\sin(\gamma) = 1$ entartet er zu einem Kreis, in Übereinstimmung mit der Tatsache, daß der zugehörige 2-sphärische Kreis dann zu dem Punkt $(1, 0, 0)$ entartet (man mache sich dies klar). Geht $\sin(\gamma)$ gegen -1, so wird der Torus unendlich groß. Lassen wir nun α von 0 bis 2π laufen, so beginnt sich der Torus in folgender Weise zu deformieren: Mit wachsendem α bläht er sich immer weiter auf, und zwar in Richtung der Achse $(\cos(\beta), \sin(\beta), 0)$. Wenn α den kritischen Wert $\gamma + \frac{\pi}{2}$ erreicht, ist der "deformierte Torus" (genauer: die Dupinsche Cyclide) unendlich groß und gewissermaßen gerade dabei, sich "umzustülpen". Wenn α über $\gamma + \frac{\pi}{2}$ hinauswächst, zieht sich die Cyclide "auf der anderen Seite" (d.h. in Richtung der Achse $(-\cos(\beta), -\sin(\beta), 0)$) wieder zusammen, bis sie bei $\alpha = \pi$ erneut zu einem symmetrischen Torus wird (allerdings womöglich mit anderem Radius als am Anfang). Man beachte, daß quasi das Innere des ursprünglichen Torus nun auf der Außenseite liegt. Je nachdem, wie nahe γ bei $-\frac{\pi}{2}$ bzw. $\frac{\pi}{2}$ liegt, vollziehen sich der Aufblähungs- und Schrumpfungsprozeß entsprechend schnell oder langsam – wenn beispielsweise γ nahe bei $-\frac{\pi}{2}$ liegt, so ist das Intervall $[0, \gamma + \frac{\pi}{2}]$ ziemlich klein und wird entsprechend schnell von α durchlaufen, was eine schnellen Aufblähung zur Folge hat; hingegen ist in diesem Fall das Intervall $[\gamma + \frac{\pi}{2}, \pi]$ verhältnismäßig groß, was eine langsame anschließende Schrumpfung zur Folge hat. Für $\alpha \approx \frac{\pi}{2}$ ist die Situation genau andersherum. Demnach gilt: Je schneller die Ausdehnung, desto langsamer das anschließende Zusammenziehen und umgekehrt.

Anschließend beginnt das Spiel in spiegelbildlicher Weise von neuem. Während α von π bis $2\pi - (\frac{\pi}{2} + \gamma) = \frac{3\pi}{2} - \gamma$ läuft, bläht sich der Torus wieder auf (erneut in Richtung der Hauptachse $(\cos(\beta), \sin(\beta), 0)$, und zwar mit derselben Geschwindigkeit, wie er sich zuvor in der zweiten Phase zusammengezogen hatte. Beim zweiten kritischen Winkel $\alpha = \frac{3\pi}{2} - \gamma$ wird er wieder unendlich groß, wobei er sich gewissermaßen "zurückstülpt" und anschließend mit dem gleichen Tempo wie in der ersten Phase zusammenschrumpft, bis er bei $\alpha = 2\pi$ seine ursprüngliche Gestalt zurückgewonnen hat.

Gleichzeitig zum eben beschriebenen Prozeß geschieht aber noch etwas: Während α wächst, beginnt sich der Cyclide nämlich auch noch um ihre Hauptachse zu drehen, und zwar mit der halben Geschwindigkeit wie α (d.h. der Winkel, den die "Grundebene" der Cyclide mit der x, y-Ebene einnimmt), ist gleich $\frac{\alpha}{2}$). Interessant ist hierbei, daß bedingt durch das "Umstülpen" beim Erreichen des kritischen Winkels $\gamma + \frac{\pi}{2}$ die Grundebene der Cyclide plötzlich um $\frac{\pi}{2}$ versetzt wirkt (was mich beim Suchen nach einer Gesetzmäßigkeit regelmäßig zur Verzweiflung brachte); dieser Effekt wird beim Zurückstülpen wieder aufgehoben.

4 Die Benutzeroberfläche von HyperView

In der nahen Zukunft wird sich die Benutzeroberfläche vermutlich noch bis zu einem gewissen Grad ändern, weswegen die genaue Bedienungsanleitung warten muß, bis nur noch marginale Modifikationen zu erwarten sind. Eine vorläufige Beschreibung findet sich auf der projektbegleitenden Webseite [5].

5 Beschreibung der Klassen

Das Herzstück von HyperView ist die Klasse `HypersphereViewer`. Sie stellt die Benutzeroberfläche zur Verfügung und verwaltet die vom Benutzer erzeugten Cycliden-Objekte. Hierbei bedient sie sich der Klasse `CTorus`, die speziell auf die Handhabung von mittels der Hopf-Abbildung erzeugten Dupinschen Cycliden zugeschnitten ist. Beide Klassen und weitere, von ihnen benutzte Klassen werden im folgenden etwas genauer beschrieben, wobei jedoch nicht auf jede Einzelheit eingegangen wird.

5.1 Die "interaktiven" Klassen

5.1.1 `HypersphereViewer`

Der `HypersphereViewer` ist von der Klasse `GenericViewer` abgeleitet, die Tobias Dyckerhoff und David Reichert im Rahmen ihres Softwarepraktikums über Quaternionenfraktale ([6]) entwickelt und verwendet haben. Sie stellt bereits die Grundfunktionalität für die graphische Anzeige mittels OpenGL zur Verfügung (insbesondere ein `gtkglarea`-Fenster, in dem die Anzeige erfolgt, sowie Routinen zur Maussteuerung, die ein interaktives Rotieren und Skalieren der angezeigten Objekte ermöglichen). Der `HypersphereViewer` erweitert den `GenericViewer` um eine Combo-Box, in der zwischen den momentan vorhandenen Tori ausgewählt werden kann, um einige Buttons zur Cycliden-Manipulation sowie um je ein Fenster, in dem die Parameter für eine neu zu erzeugende oder bereits existierende Cyclide angegeben bzw. verändert werden können. Diese Fenster sind von den zu diesem Zweck geschaffenen Klassen `AddTorusWindow` bzw. `TorusParamsWindow`, welche beide auf die Klasse `TorusParamsFrame` zurückgreifen, um die Bearbeitung der Cycliden-Parameter zu ermöglichen.

5.1.2 `Gtk_AddTorusWindow`

Eine von `Gtk::Window` abgeleitete Klasse, die ein Fenster zur Darstellung bzw. Änderung der Parameter für neu zu erzeugende Cycliden implementiert. Sie benutzt die Klasse `Gtk_TorusParamsFrame`, um den inneren Rahmen darzustellen, in dem die Parameter bearbeitet werden können.

Soll eine neue Cyclide dargestellt werden, wird das Signal `add_torus_request` ausgesendet. Der `HypersphereViewer` fängt dieses Signal ab und ruft ggf. die Funktion `HypersphereViewer::add_new_torus` auf, die eine Cyclide mit den angegebenen Parametern erzeugt.

Ferner wird die von `Gtk::Window` vererbte Methode `delete_event_impl` überschrieben, da das Fenster nicht zerstört sondern beim Schließen lediglich unsichtbar gemacht werden soll.

5.1.3 `Gtk_TorusParamsWindow`

Diese ist der Klasse `Gtk_AddTorusWindow` sehr ähnlich. Es fehlen bei ihr allerdings naturgemäß die Buttons "OK" und "Cancel", da das Parameterfenster nur die Parameter für schon existierende Cycliden anzeigt und das Fenster daher lediglich auf Benutzerbefehl hin ein- und ausgeblendet wird. Ferner verfügt sie über Methoden, die die Anzeige verändern, wenn der Benutzer über die Combo-Box im Hauptfenster eine andere Cyclide auswählt. Diese Klasse ist zum momentanen Zeitpunkt noch nicht voll implementiert.

5.1.4 `Gtk_TorusParamsFrame`

Stellt dem Benutzer diverse Widgets zur Verfügung, um die Cycliden-Parameter zu ändern; die eigentliche Verarbeitung der Daten geschieht durch die übergeordneten Klassen `AddTorusWindow` bzw. `TorusParamsWindow`.

5.2 Die "mathematischen" Klassen

Wir kommen nun zu den Klassen, die die auftretenden mathematischen Objekte modellieren und behandeln diese mit wachsender Komplexität.

5.2.1 `Vector3`

`Vector3` implementiert, wie nicht anders zu erwarten, die elementaren Funktionen im Umgang mit Vektoren des $\mathbb{R}^3$. Wir verwalten damit in der Regel Punkte des $\mathbb{R}^3$, indem wir den Vektorraum $\mathbb{R}^3$ mit dem entsprechenden affinen Raum identifizieren. (Diese Klasse wurde aus dem QuatFract-Projekt übernommen und um einige Kleinigkeiten erweitert).

5.2.2 `QuatMath`

Dies ist eine Klasse, die ebenfalls aus dem QuatFract-Projekt stammt (vgl. [6]). Sie stellt grundlegende mathematische Routinen zur Handhabung von Quaternionen zur Verfügung. Für unsere Zwecke ist sie vor allem interessant, weil Quaternionen außerordentlich geschickt für die Drehung von Vektoren im $\mathbb{R}^3$ eingesetzt werden können (siehe hierzu auch [2]). Die entsprechende Funktion in `QuatMath` heißt `q_rotate` und findet insbesondere in den Klassen `CTorus` und `Circle3d` rege Anwendung.

In der ursprünglich mit QuatFract veröffentlichten Version von `QuatMath` befand sich übrigens ein kleiner Bug in der erwähnten Funktion `q_rotate` (es lief auf einen fehlenden Faktor $\frac{1}{2}$ hinaus, wodurch allerdings Drehungen falsch berechnet wurden). In der mit HyperView ausgelieferten Version ist dieser verbessert, was jedoch geringfügige Auswirkungen auf die Drehgeschwindigkeit im Ansichtsfenster bei der Maussteuerung hat, weil die Funktion hier gleichfalls verwendet wird. Dies wird vielleicht bei Gelegenheit noch ausgeglichen werden.

5.2.3 `Circle3d`

Wie bereits im ersten Teil der Dokumentation gesehen, treten Kreise im $\mathbb{R}^3$ bei unseren Untersuchungen überall auf. Die Klasse `Circle3d` modelliert einen solchen Kreis. Er wird spezifiziert durch seinen Mittelpunkt, seinen Radius und den Normalenvektor der Ebene, in der er liegt. Momentan wird ihm auch noch eine Farbe zugeordnet, aber es wird geplant, dies ganz zu streichen und stattdessen die übergeordnete CTorus-Struktur die Färbung komplett organisieren zu lassen.

Die vorhandenen Methoden ermöglichen die Abfrage und das Setzen von Radius und Normalenvektor. Die Methode `draw` tut sinnigerweise nichts anderes, als den Kreis zu zeichnen, wobei auf die von der GLU-Bibliothek bereitgestellte Möglichkeit, Quadriken zu zeichnen, zurückgegriffen wird. Des weiteren existieren die Methoden `turn_normal` (dreht den Normalenvektor um), `get_points` (liefert eine Menge von gleichmäßig auf dem Kreis verteilten Punkten), `get_point_from_parameter` (liefert einen bestimmten, durch einen Parameter spezifizierten Punkt auf dem Kreis) und `get_intersection` (liefert einen Schnittpunkt des Kreises mit einer gewissen Ebene), die für interne Zwecke gebraucht werden. Schließlich implementiert `pre_image` eine Funktion, die aus einem Punkt auf der S^2 mit Hilfe des in Abschnitt 6.2 angegebenen Algorithmus die stereographische Projektion des entsprechenden Hopf-Kreises berechnet.

5.2.4 SPlane

Wie oben beschrieben, erhalten wir eine Dupinsche Cyclide, wenn wir die Urbildmenge unter der Hopf-Abbildung eines beliebigen 2-sphärischen Kreises stereographisch in den $\mathbb{R}^3$ projizieren. Ein solcher 2-sphärischer Kreis läßt sich am geschicktesten beschreiben als die Schnittmenge einer Ebene mit der S^2. Die Klasse `SPlane` dient zur Modellierung einer solchen Ebene. Sie wird spezifiziert durch einen Normalenvektor und den Abstand der Ebene zum Ursprung (in Richtung des Normalenvektors). Die Methode `get_intersecting_circle` liefert den Schnittkreis mit der S^2 zurück (als `Circle3d`-Struktur; vgl. auch 6.1). Im Verlaufe des Projektes hat es sich als nützlich erwiesen, daß auch Schnittkreise mit beliebigen anderen Kugeln behandelt werden können, weswegen diese Methode überlagert wurde, um dies zu leisten (anfangs mußte auch der Abstand vom Ursprung ≤ 1 sein, was später geändert wurde).

5.2.5 CTorus

Die wichtigste Klasse im Umgang mit den Cycliden ist zweifellos `CTorus`. Sie modelliert eine Cyclide, die die Projektion einer Familie von Hopf-Kreisen darstellt, welche wiederum als Urbildmenge eines 2-sphärischen Kreises unter der Hopf-Abbildung entstehen. Intern wird als definierendes Element der Cyclide dieser 2-sphärische Kreis gespeichert, und zwar in Form der zugehörigen Schnittebene mit der S^2. Ferner wird die gewünschte Darstellungsmethode der Cyclide gepspeichert; sie kann als `CIRCS` (als Schar von Kreisen), `FOLIATED` (mit Bänderung), `MESH` (als Gitternetz) und `SOLID` (in Festkörperansicht) gewählt und über `set_drawmethod` geändert werden. Auch die Wahl des Färbungsstils (`UNI` – einfarbig, wobei noch durch `set_color` eine Farbe festzulegen ist, bzw. `SMOOTH` – mit weichem Übergang) ist möglich. Gezeichnet schließlich wird mit `draw`.

Durch die Lage des 2-sphärischen Kreises auf der S^2 lassen sich im Prinzip Lage und Gestalt der Cyclide schon genau bestimmen (vgl. Ende von Abschnitt 3.5). Entsprechende Daten (u.a. Lage der "Hauptachse", Grad der "Aufblähung") werden intern ermittelt und in privaten Variablen gespeichert. Sie finden Verwendung bei der MESH- bzw. SOLID-Darstellung der Cyclide. An dieser Stelle vielleicht noch ein Wort zum “Grad der Aufblähung”. Man stelle sich einen Längsschnitt durch eine Cyclide vor, wobei die Schnittebene durch die Hauptachse der Cyclide und senkrecht zu ihrer Grundebene verläuft. Das Resultat sind zwei Kreise K_1 und K_2 (in gewissem Sinn der kleinste und der größte Kreis, die in der Cyclide auftreten; siehe Abbildung 4 links).

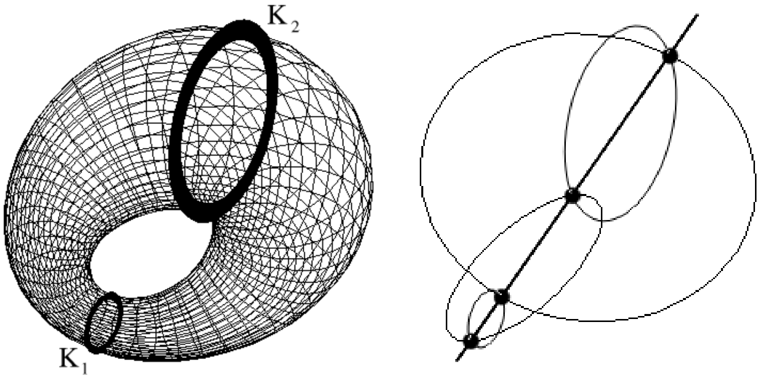

Abbildung 4: Die Kreise K_1 und K_2 (links) und ihre Schnittpunkte mit der Hauptachse der Cyclide (rechts)

Es sind die Radien dieser beiden Kreise, die intern ermittelt und gespeichert werden. Dies wiederum funktioniert empirisch, indem (wie bei der CIRCS-Darstellung, s.u.) die Projektionen der Hopf-Kreise zu zwei geeignet auf dem ursprünglichen 2-sphärischen Kreis gewählten Punkten und anschließend deren Schnittpunkte mit der Hauptachse der Cyclide berechnet werden. Die resultierenden vier Punkte stimmen genau mit den Schnittpunkte von K_1 und K_2 mit der Hauptachse überein (zur Erläuterung vgl. Abb. 4).

In [7] wird am Ende des letzten Kapitels angedeutet, daß man mit Hilfe dieser Daten eine genaue Parametrisierung der Cyclide angeben kann. In CTorus wird nun diese Parametrisierung mittels vier Bézier-Flächen realisiert, die zusammengenommen die gesamte Oberfläche abdecken. Die Methode calc_control_point_parameters berechnet die entsprechenden Kontrollpunkte der Oberfläche (siehe auch Abschnitt 6.3). Bei der Programmierung der Bézier-Darstellung war für mich unter anderem der Artikel [8] (insbesondere Abschnitt 4.2) hilfreich.

Ferner beinhaltet CTorus eine private Member-Variable points_on_S2 vom Typ vector<Vector3> (Anmerkung: “vector” ist eine Struktur aus der von mir verwendeten Standard Template Library, siehe auch [9]). Die CIRCS-Darstellung der Cyclide erfolgt durch Anzeigen einer Schar von Kreisen, welche die Projektion von Hopf-Kreisen sind, die wiederum als Urbilder von gewissen Punkten auf dem zugrundeliegenden 2-sphärischen Kreis entstehen. Diese Punkte sind auf dem Kreis gleichverteilt (nebenbei bemerkt wird diese Gleichverteilung durch fortgesetzte Drehung eines Radiusvektors durch die oben erwähnte Funktion q_rotate erzielt) und werden nach der Berechnung in points_on_S2 gespeichert; ihre Anzahl ist durch die private Variable num_circles festgelegt. Die zugehörigen Projektionen der Hopf-Kreise, die letztendlich zur Anzeige dienen, werden in einer `vector<Circle3d>`-Struktur namens `torus` abgelegt. Bei Änderung der Anzahl der darzustellenden Kreise durch den Benutzer erfolgt ggf. eine Neuberechnung beider Strukturen.

Es ist zum gegenwärtigen Zeitpunkt noch nicht völlig klar, wie die gebänderte Darstellung der Cyclide erfolgen wird, aber vermutlich wird sie ebenfalls von den in `torus` gespeicherten Kreisen Gebrauch machen und diese evtl. durch simple Polygonzüge oder durch Spline-Interpolationen miteinander verbinden.

6 Algorithmen

In diesem Abschnitt sollen einige Berechnungsverfahren, die in den Klassen von HyperView zur Anwendung kommen, noch ein wenig eingehender erläutert werden, als dies bisher geschehen ist. Teilweise sind diese so simpel, daß der Ausdruck Algorithmus fast schon eine Übertreibung darstellt, aber in Ermangelung eines besseren Wortes heißt der Abschnitt jetzt nunmal so.

6.1 Schnittkreis zwischen Kugel und Ebene

Eine Berechnung, die immer wieder durchgeführt werden muß ist die des Schnittkreises zwischen einer Ebene und einer Kugel. Diese Funktion ist in der Klasse `SPlane` implementiert. Gegeben ist eine `SPlane`-Struktur (diese enthält den Normalenvektor $n = (n_1, n_2, n_3)$ der Ebene sowie deren Abstand d vom Ursprung) sowie der Mittelpunkt $M = (M_1, M_2, M_3)$ und Radius r einer Kugel. Gesucht sind Mittelpunkt und Radius des entsprechenden Schnittkreises. Da man Aufgaben dieses simplen Typs zur Genüge im Gymnasium behandelt, stellt die Lösung keine Probleme dar. Die Implementierung in `SPlane` folgt genau dem bekannten elementargeometrischen Lösungsverfahren: Die Gerade durch M mit Richtungsvektor n schneidet die Ebene im Mittelpunkt M' des Schnittkreises. Sein Radius ermittelt sich über den Satz von Pythagoras aus dem Radius der Kugel und der Länge der Strecke $\overline{MM'}$.

Gelegentlich passiert es, daß wir den Schnittkreis einer Ebene mit einer *vierdimensionalen* Kugel bestimmen müssen. In diesem Fall ermitteln wir vorab den durch ihren Mittelpunkt und die gegebene Ebene aufgespannten dreidimensionalen Raum U (sollte der Unterraum nicht drei-, sondern nur zweidimensional sein, also M in der Ebene liegen, so läßt sich der Schnittkreis direkt ablesen). Anschließend verfahren wir wie zuvor. Ein in U liegender Normalenvektor zu E läßt sich hierbei etwa mit Hilfe des Schmidtschen Orthogonalisierungsverfahrens aus einem Vektor gewinnen, der die aufspannenden Vektoren von E zu einer Basis von U ergänzt.

6.2 Stereographische Projektion eines Hopf-Kreises

Wir haben nun zur Genüge gesehen, daß bei der `CIRCS`-Darstellung der Cyclide stereographische Projektionen von Hopf-Kreisen berechnet werden müssen. Wie in [2] erwähnt kann man zu einem Punkt auf der S^2 eine Parametrisierung des entsprechenden Hopf-Kreises direkt angeben (vgl. auch Formel (1) in

Abschnitt 3.2). Diese Parametrisierung wird mit Hilfe einer Quaternionenmultiplikation mit dem Quaternion

$$r = r_1 i + r_2 j + r_3 k = \begin{pmatrix} 0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \frac{1}{\sqrt{2(1+p_1)}} \cdot \begin{pmatrix} 0 \\ 1+p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

beschrieben (und intern in HyperView auch mit Hilfe der Klasse `QuatMath` so gehandhabt), wobei p_1, p_2 und p_3 die Koordinaten des Urbildpunktes waren. Man kann diese Quaternionenmultiplikation jedoch gleichermaßen als eine orthogonale lineare Abbildung in $\mathbb{R}^4$ darstellen. Die beschreibende Matrix lautet

$$A = \begin{pmatrix} 0 & -r_1 & -r_2 & -r_3 \\ r_1 & 0 & -r_3 & r_2 \\ r_2 & r_3 & 0 & -r_1 \\ r_3 & -r_2 & r_1 & 0 \end{pmatrix}$$

Die Vektoren $e_1 = (1,0,0,0)$ und $e_2 = (0,1,0,0)$ bilden eine Orthogonalbasis des $\mathbb{R}^2 \subseteq \mathbb{R}^4$, und die ihnen entsprechenden Punkte liegen auf dem komplexen Einheitskreis. Gleiches gilt dann natürlich für die Vektoren $A \cdot e_1$ und $A \cdot e_2$ in bezug auf den Bildkreis. Ist daher ein Punkt $P \in S^2$ gegeben, so können wir die Ebene E, deren Schnitt mit S^3 den Hopf-Kreis $H = h^{-1}(P)$ zu P liefert, durch die zwei aufspannenden Vektoren

$$\begin{pmatrix} 0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} -r_1 \\ 0 \\ r_3 \\ -r_2 \end{pmatrix}$$

direkt angeben (aus Formel (1) ist klar, daß die Ebene durch den Ursprung verläuft).

Wir betrachten nun den von E und dem Projektionszentrum $Z = (0,0,0,1)$ aufgespannten dreidimensionalen Unterraum U (wir wollen annehmen, daß $Z \notin E$ gilt; sonst ist das Bild des einzigen Hopfkreises, der durch Z verläuft, einfach die x-Achse) und beschränken unsere Überlegungen auf U, was ohne weiteres möglich ist, weil sowohl H als auch $\pi_4(H)$ in U liegen. Der Schnitt von U mit S^3 ist eine dreidimensionale Kugel K.

Nun wenden wir die Überlegungen aus 3.3 und 3.4 an. Das Bild von E unter der stereographischer Projektion π_4 ist eine in U enthaltene dreidimensionale Kugel E', und $\pi_4(K)$ ist eine in U enthaltene Ebene K' (da die Kugel K nach Konstruktion durch das Projektionszentrum N verläuft). Wegen $H = E \cap K$ folgt demnach $H' := \pi_4(H) = E' \cap K'$. Sowohl die Ebene K' als auch die Kugel E' können leicht explizit angegeben werden (vgl. Abschnitt 3.4), so daß es mit Hilfe elementarster Gymnasialgeometrie wie in Abschnitt 6.1 beschrieben sofort möglich ist, Mittelpunkt und Radius ihres Schnittkreises H' (und damit der Projektion des Hopfkreises H) zu berechen. Es stellt sich erstaunlicherweise heraus, daß die Darstellung von H durch r_1, r_2 und r_3 besonders schlicht und elegant wird. Für den Mittelpunkt von H erhält man so etwa den überraschend einfachen Ausdruck $(\frac{r2}{r1}, \frac{r3}{r1}, 0)$ und der Normalenvektor der H enthaltenden Ebene entpuppt sich als

$$\begin{pmatrix} 0 \\ r1 \\ r2 \end{pmatrix} \times \begin{pmatrix} -r1 \\ 0 \\ r3 \end{pmatrix} = r_1 \cdot \begin{pmatrix} r_3 \\ -r_2 \\ r_1 \end{pmatrix}$$

Schließlich ist der Radius des Schnittkreises gerade gleich $\frac{1}{||r_1||}$. Die Eleganz dieses Ergebnisses ist nicht von der Hand zu weisen und hat mich persönlich, als ich die Werte das erste Mal berechnete, ausgesprochen überrascht.

6.3 Gitternetzmodellierung mit Bézier-Flächen

Die Darstellungen `MESH` und `SOLID` erfordern eine vollkommen andere Herangehensweise als `CIRCS` und `FOLIATED`. Die Berechnung der einzelnen Hopf-Kreis-Projektionen macht hier keinen Sinn, weil die Oberfläche als Ganzes dargestellt werden soll. Zu diesem Zweck wird die Cyclide mit Hilfe von Bézier-Flächen modelliert, wobei die in OpenGL eingebaute Unterstützung von Bézier-Flächen ("Evaluators") zum Einsatz kommt. Für die korrekte Kontrollpunkt-Berechnung der Flächen ist die Methode `CTorus::calc_control_point_parameters` zuständig. Sie greift in Teilen auf die gegen Ende von Abschnitt 3.5 beschriebenen empirischen Beobachtungen zurück.

`calc_control_point_parameters` ermittelt zunächst die dort genannten Daten und berechnet anschließend die Evaluator-Kontrollpunkte so, daß die vier Surface-Patches mit dem Parameterbereich $[0,1] \times [0,1]$ jeweils ein Viertel der Cycliden-Oberfläche abdecken (in Abb. 5 sind die einzelnen Teile getrennt dargestellt). Anschließend wird noch eine entsprechende Translation durchgeführt, so daß die Flächen nicht nur die richtige Gestalt haben sondern sich auch an der richtigen Position befinden. Die Bestimmung des Translationsvektor erfolgt hierbei ebenfalls empirisch, indem testweise ein Kreis aus der `CIRCS`-Darstellung mit der Lage der Bézier-Patches verglichen wird.

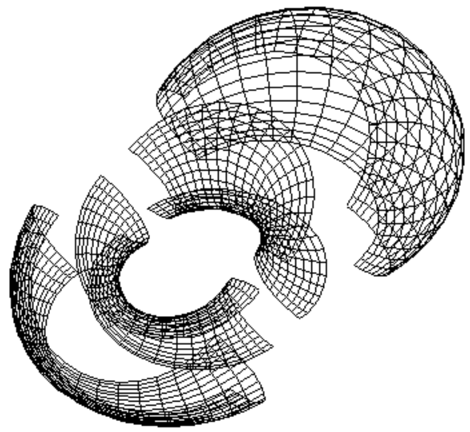

Abbildung 5: Die vier Bézier-Patches der `MESH`-Darstellung

Literatur

[1] Thomas Banchoff. http://www.math.brown.edu/~banchoff/.

[2] David Lyons. An Elementary Introduction to the Hopf Fibration.
http://csunix1.lvc.edu/~lyons/notes/hopf_pre.pdf.

[3] Horst Knörrer. *Geometrie.* Vieweg, 1996.

[4] Harold S. M. Coxeter. *Unvergängliche Geometrie.* Birkhäuser, 1963.

[5] Maximilian Albert. Webseite zum Softwarepraktikum "HyperView".
http://pille.iwr.uni-heidelberg.de/albert/.

[6] Tobias Dyckerhoff and David Reichert. Softwarepraktikum über die Darstellung von Quaternionenfraktalen.
http://pille.iwr.uni-heidelberg.de/quatfract, 2002.

[7] Vladimir Rovenski. *Geometry of Curves and Surfaces with MAPLE.* Birkhäuser, 2000.

[8] Rimvydas Krasauskas. Shape of toric surfaces.
http://www.maf.vu.lt/katedros/cs2/cagl/pdf/sh_ts.pdf.

[9] The Standard Template Library.
http://www.sgi.com/tech/stl/.

[10] The GNU General Public License.
http://www.gnu.org/copyleft/gpl.html.