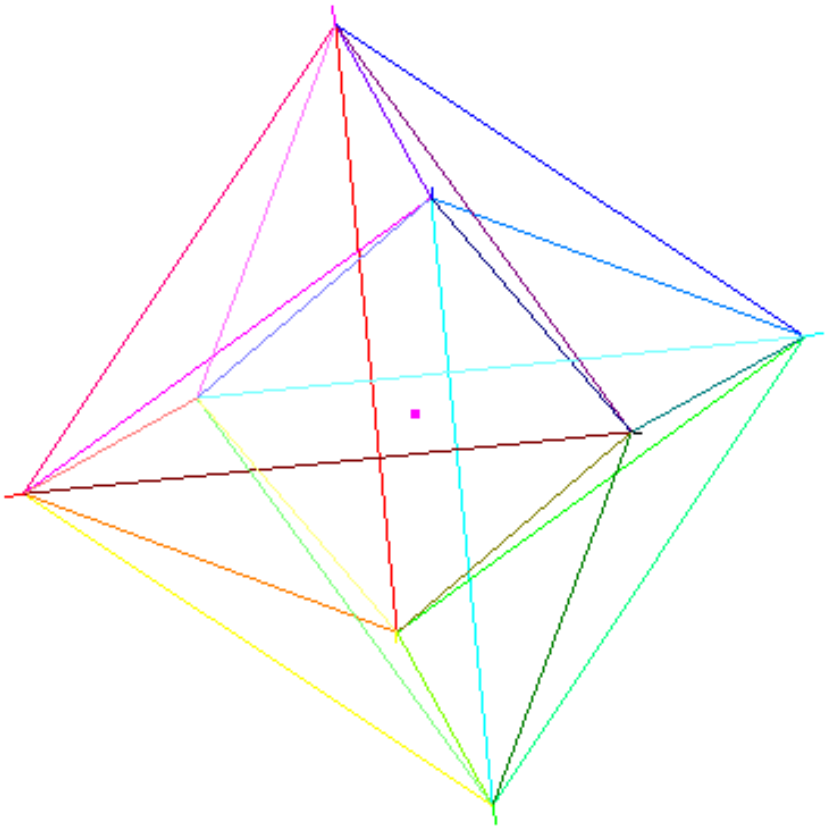
Ein Programm zur Darstellung rotierender vierdimensionaler Körper

Linus Wigger G4E



Maturaarbeit Alte Kantonsschule Aarau
Betreuende Lehrperson: Peter Hänsli
Abgegeben am 25. Oktober 2013

Inhaltsverzeichnis

1 Einleitung **3**
1.1 Zusammenfassung . . . 3
1.2 Motivation . . . 3
1.3 Dank . . . 3
1.4 Ziele . . . 3

2 Theoretische Grundlagen **4**
2.1 Der vierdimensionale Raum . . . 4
2.1.1 Vom Dreidimensionalen ins Vierdimensionale . . . 4
2.1.2 Einfache vierdimensionale Körper . . . 4
2.2 Matrizenrechnung . . . 6
2.3 Drehmatrizen . . . 7

3 Vorgehen **11**

4 Resultate **12**
4.1 Das Programm . . . 12
4.2 Bildbeispiele . . . 12

5 Diskussion/Interpretation **17**

6 Fazit/Schlusswort **18**

7 Abbildung- und Tabellenverzeichnis **19**

8 Quellenverzeichnis **20**

A Bedienungsanleitung **21**
A.1 Die verschiedenen Elemente . . . 21
A.2 Objekte darstellen und drehen . . . 22
A.3 Objekte selber erstellen . . . 23

B Quellcode **24**

C Anti-Plagiat-Erklärung **47**

1 Einleitung

1.1 Zusammenfassung

Bei dieser Maturaarbeit ging es darum, ein Programm zu erstellen, mit dem man rotierende vierdimensionale Körper darstellen kann. Im theoretischen Teil wird der vierdimensionale Raum erläutert und es werden einige der einfacheren vierdimensionalen Formen, z.B. der Tesserakt, ein vierdimensionaler Hyperwürfel, dargestellt. Er beinhaltet auch die Grundlagen der Matrizenrechnung und zeigt die Drehmatrizen, die im Programm verwendet werden.

Das Programm wurde mit Visual Basic 2010 erstellt und wird im Resultate-Teil genauer vorgestellt. Es wird 4Dreh genannt und kann vierdimensionale Körper als Gitterstrukturen anzeigen. Man kann das ausgewählte Objekt beliebig um alle sechs Koordinatenebenen rotieren und ausserdem selber Objekte erschaffen und speichern.

Im Teil Diskussion/Interpretation wird aufgezeigt, was es bereits an ähnlichen Programmen gibt. Dort werden auch Schwierigkeiten beim Erstellen der Arbeit aufgezeigt.

In den Anhängen ist hauptsächlich die Bedienungsanleitung von Bedeutung. Sie führt den Leser in die Benutzung von 4Dreh ein.

1.2 Motivation

Bei der Wahl eines Themas für Maturaarbeit kam ich sehr bald auf die Idee, etwas mit dem vierdimensionalen Raum zu nehmen. Das liegt daran, dass ich nicht ein Thema wählen wollte, welches man überall antrifft, denn das wäre meiner Meinung nach langweilig. Ausserdem kann man dadurch schöne Bilder erzeugen und an geometrischen Formen hatte ich schon als kleines Kind viel Freude. Es war mir auch von Anfang an klar, dass ich irgendetwas programmieren wollt. So gelangte ich zu diesem Thema.

1.3 Dank

Einen wichtigen Beitrag zum Gelingen meiner Arbeit lieferte Herr Herbert Hunziker, mit dessen Skript[1] ich Visual Basic erlernte. Der grösste Teil meines Dankes gebührt ihm. Des Weiteren danke ich meinen Eltern für ihre Hilfe bei der Arbeit mit LaTeXund Herrn Hänsli dafür, dass er meine Arbeit betreut hat.

1.4 Ziele

Das Ziel dieser Arbeit ist die Erstellung eines Computerprogramms, mit dem man vierdimensionale Objekte darstellen kann. Es sollte ebenfalls möglich sein, die Objekte drehend anzuzeigen. Mit dem Programm können natürlich auch ein- bis dreidimensionale Körper gezeigt werden. Eine mögliche Erweiterung wäre die Ausweitung des Programms auf beliebig viele Dimensionen.

2 Theoretische Grundlagen

2.1 Der vierdimensionale Raum

2.1.1 Vom Dreidimensionalen ins Vierdimensionale

Der vierdimensionale Raum entsteht, wenn man dem bekannten dreidimensionalen Raum noch eine Dimension hinzufügt. Alle Punkte, die vorher durch drei Werte, nämlich ihre Position in x-, y- und z-Richtung, gegeben waren, benötigen zusätzlich als vierten Wert eine Position in der neuen w-Richtung. Die Vorstellung des vierdimensionalen Raums gestaltet sich für uns Menschen sehr schwierig. In der Physik wird oft die Zeit als vierte Dimension genommen, nicht zuletzt weil Albert Einstein in seiner speziellen Relativitätstheorie die Zeit so verwendete[6, Kap. 17][7, :223]. Im Rahmen dieser Arbeit ist die vierte Dimension jedoch lediglich eine weitere räumliche Dimension, da Einsteins vierdimensionale Raumzeit die vierte Dimension nicht gleich behandelt wie die anderen drei Dimensionen. Es ist jedoch unmöglich, sich einen vierdimensionalen Raum wirklich vorzustellen, da wir nur drei Dimensionen kennen. Man kann zumindest eine Idee davon bekommen, wie dass der vierdimensionale Raum funktioniert, indem man den Schritt vom Dreidimensionalen ins Vierdimensionale vergleicht mit dem Schritt vom Eindimensionalen ins Zweidimensionale und vom Zweidimensionalen ins Dreidimensionale. Wenn man zum Beispiel aus einer Strecke ein Quadrat macht, fügt man an beide Enden eine neue Strecke an. Diese ist parallel zur neuen Achse und gleich lang wie die ursprüngliche Strecke. Die Endpunkte dieser Strecken werden durch die Anfangsform (in diesem Fall eine Strecke) verbunden und es entsteht ein Quadrat. Ähnlich kann man von einem Quadrat zu einem Würfel kommen. An jede Ecke des Quadrats wird senkrecht eine neue Linie angefügt, die in die dritte Dimension geht. Die Endpunkte werden durch ein zweites Quadrat verbunden. Man erhält einen Würfel. Es funktioniert genauso, wenn man aus einem Würfel einen Tesserakt (vierdimensionaler Hyperwürfel) macht. An jede der acht Ecken des Würfels wird eine weitere Kante angefügt, welche sich nur in die neue vierte Dimension ausdehnt. Die neu entstandenen Ecken werden durch einen Würfel miteinander verbunden und ein Tesserakt entsteht.

Im dreidimensionalen Raum sind für die verschiedenen Richtungen(x, y, z) die Bezeichnungen vorne und hinten, links und rechts und oben und unten oder Norden und Süden, Westen und Osten und oben und unten gebräuchlich. Für die w-Richtung wurden von Charles Howard Hinton die Begriffe "ana" und "kata" eingeführt [4]. Er führte ebenfalls das Wort "Tesserakt" für den vierdimensionalen Hyperwürfel ein[5].

2.1.2 Einfache vierdimensionale Körper

Eine der einfachsten vierdimensionalen Figuren ist der Tesserakt. Ein Tesserakt hat 16 Ecken, doppelt so viele wie ein Würfel. Dies kann man in der Abbildung 1 erkennen, wo der Tesserakt durch einen inneren und einen äusseren Würfel dargestellt wird. Ebenfalls sichtbar sind die 32 Kanten, die den jeweils 12 Kanten des inneren und äusseren Würfels und den 8 Verbindungen vom inneren zum äusseren Würfel entsprechen. Hier muss man beachten, dass alle Kanten gleich lang sind, auch wenn es auf der Abbildung 1 nicht so aussieht. Ein Tesserakt hat ausserdem 24 quadratische Flächen. Diese, wie auch 6 der 8 würfelförmigen Zellen, sind in der Abbildung 1 nicht auf den ersten Blick zu erkennen, da sie auf der Abbildung trapezförmig aussehen.

Abbildung 2 zeigt das Netz eines Tesserakts. Da der Tesserakt vier Dimensionen hat, ist seine Abwicklung dreidimensional, ähnlich wie ein dreidimensionaler Körper ein zweidimensionales Netz hat. Man erkennt auf der Abbildung 2 gut, dass alle Zellen würfelförmig und alle Flächen quadratisch sind.

Eine andere vierdimensionale Form ist das Pentachoron oder Fünfzeller. So wie ein Tesserakt ein vierdimensionaler Würfel ist, ist ein Pentachoron ein vierdimensionaler Tetraeder. Ein Pentachoron entsteht, indem man einem Tetraeder eine fünfte Ecke, deren Abstand von den anderen Ecken der

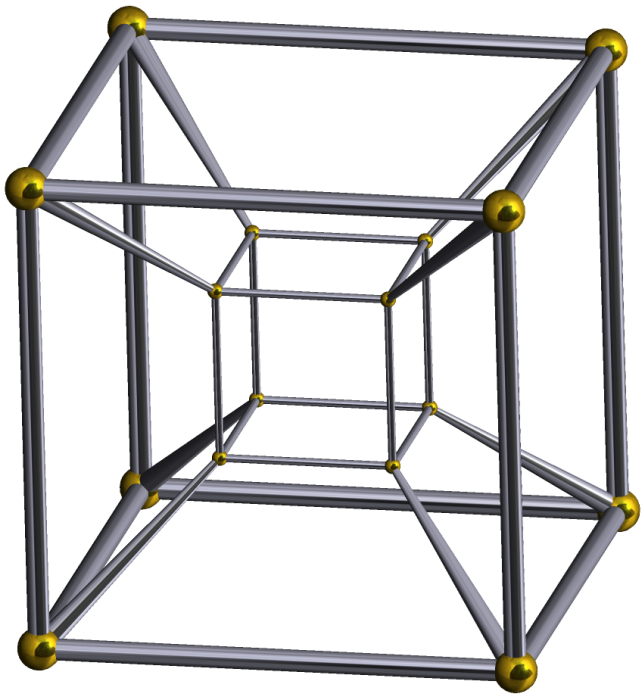


Abbildung 1: Schlegeldiagramm eines Tesserakts (Quelle: Wikipedia[10])

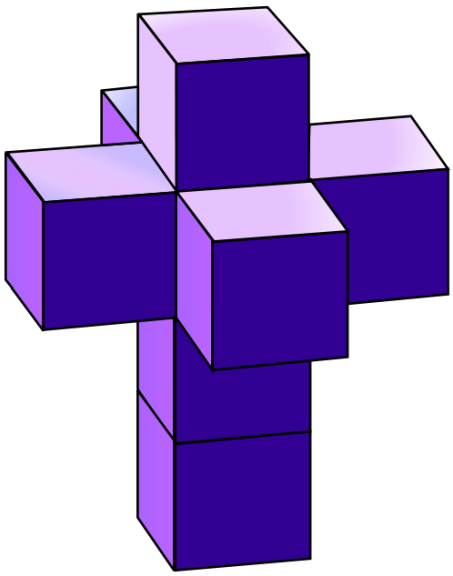


Abbildung 2: Netz eines Tesserakts (Quelle: Wikipedia[11])

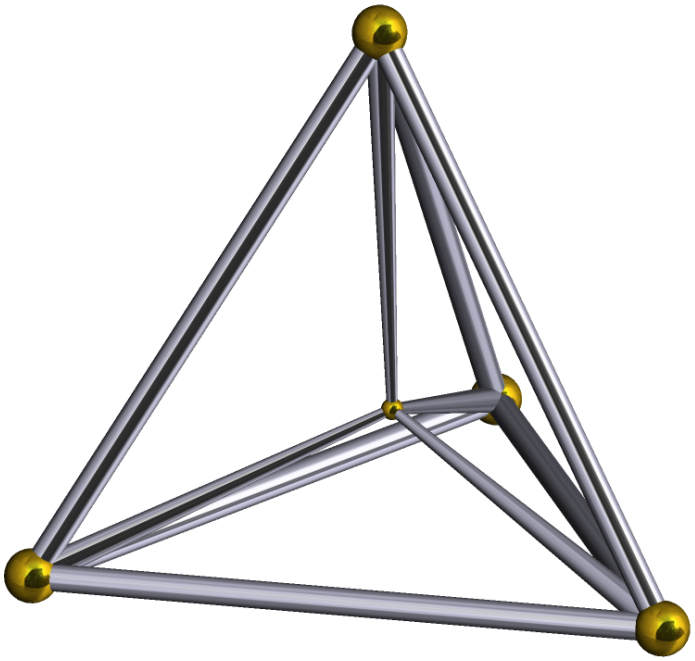


Abbildung 3: Schlegeldiagramm eines Pentachorons (Quelle: Wikipedia[12])

Seitenlänge entspricht, in der vierten Dimension hinzufügt. Ein Pentachoron hat 5 Ecken, 10 Kanten, 10 dreieckige Flächen und 5 tetraederförmige Zellen. Die Abbildung 3 ist ein Schlegeldiagramm eines Pentachorons. Es sieht aus wie ein Tetraeder mit einer weiteren Ecke in der Mitte.

Die vierdimensionale Form, die man sich am besten vorstellen kann, ist sicherlich die Hyperkugel. Genau wie die dreidimensionale Kugel und der zweidimensionale Kreis besitzt sie einen Mittelpunkt und einen Radius. Alle Punkte, deren Abstand vom Mittelpunkt gleich dem Radius sind, liegen auf der sogenannten 3-Sphäre, welche sozusagen das "Obervolumen" oder das Äquivalent der Oberfläche im vierdimensionalen Raum ist.

Auf dem Titelbild ist ein 16-Zeller zu sehen. Er besteht aus 16 tetraederförmigen Zellen. Ein 16-Zeller kann ähnlich konstruiert werden wie ein Oktaeder. Um in einem dreidimensionalen Koordinatensystem einen Oktaeder zu erhalten, kann man alle drei Achsen mit einer Kugel, deren Mittelpunkt im Ursprung liegt, (der Umkugel des Oktaeders) schneiden. Die so erhaltenen Punkte sind die sechs Ecken eines Oktaeders. Um die Ecken eines 16-Zellers zu erhalten muss man in einem vierdimensionalen Koordinatensystem die vier Achsen mit einer Hyperkugel schneiden und erhält so die Ecken des 16-Zellers. Ein 16-Zeller besitzt 8 Ecken, 24 Kanten, 32 dreieckige Flächen und 16 tetraederförmige Zellen.

2.2 Matrizenrechnung

Um Drehungen mathematisch durchzuführen benötigt man Rotationsmatrizen (Drehmatrizen). Matrizen sind rechteckige Anordnungen von Elementen (meist Zahlen). Eine Matrix mit m Zeilen und n Spalten bezeichnet man als $m \times n$-Matrix. Eine 2×3-Matrix sähe also so aus:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \tag{1}$$

Wobei $a_{m,n}$ für beliebige Zahlen stehen. Man kann zwei Matrizen des gleichen Typs (d.h. zwei Matrizen, die dieselbe Anzahl Zeilen und Spalten haben) addieren, indem man jeweils die Elemente an denselben Stellen zusammenzählt.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} \\ a_{21}+b_{21} & a_{22}+b_{22} \\ a_{31}+b_{31} & a_{32}+b_{32} \end{pmatrix} \tag{2}$$

Die Multiplikation zweier Matrizen ist komplizierter. Um zwei Matrizen miteinander multiplizieren zu können, muss die vordere vordere Matrix gleich viele Spalten haben wie die hintere Zeilen hat. Die resultierende Matrix hat so viele Zeilen wie die vordere und so viele Spalten wie die hintere. Kurz gesagt, wenn man eine $m \times n$-Matrix mit einer $n \times k$-Matrix multipliziert, erhält man eine $m \times k$-Matrix. Ein Beispiel dazu: Es werden eine 3×2-Matrix und eine 2×3-Matrix miteinander multipliziert.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{pmatrix} \tag{3}$$

Das Ergebnis ist eine 3×3-Matrix.

$$\begin{pmatrix} a_{11}*b_{11}+a_{12}*b_{21} & a_{11}*b_{12}+a_{12}*b_{22} & a_{11}*b_{13}+a_{12}*b_{23} \\ a_{21}*b_{11}+a_{22}*b_{21} & a_{21}*b_{12}+a_{22}*b_{22} & a_{21}*b_{13}+a_{22}*b_{23} \\ a_{31}*b_{11}+a_{32}*b_{21} & a_{31}*b_{12}+a_{32}*b_{22} & a_{31}*b_{13}+a_{32}*b_{23} \end{pmatrix} \tag{4}$$

Um ein Element in der neuen Matrix zu berechnen, muss man alle Elemente der entsprechenden Zeile der ersten Matrix mit den Elementen in der entsprechenden Spalte der zweiten Matrix miteinander multiplizieren und diese Produkte zusammenzählen.

$$c_{ij} = a_{i1} * b_{1j} + a_{i2} * b_{2j} + ... + a_{in} * b_{ni} \tag{5}$$

Bei der obigen Formel steht n für die Anzahl Spalten der ersten Matrix, die gleich der Anzahl Zeilen der zweiten Matrix ist.

Einige wichtige Eigenschaften der Matrixmultiplikation ist, dass das Assoziativgesetz gilt, d.h. für die Matrizen A, B und C gilt[8]:

$$(A * B) * C = A * (B * C) \tag{6}$$

Sie ist allerdings nicht kommutativ, also gilt allgemein:

$$A * B \neq B * A \tag{7}$$

Dies kann man sehr leicht selbst erkennen. A sei eine $m \times n$-Matrix und B eine $n \times k$-Matrix. $A * B$ ergibt eine $m \times k$-Matrix, aber $B * A$ ist nicht einmal definiert (ausser wenn $m = k$ ist).

2.3 Drehmatrizen

Eine zweidimensionale Drehmatrix, die die xy-Ebene um den Winkel α um den Ursprung dreht, sieht folgendermassen aus:

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{8}$$

Um zu verstehen, wieso die Matrix (8) eine Drehmatrix ist, multipliziert man sie am besten mit den Einheitsvektoren. In diesem Beispiel wird für den Winkel 30° eingesetzt.

$$\begin{pmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{pmatrix} * \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos 30° \\ \sin 30° \end{pmatrix} \tag{9}$$

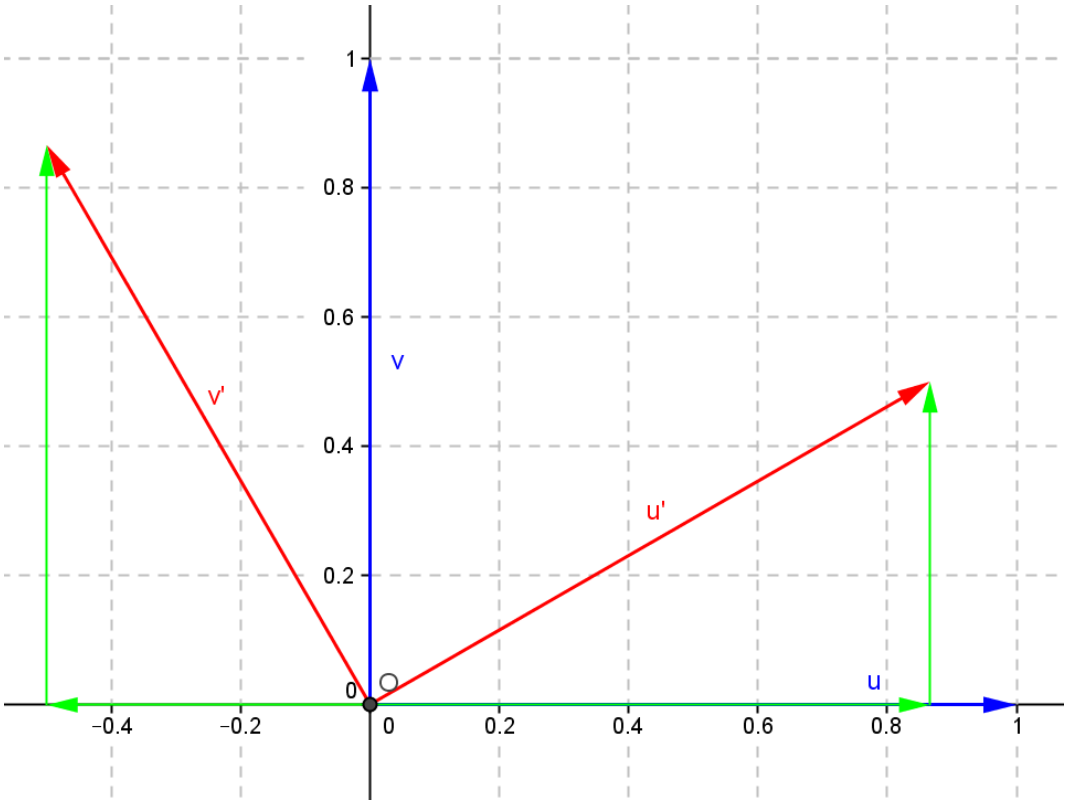


Abbildung 4: Um 30° gedrehte Einheitsvektoren im Koordinatensystem

$$\begin{pmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -\sin 30° \\ \cos 30° \end{pmatrix} \tag{10}$$

In der Abbildung 4 werden diese Vektoren in ein Koordinatensystem eingesetzt. Die Vektoren aus Gleichung (9) werden u und u' genannt und sind blau, die Vektoren aus Gleichung (10) heissen v und v' und werden rot gezeichnet. Die grünen Vektoren zeigen die einzelnen Komponenten der Bildvektoren u' und v'. Man sieht, dass u' und v' den um 30° gedrehten Einheitsvektoren entsprechen. Es ist zu beachten, dass nicht jede Matrix eine Drehung darstellt. In der Ebene sind nur Matrizen der Form (8) Drehungen um den Ursprung.

Im dreidimensionalen Raum gibt es drei einfache Drehmatrizen. Dies liegt daran, dass in der Ebene nur um einen Punkt gedreht wird. Im Raum dreht man jedoch um eine Achse und im dreidimensionalen Raum gibt es drei verschiedene Achsen. Die dadurch entstehenden Matrizen kann man miteinander verknüpfen um so jede beliebige Drehung darzustellen.

$$\begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

Die Matrix (11) stellt die gleiche Drehung dar wie die Matrix (8), einfach im Raum mit der z-Achse als Drehzentrum. Eine Drehung um die x-Achse wird dargestellt mit der Matrix (12) und eine Drehung um die y-Achse mit der Matrix (13).

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{12}$$

$$\begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{13}$$

Um die richtigen Vorzeichen bei der Matrix (13) herauszufinden, kann man das Ganze an der rechten Hand ausprobieren. In einem Rechtssystem verhalten sich die x-, y- und z-Achsen wie Daumen, Zeigefinger und Mittelfinger der rechten Hand. Gedreht wird im Gegenuhrzeigersinn. Angenommen, der Drehwinkel betrage zwischen 0° und 90°, so würde ein Vektor parallel zur x-Achse nach der Drehung in positive x-Richtung und negative z-Richtung zeigen, während ein Vektor, der parallel zur z-Achse ist, nach der Drehung in die positive x-Richtung und positive z-Richtung zeigt. Diese verschiedenen Drehmatrizen können miteinander verkettet werden. So wird zum Beispiel in der folgenden Multiplikation der Vektor mit den Komponenten a, b und c zuerst 45° um die x-Achse gedreht und danach 60° um die y-Achse.

$$\begin{pmatrix} \cos 60° & 0 & \sin 60° \\ 0 & 1 & 0 \\ -\sin 60° & 0 & \cos 60° \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos 45° & -\sin 45° \\ 0 & \sin 45° & \cos 45° \end{pmatrix} * \begin{pmatrix} a \\ b \\ c \end{pmatrix} \tag{14}$$

Zu beachten ist, dass die zuerst ausgeführte Drehung rechts ist, direkt vor dem Vektor. Wegen dem Assoziativgesetz kann man es so betrachten, dass der Term oben die Drehung um die y-Achse darstellt und der gedrehte Vektor der bereits um die x-Achse rotierte ursprüngliche Vektor ist.

Im vierdimensionalen Raum wird es noch einmal komplizierter, weil man es sich nicht mehr so gut vorstellen kann. In der Ebene wurde um einen Punkt gedreht, im dreidimensionalen Raum um eine Gerade. Im vierdimensionalen Raum dreht man daher um eine Fläche [2]. Die vier Achsen spannen zusammen insgesamt sechs Ebenen auf und jede davon hat ihre eigene Drehmatrix.

$$R_{xy} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\alpha & -\sin\alpha \\ 0 & 0 & \sin\alpha & \cos\alpha \end{pmatrix} \tag{15}$$

$$R_{xz} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & 0 & \sin\beta \\ 0 & 0 & 1 & 0 \\ 0 & -\sin\beta & 0 & \cos\beta \end{pmatrix} \tag{16}$$

$$R_{xw} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{17}$$

$$R_{yz} = \begin{pmatrix} \cos\delta & 0 & 0 & \sin\delta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\delta & 0 & 0 & \cos\delta \end{pmatrix} \tag{18}$$

$$R_{yw} = \begin{pmatrix} \cos\varepsilon & 0 & \sin\varepsilon & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varepsilon & 0 & \cos\varepsilon & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{19}$$

$$R_{zw} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{20}$$

Die Matrix R_{xz} zum Beispiel dreht den vierdimensionalen Raum um die xz-Ebene. Alle Punkte mit y- oder w-Positionen ungleich null verändern ihre Koordinaten. Abgesehen davon, dass um eine Ebene gedreht wird, funktioniert alles genauso wie im dreidimensionalen Raum. Die Matrizen (15) bis (20) werden im Programm verwendet, um die Punkte für das Bild zu berechnen.

3 Vorgehen

Zur Erstellung des Programms wurde hauptsächlich eine Umgebung benötigt, mit der man bewirken kann, dass auf einer Bildfläche Linien angezeigt werden und der Benutzer irgendwo die verschiedenen Werte beeinflussen kann. Dazu eignet sich Visual Basic, da es sehr einfach zu lernen ist und einen integrierten GUI-Editor besitzt. Für die Gestaltung des Programms wurde Visual Basic 2010 Express, das sich kostenlos herunterladen lässt [3], auf einem Computer mit Windows 7 verwendet. Bei der Programmierung wurden als erstes die Hauptelemente des Programms erstellt. Das bedeutet: Eine Bildfläche, auf der das Objekt angezeigt werden soll, die Eingabefelder, mit denen man die Drehung steuert und ein Weg, Formen einzugeben. Das Programm funktionierte anfangs nur in drei Dimensionen. Im nächsten Schritt wurden den Punkten vierdimensionale Koordinaten gegeben, sowie die nötigen Elemente hinzugefügt, um auch Drehungen im vierdimensionalen Raum zu verwenden. Dies setzte die im vorigen Kapitel behandelten Drehmatrizen voraus. Es folgten die Projektionseinstellungen und mehr Freiheit beim Einstellen von Punkten und Verbindungen. Gegen den Schluss kam auch die Möglichkeit, Objekte zu speichern, sowie eine Auswahl an Vorlagen. Nachdem noch einige Fehler behoben worden waren, erhielt das Programm schliesslich den Namen "4Dreh". Der Name soll anzeigen, dass es sich um ein Programm handelt, das im vierdimensionalen Raum etwas gedreht wird.

4 Resultate

4.1 Das Programm

Das fertige Programm kann Gitterstrukturen im vierdimensionalen Raum anzeigen und kontinuierlich rotieren. Mithilfe verschiedener Eingabefenster kann man eine breite Auswahl an Parametern verändern. Die vollständige Liste aller beeinflussbaren Teile des Programms wird hier noch einmal aufgeführt:

- Drehung um alle sechs Koordinatenebenen
- Projektion des vierdimensionalen Raums auf die zweidimensionale Bildfläche
- Punkte und Verbindungslinien zwischen Punkten
- Bildfrequenz

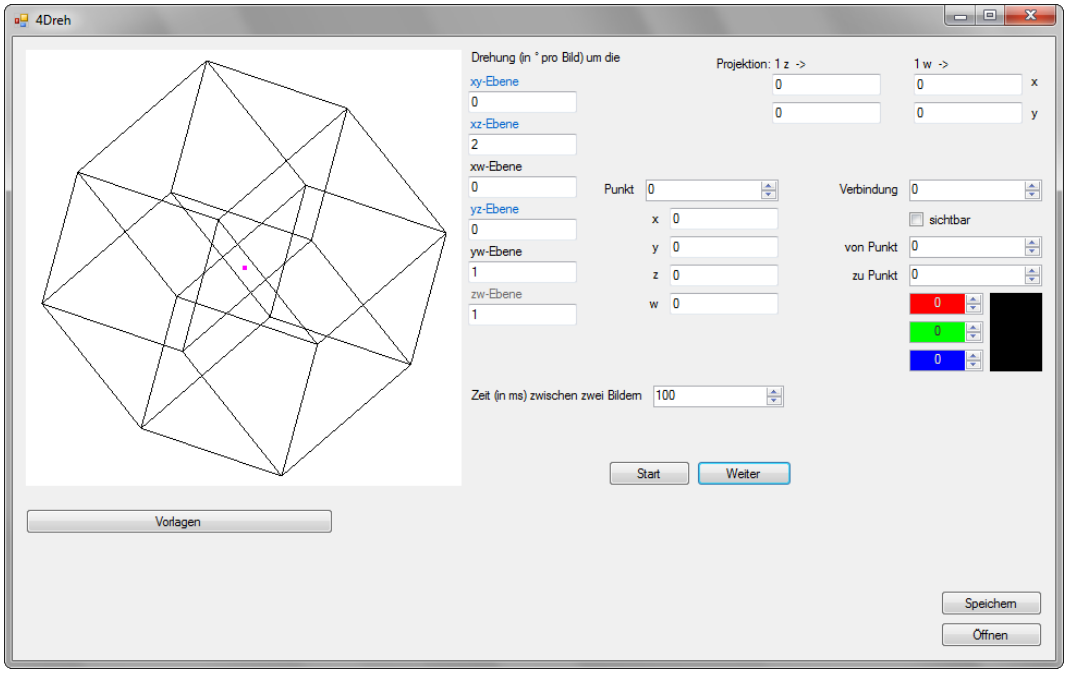


Abbildung 5: 4Dreh in Aktion

Das Programm kann maximal 61 Punkte und 151 Verbindungslinien anzeigen (diese Zahlen wurden willkürlich festgelegt). Die momentan ausgewählten Punkte und Linien werden auf dem Bild gekennzeichnet. Das Programm stellt einige Grundformen zur Verfügung, nämlich Quadrat, Würfel, Tesserakt, gleichseitiges Dreieck, Tetraeder und Pentachoron. Man kann die aktuellen Formen jederzeit speichern und bereits gespeicherte Formen laden. Die Abbildung 5 zeigt das Fenster beim Start der Anwendung. Die Drehungen werden von oben nach unten durchgeführt, d.h. zuerst findet die Drehung um die xy-Ebene statt, es folgt die Drehung um die xz-Ebene und zuletzt wird um die zw-Ebene gedreht. Mit jedem weiteren Bild werden alle Drehungen an den schon gedrehten Punkten ausgeführt.

4.2 Bildbeispiele

Auf der Abbildung 6 wird das Schrägbild eines Tesserakts dargestellt. Die verwendete Projektion ist $1z \to -0.3x; -0.3y$ und $1w \to 0.4x; 0.2y$. Die Farbe der Kanten zeigt an, zu welcher Achse sie parallel

sind. Die schwarzen Kanten sind parallel zur x-Achse, die blauen zur y-Achse, die roten zur z-Achse und die grünen zur w-Achse. Mit diesen Einstellungen kann man sehr gut die Drehung um die einzelnen Koordinatenebenen zeigen, da jeweils zwei Sorten Kanten nur ihre Position, nicht aber ihre Richtung ändern. Die Abbildung 7 verdeutlicht dies, es werden eine Drehung um die xz-Ebene und eine um die yw-Ebene gezeigt. Bei beiden Seiten erfolgt mit jedem Schritt eine Drehung um 30°.

Die Abbildungen 8, 9 und 10 zeigen die Rotation der im Programm vorhandenen Dreieck-Analoge. Abbildung 8 zeigt ein Dreieck, das mit jedem Schritt um 20° gedreht wird.

Auf den Abbildungen 9 und 10 werden die z- und w-Werte der Punkte ignoriert, die Eckpunkte der Körper werden senkrecht auf die xy-Ebene projiziert. Auf der Abbildung 9 wird ein Tetraeder mit jedem Schritt zuerst 10° um die x-Achse (xw-Ebene), dann 10° um die y-Achse (yw-Ebene) und 10° um die z-Achse (zw-Ebene) gedreht.

Bei der Abbildung 10 wird ein Pentachoron mit jedem Schritt um 5° um alle sechs Ebenen rotiert.

Die Abbildungen 11 und 12 zeigen ein interessantes Phänomen: Ein in eine höhere Dimension gedrehter Körper erscheint verformt. Abbildung 11 zeigt ein Quadrat, das in die dritte Dimension gedreht wird. Die Projektion davon sieht aus wie ein Parallelogramm, nicht wie ein Quadrat.

Dasselbe gilt für den Würfel in Abbildung 12. Der Würfel wird in die vierte Dimension gedreht und die Projektion ist ein schiefes Prisma.

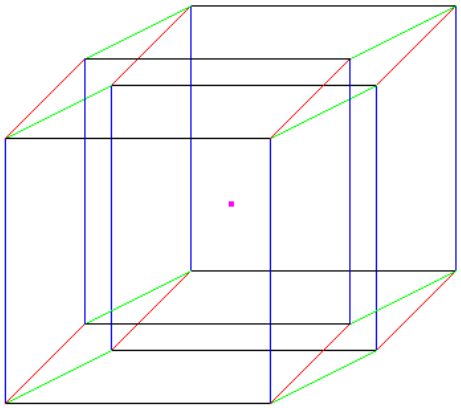


Abbildung 6: Schrägbild eines Tesserakts

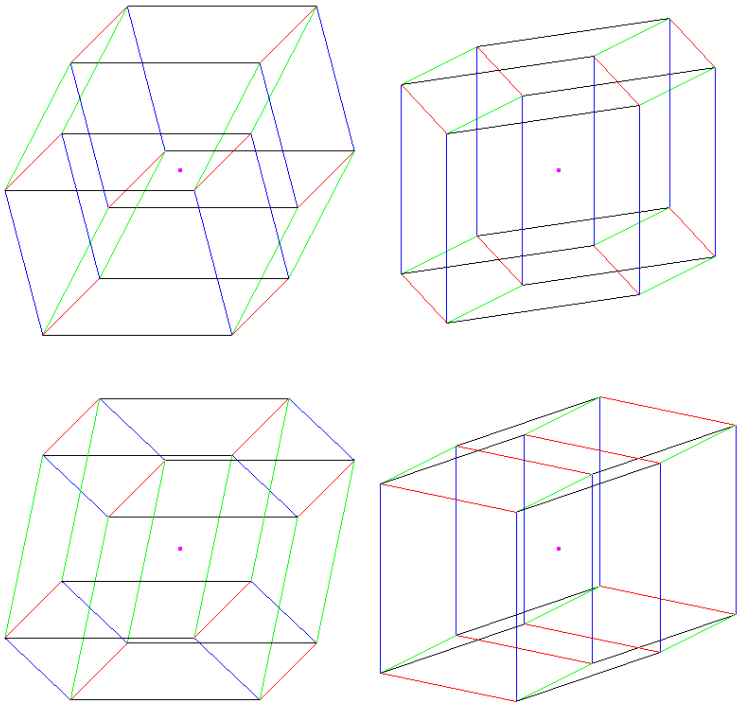


Abbildung 7: Links: Drehung um xz-Ebene, rechts: Drehung um yw-Ebene

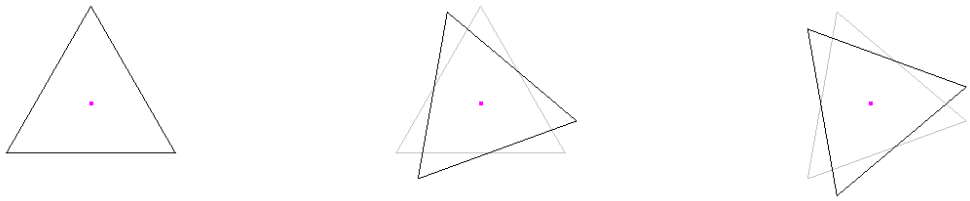


Abbildung 8: Drehung eines Dreiecks

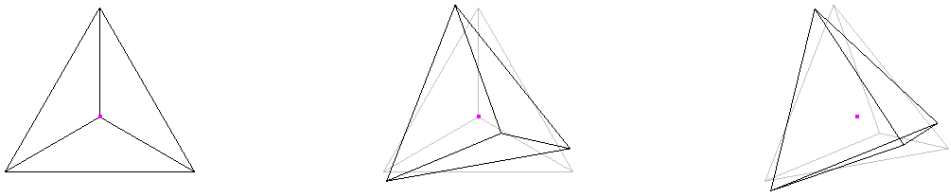


Abbildung 9: Drehung eines Tetraeders

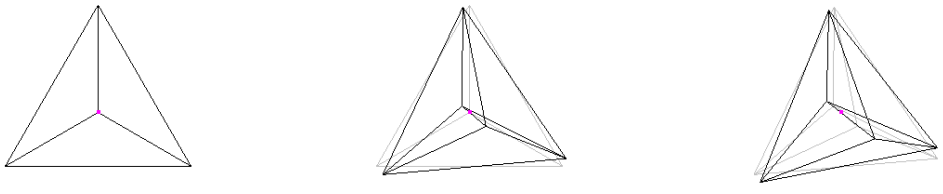


Abbildung 10: Drehung eines Pentachorons

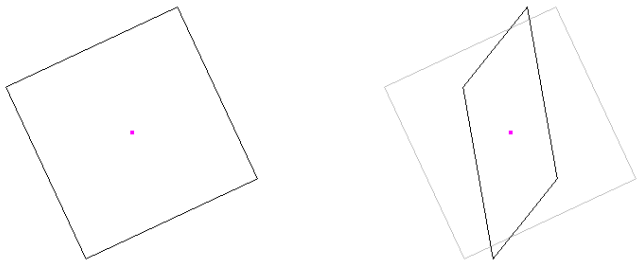


Abbildung 11: Ein Quadrat wird in die dritte Dimension gedreht

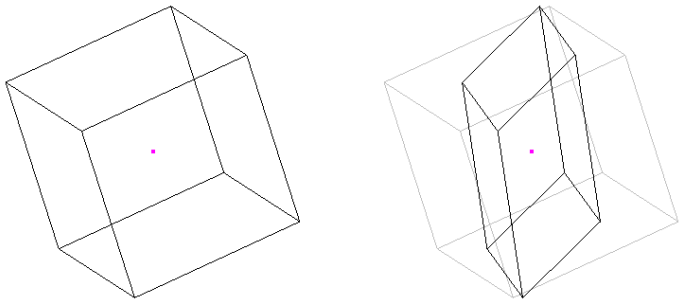


Abbildung 12: Ein Würfel wird in die vierte Dimension gedreht

5 Diskussion/Interpretation

Es gibt bereits verschiedene ähnliche Anwendungen, im Internet findet man mehrere Seiten, die entweder direkt drehbare 4D-Objekte anzeigen oder ein herunterladbares Programm zur Verfügung stellen. Besonders hervorzuheben ist das Programm Stella4D [9], sichtbar in der Abbildung 13 wo gerade ein Tesserakt gezeigt wird. Die volle Version ist zwar kostenpflichtig, doch schon die Demoversion beinhaltet eine grosse Auswahl an drei- und vierdimensionalen Körpern, die dreidimensional dargestellt werden und die man direkt mit der Maus in Rotation versetzen kann. Man kann bei Stella4D Körper auf eine Vielzahl von Arten bearbeiten und anzeigen. Wie auch die meisten anderen derartigen Programme behandelt es die vierte Dimension speziell. Es gibt eigens eine Option dafür, das Objekt in der vierten statt der dritten Dimension zu drehen. Es ist natürlich auch schwierig ist, mit einer zweidimensionalen Mausbewegung eine vierdimensionale Drehung zu definieren. Bei 4Dreh, dem Programm zu dieser Arbeit, wird die w-Richtung exakt wie die z-Richtung behandelt.

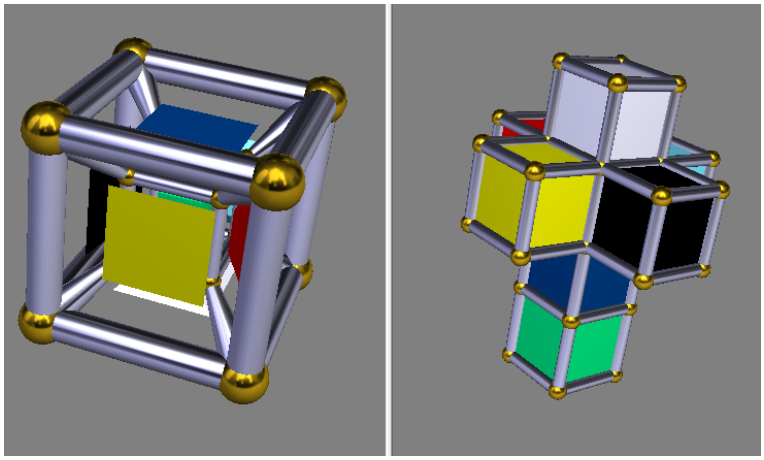


Abbildung 13: Bildfläche von Stella4D

6 Fazit/Schlusswort

Das Ziel der Arbeit, die Erstellung eines Programms, mit dem man vierdimensionale Objekte drehend darstellen kann, wurde erreicht. Das fertige Programm kann ein- bis vierdimensionale Körper anzeigen und gibt ausserdem die Möglichkeit, eigene Polyeder zu definieren.

Es wurde bereits erwähnt, dass eine mögliche Erweiterung der Arbeit die Ausweitung des Programms auf beliebig viele Dimensionen war. Dies wurde aus einer Vielzahl von Gründen nicht durchgeführt. Zum einen würde man immer mehr Eingabefelder benötigen. Abgesehen von der Eingabe der Koordinaten, würde vor allem die Anzahl der möglichen Drehzentren immer schneller wachsen. Im dreidimensionalen Raum gibt es drei Koordinatenachsen und im vierdimensionalen sechs Koordinatenebenen. Im fünfdimensionalen hätte man zehn Koordinatenräume, um die gedreht wird, und in höheren Dimensionen würde es nur noch schlimmer. Auch der Speicherplatz würde immer schneller wachsen. Um einen Würfel anzuzeigen benötigt man 8 Punkte und 12 Verbindungen. Ein Tesserakt hat schon 16 Punkte und 32 Verbindungen und ein einfacher fünfdimensionaler Hyperwürfel bräuchte schon 32 Punkte und 80 Verbindungslinien. Diese Probleme sind noch lösbar, aber wenn man bedenkt, wie unübersichtlich auch nur ein gedrehtes dreidimensionales Objekt werden kann, so könnte man ein fünfdimensionales Objekt kaum noch darstellen ohne komplett den Überblick zu verlieren. Die Tabelle 1 zeigt das Wachstum der Anzahl Grenzelemente eines n-dimensionalen Hyperwürfels. Am wichtigsten für das Programm 4Dreh ist die Zeile "Kanten", da es diese zeichnet. Ein sechsdimensionaler Hyperwürfel übersteigt die Maximalanzahl Punkte und Verbindungen des Programms.

Dimensionen des Hyperwürfels	0	1	2	3	4	5	6	n
Ecken	1	2	4	8	16	32	64	2^n
Kanten	0	1	4	12	32	80	192	$n2^{n-1}$
Flächen	0	0	1	6	24	80	240	$n(n-1)2^{n-3}$

Tabelle 1: Grenzelemente von Hyperwürfeln

7 Abbildung- und Tabellenverzeichnis

Abbildungsverzeichnis

1 Tesserakt 5
2 Netz eines Tesserakts 5
3 Pentachoron 6
4 Gedrehte Einheitsvektoren im Koordinatensystem 8
5 4Dreh 12
6 Schrägbild eines Tesserakts 14
7 Drehung eines Tesserakts um die xz- bzw. die yw-Ebene 14
8 Drehung eines Dreiecks 15
9 Drehung eines Tetraeders 15
10 Drehung eines Pentachorons 15
11 Drehung eines Quadrats in eine höhere Dimension 16
12 Drehung eines Würfels in eine höhere Dimension 16
13 Bildfläche von Stella4D 17
14 Elemente von 4Dreh 21

Tabellenverzeichnis

1 Grenzelemente von Hyperwürfeln 18

8 Quellenverzeichnis

Literatur

[1] Hunziker, Herbert, 2011/2012, Einführung Programmieren mit Visual Basic 2010, Alte Kantonsschule Aarau

[2] Grave, Bernd, http://www.uni-math.gwdg.de/bgr/drehungen.php, besucht: 24.3.2013

[3] Microsoft Visual Studio, http://www.microsoft.com/visualstudio/deu/products/visual-studio-express-products, besucht: 14.10.2013

[4] Rucker, Rudolf, 2005, Spaceland Notes, http://www.rudyrucker.com/pdf/spacelandnotesposted.pdf, besucht: 19.10.2013

[5] Hinton, Charles, 1888, A New Era of Thought, London, Swan Sonnenschein & Co. Ltd.

[6] Feynman, Richard, 1974, Feynman Vorlesungen über Physik: Hauptsächlich Mechanik, Strahlung und Wärme, Bd. 1 Teil 1, deutsch-englische BILINGUA-Ausgabe, München, R. Oldenbourg Verlag GmbH

[7] Born, Max, 1920, Die Relativitätstheorie Einsteins, Bd. 3, Berlin, Verlag von Julius Springer

[8] Matse-Hamburg, http://matse-hamburg.wikispaces.com/Beweis+des+Assoziativgesetzes+für+Matrizenmultiplikation, besucht: 20.10.2013

[9] Webb, Robert, http://www.software3d.com/Stella.php, besucht: 20.10.2013

[10] http://en.wikipedia.org/wiki/File:Schlegel_wireframe_8-cell.png, besucht: 21.10.2013

[11] http://commons.wikimedia.org/wiki/File:Tesseract2.svg, besucht: 21.10.2013

[12] http://commons.wikimedia.org/wiki/File:Schlegel_wireframe_5-cell.png, besucht: 21.10.2013

A Bedienungsanleitung

A.1 Die verschiedenen Elemente

Auf der Abbildung 14 sieht man alle Teile des Programms 4Dreh. Die zueinander gehörenden Elemente sind mit derselben Farbe markiert.

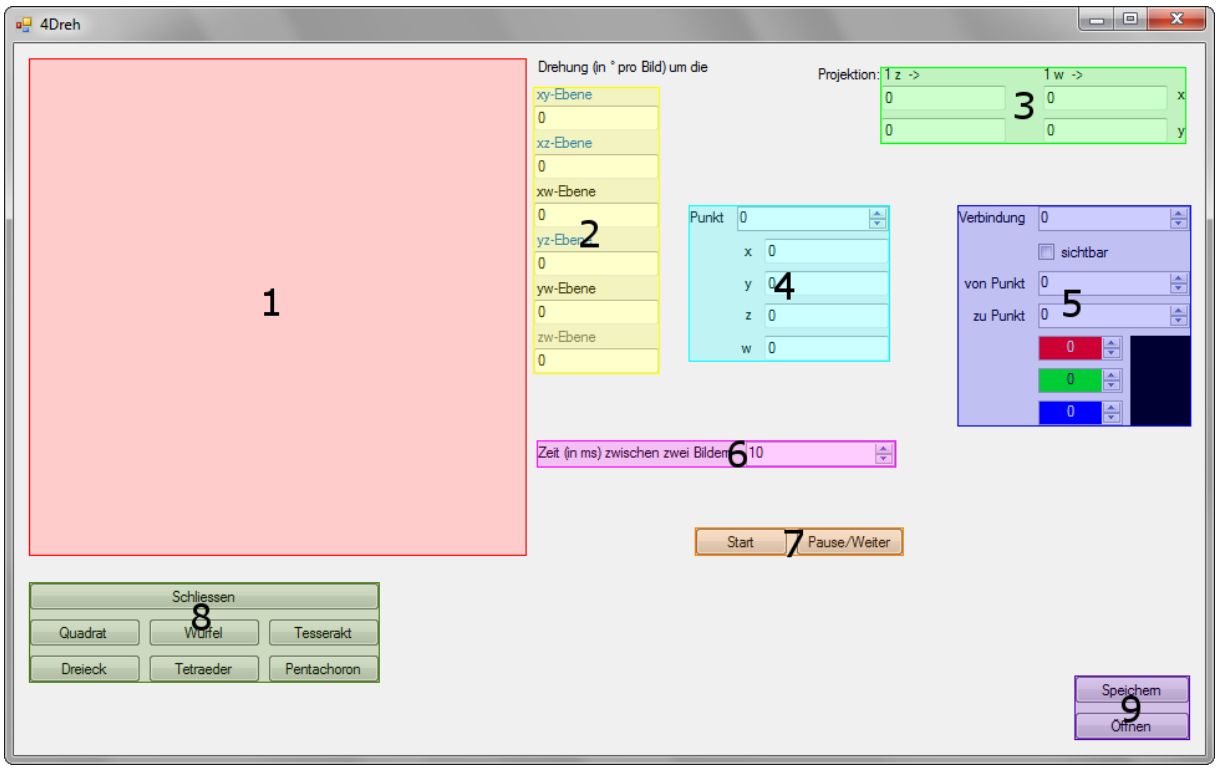


Abbildung 14: Alle Elemente von 4Dreh

1. **Bildfläche**: Hier wird das Bild angezeigt. Die x-Achse zeigt stets nach rechts, die y-Achse stets nach oben. Wie dass die z- und w-Achsen liegen, hängt von der Projektion ab.

2. **Eingabefelder zur Rotation**: Die Werte in diesen Feldern geben an, wie stark das Objekt um die Koordinatenebenen gedreht wird. Die Farbe der Beschriftung zeigt, in wie vielen Dimensionen die Drehung stattfindet: Grau (Drehung um die zw-Ebene) bedeutet, dass nur zweidimensional gedreht wird. Es wird direkt die Bildfläche rotiert. Wenn man ein zweidimensionales Objekt dreht, benötigt man nur dieses Feld. Schwarz (Drehung um die xw-Ebene und die yw-Ebene) zeigt Drehungen im dreidimensionalen Raum an, d.h. wenn man einen dreidimensionalen Körper dreht benötigt man diese Felder und das Graue. Die restlichen Drehungen sind blau markiert und drehen das Objekt im vierdimensionalen Raum. Bei der Drehung werden die Eingabefelder von oben nach unten durchgegangen. Die obersten Drehungen werden zuerst ausgeführt.

3. **Projektionsvorschrift**: Mit den vier Eingabefeldern oben rechts legt man die Parameter für die Projektion (es handelt sich um eine einfache parallele Projektion) vom drei- und vierdimensionalen Raum auf die Bildebene fest. Die linken Felder kontrollieren, wie die Position eines Punktes in z-Richtung dargestellt wird, die rechten bewirken dasselbe für die w-Richtung. Die oben eingegebenen Zahlen sagen aus, wie viel ein Punkt für jede Einheit in z- bzw. w-Richtung auf dem Bild in x-Richtung verschoben wird. Ein dreidimensionales Koordinatensystem, so wie

man es normalerweise auf Papier zeichnet, hätte z.B. die Einstellung $1z \rightarrow -0.5x; -0.5y$. der Punkt $A(0|0|100|0)$ hätte dadurch auf der Bildfläche die Koordinaten $A'(-50|-50|0|0)$.

4. **Punktedefinitionen**: Hier kann man die Anfangskoordinaten von maximal 61 Punkten festlegen, um damit eigene Objekte zu kreieren. Das oberste Element legt fest, welcher Punkt dass ausgew¨hlt ist. Dieser momentan ausgewählte Punkt wird auf dem Bild pink markiert (Hinweis: Wenn das Bild pausiert ist, muss man es weiterlaufen lassen, damit die Markierung angezeigt wird. Dasselbe gilt für die Verbindungen). Der Punkt 0 wird normalerweise für den Ursprung reserviert.

5. **Verbindungsdefinitionen** Mit diesen Elementen kann man bis zu 151 Verbindungen festlegen. Die Verbindungen werden als gerade Strecke zwischen zwei Punkten angezeigt. Das erste Element nach der Auswahl der Verbindung ist ein Kästchen, mit dem man festlegen kann ob die momentan ausgewählte Verbindung auf dem Bild sichtbar ist. Die aktuelle Verbindung wird allerdings immer fett auf dem Bild markiert, auch wenn sie als nicht sichtbar gekennzeichnet wurde. Die nächsten zwei Felder geben an, welche zwei Punkte miteinander verbunden werden. Die dazu verwendeten Nummern sind dieselben, die man bei der Auswahl der Punkte links davon verwendet. Schliesslich kann man die Farbe der Verbindung festlegen, indem man ihre Rot-, Grün- und Blauwerte angibt. Das Feld daneben zeigt eine Vorschau der Farbe an.

6. **Bildfrequenz**: Hier kann man festlegen, wie viel Zeit zwischen zwei Einzelbildern vergeht. Längere Pausen können hilfreich sein, wenn man eine sehr genaue Drehung durchführen will.

7. **Animationssteuerung**: Der "Start"-Knopf startet die Animation, ausgehend von den gegebenen Punkten und Verbindungen. Wenn die Koordinaten eines Punkts verändert werden, muss ebenfalls der "Start"-Knopf gedrückt werden, ansonsten werden dieselben Punkte wie vorher verwendet. Neben dem "Start"-Knopf ist der "Pause"-/"Weiter"-Knopf. Mit "Pause" kann man die Animation jederzeit anhalten, um die Drehung, die Projektion oder einzelne Verbindungen zu verändern. Wenn man dabei jedoch die Koordinaten von Punkten verändert, muss man, wie vorher erwähnt, den "Start"-Knopf drücken, damit die neuen Punkte auch verwendet werden. Mit "Weiter" kann man die Animation weiterführen. Die während der Pause durchgeführten Veränderungen (abgesehen von Punkten) werden sofort angewendet.

8. **Vorlagen**: Dieser Knopf führt, wenn man ihn anklickt, zu einer Auswahl an einfachen Formen. Wenn eine dieser Formen ausgewählt wird, werden alle momentan eingestellten Punkte und Verbindungen gelöscht. Um die Form anzuzeigen, muss der "Start"-Knopf gedrückt werden. Die wählbaren Formen sind: Quadrat(2d), Würfel(3d), Tesserakt(4d), gleichseitiges Dreieck(2d), Tetraeder(3d) und Pentachoron(4d).

9. **Datenspeicherung**: Die letzten beiden Knöpfe werden verwendet, um die aktuellen Einstellungen zu speichern und vorhandene Dateien zu laden. Die eingestellten Parameter werden in einer Textdatei gespeichert.

A.2 Objekte darstellen und drehen

In diesem Abschnitt wird beschrieben, wie dass man in diesem Programm mit einem gewöhnlichen Objekt umgeht. Als erstes braucht man das Objekt. Dazu wählt man für diese Anleitung einmal das Quadrat bei den Standardformen und drückt darauf den "Start"-Knopf. Auf der Bildfläche sollte nun ein schwarzes Quadrat sichtbar sein. Um das Quadrat zu drehen, muss man das Bild pausieren ("Pause"-Knopf). Wie im letzten Teil beschrieben braucht man für die Drehung eines Quadrats nur das unterste Feld (grau angeschrieben), die Drehung um die zw-Ebene. Setzt man hier einen Wert ein (nicht null) und drückt den "Weiter"-Knopf. Jetzt dreht sich das Quadrat um den Ursprung.

Als nächstes kommt die Darstellung eines Würfels. Wenn man bei den Standardformen einfach den Würfel wählt und die Animation startet, sieht er genauso aus wie das Quadrat. Man kann die schwarzen Drehungen (um die xw- und yw-Ebene) aktivieren und dadurch den Würfel erkennen. Nun will man aber vielleicht auf den ersten Blick sehen, dass es sich hier nicht nur um ein Quadrat handelt, ohne etwas drehen zu müssen. In diesem Fall muss man rechts oben die Projektion verändern. Die linken beiden Felder dort sind überschrieben mit "$1z- >$ ". Damit kann man bestimmen, wie dass die z-Richtung auf dem Bild festgelegt werden soll. Wenn man in beide -0.5 einsetzt und das Bild anzeigt, erhält man das Schrägbild eines Würfels. Wenn man das Minuszeichen weglässt erhält man zwar dasselbe Bild, allerdings hat man dann ein Linkssystem. Wenn man das Schrägbild dreht, kann es sein, dass seine Form recht seltsam aussieht. Dies kann man nicht ändern, es ist eine Folge der Projektion. Wenn man ein Objekt dreht, ist es normalerweise besser, die Projektion auf $1z \to 0x; 0y$ und $1w \to 0x; 0y$, damit die Form nicht verfälscht wird.

Wenn man einen Tesserakt darstellen und drehen will, muss man praktisch dasselbe tun wie beim Würfel. Die einzigen Unterschiede sind, dass jetzt auch die blauen Rotationen verwendet werden und bei der Projektion alle vier Eingabefelder von Bedeutung sind.

A.3 Objekte selber erstellen

Dieser Teil ist eine Anleitung, mit deren Hilfe man sein erstes Objekt erstellen kann. Mit den Einstellungen im mittleren Bereich des Fensters kann man selbst Punkte und Verbindungen definieren. Um dies zu tun, muss man zuerst sicherstellen, dass die Animation pausiert ist. Für die Erstellung von Objekten ist es normalerweise von Vorteil, wenn alle Drehwinkel 0° sind.

Das Objekt, das für dieses Beispiel erstellt werden soll, ist ein vierdimensionales Koordinatensystem. Dazu benötigt man fünf Punkte: Den Ursprung, einen Punkt auf der x-Achse, einen auf der y-Achse, einen auf der z-Achse und einen auf der w-Achse. Man wählt also den Punkt 0 und setzt seine Koordinaten auf $(0|0|0|0)$. Als nächstes setzt man beim Punkt 1 die x-Position auf 100 und alle anderen auf 0. Beim Punkt 2 wird die y-Position auf 100 gesetzt und analog dazu beim Punkt 3 die z-Position und beim Punkt 4 die w-Position. Man hat nun die Punkte $P_0(0|0|0|0)$, $P_1(100|0|0|0)$, $P_2(0|100|0|0)$, $P_3(0|0|100|0)$ und $P_4(0|0|0|100)$.

Für ein vierachsiges Koordinatensystem werden vier Verbindungen benötigt. Die Verbindung 0 sollte ausgelassen werden, damit man bei der Drehung später immer eine Verbindung zur Auswahl hat, die nicht im Bild markiert wird und dort stört. Die Verbindungen 1 bis 4 werden nun also auf sichtbar gestellt. Die Verbindung 1 verbindet die Punkte 0 und 1, Verbindung 2 die Punkte 0 und 2 usw.

Nun ist das Koordinatensystem fertig. Wenn man die Projektion entsprechend anpasst, sind auf dem Bild vier Linien sichtbar. Um sie besser unterscheiden zu können, eignet es sich, ihnen verschiedene Farben zu geben. Als letztes muss man das fertige Objekt speichern. Dazu muss man den "Speichern"-Knopf drücken und dann Speicherort und Name der Datei festlegen.

An diesem Koordinatensystem kann man sehr gut die verschiedenen Drehungen zeigen. Es werden jeweils zwei Achsen gedreht, während die anderen beiden fest an ihrem Ort bleiben.

B Quellcode

```
Public Class VierDreh
    Dim x(60) As Single
    Dim y(60) As Single
    Dim z(60) As Single
    Dim w(60) As Single
    Dim x_j(60) As Single
    Dim y_j(60) As Single
    Dim z_j(60) As Single
    Dim w_j(60) As Single
    Dim vis_x(60) As Single
    Dim vis_y(60) As Single
    Dim line_a(150) As Integer
    Dim line_b(150) As Integer
    Dim there(150) As Boolean
    Dim red(150) As Integer
    Dim green(150) As Integer
    Dim blue(150) As Integer
    Dim p As Single
    Dim g As Graphics
    Dim ofd As New SaveFileDialog
    Dim dfo As New OpenFileDialog

    Private Sub Button1_Click(ByVal sender As System.Object,
                              ByVal e As System.EventArgs) Handles
                                 Button_Start.Click
        Timer.Enabled = True
        For i = 0 To 60
            x_j(i) = x(i)
            y_j(i) = y(i)
            z_j(i) = z(i)
            w_j(i) = w(i)
        Next
        g = PictureBox_Bild.CreateGraphics
        g.TranslateTransform(200, 200)
        g.ScaleTransform(1, -1)
        Call play()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
       System.EventArgs) Handles Timer.Tick
        For i = 0 To 60
            ' ZW /   um XY
            p = z_j(i)
            z_j(i) = p * Math.Cos(TextBox_XY.Text * Math.PI / 180) +
               w_j(i) * Math.Sin(TextBox_XY.Text * Math.PI / 180)
            w_j(i) = -p * Math.Sin(TextBox_XY.Text * Math.PI / 180) +
               w_j(i) * Math.Cos(TextBox_XY.Text * Math.PI / 180)
            ' YW /   um XZ
```

```
        p = y_j(i)
        y_j(i) = p * Math.Cos(TextBox_XZ.Text * Math.PI / 180) +
           w_j(i) * Math.Sin(TextBox_XZ.Text * Math.PI / 180)
        w_j(i) = -p * Math.Sin(TextBox_XZ.Text * Math.PI / 180) +
           w_j(i) * Math.Cos(TextBox_XZ.Text * Math.PI / 180)
        ' YZ /   um XW
        p = y_j(i)
        y_j(i) = p * Math.Cos(TextBox_XW.Text * Math.PI / 180) -
           z_j(i) * Math.Sin(TextBox_XW.Text * Math.PI / 180)
        z_j(i) = p * Math.Sin(TextBox_XW.Text * Math.PI / 180) +
           z_j(i) * Math.Cos(TextBox_XW.Text * Math.PI / 180)
        ' XW /   um YZ
        p = x_j(i)
        x_j(i) = p * Math.Cos(TextBox_YZ.Text * Math.PI / 180) -
           w_j(i) * Math.Sin(TextBox_YZ.Text * Math.PI / 180)
        w_j(i) = p * Math.Sin(TextBox_YZ.Text * Math.PI / 180) +
           w_j(i) * Math.Cos(TextBox_YZ.Text * Math.PI / 180)
        ' XZ /   um YW
        p = x_j(i)
        x_j(i) = p * Math.Cos(TextBox_YW.Text * Math.PI / 180) +
           z_j(i) * Math.Sin(TextBox_YW.Text * Math.PI / 180)
        z_j(i) = -p * Math.Sin(TextBox_YW.Text * Math.PI / 180) +
           z_j(i) * Math.Cos(TextBox_YW.Text * Math.PI / 180)
        ' XY /   um ZW
        p = x_j(i)
        x_j(i) = p * Math.Cos(TextBox_ZW.Text * Math.PI / 180) -
           y_j(i) * Math.Sin(TextBox_ZW.Text * Math.PI / 180)
        y_j(i) = p * Math.Sin(TextBox_ZW.Text * Math.PI / 180) +
           y_j(i) * Math.Cos(TextBox_ZW.Text * Math.PI / 180)

        vis_x(i) = x_j(i) + TextBox_ZinX.Text * z_j(i) +
           TextBox_Winx.Text * w_j(i)
        vis_y(i) = y_j(i) + TextBox_ZinY.Text * z_j(i) +
           TextBox_WinY.Text * w_j(i)
    Next
    g.Clear(Color.White)
    For i = 0 To 150
        If there(i) = True Then
            g.DrawLine(New Pen(Color.FromArgb(red(i), green(i),
               blue(i))), vis_x(line_a(i)), vis_y(line_a(i)),
               vis_x(line_b(i)), vis_y(line_b(i)))
        End If
    Next
    'Momentan ausgew hlter Punkt und momentan ausgew hlte
       Verbindung
    g.FillEllipse(New SolidBrush(Color.FromArgb(255, 0, 255)),
       vis_x(NumericUpDown_Punkt.Value) - 2, vis_y(
       NumericUpDown_Punkt.Value) - 2, 5, 5)
    g.DrawLine(New Pen(Color.FromArgb(red(NumericUpDown_Verbindung
       .Value), green(NumericUpDown_Verbindung.Value), blue(
```

```
        NumericUpDown_Verbindung.Value)), 2), vis_x(line_a(
        NumericUpDown_Verbindung.Value)), vis_y(line_a(
        NumericUpDown_Verbindung.Value)), vis_x(line_b(
        NumericUpDown_Verbindung.Value)), vis_y(line_b(
        NumericUpDown_Verbindung.Value)))
End Sub

Sub pause()
    Timer.Enabled = False
    TextBox_ZW.Enabled = True
    TextBox_YW.Enabled = True
    TextBox_YZ.Enabled = True
    TextBox_Xpos.Enabled = True
    TextBox_Ypos.Enabled = True
    TextBox_Zpos.Enabled = True
    TextBox_Wpos.Enabled = True
    TextBox_XW.Enabled = True
    TextBox_XZ.Enabled = True
    TextBox_XY.Enabled = True
    TextBox_ZinX.Enabled = True
    TextBox_ZinY.Enabled = True
    TextBox_Winx.Enabled = True
    TextBox_WinY.Enabled = True
    NumericUpDown_LineA.Enabled = True
    NumericUpDown_LineB.Enabled = True
    NumericUpDown_Zeit.Enabled = True
    Button_PauseWeiter.Text = "Weiter"
End Sub

Sub play()
    Timer.Enabled = True
    TextBox_ZW.Enabled = False
    TextBox_YW.Enabled = False
    TextBox_YZ.Enabled = False
    TextBox_Xpos.Enabled = False
    TextBox_Ypos.Enabled = False
    TextBox_Zpos.Enabled = False
    TextBox_Wpos.Enabled = False
    TextBox_XW.Enabled = False
    TextBox_XZ.Enabled = False
    TextBox_XY.Enabled = False
    TextBox_ZinX.Enabled = False
    TextBox_ZinY.Enabled = False
    TextBox_Winx.Enabled = False
    TextBox_WinY.Enabled = False
    NumericUpDown_LineA.Enabled = False
    NumericUpDown_LineB.Enabled = False
    NumericUpDown_Zeit.Enabled = False
    Button_PauseWeiter.Text = "Pause"
End Sub
```

```
Private Sub NumericUpDown1_ValueChanged(ByVal sender As System.
   Object, ByVal e As System.EventArgs) Handles
   NumericUpDown_Punkt.ValueChanged
    TextBox_Xpos.Text = x(NumericUpDown_Punkt.Value)
    TextBox_Ypos.Text = y(NumericUpDown_Punkt.Value)
    TextBox_Zpos.Text = z(NumericUpDown_Punkt.Value)
    TextBox_Wpos.Text = w(NumericUpDown_Punkt.Value)
End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
   System.EventArgs) Handles MyBase.Load
    For i = 0 To 60
        x(i) = 0
        y(i) = 0
        z(i) = 0
        w(i) = 0
    Next
    For i = 0 To 150
        line_a(i) = 0
        line_b(i) = 0
        there(i) = False
        red(i) = 0
        green(i) = 0
        blue(i) = 0
    Next
End Sub

Private Sub TextBox4_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_Xpos.TextChanged
    If TextBox_Xpos.Text.Length = 0 Then
        TextBox_Xpos.Text = "0"
    End If
    x(NumericUpDown_Punkt.Value) = TextBox_Xpos.Text
End Sub

Private Sub TextBox5_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_Ypos.TextChanged
    If TextBox_Ypos.Text.Length = 0 Then
        TextBox_Ypos.Text = "0"
    End If
    y(NumericUpDown_Punkt.Value) = TextBox_Ypos.Text
End Sub

Private Sub TextBox6_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_Zpos.TextChanged
    If TextBox_Zpos.Text.Length = 0 Then
        TextBox_Zpos.Text = "0"
    End If
    z(NumericUpDown_Punkt.Value) = TextBox_Zpos.Text
```

```
End Sub

Private Sub TextBox7_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_Wpos.TextChanged
    If TextBox_Wpos.Text.Length = 0 Then
        TextBox_Wpos.Text = "0"
    End If
    w(NumericUpDown_Punkt.Value) = TextBox_Wpos.Text
End Sub

'Verbindungen
Private Sub NumericUpDown2_ValueChanged(ByVal sender As System.
   Object, ByVal e As System.EventArgs) Handles
   NumericUpDown_Verbindung.ValueChanged
    NumericUpDown_LineA.Value = line_a(NumericUpDown_Verbindung.
       Value)
    NumericUpDown_LineB.Value = line_b(NumericUpDown_Verbindung.
       Value)
    NumericUpDown_Rot.Value = red(NumericUpDown_Verbindung.Value)
    NumericUpDown_Gruen.Value = green(NumericUpDown_Verbindung.
       Value)
    NumericUpDown_Blau.Value = blue(NumericUpDown_Verbindung.Value
       )
    PictureBox_Farbvorschau.BackColor = Color.FromArgb(
       NumericUpDown_Rot.Value, NumericUpDown_Gruen.Value,
       NumericUpDown_Blau.Value)
    CheckBox_There.Checked = there(NumericUpDown_Verbindung.Value)
End Sub

Private Sub CheckBox1_CheckedChanged(ByVal sender As System.Object
   , ByVal e As System.EventArgs) Handles CheckBox_There.
   CheckedChanged
    there(NumericUpDown_Verbindung.Value) = CheckBox_There.Checked
End Sub

Private Sub NumericUpDown4_ValueChanged(ByVal sender As System.
   Object, ByVal e As System.EventArgs) Handles
   NumericUpDown_LineB.ValueChanged
    line_b(NumericUpDown_Verbindung.Value) = NumericUpDown_LineB.
       Value
End Sub

Private Sub NumericUpDown3_ValueChanged(ByVal sender As System.
   Object, ByVal e As System.EventArgs) Handles
   NumericUpDown_LineA.ValueChanged
    line_a(NumericUpDown_Verbindung.Value) = NumericUpDown_LineA.
       Value
End Sub

Private Sub NumericUpDown5_ValueChanged(ByVal sender As System.
```

```
    Object, ByVal e As System.EventArgs) Handles NumericUpDown_Rot.
    ValueChanged
     red(NumericUpDown_Verbindung.Value) = NumericUpDown_Rot.Value
     PictureBox_Farbvorschau.BackColor = Color.FromArgb(
        NumericUpDown_Rot.Value, NumericUpDown_Gruen.Value,
        NumericUpDown_Blau.Value)
End Sub

Private Sub NumericUpDown6_ValueChanged(ByVal sender As System.
    Object, ByVal e As System.EventArgs) Handles
    NumericUpDown_Gruen.ValueChanged
     green(NumericUpDown_Verbindung.Value) = NumericUpDown_Gruen.
        Value
     PictureBox_Farbvorschau.BackColor = Color.FromArgb(
        NumericUpDown_Rot.Value, NumericUpDown_Gruen.Value,
        NumericUpDown_Blau.Value)
End Sub

Private Sub NumericUpDown7_ValueChanged(ByVal sender As System.
    Object, ByVal e As System.EventArgs) Handles NumericUpDown_Blau
    .ValueChanged
     blue(NumericUpDown_Verbindung.Value) = NumericUpDown_Blau.
        Value
     PictureBox_Farbvorschau.BackColor = Color.FromArgb(
        NumericUpDown_Rot.Value, NumericUpDown_Gruen.Value,
        NumericUpDown_Blau.Value)
End Sub

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles Button_PauseWeiter.Click
     If Timer.Enabled = True Then
         Call pause()
     Else
         Call play()
     End If
End Sub

' Speichern und Laden
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles Button_Speichern.Click
     Dim path As String
     If ofd.ShowDialog() = DialogResult.OK Then
         path = ofd.FileName & ".txt"


         Dim sw As System.IO.StreamWriter = IO.File.CreateText(path
            )
         For i = 0 To 60
             sw.WriteLine(x(i))
             sw.WriteLine(y(i))
```

```
            sw.WriteLine(z(i))
            sw.WriteLine(w(i))
        Next
        For i = 0 To 150
            sw.WriteLine(line_a(i))
            sw.WriteLine(line_b(i))
            sw.WriteLine(there(i))
            sw.WriteLine(red(i))
            sw.WriteLine(green(i))
            sw.WriteLine(blue(i))
        Next
        sw.WriteLine(TextBox_ZinX.Text)
        sw.WriteLine(TextBox_ZinY.Text)
        sw.WriteLine(TextBox_Winx.Text)
        sw.WriteLine(TextBox_WinY.Text)
        sw.Close()
    End If
End Sub

Private Sub Button5_Click(ByVal sender As System.Object, ByVal e
   As System.EventArgs) Handles Button_Laden.Click
    Dim path As String
    If dfo.ShowDialog() = DialogResult.OK Then
        path = dfo.FileName

        Dim sr As System.IO.StreamReader = IO.File.OpenText(path)
        For i = 0 To 60
            x(i) = sr.ReadLine
            y(i) = sr.ReadLine
            z(i) = sr.ReadLine
            w(i) = sr.ReadLine
        Next
        For i = 0 To 150
            line_a(i) = sr.ReadLine
            line_b(i) = sr.ReadLine
            there(i) = sr.ReadLine
            red(i) = sr.ReadLine
            green(i) = sr.ReadLine
            blue(i) = sr.ReadLine
        Next
        TextBox_ZinX.Text = sr.ReadLine
        TextBox_ZinY.Text = sr.ReadLine
        TextBox_Winx.Text = sr.ReadLine
        TextBox_WinY.Text = sr.ReadLine
        sr.Close()
    End If
End Sub

Private Sub Button11_Click(ByVal sender As System.Object, ByVal e
   As System.EventArgs) Handles Button_Formen.Click
```

```
    If Button_Formen.Text = "Vorlagen" Then
        Button_Tesserakt.Visible = True
        Button_Pentachoron.Visible = True
        Button_Wuerfel.Visible = True
        Button_Tetraeder.Visible = True
        Button_Quadrat.Visible = True
        Button_Dreieck.Visible = True
        Button_Formen.Text = "Schliessen"
    Else
        Button_Tesserakt.Visible = False
        Button_Pentachoron.Visible = False
        Button_Wuerfel.Visible = False
        Button_Tetraeder.Visible = False
        Button_Quadrat.Visible = False
        Button_Dreieck.Visible = False
        Button_Formen.Text = "Vorlagen"
    End If
End Sub


Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
   As System.EventArgs) Handles Button_Tesserakt.Click
    For i = 0 To 60
        x(i) = 0
        y(i) = 0
        z(i) = 0
        w(i) = 0
    Next
    For i = 0 To 150
        line_a(i) = 0
        line_b(i) = 0
        there(i) = False
        red(i) = 0
        green(i) = 0
        blue(i) = 0
    Next
    'Tesserakt, Mittelpunkt im Ursprung, Seitenl nge 200
    x(1) = 100
    y(1) = 100
    z(1) = 100
    w(1) = 100

    x(2) = -100
    y(2) = 100
    z(2) = 100
    w(2) = 100

    x(3) = 100
    y(3) = -100
    z(3) = 100
```

```
w(3) = 100

x(4) = 100
y(4) = 100
z(4) = -100
w(4) = 100

x(5) = 100
y(5) = 100
z(5) = 100
w(5) = -100

x(6) = -100
y(6) = -100
z(6) = 100
w(6) = 100

x(7) = -100
y(7) = 100
z(7) = -100
w(7) = 100

x(8) = -100
y(8) = 100
z(8) = 100
w(8) = -100

x(9) = -100
y(9) = -100
z(9) = -100
w(9) = 100

x(10) = -100
y(10) = -100
z(10) = 100
w(10) = -100

x(11) = -100
y(11) = 100
z(11) = -100
w(11) = -100

x(12) = -100
y(12) = -100
z(12) = -100
w(12) = -100

x(13) = 100
y(13) = 100
z(13) = -100
```

```
w(13) = -100

x(14) = 100
y(14) = -100
z(14) = 100
w(14) = -100

x(15) = 100
y(15) = -100
z(15) = -100
w(15) = 100

x(16) = 100
y(16) = -100
z(16) = -100
w(16) = -100

line_a(1) = 1
line_b(1) = 2
there(1) = True
line_a(2) = 1
line_b(2) = 3
there(2) = True
line_a(3) = 1
line_b(3) = 4
there(3) = True
line_a(4) = 1
line_b(4) = 5
there(4) = True

line_a(5) = 3
line_b(5) = 15
there(5) = True
line_a(6) = 3
line_b(6) = 6
there(6) = True
line_a(7) = 3
line_b(7) = 14
there(7) = True

line_a(8) = 6
line_b(8) = 2
there(8) = True
line_a(9) = 6
line_b(9) = 9
there(9) = True
line_a(10) = 6
line_b(10) = 10
there(10) = True
```

```
line_a(11) = 2
line_b(11) = 7
there(11) = True
line_a(12) = 2
line_b(12) = 8
there(12) = True

line_a(13) = 4
line_b(13) = 15
there(13) = True
line_a(14) = 4
line_b(14) = 7
there(14) = True
line_a(15) = 4
line_b(15) = 13
there(15) = True

line_a(16) = 15
line_b(16) = 9
there(16) = True
line_a(17) = 15
line_b(17) = 16
there(17) = True

line_a(18) = 9
line_b(18) = 7
there(18) = True
line_a(19) = 9
line_b(19) = 12
there(19) = True

line_a(20) = 7
line_b(20) = 11
there(20) = True

line_a(21) = 5
line_b(21) = 14
there(21) = True
line_a(22) = 5
line_b(22) = 8
there(22) = True
line_a(23) = 5
line_b(23) = 13
there(23) = True

line_a(24) = 14
line_b(24) = 10
there(24) = True
line_a(25) = 14
line_b(25) = 16
```

```
        there(25) = True

        line_a(26) = 10
        line_b(26) = 8
        there(26) = True
        line_a(27) = 10
        line_b(27) = 12
        there(27) = True

        line_a(28) = 8
        line_b(28) = 11
        there(28) = True

        line_a(29) = 13
        line_b(29) = 16
        there(29) = True
        line_a(30) = 13
        line_b(30) = 11
        there(30) = True

        line_a(31) = 16
        line_b(31) = 12
        there(31) = True

        line_a(32) = 12
        line_b(32) = 11
        there(32) = True
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e
       As System.EventArgs) Handles Button_Pentachoron.Click
        For i = 0 To 60
            x(i) = 0
            y(i) = 0
            z(i) = 0
            w(i) = 0
        Next
        For i = 0 To 150
            line_a(i) = 0
            line_b(i) = 0
            there(i) = False
            red(i) = 0
            green(i) = 0
            blue(i) = 0
        Next
        ' Pentachoron, Mittelpunkt im Ursprung, Seitenl nge 200
        x(1) = 100
        y(1) = -57.735026918965
        z(1) = -40.824829046387
        w(1) = -63.245553203367
```

```
x(2) = -100
y(2) = -57.735026918965
z(2) = -40.824829046387
w(2) = -63.245553203367

x(3) = 0
y(3) = 115.47005383793
z(3) = -40.824829046387
w(3) = -63.245553203367

x(4) = 0
y(4) = 0
z(4) = 122.47448713916
w(4) = -63.245553203367

x(5) = 0
y(5) = 0
z(5) = 0
w(5) = 94.868329805053

line_a(1) = 1
line_b(1) = 2
there(1) = True

line_a(2) = 1
line_b(2) = 3
there(2) = True

line_a(3) = 1
line_b(3) = 4
there(3) = True

line_a(4) = 1
line_b(4) = 5
there(4) = True

line_a(5) = 2
line_b(5) = 3
there(5) = True

line_a(6) = 2
line_b(6) = 4
there(6) = True

line_a(7) = 2
line_b(7) = 5
there(7) = True

line_a(8) = 3
```

```
        line_b(8) = 4
        there(8) = True

        line_a(9) = 3
        line_b(9) = 5
        there(9) = True

        line_a(10) = 4
        line_b(10) = 5
        there(10) = True
    End Sub

    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e
       As System.EventArgs) Handles Button_Tetraeder.Click
        For i = 0 To 60
            x(i) = 0
            y(i) = 0
            z(i) = 0
            w(i) = 0
        Next
        For i = 0 To 150
            line_a(i) = 0
            line_b(i) = 0
            there(i) = False
            red(i) = 0
            green(i) = 0
            blue(i) = 0
        Next
        ' Tetraeder, Seitenl nge 200, Mittelpunkt im Ursprung,
           DIESMAL RICHTIG!
        x(1) = 100
        y(1) = -57.735026918965
        z(1) = -40.824829046387
        w(1) = 0    'der ist nur dreidimensional

        x(2) = -100
        y(2) = -57.735026918965
        z(2) = -40.824829046387
        w(2) = 0

        x(3) = 0
        y(3) = 115.47005383793
        z(3) = -40.824829046387
        w(3) = 0

        x(4) = 0
        y(4) = 0
        z(4) = 122.47448713916
        w(4) = 0
```

```
        line_a(1) = 1
        line_b(1) = 2
        there(1) = True

        line_a(2) = 1
        line_b(2) = 3
        there(2) = True

        line_a(3) = 1
        line_b(3) = 4
        there(3) = True

        line_a(4) = 2
        line_b(4) = 3
        there(4) = True

        line_a(5) = 2
        line_b(5) = 4
        there(5) = True

        line_a(6) = 3
        line_b(6) = 4
        there(6) = True
    End Sub

    Private Sub Button10_Click(ByVal sender As System.Object, ByVal e
       As System.EventArgs) Handles Button_Dreieck.Click

        For i = 0 To 60
            x(i) = 0
            y(i) = 0
            z(i) = 0
            w(i) = 0
        Next
        For i = 0 To 150
            line_a(i) = 0
            line_b(i) = 0
            there(i) = False
            red(i) = 0
            green(i) = 0
            blue(i) = 0
        Next
        ' Gleichseitiges Dreieck
        x(1) = 100
        y(1) = -57.735026918965
        z(1) = 0
        w(1) = 0

        x(2) = -100
        y(2) = -57.735026918965
```

```
    z(2) = 0
    w(2) = 0

    x(3) = 0
    y(3) = 115.47005383793
    z(3) = 0
    w(3) = 0

    line_a(1) = 1
    line_b(1) = 2
    there(1) = True

    line_a(2) = 1
    line_b(2) = 3
    there(2) = True

    line_a(3) = 2
    line_b(3) = 3
    there(3) = True
End Sub

Private Sub Button7_Click(ByVal sender As System.Object, ByVal e
   As System.EventArgs) Handles Button_Wuerfel.Click
    For i = 0 To 60
        x(i) = 0
        y(i) = 0
        z(i) = 0
        w(i) = 0
    Next
    For i = 0 To 150
        line_a(i) = 0
        line_b(i) = 0
        there(i) = False
        red(i) = 0
        green(i) = 0
        blue(i) = 0
    Next
    'W rfel , Mittelpunkt im Ursprung, Seitenl nge 200
    x(1) = 100
    y(1) = 100
    z(1) = 100
    w(1) = 0

    x(2) = -100
    y(2) = 100
    z(2) = 100
    w(2) = 0

    x(3) = 100
    y(3) = -100
```

```
z(3) = 100
w(3) = 0

x(4) = 100
y(4) = 100
z(4) = -100
w(4) = 0

x(5) = -100
y(5) = -100
z(5) = 100
w(5) = 0

x(6) = 100
y(6) = -100
z(6) = -100
w(6) = 0

x(7) = -100
y(7) = 100
z(7) = -100
w(7) = 0

x(8) = -100
y(8) = -100
z(8) = -100
w(8) = 0

line_a(1) = 1
line_b(1) = 2
there(1) = True

line_a(2) = 1
line_b(2) = 3
there(2) = True

line_a(3) = 1
line_b(3) = 4
there(3) = True

line_a(4) = 2
line_b(4) = 5
there(4) = True

line_a(5) = 2
line_b(5) = 7
there(5) = True

line_a(6) = 3
line_b(6) = 5
```

```
        there(6) = True

        line_a(7) = 3
        line_b(7) = 6
        there(7) = True

        line_a(8) = 4
        line_b(8) = 6
        there(8) = True

        line_a(9) = 4
        line_b(9) = 7
        there(9) = True

        line_a(10) = 5
        line_b(10) = 8
        there(10) = True

        line_a(11) = 6
        line_b(11) = 8
        there(11) = True

        line_a(12) = 7
        line_b(12) = 8
        there(12) = True
    End Sub

    Private Sub Button9_Click(ByVal sender As System.Object, ByVal e
       As System.EventArgs) Handles Button_Quadrat.Click
        For i = 0 To 60
            x(i) = 0
            y(i) = 0
            z(i) = 0
            w(i) = 0
        Next
        For i = 0 To 150
            line_a(i) = 0
            line_b(i) = 0
            there(i) = False
            red(i) = 0
            green(i) = 0
            blue(i) = 0
        Next
        'Quadrat, Mittelpunkt im Ursprung, Seitenl nge 200
        x(1) = 100
        y(1) = 100
        z(1) = 0
        w(1) = 0

        x(2) = -100
```

```
    y(2) = 100
    z(2) = 0
    w(2) = 0

    x(3) = 100
    y(3) = -100
    z(3) = 0
    w(3) = 0

    x(4) = -100
    y(4) = -100
    z(4) = 0
    w(4) = 0

    line_a(1) = 1
    line_b(1) = 2
    there(1) = True

    line_a(2) = 1
    line_b(2) = 3
    there(2) = True

    line_a(3) = 2
    line_b(3) = 4
    there(3) = True

    line_a(4) = 3
    line_b(4) = 4
    there(4) = True
End Sub

Private Sub NumericUpDown8_ValueChanged(ByVal sender As System.
   Object, ByVal e As System.EventArgs) Handles NumericUpDown_Zeit
   .ValueChanged
    Timer.Interval = NumericUpDown_Zeit.Value
End Sub

Private Sub TextBox1_KeyPress(ByVal sender As Object, _
                              ByVal e As System.Windows.Forms.
                                 KeyPressEventArgs) _
                                      Handles TextBox_ZW.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub

Private Sub TextBox2_KeyPress(ByVal sender As Object, _
                              ByVal e As System.Windows.Forms.
```

```
                                    KeyPressEventArgs) _
                                        Handles TextBox_YW.KeyPress

        If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
            Chr(8) Then
            e.Handled = True
        End If
    End Sub

    Private Sub TextBox3_KeyPress(ByVal sender As Object, _
                                  ByVal e As System.Windows.Forms.
                                    KeyPressEventArgs) _
                                        Handles TextBox_YZ.KeyPress

        If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
            Chr(8) Then
            e.Handled = True
        End If
    End Sub
    Private Sub TextBox4_KeyPress(ByVal sender As Object, _
                                  ByVal e As System.Windows.Forms.
                                    KeyPressEventArgs) _
                                        Handles TextBox_Xpos.KeyPress

        If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
            Chr(8) Then
            e.Handled = True
        End If
    End Sub
    Private Sub TextBox5_KeyPress(ByVal sender As Object, _
                                  ByVal e As System.Windows.Forms.
                                    KeyPressEventArgs) _
                                        Handles TextBox_Ypos.KeyPress

        If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
            Chr(8) Then
            e.Handled = True
        End If
    End Sub
    Private Sub TextBox6_KeyPress(ByVal sender As Object, _
                                  ByVal e As System.Windows.Forms.
                                    KeyPressEventArgs) _
                                        Handles TextBox_Zpos.KeyPress

        If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
            Chr(8) Then
            e.Handled = True
        End If
    End Sub
    Private Sub TextBox7_KeyPress(ByVal sender As Object, _
```

```
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_Wpos.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox8_KeyPress(ByVal sender As Object, _
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_XW.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox9_KeyPress(ByVal sender As Object, _
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_XZ.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox10_KeyPress(ByVal sender As Object, _
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_XY.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox11_KeyPress(ByVal sender As Object, _
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_ZinX.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox12_KeyPress(ByVal sender As Object, _
```

```
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_ZinY.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox13_KeyPress(ByVal sender As Object, _
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_Winx.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub
Private Sub TextBox14_KeyPress(ByVal sender As Object, _
                                ByVal e As System.Windows.Forms.
                                   KeyPressEventArgs) _
                                        Handles TextBox_WinY.KeyPress

    If "1234567890.-".Contains(e.KeyChar) = False And e.KeyChar <>
        Chr(8) Then
        e.Handled = True
    End If
End Sub

Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_ZW.TextChanged
    If TextBox_ZW.Text.Length = 0 Then
        TextBox_ZW.Text = "0"
    End If
End Sub

Private Sub TextBox2_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_YW.TextChanged
    If TextBox_YW.Text.Length = 0 Then
        TextBox_YW.Text = "0"
    End If
End Sub

Private Sub TextBox3_TextChanged(ByVal sender As System.Object,
   ByVal e As System.EventArgs) Handles TextBox_YZ.TextChanged
    If TextBox_YZ.Text.Length = 0 Then
        TextBox_YZ.Text = "0"
    End If
End Sub
```

```
    Private Sub TextBox8_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_XW.TextChanged
        If TextBox_XW.Text.Length = 0 Then
            TextBox_XW.Text = "0"
        End If
    End Sub

    Private Sub TextBox9_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_XZ.TextChanged
        If TextBox_XZ.Text.Length = 0 Then
            TextBox_XZ.Text = "0"
        End If
    End Sub

    Private Sub TextBox10_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_XY.TextChanged
        If TextBox_XY.Text.Length = 0 Then
            TextBox_XY.Text = "0"
        End If
    End Sub

    Private Sub TextBox11_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_ZinX.TextChanged
        If TextBox_ZinX.Text.Length = 0 Then
            TextBox_ZinX.Text = "0"
        End If
    End Sub

    Private Sub TextBox12_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_ZinY.TextChanged
        If TextBox_ZinY.Text.Length = 0 Then
            TextBox_ZinY.Text = "0"
        End If
    End Sub

    Private Sub TextBox13_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_Winx.TextChanged
        If TextBox_Winx.Text.Length = 0 Then
            TextBox_Winx.Text = "0"
        End If
    End Sub

    Private Sub TextBox14_TextChanged(ByVal sender As System.Object,
       ByVal e As System.EventArgs) Handles TextBox_WinY.TextChanged
        If TextBox_WinY.Text.Length = 0 Then
            TextBox_WinY.Text = "0"
        End If
    End Sub
End Class
```

C Anti-Plagiat-Erklärung

Ich erkläre hiermit, dass

- diese Arbeit weder abgeschrieben noch kopiert oder aus dem Internet übernommen wurde,
- der Quellennachweis korrekt und vollständig ist,
- die dargestellten Daten und Resultate selber und korrekt erhoben und verarbeitet wurden.

Name: Vorname:

Ort: Datum:

Unterschrift: