

Zentralprojektion auf dem PC

Maturarbeit von Christoph Leuzinger, 4sb

**Kantonsschule Schaffhausen
Fach Mathematik, 2000/2001**

Betreut durch Christoph Stamm

Abstract

Zur Darstellung eines Körpers in Zentralperspektive wird ein Computerprogramm entwickelt. Mit Methoden der Vektorgeometrie werden Punkte projiziert. Die Flächen des Körpers werden durch Punktgitter gerastert. Ein Kriterium zur Sichtbarkeit von Punkten und die Schrittweite der Punktgitter werden untersucht. Die Programmierung erfolgt in *PASCAL*.

Inhaltsverzeichnis

1 Einleitung	3
1.1 Camera obscura	3
1.2 Das Auge	3
1.3 Anwendungen heute	4
1.4 Zentralprojektion	4
1.5 Aus dem Raum in die Ebene	4
2 Theorie	5
2.1 Zentralprojektion	5
2.1.1 Zur Zentralprojektion mit Vektorgeometrie notwendige Elemente	5
2.1.2 Projektion von Punkten	5
2.1.3 Projektion von Geraden	6
2.1.4 Projektion von Parallelogrammen	6
2.1.4.1 Sichtbarkeit	7
3 Umsetzung	8
3.1 Programmiersprache	8
3.2 Grundgedanke	8
3.3 Eingabe der zur Zentralprojektion notwendigen Elemente	8
3.3.1 Projektionsebene	8
3.3.2 Augpunkt	9
3.3.3 Das Objekt	9
3.4 Umsetzung der Methoden zur Projektion von Körpern	10
3.4.1 Projektion von Punkten	10
3.4.1.1 Spiegelung	11
3.4.2 Projektion von Geraden	11
3.4.3 Projektion von Parallelogrammen	12
3.4.4 Sichtbarkeit von Punkten	14
3.5 Koordinatentransformation	14
3.5.1 Ursprung des Bildschirmsystems	14
3.5.2 Basisvektoren des Bildschirmsystems	14
3.5.3 Koordinaten im Bildschirmsystem	16
3.5.3.1 Bestimmung der Koordinaten	17
3.5.4 Bildschirmnullpunkt des PCs	17
3.5.5 Skalierung des Bildausschnitts	18
3.5.6 Verschieben des Bildschirmsystems	18
3.6 Schrittweite bei der Projektion von Geraden und Flächen	18
4 Resultate	20
4.1 Untersuchte Parameter	20
4.2 Bildserien	20
4.2.1 Variation der Höhe des Projektionszentrums	20
4.2.2 Zunehmende Entfernung vom Objekt	23
4.2.3 Kreisbewegung	25
4.2.4 Transparenzeffekt	28
5 Schluss	31
5.1 Zusammenfassung	31
5.2 Ausblick: Alternative Projektionsverfahren	31
5.3 Literatur- und Quellenverzeichnis	31
Anhang A: Vektorgeometrie im Raum	32
Anhang B: Programmcode	35

1 Einleitung

1.1 Camera obscura

Schon vor über 2000 Jahren machte der griechische Philosoph *Aristoteles* während einer Sonnenfinsternis die Beobachtung, dass die Sonne mehrfach im Schatten eines Baumes erschien. Er folgerte, dass diese Abbilder durch Lücken im Blattwerk des Baumes zustande kamen [1].

Dieses Phänomen griffen findige Bastler auf und bauten eine sog. *camera obscura*. Diese besteht aus einem Kasten, in den kein Licht eindringen kann (lat. *obscurus* = dunkel), ausser durch ein kleines Loch in der Vorderseite des Kastens. Die dort einfallenden Lichtstrahlen treffen auf der Rückseite auf und geben ein spiegelverkehrtes Abbild der Umwelt ab.

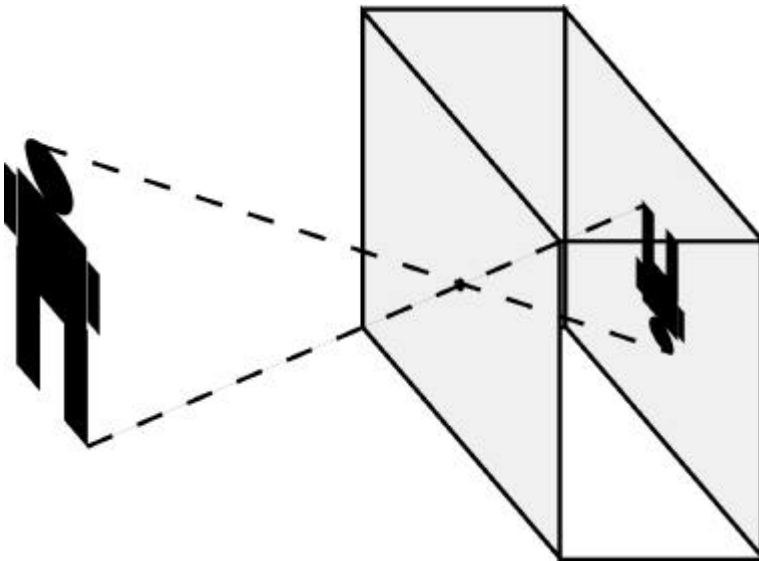


Abb. 1.1:

Prinzip der *camera obscura*

Im 19. Jahrhundert wurden Häuser als Platz für Touristen zu solchen Kammern umfunktioniert, die als Attraktion dienten [2]. Auch wurde das Verfahren als Hilfe zum Zeichnen verwendet, indem das Abbild an der Wand nachgezeichnet wurde.

1.2 Das Auge

Das Prinzip ist aber schon viel älter. Das menschliche Auge verwendet ein ähnliches Prinzip zur Bilderzeugung.

Die einfallenden Lichtstrahlen gelangen durch die Pupille auf die Netzhaut. (Davor werden sie allerdings noch an der Linse gebrochen.)

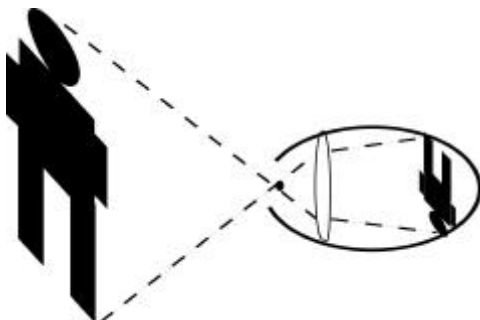


Abb. 1.2:

Bilderzeugung beim menschlichen Auge

1.3 Anwendungen heute

Diese Anwendung ist auch heute noch verbreitet. Man kennt sie von der Fotokamera, aus der Architektur [3] und speziell auch im Zusammenhang mit Computern aus Simulatoren oder Spielen. Man kann zum Beispiel in einem Flugsimulator die Welt so sehen, als säße man tatsächlich im Cockpit und schaute zum Boden hinunter. In Tat und Wahrheit befindet sich das Bild aber auf der Bildschirmfläche. Der Computer übernimmt hier die Darstellung des Raums in einer Ebene.

1.4 Zentralprojektion

Dieses Verfahren hat als Grundlage die Zentralprojektion. Von einem Originalbild wird ein Abbild erstellt, indem man Strahlen von dem Original durch ein Punkt laufen lässt und so auf einer Fläche ein Abbild erstellt.

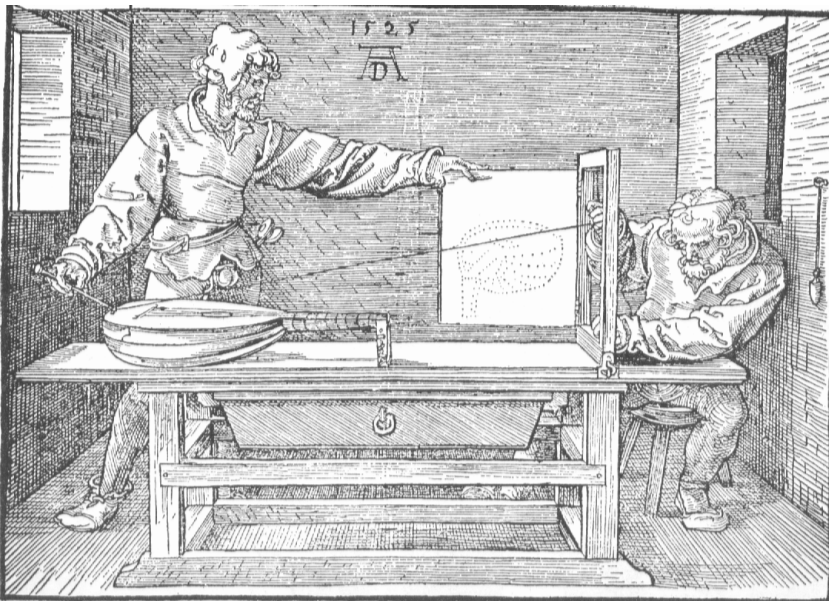


Abb. 1.3:

„Der Zeichner der Laute“
A. Dürer, Holzschnitt, 1525

Die Abb. 1.3 zeigt, wie die beiden Zeichner ein Abbild einer Laute erstellen. Die an der Wand befestigte Schnur wird zum zu zeichnenden Punkt gespannt. Danach wird mit einer horizontalen und einer vertikalen Schnur am Rahmen der Punkt fixiert, durch den die lange Schnur durch den Rahmen geht. Die Fläche wird nachher in den Rahmen geklappt und der Punkt darauf gezeichnet.

1.5 Aus dem Raum in die Ebene

Ob eine Schnur durch einen Rahmen gespannt wird oder Lichtstrahlen auf eine Wand treffen: All diesen Verfahren und Anwendungen ist der Zweck gemeinsam, einen Gegenstand im Raum in eine Ebene abzubilden. Die Körper und Punkte, die im Raum sind, werden in diese Ebene übergeführt. Das ist auch der Zweck des Programms, das dieser Arbeit zugrunde liegt.

2 Theorie

2.1 Zentralprojektion

Das Programm, das die Gedanken aus der Einleitung zum Vorbild hat, setzt das Prinzip Zentralprojektion mit Hilfe der Vektorgeometrie (siehe *Literaturverzeichnis* [4], [5], [6] und *Anhang A*) um.

Das Grundprinzip der Zentralprojektion ist dasjenige, das auf den *Abb. 1.1, 1.2* und *1.3* zu sehen ist. Ein Strahl wird von einem Punkt im Raum aus durch ein Projektionszentrum gelegt und mit einer Ebene geschnitten, auf die projiziert wird. So entsteht ein perspektivische Abbild des Körpers im Raum.

2.1.1 Zur Zentralprojektion mit Vektorgeometrie notwendige Elemente

Um eine Zentralprojektion durchzuführen, müssen folgende Elemente in einem kartesischen, dreidimensionalen System gegeben sein: das *Projektionszentrum* P (oder der *Augpunkt*, wie er auch genannt wird) und die *Projektionsebene* π (auch *Bildebene*).

2.1.2 Projektion von Punkten

Als erster Schritt soll ein einzelner Punkt projiziert werden.

Gegeben sind die in *2.1.1* erwähnten Elemente und der zu projizierende *Punkt* X .

Die Strahlen, die durch das Projektionszentrum P laufen, sind *Geraden*. Sie gehen vom Punkt X im Raum aus, laufen durch den Augpunkt und treffen in einem *Spurpunkt* X' (*Durchstosspunkt*, *Bildpunkt*) auf die Projektionsebene π . Dieser Spurpunkt ist die Projektion des Punktes X .

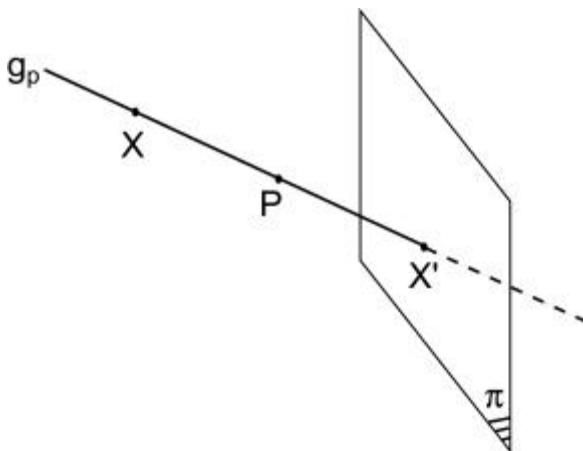


Abb. 2.1:

Elemente der *Zentralprojektion* mit Hilfe der Vektorgeometrie

Die Darstellung in *Abb. 2.1* zeigt den Augpunkt P , der zwischen dem Punkt im Raum X und der Projektionsebene π liegt. Denkbar wäre auch die Situation, in der die Projektionsebene π zwischen dem Punkt X und dem Augpunkt P liegt (siehe *Abb. 1.3*); in dieser Arbeit wird aber ausschliesslich von der in *Abb. 2.1* dargestellten Situation ausgegangen. Somit ergibt sich, wie bei der *camera obscura*, ein spiegelverkehrtes Abbild.

Der erste Schritt, um beliebige Körper im Raum zu projizieren, ist die Projektion von Punkten im Raum.

Abb. 2.1 zeigt, wie eine Projektionsgerade g_p , die von dem zu projizierenden Punkt X durch den Augpunkt geht und die Projektionsebene im Bildpunkt X' durchstösst.

Die *Parameterdarstellung* der Projektionsgeraden g_p lautet:

$$g_p: \vec{OX}' = \vec{OX} + t \vec{XP} \quad (1)$$

Wobei \vec{OX}' der Ortsvektor des Bildpunktes X' , \vec{OX} der Ortsvektor des Punktes X , t der Parameter der Projektionsgeraden g_p und \vec{XP} der Verbindungsvektor zwischen X und P und auch der Richtungsvektor der Geraden ist.

2.1.3 Projektion von Geraden

Somit können einzelne Punkte projiziert werden. Der nächste Schritt ist die Projektion von geraden Strecken.

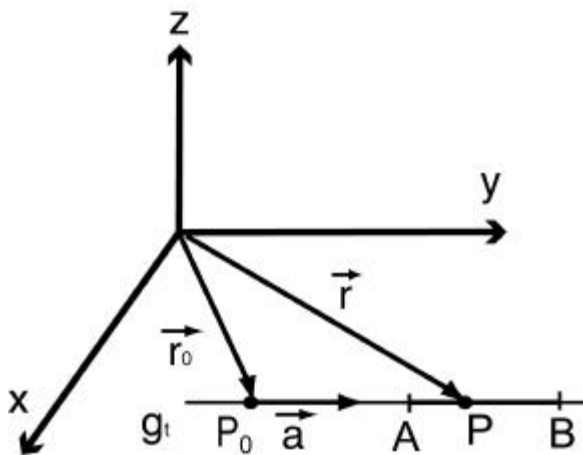


Abb. 2.2:

Die Strecke AB liegt auf der Geraden g_t , die durch den Trägerpunkt P_0 und den Richtungsvektor \vec{a} dargestellt wird.

Neben den zur Zentralprojektion notwendigen Elementen ist ein Anfangspunkt A und ein Endpunkt B einer Strecke gegeben. Diese Strecke liegt auf einer Geraden g_t , deren Parameterdarstellung wie folgt lautet:

$$g_t: \vec{r} = \vec{r}_0 + t \vec{a} \quad (2)$$

Wobei \vec{r} der Ortsvektor eines Punktes auf der Geraden, \vec{r}_0 der Ortsvektor des Trägerpunktes P_0 der Geraden, t der Parameter und \vec{a} der Richtungsvektor der Geraden ist.

Wenn man die Strecke AB als *Reihe von Punkten* betrachtet, kann man beliebige Punkte P der Strecke „anfahen“ und, wie oben beschrieben, projizieren.

2.1.4 Projektion von Parallelogrammen

Mit der Möglichkeit, Geraden zu projizieren, lassen sich Körper als Gitternetzkonstruktionen darstellen. Mit Flächen hingegen lassen sich Körper besser darstellen. In dieser Arbeit werden als Flächen ausschliesslich Parallelogramme verwendet.

Für die Darstellung von Parallelogrammen kann ebenfalls die Parameterdarstellung gewählt werden:

$$p: \vec{r} = \vec{r}_0 + t_1 \vec{a}_1 + t_2 \vec{a}_2 \quad (3)$$

Wobei \vec{r} der Ortsvektor eines Punktes des Parallelogramms, \vec{r}_0 der Ortsvektor des Trägerpunktes $P_0 = B$ des Parallelogramms, t_1 der Parameter für den ersten Richtungsvektor \vec{a}_1 und t_2 der Parameter für den zweiten Richtungsvektor \vec{a}_2 ist.

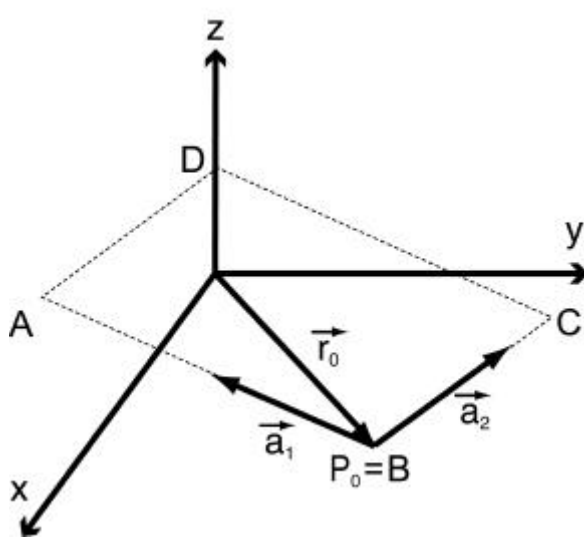


Abb. 2.3:

Das Parallelogramm ABCD besteht aus vier Strecken, die die Richtungsvektoren \vec{a}_1 und \vec{a}_2 haben.

$$AB \parallel DC, AD \parallel BC$$

$$AB = DC, AD = BC$$

Somit kann mit geeigneter Wahl von t_1 und t_2 ein beliebiger Punkt angefahren und projiziert werden.

2.1.4.1 Sichtbarkeit

Eine Besonderheit, die vor allem bei der Projektion von Flächen auftritt, ist das Problem der Sichtbarkeit.

Es kann die Situation eintreten, dass zwei oder mehrere verschiedene Punkte im Raum auf derselben Projektionsgeraden zu liegen kommen und folglich auf der Projektionsebene am gleichen Ort ihren Bildpunkt haben. Es gilt herauszufinden, welcher der beiden Punkte sichtbar ist.

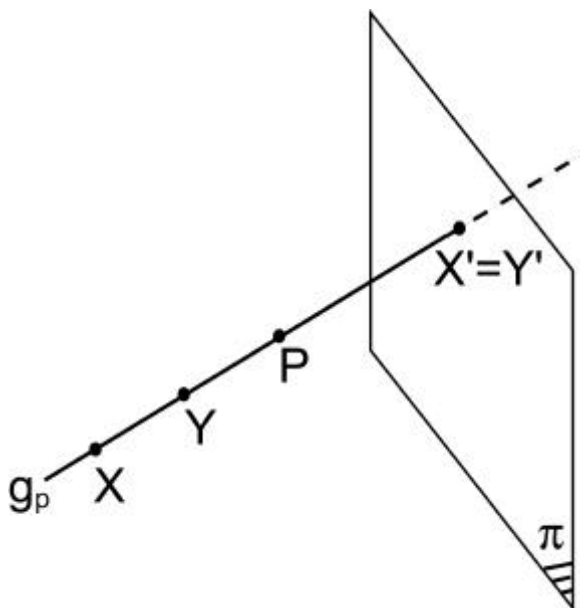


Abb. 2.4:

Die Punkte X und Y liegen auf derselben Projektionsgeraden g_p und erscheinen in der Projektionsebene als ein Punkt.

Als Kriterium eignet sich die Distanz des zu projizierenden Punktes zum Projektionszentrum: Derjenige Punkt ist sichtbar, der näher beim Projektionszentrum liegt.

3 Umsetzung

Die oben beschriebenen Methoden werden nun für ein Programm verwendet, das einen gegebenen Körper im (virtuellen) Raum auf dem Bildschirm darstellen soll. Als Körper wird ausschliesslich das Objekt „Haus“ verwendet (siehe 3.3.2 *Das Objekt*).

3.1 Programmiersprache

Das Programm ist in *PASCAL* geschrieben (Quellcode siehe *Anhang B*). Zur Übersetzung wurde der *Free Pascal Compiler* [7] und die dazugehörigen Units zur Ein- und Ausgabe (*Crt*) sowie zur Bedienung der Graphikhardware (*Graph*) verwendet.

3.2 Grundgedanke

Die Aufgabe des Programms ist es, Körper im Raum zu projizieren. Ein Körper kann als Kombination von Punkte, Geraden oder Flächen betrachtet werden. Das heisst, dass die oben erwähnten Methoden nun umgesetzt werden müssen. So erhält man eine Projektion in der Projektionsebene.

Da Bildschirm des Computers im Gegensatz zur Projektionsebene aber nicht unendlich weit ausgedehnt ist, muss ein Ausschnitt aus der Projektionsebene gewählt werden, der anschliessend auf dem Bildschirm dargestellt wird.

Der Bildschirm hat ein anderes Koordinatensystem als die Projektionsebene. Die Koordinaten aus der Ebene müssen in das Koordinatensystem des Bildschirms überführt, eine *Koordinatentransformation* durchgeführt werden.

3.3 Eingabe der zur Zentralprojektion notwendigen Elemente

Die zur Zentralprojektion notwendigen Elemente, Projektionsebene, Augpunkt und Objekt, müssen gegeben sein. Daher müssen sie eingegeben werden.

3.3.1 Projektionsebene

Zur Beschreibung der Projektionsebene wird die *allgemeine Ebenengleichung* verwendet:

$$Ax + By + Cz + D = 0 \quad (4)$$

Wobei A, B, C und D die Parameter der Projektionsbene und x, y und z die Koordinaten eines Punktes im Raum sind. Jeder Punkt, dessen Koordinaten die Gleichung erfüllen, liegt in der Projektionsebene.

Ein bessere Möglichkeit zur Bestimmung der Parameter A, B, C und D für die allgemeine Ebenengleichung der Projektionsebene als deren direkte Eingabe ist jene, die aus dem *Normalenvektor* \vec{n} und dem Ortsvektor des *Trägerpunkts* \vec{OT} der Projektionsebene diese Parameter berechnet:

$$\vec{n} = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} = \begin{pmatrix} A \\ B \\ C \end{pmatrix} \quad A = x_n, B = y_n, C = z_n \quad (5)$$

$$D = -B(Ax_{OT} + ABy_{OT} + ACE_{OT}) \quad (6)$$

Wobei $\vec{OT} = \begin{pmatrix} x_{OT} \\ y_{OT} \\ z_{OT} \end{pmatrix} \quad (7)$

Beachtet werden muss, dass der Normalenvektor der Projektionsebene auf die dem Objekt entgegengesetzte Seite zeigt, da sonst der *Basisvektor* des Koordinatensystems der Bildschirmfläche auf die falsche Seite zeigt (siehe 3.5.2 *Basisvektoren des Bildschirmsystems*).

Somit kann das Programm als Eingabe den Normalenvektor und den Trägerpunkt der Projektionsebene verlangen, um die allgemeine Ebenengleichung aufzustellen.

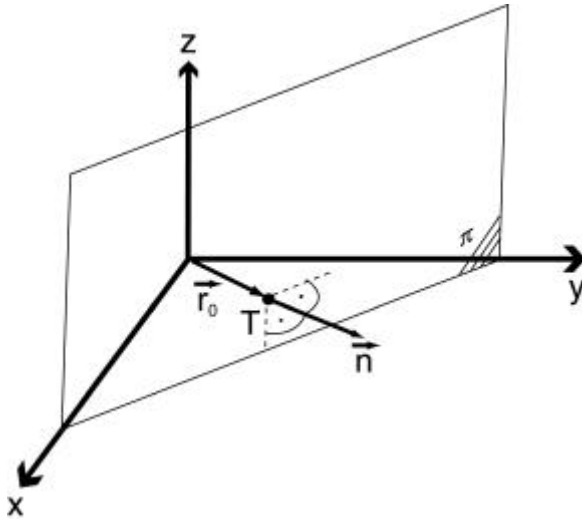


Abb. 3.1:

Mit dem Normalenvektor \vec{n} und dem Ortsvektor \vec{r}_0 seines Trägerpunktes T kann die Projektionsebene bestimmt werden.

3.3.2 Augpunkt

Neben der Projektionsebene muss auch der Augpunkt P eingegeben werden. Dies geschieht mit der Eingabe der Komponenten seines Ortsvektors \vec{OP} . Danach sind alle zur Zentralprojektion nötigen Elemente gegeben, bis auf den zu projizierenden Körper.

3.3.3 Das Objekt

Das in dieser Arbeit ausschliesslich verwendete Objekt ist ein „Haus“. Der Körper lässt sich als ein Quader mit einem darauf gesetzten Prisma beschreiben.

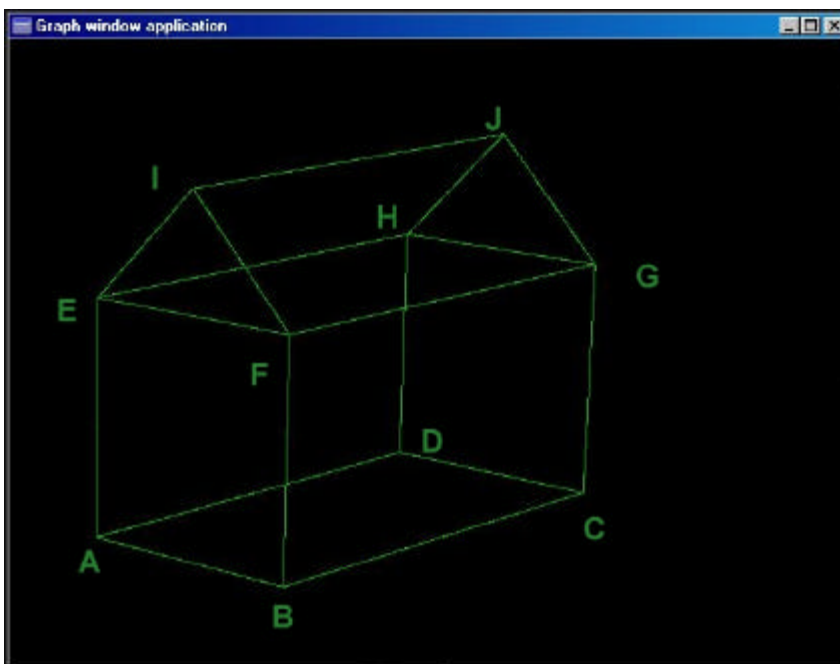


Abb. 3.2:

Das Objekt „Haus“ wird definiert durch die fünf Punkte A, B, D, E und I.

Um das Objekt „Haus“ zu definieren, müssen fünf Punkte A, B, D, E und I eingegeben werden. In dieser Arbeit wird jeweils dasselbe Objekt mit gleichbleibenden Koordinaten der Eckpunkte verwendet. Die Ortsvektoren der ersten fünf Punkte lauten:

$$\vec{OA} = \begin{pmatrix} 80 \\ 120 \\ 120 \end{pmatrix} \quad (8) \quad \vec{OB} = \begin{pmatrix} 80 \\ 200 \\ 120 \end{pmatrix} \quad (9) \quad \vec{OD} = \begin{pmatrix} 80 \\ 120 \\ 120 \end{pmatrix} \quad (10) \quad \vec{OE} = \begin{pmatrix} 80 \\ 120 \\ 200 \end{pmatrix} \quad (11) \quad \vec{OI} = \begin{pmatrix} 80 \\ 160 \\ 240 \end{pmatrix} \quad (12)$$

Die restlichen Punkte C, F, G, H und J lassen sich wie folgt berechnen:

$$\vec{OC} = \vec{OB} - \vec{BA} = \begin{pmatrix} 80 \\ 200 \\ 120 \end{pmatrix} \quad (13)$$

$$\vec{OF} = \vec{OB} - \vec{BA} = \begin{pmatrix} 80 \\ 200 \\ 200 \end{pmatrix} \quad (14)$$

$$\vec{OG} = \vec{OC} - \vec{CA} = \begin{pmatrix} 80 \\ 200 \\ 200 \end{pmatrix} \quad (15)$$

$$\vec{OH} = \vec{OD} - \vec{DA} = \begin{pmatrix} 80 \\ 120 \\ 200 \end{pmatrix} \quad (16)$$

$$\vec{OJ} = \vec{OI} - \vec{IA} = \begin{pmatrix} 80 \\ 160 \\ 240 \end{pmatrix} \quad (17)$$

Wie schon erwähnt, liegt das Projektionszentrum in dieser Arbeit immer zwischen dem Objekt und der Projektionsebene.

3.4 Umsetzung der Methoden zur Projektion von Körpern

Die im Theorieteil beschriebenen Methoden zur Projektion von Punkten, Geraden und Flächen sollen nun mit dem Programm umgesetzt werden.

3.4.1 Projektion von Punkten

Um den Bildpunkt eines Punktes zu berechnen, muss der Durchstoßpunkt der Projektionsgeraden dieses Punktes mit der Projektionsebene bestimmt werden.

Zuerst wird die Parameterdarstellung für die Projektionsgerade bestimmt:

$$g_p: \vec{OX'} = \vec{OX} + t_d \vec{XP} \quad (18)$$

$\vec{OX'}$ ist der Ortsvektor des Bildpunktes X' , \vec{OX} ist der Ortsvektor des Punktes X und t_d ist der Parameter der Geraden. Die Gerade hat als Trägerpunkt den zu projizierenden Punkt X . Der Richtungsvektor der Projektionsgeraden ist der Verbindungsvektor \vec{XP} zwischen dem zu projizierenden Punkt X und dem Projektionszentrum P :

$$\vec{XP} = \vec{OP} - \vec{OX} \quad (19)$$

Wobei
$$\vec{OX} = \begin{pmatrix} x_{OX} \\ y_{OX} \\ z_{OX} \end{pmatrix} \quad (20)$$

Um den Bildpunkt X' zu erhalten, muss nun der Durchstosspunkt der Projektionsgeraden mit der Projektionsebene berechnet werden. Dies geschieht, indem t_d berechnet wird. In die Parameterdarstellung der Projektionsgeraden eingesetzt erhält man einen Punkt, der in der Ebene liegt.

X' soll in der Projektionsebene liegen, also können die Komponenten von \vec{OX}' in die allgemeine Ebenengleichung der Projektionsebene eingesetzt werden:

$$A x_{OX'} + B y_{OX'} + C z_{OX'} + D = 0 \quad (21)$$

Wenn man die Gleichung (18) komponentenweise in Gleichung (21) einsetzt, ergibt sich folgende Gleichung:

$$A (x_{OX} + t_d E_{XP}) + B (y_{OX} + t_d E_{YP}) + C (z_{OX} + t_d E_{ZP}) + D = 0 \quad (22)$$

Diese lässt sich nach t_d auflösen:

$$A E_{d,XP} + B E_{d,YP} + C E_{d,ZP} = -B (A x_{OX} + B y_{OX} + C z_{OX} + D) \quad (23)$$

$$t_d = \frac{-(A x_{OX} + B y_{OX} + C z_{OX} + D)}{A E_{XP} + B E_{YP} + C E_{ZP}} \quad (24)$$

Nachdem t_d in (18) eingesetzt ist, ist der Durchstosspunkt X' bestimmt, d. h., der Punkt ist in die Projektionsebene projiziert.

3.4.1.1 Spiegelung

Weil der Augpunkt zwischen dem Punkt und der Projektionsebene liegt, entsteht bei der Projektion ein spiegelverkehrtes Bild (siehe *Abb. 1.1*).

Damit das Bild nicht spiegelverkehrt erscheint, wird jeder Punkt, der in die Projektionsebene projiziert wird, am Ursprung des Koordinatensystems der Bildschirmfläche gespiegelt (siehe *3.5.1 Ursprung des Bildschirmsystems*).

$$\vec{OX''} = \vec{OP'} + \vec{AX'P'} \quad (25)$$

P' ist der Ursprung des Koordinatensystems der Bildschirmfläche. $\vec{AX'P'}$ ist der Verbindungsvektor zwischen dem projizierten Punkt X' und dem Ursprung des Bildschirmsystems P'. X'' ist der am Ursprung gespiegelte Punkt X'.

3.4.2 Projektion von Geraden

Als nächstes sollen gerade Strecken projiziert werden. Damit lassen sich Körper als Gitternetzkonstruktionen darstellen (siehe *Abb. 3.2*).

Eine Strecke kann als eine Reihe von Punkten betrachtet werden. Aus dieser Reihe können nun beliebige Punkte projiziert werden. Wenn man eine Schrittweite wählt und mit dieser die Reihe von Punkten „abfährt“, wird in regelmässigen Abständen ein Punkt dargestellt, was als eine Strecke in Erscheinung tritt. (Siehe auch *2.1.3 Projektion von Geraden*.)

Die Projektion einer Strecke erfordert, dass die Ortsvektoren des Anfangs- und Endpunkts bekannt sind.

$$\vec{OA} = \begin{pmatrix} x_{OA} \\ y_{OA} \\ z_{OA} \end{pmatrix} \quad (26) \quad \vec{OB} = \begin{pmatrix} x_{OB} \\ y_{OB} \\ z_{OB} \end{pmatrix} \quad (27)$$

Um eine Strecke zu projizieren wird die Parameterdarstellung der Geraden verwendet, auf der die Strecke liegt:

$$g: \vec{r} = \vec{r}_0 + t \vec{a} \quad (28)$$

Dabei gibt \vec{r} den Ortsvektor eines Punktes auf der Geraden g an; \vec{r}_0 ist der Ortsvektor des Trägerpunktes der Geraden. \vec{a} ist deren Richtungsvektor. Mit dem Parameter t kann ein Punkt auf der Geraden "angefahren" werden. Der Trägerpunkt der Gerade ist der Anfangspunkt, der gegeben ist.

$$\vec{r}_0 = \vec{OA} \quad (29)$$

Der Richtungsvektor ist kollinear zum Verbindungsvektor zwischen Anfangs- und Endpunkt der Strecke:

$$\vec{AB} = \vec{OB} - \vec{OA} = c \vec{a} \quad (30)$$

Wobei c konstant ist. Der Betrag des Richtungsvektors ist 1:

$$\vec{a} = \frac{\vec{AB}}{|\vec{AB}|} \quad (31)$$

Da der Richtungsvektor \vec{a} der Geraden den Betrag 1 hat, kann man den Parameter t von 0 bis $\frac{|\vec{AB}|}{|\vec{a}|}$ mit der Schrittweite 1 laufen lassen und \vec{r} für diese Punkte berechnen. Diese Punkte werden anschliessend projiziert (siehe 3.4.1).

3.4.3 Projektion von Parallelogrammen

Damit Objekte aber nicht nur als Gitternetzlinien dargestellt werden, müssen Flächen projiziert werden können. Um Parallelogramme zu projizieren, wird wiederum auf die Projektion von Strecken zurückgegriffen.

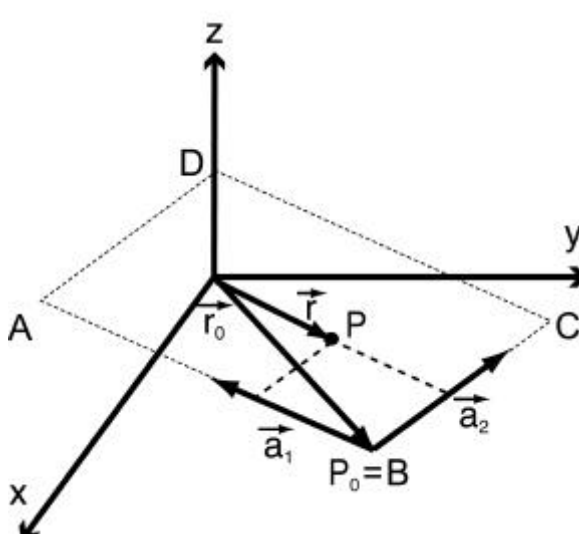


Abb. 3.3:

Im Parallelogramm ABCD kann ein Punkt P mit zwei Richtungsvektoren \vec{a}_1 und \vec{a}_2 angefahren werden.

Man kann sich nämlich zwei sich schneidende Geraden im Raum vorstellen, die so eine Ebene aufspannen. Jede dieser Geraden hat einen Richtungsvektor.

Um Punkte der Fläche eines Parallelogramms zu projizieren, wird folgende Methode angewendet: Man berechnet den Richtungsvektor \vec{a}_1 der ersten Geraden g_1 . Der Ortsvektor des Trägerpunkts der Geraden g_1 ist $\vec{r}_0 = \vec{OB}$. Die Parameterdarstellung von g_1 lautet also:

$$g_1: \vec{r}_1 = \vec{OB} + t_1 \vec{a}_1 \quad (32)$$

Der Richtungsvektor \vec{a}_1 ist kollinear zum Verbindungsvektor \vec{BA} und hat den Betrag 1:

$$\vec{a}_1 = \frac{\vec{BA}}{|\vec{BA}|} \quad (33)$$

Die Parameterdarstellung der Geraden g_2 und deren Richtungsvektor lauten entsprechend:

$$g_2: \vec{r}_2 = \vec{OB} + t_2 \vec{a}_2 \quad (34)$$

$$\vec{a}_2 = \frac{\vec{BC}}{|\vec{BC}|} \quad (35)$$

Wenn man den Parameter der ersten Geraden t_1 von 0 nach $|\vec{BA}|$ mit einer bestimmten Schrittweite laufen lässt, erhält man in regelmässigen Abständen einen Punkt auf der Strecke BA. Dieser Punkt wird nun jeweils als Anfangspunkt für eine zu Gerade g_2 parallele Gerade verwendet.

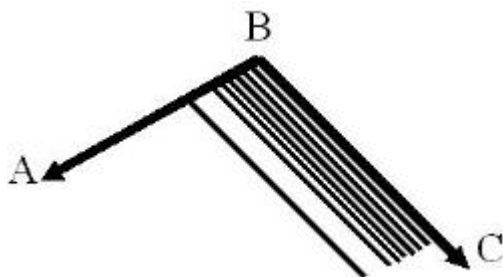


Abb. 3.4:

Parallele Strecken stellen eine Fläche dar.

Für jeden der auf der Geraden g_1 angefahrenen Punkt wird eine zu g_2 parallele Strecke gezeichnet. Somit lässt sich das Parallelogramm in folgender Parameterdarstellung formulieren:

$$p: \vec{r}_p = \vec{OB} + t_1 \vec{a}_1 + t_2 \vec{a}_2 \quad (36)$$

Wobei \vec{r}_p der Ortsvektor eines Punktes in der Fläche des Parallelogramms ist.

Während t_1 von 0 bis $|\vec{BA}|$ läuft, lässt man t_2 für jedes t_1 von 0 bis $|\vec{BC}|$ laufen. Für jede der beiden Geraden wäre eine andere Schrittweite denkbar, im Programm wird aber für beide die gleiche verwendet.

3.4.4 Sichtbarkeit von Punkten

Verschiedene Punkte im Raum können, besonders bei Flächen, den gleichen Bildpunkt besitzen. Um zu bestimmen, welcher Punkt welchen verdeckt, kann als Kriterium die Distanz des Punktes zum Projektionszentrum verwendet werden.

Die Erklärung an einem Beispiel:

$$X = (0 \ 0 \ 1) \quad Y = (1 \ 0 \ 1) \quad P = (1 \ 0 \ 0)$$

Diese drei Punkte liegen alle auf derselben Geraden, welche zugleich die Projektionsgerade von X und Y ist. Somit werden X und Y an die gleiche Stelle in der Projektionsebene projiziert. Da nun aber Y näher bei P liegt ($|XP| > |YP|$), wird nur Y gezeichnet werden.

Damit aber festgestellt werden kann, ob an der fraglichen Stelle schon ein Bildpunkt existiert und welche Distanz zum Projektionszentrum er hat, muss der Wert der Distanz gespeichert werden. Das Programm legt die Daten für den den Bildschirm betreffenden Ausschnitt der Projektionsebene im Speicher ab und prüft jeweils, ob der neue Bildpunkt sichtbar ist.

3.5 Koordinatentransformation

Jetzt können zwar Punkte, Geraden und Flächen in die Bildebene projiziert werden, doch lässt sich dies noch nicht auf dem Bildschirm darstellen. Die Projektionsebene hat ein anderes Koordinatensystem als der Bildschirm. Zwar sind beide zweidimensional, aber haben nicht den gleichen Ursprung und nicht die gleichen Basisvektoren.

Damit die Punkte auf dem Bildschirm dargestellt werden können, müssen sie in sein System überführt werden.

Da der Bildschirm nicht wie die Bildebene unendlich ausgedehnt ist, kann jeweils nur ein Ausschnitt von ihr dargestellt werden.

3.5.1 Ursprung des Bildschirmsystems

Als erster Schritt zur Bestimmung dieses Systems wird sein Ursprung festgelegt.

Als Ursprung wurde derjenige Punkt in der Ebene gewählt, den man erhält, wenn man das Lot des Projektionszentrums auf die Projektionsebene fällt.

Um den Durchstoßpunkt der Geraden des Lots mit der Projektionsebene zu bestimmen, kann man gleich vorgehen, wie bei der Projektion eines Punktes.

$$OP' = OP + t_d \cdot \vec{n} \quad (37)$$

Wobei \vec{n} der Normalenvektor der Projektionsebene ist.

t_d lässt sich wiederum wie folgt berechnen (siehe Gleichung (24)):

$$t_d = \frac{(A x_{OP} \cdot AB + y_{OP} \cdot AC + z_{OP} \cdot AD)}{(A x_n \cdot AB + y_n \cdot AC + z_n \cdot AD)} \quad (38)$$

P' ist ein Punkt im dreidimensionalen System, der in der Projektionsebene liegt; gleichzeitig ist er aber ein Punkt im zweidimensionalen System der Ebene und der Ursprung des Koordinatensystems der Bildschirmfläche.

3.5.2 Basisvektoren des Bildschirmsystems

Damit die Koordinaten eines Punktes im *Koordinatensystem der Bildschirmfläche* angegeben werden können, müssen die *Basisvektoren* des Systems bestimmt werden.

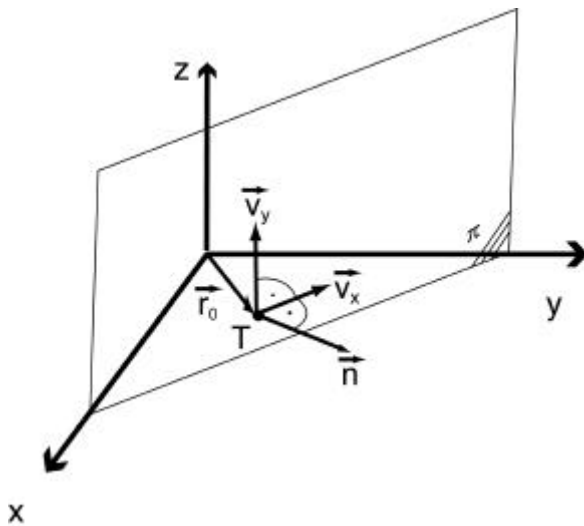


Abb. 3.5:

\vec{v}_x und \vec{v}_y sind die Basisvektoren für das zweidimensionale Koordinatensystem der Bildschirmfläche.

Die Kriterien für diese sind, dass sie

- 1.) in der Projektionsebene liegen
- 2.) senkrecht zueinander stehen
- 3.) den Betrag 1 haben

Zuerst muss einer der beiden Basisvektoren gefunden werden. Im Programm wird zuerst ein Vektor \vec{v}_x gesucht, der kollinear zum Basisvektor der x-Achse des Koordinatensystems der Bildschirmfläche ist.

$$\vec{v}_x = \begin{pmatrix} x_{vx} \\ y_{vx} \\ z_{vx} \end{pmatrix} \quad (39)$$

Die z-Komponente dieses Vektors wird 0 gesetzt. (Es wäre auch ein anderer, beliebiger Wert denkbar; das Bild auf dem Bildschirm wäre dann „geneigt“.) Somit ist \vec{v}_x zur x-y-Ebene des dreidimensionalen Koordinatensystems parallel.

$$z_{vx} = 0 \quad (40)$$

Im Allgemeinen, d.h., wenn die Projektionsebene nicht parallel zu einer der Koordinatenebenen des dreidimensionalen Systems ist, lässt sich der gesuchte Vektor \vec{v}_x folgendermassen berechnen:

$$x_{vx} = B y_n = B B \quad (41)$$

$$y_{vx} = x_n = A \quad (42)$$

Das lässt sich am einfachsten an einem Beispiel überprüfen.

Der Normalenvektor der Projektionsebene steht senkrecht auf \vec{v}_x , da dieser in der Projektionsebene liegt.

Mit $n = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$ und $v_x \cdot n = 0$ ergibt sich $v_x = \begin{pmatrix} B2 \\ 1 \\ 0 \end{pmatrix}$.

Da der Normalenvektor in die dem Objekt entgegengesetzte Richtung zeigt, zeigt der Basisvektor der x-Achse des Bildschirmsystems immer nach „rechts“.

Spezialfälle:

a.) Projektionsebene ist parallel zur y-z-Ebene

$$x_{vx} = 0 \quad (43)$$

$$y_{vx} = x_n = A \quad (44)$$

b.) Projektionsebene ist parallel zur x-z-Ebene

$$x_{vx} = B y_n = BB \quad (45)$$

$$y_{vx} = 0 \quad (46)$$

c.) Projektionsebene ist parallel zur x-y-Ebene, Objekt liegt unter der Projektionsebene

$$x_{vx} = 0 \quad (47)$$

$$y_{vx} = z_n = C \quad (48)$$

d.) Projektionsebene ist parallel zur x-y-Ebene, Objekt liegt über der Projektionsebene

$$x_{vx} = 0 \quad (49)$$

$$y_{vx} = B z_n = BC \quad (50)$$

Somit ist ein zum gesuchten Basisvektor der x-Achse des Koordinatensystems der Bildschirmfläche kollinearer Vektor gegeben. Den Basisvektor mit der Länge 1 erhält man wie folgt:

$$b_x = \frac{v_x}{|v_x|} \quad (51)$$

Den zweiten Einheitsvektor b_y kann man mit Hilfe des Vektorprodukts berechnen. Ein zu b_x kollinearer Vektor lautet:

$$v_y = n \times v_x \quad (52)$$

Der Basisvektor b_y hat den Betrag 1 und lautet daher:

$$b_y = \frac{v_y}{|v_y|} \quad (53)$$

Somit sind die beiden Basisvektoren für das 2D-System berechnet.

3.5.3 Koordinaten im Bildschirmsystem

Nachdem die Basisvektoren des Bildschirmsystems bekannt sind, kann man diese verwenden, um die Koordinaten von Punkten in diesem System zu bestimmen.

3.5.3.1 Bestimmung der Koordinaten

Gegeben sind der Ursprung des Koordinatensystems der Bildschirmfläche O' und ein in der Projektionsebene liegender Punkt X'' .

Der Verbindungsvektor zwischen O' und X'' (der dem Ortsvektor von X'' im Bildschirmsystem entspricht) lässt sich als Linearkombination aus \vec{b}_x und \vec{b}_y darstellen.

$$\vec{OX''} = c_1 \vec{b}_x + c_2 \vec{b}_y \quad (54)$$

Mit Hilfe des Skalarprodukts lässt sich der Winkel, den der Verbindungsvektor $\vec{OX''}$ und $c_1 \vec{b}_x$ einschliessen, berechnen:

$$\cos \alpha = \frac{\vec{b}_x \cdot \vec{OX''}}{|\vec{b}_x| |\vec{OX''}|} \quad (55)$$

Trigonometrisch lässt sich der Winkel α aber auch so berechnen:

$$\cos \alpha = \frac{c_1 |\vec{b}_x|}{|\vec{OX''}|} \quad (56) \quad \text{oder} \quad c_1 |\vec{b}_x| = \cos \alpha |\vec{OX''}| \quad (57)$$

Durch Einsetzen von (55) in (57)

$$c_1 |\vec{b}_x| = \frac{\vec{b}_x \cdot \vec{OX''}}{|\vec{OX''}|} |\vec{OX''}| \quad (58)$$

$$c_1 = \vec{b}_x \cdot \vec{OX''} \quad (59)$$

Die x-Koordinate im Bildschirmsystem lautet:

$$x_B = c_1 \quad (60)$$

Analog lässt sich auch die y-Koordinate berechnen, allerdings muss man darauf achten, das Vorzeichen zu wechseln, wenn man den Winkel zwischen den beiden Vektoren mit Hilfe des Skalarprodukts berechnen will. Gleichung (40) analog zu (39):

$$c_2 = -\vec{b}_y \cdot \vec{OX''} \quad (61)$$

$$y_B = c_2 \quad (62)$$

Damit sind die Koordinaten in das Koordinatensystem der Bildschirmfläche überführt worden. Der Punkt X_B ist ein Punkt im Koordinatensystem des Bildschirms:

$$X_B = (c_1 \mid c_2) = (\vec{b}_x \cdot \vec{OX''} \mid -\vec{b}_y \cdot \vec{OX''}) \quad (63)$$

3.5.4 Bildschirmnullpunkt des PCs

Weil der Bildschirmnullpunkt des PCs links oben statt links unten liegt, muss der Wert der y-Koordinate des Punktes angepasst werden. Somit müssen die Koordinaten des Punktes $(x_B \mid y_B)$ lauten:

$$(x_B \mid A_y - y_B)$$

Wobei A_y der Wert der vertikalen Auflösung des Bildschirms ist. D. h. im Beispiel mit der Auflösung 640 x 480 Punkten für den Punkt (200 | 200):

$$(200 \mid 480 - 200) = (200 \mid 280)$$

3.5.5 Skalierung des Bildausschnitts

Neben dem Fall, dass die Projektion neben dem Ausschnitt liegt und deshalb das Bildschirmsystem verschoben werden muss (siehe 3.5.6 *Verschieben des Bildschirmsystems*), kann es auch vorkommen, dass der Ausschnitt zu klein oder zu gross ist.

Um die Grösse des Ausschnitts zu verändern, kann dieser *skaliert* werden. D. h., die Koordinaten der Punkte im Bildschirmsystem werden mit einem Faktor multipliziert.

Der Faktor berechnet sich wie folgt:

$$q_x = \frac{A_x}{S_x} \quad (64) \quad \text{bzw.} \quad q_y = \frac{A_y}{S_y} \quad (65)$$

Wobei q_x und q_y die Skalierungsfaktoren für die x- bzw. y-Koordinaten sind. A_x und A_y sind die Werte für die horizontale bzw. vertikale Auflösung des Bildschirms. S_x und S_y sind die Werte für die gewünschte horizontale bzw. vertikale Grösse des Ausschnitts.

Die Koordinaten des Punkts (x_B | y_B) im skalierten Bildausschnitt lauten also:

$$\{x_B q_x \mid y_B q_y\}$$

Man könnte die Grösse des Ausschnitts auch verändern, indem man die Werte für den Augpunkt oder die Projektionsebene ändert. Möchte man aber diese nicht verändern, ist die Skalierung des Bildschirmsystems eine Lösung.

3.5.6 Verschieben des Bildschirmsystems

Die Projektion eines Objekts liegt nicht unbedingt im Ausschnitt der Projektionsebene, der auf dem Bildschirm sichtbar wird, wenn man das auf die Projektionsebene gefällte Lot des Projektionszentrums als Ursprung des Bildschirmsystems verwendet.

Damit das Bildschirmsystem (und damit der auf dem Bildschirm dargestellte Ausschnitt der Projektionsebene) verschoben werden kann, kann zu den Koordinaten der Punkte im Bildschirmsystem noch ein Wert addiert oder subtrahiert werden. Dieser ist für alle Punkte gleich.

$$X_B = \{x_B o_x \mid y_B o_y\}$$

Wobei o_x und o_y die Werte zur Verschiebung des Bildschirmsystems in x- bzw. y-Richtung sind.

Bemerkung: Die eingegebenen Werte für die Verschiebung werden ebenfalls mit den Skalierungsfaktoren multipliziert.

Wie beim Skalieren des Bildschirmsystems kann auch die Verschiebung des Bildausschnitts mit dem Verändern der Werte für Augpunkt und Projektionsebene geschehen.

3.6 Schrittweite bei der Projektion von Geraden und Flächen

Bisher wurde als Schrittweite bei der Projektion von Geraden oder Flächen jeweils die Schrittweite 1 angenommen. Dies kann aber dazu führen, dass die Projektionen auf dem Bildschirm nicht „dicht“ erscheinen. D. h., es gibt Abstände zwischen den projizierten Punkten einer Geraden oder Fläche, oder die Punkte erscheinen im gleichen Bildpunkt. Die Schrittweite kann deshalb angepasst werden.

Die Schrittweite, welche Geraden als durchgezogene Linien und Flächen als ausgefüllte Vierecke erscheinen lassen soll, berechnet sich wie folgt:

$$s = \frac{\overline{A'B'}}{\overline{AB}} \quad (66)$$

Wobei s die Schrittweite ist. AB ist eine Strecke im Raum, $A'B'$ deren Projektion. Die Koordinaten von A' bzw. B' sind bezüglich des Koordinatensystems der Bildschirmfläche und sind gemäss der

Skalierung des Bildausschnitts mit einem Faktor multipliziert, sodass dies für die Schrittweite berücksichtigt ist.

Dennoch erscheinen Lücken, wenn mit dieser Schrittweite projiziert wird. Dies ist darauf zurückzuführen, dass die Koordinaten gerundet werden, bevor die Punkte auf den Bildschirm gezeichnet werden.

4 Resultate

4.1 Untersuchte Parameter

Die Parameter, die im Folgenden verändert werden und deren Einfluss anhand von Bildern anschaulich dargestellt werden soll, sind die Werte für die Parameter der Projektionsebene und das Projektionszentrum.

Die Skalierungsfaktoren und die Werte für die Verschiebung des Ausschnitts werden benutzt, damit das Bild auf dem Bildschirm sichtbar wird.

4.2 Bildserien

Die Resultate der Untersuchungen sind verschiedene Aufnahmen von Projektionen des Objektes „Haus“. Es sind vier Bildserien:

- 1.) Nur die Höhe des Projektionszentrums variiert, die Blickrichtung (\hat{n} = Normalenvektor der Projektionsebene) ist waagrecht, während sich der Blickwinkel auf das Haus ändert;
- 2.) die Entfernung des Projektionszentrums und der Projektionsebene vom Objekt nimmt zu, die Höhe des Projektionszentrums bleibt gleich, die Blickrichtung zeigt zum Ursprung des 3D-Systems;
- 3.) die Entfernung zum Objekt und die Höhe des Projektionszentrums bleiben gleich, das Projektionszentrum bewegt sich auf einer Kreisbahn in 20° -Schritten;
- 4.) die Schrittweite beim Projizieren von Flächen nimmt zu.

Anhand der folgenden vier Bildserien soll der Einfluss und die Wirkung dieser Parameter diskutiert werden.

4.2.1 Variation der Höhe des Projektionszentrums

Da nur die Höhe des Projektionszentrums variiert wird und die Blickrichtung dieselbe bleibt (die Projektionsebene steht in allen Fällen senkrecht zur x-y-Ebene des dreidimensionalen Koordinatensystems), erscheint das Bild umso verzerrter, je extremer der Betrag der Höhe wird.

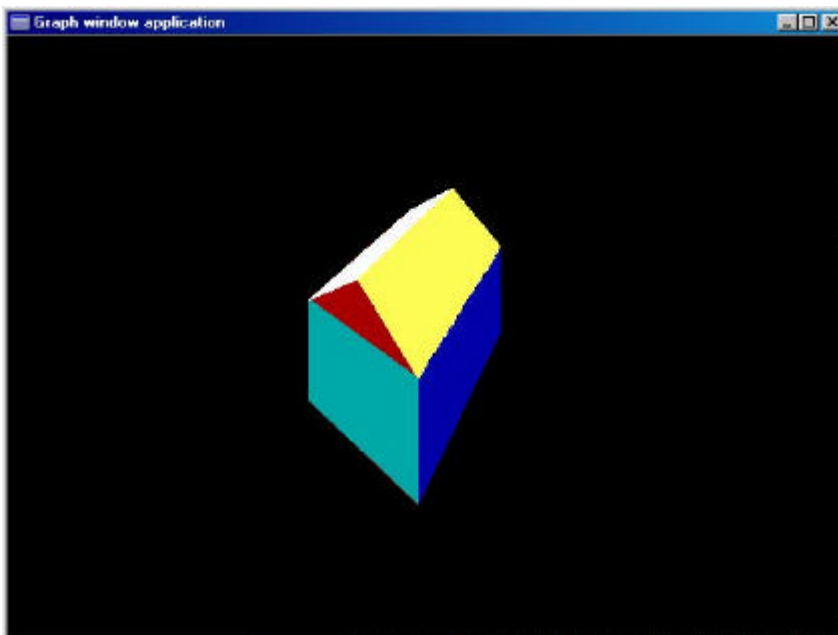


Abb. 4.1.1:

Höhe: 450
Augpunkt = (300|300|450)
Trägerpunkt der Proj.-Ebene
= (400|400|0)
Normalenvektor der Proj.-
Ebene = (1,1,0)

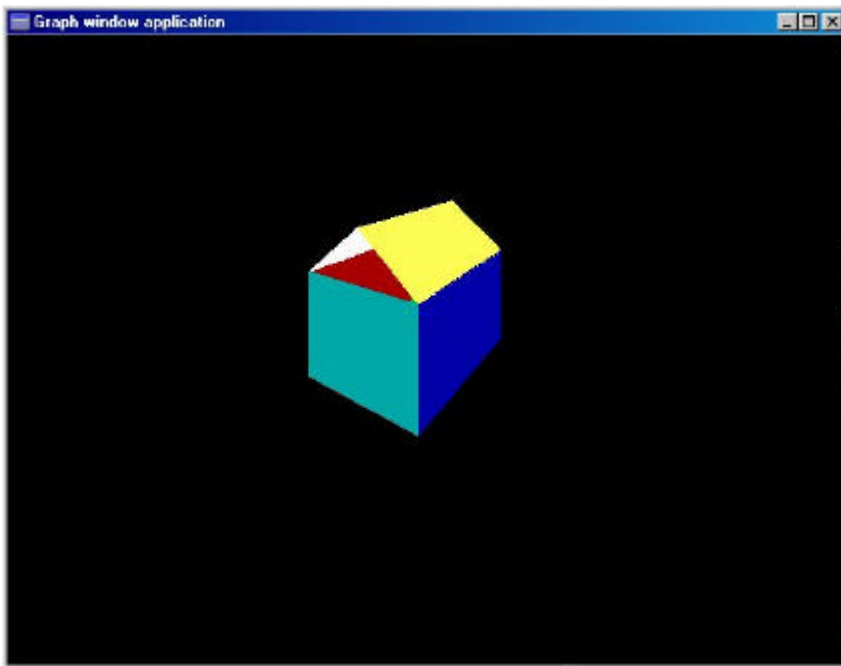


Abb. 4.1.2:

Höhe: 300
Augpunkt = (300|300|300)
Trägerpunkt der Proj.-Ebene
= (400|400|0)
Normalenvektor der Proj.-
Ebene = (1,1,0)

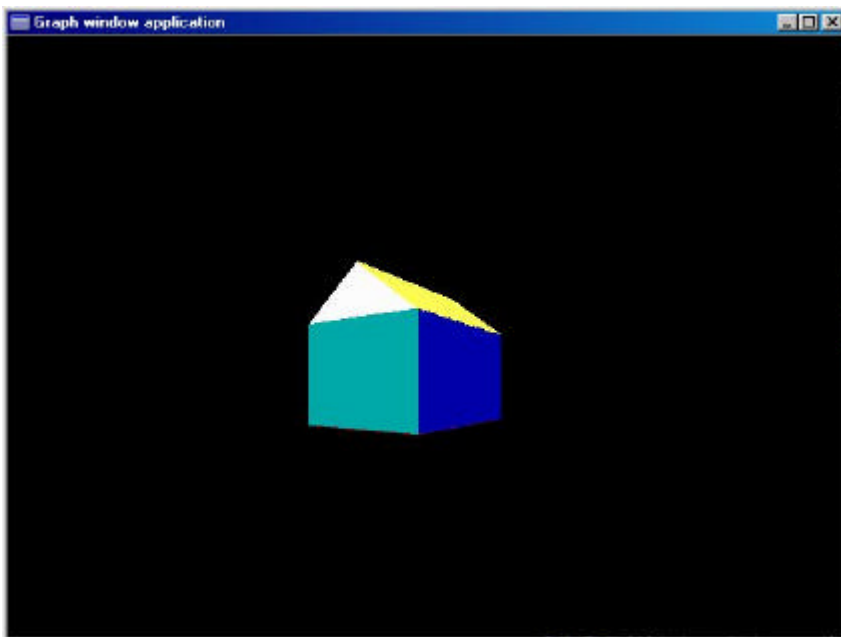


Abb. 4.1.3:

Höhe: 150
Augpunkt = (300|300|150)
Trägerpunkt der Proj.-Ebene
= (400|400|0)
Normalenvektor der Proj.-
Ebene = (1,1,0)

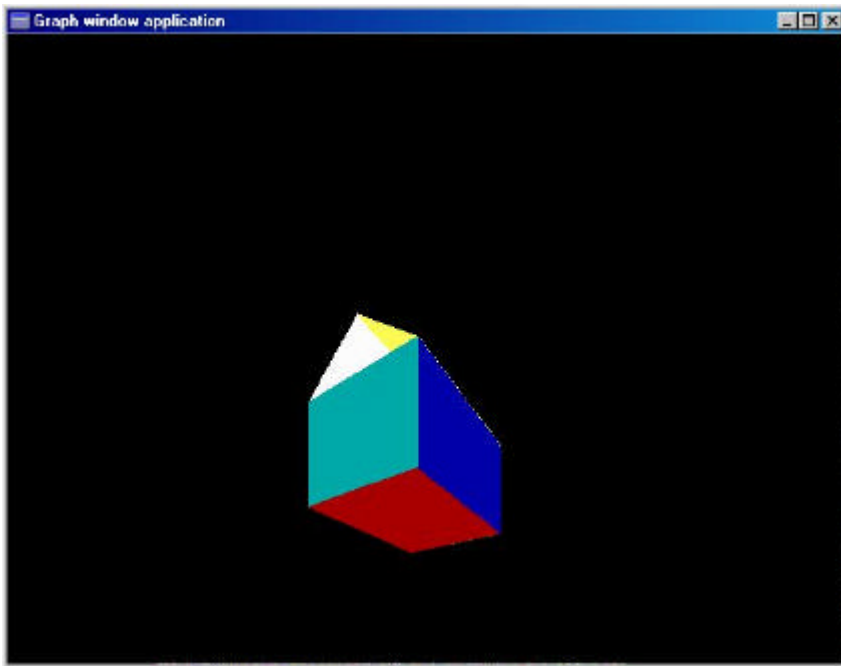


Abb. 4.1.4:
Höhe: 0
Augpunkt = (300|300|0)
Trägerpunkt der Proj.-Ebene
= (400|400|0)
Normalenvektor der Proj.-
Ebene = (1,1,0)

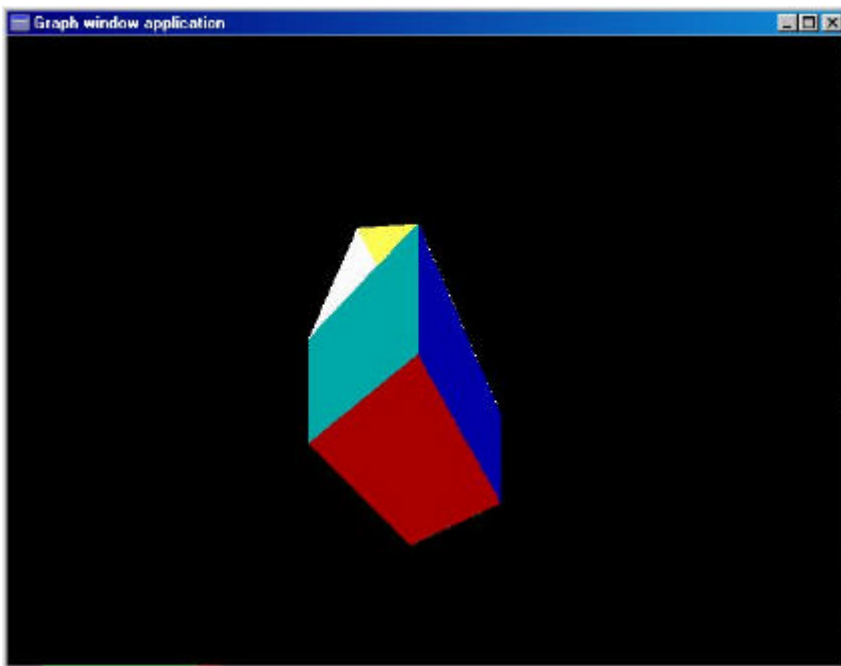


Abb. 4.1.5:
Höhe: -150
Augpunkt = (300|300|-150)
Trägerpunkt der Proj.-Ebene
= (400|400|0)
Normalenvektor der Proj.-
Ebene = (1,1,0)

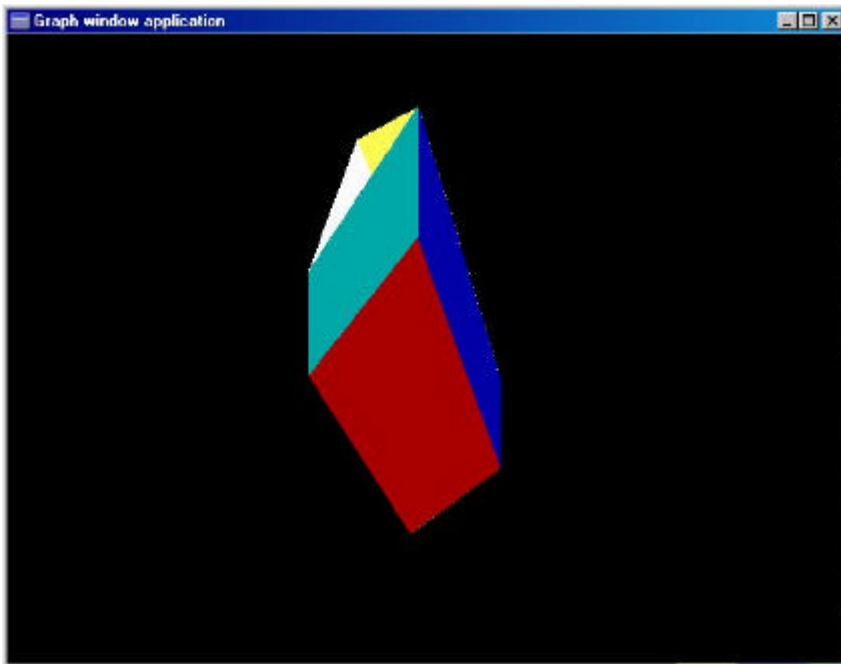


Abb. 4.1.6:

Höhe: -300
 Augpunkt = (300|300|-300)
 Trägerpunkt der Proj.-Ebene = (400|400|0)
 Normalenvektor der Proj.-Ebene = (1,1,0)

4.2.2 Zunehmende Entfernung vom Objekt

Diese Serie erzeugt weniger verzerrte Bilder, weil der Blick gezielt zum Objekt gerichtet ist.

Bei kleiner Entfernung erscheint das Haus noch sehr perspektivisch, aber mit der Entfernung nimmt dieser Eindruck ab, bis die Seitenwände schliesslich so klein erscheinen, dass man keine Perspektive mehr erkennen kann.

Während die Entfernung zunimmt, bleibt die Höhe des Projektionszentrums konstant, was dazu führt, dass der Blickwinkel immer flacher wird. Dies äussert sich darin, dass man am Schluss nicht mehr in das Haus hineinsehen kann und nur noch die vordere Wand sieht. Ausserdem sieht man nicht mehr auf das Dach, sondern blickt geradewegs auf die Dachkanten.

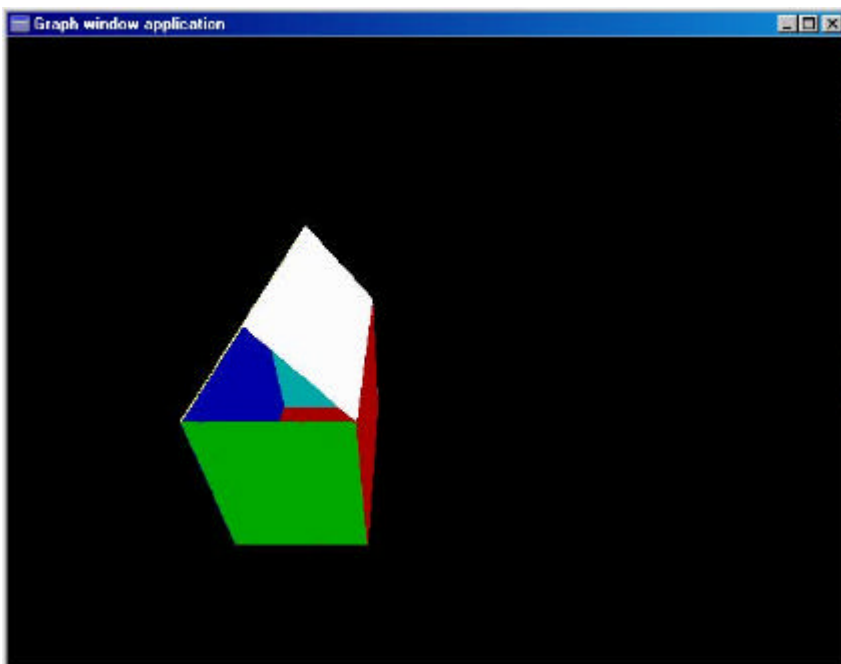


Abb. 4.2.1:

Entfernung: 250
 Augpunkt = (-250|100|300)
 Trägerpunkt der Proj.-Ebene = (-500|0|600)
 Normalenvektor der Proj.-Ebene = (-500,0,600)

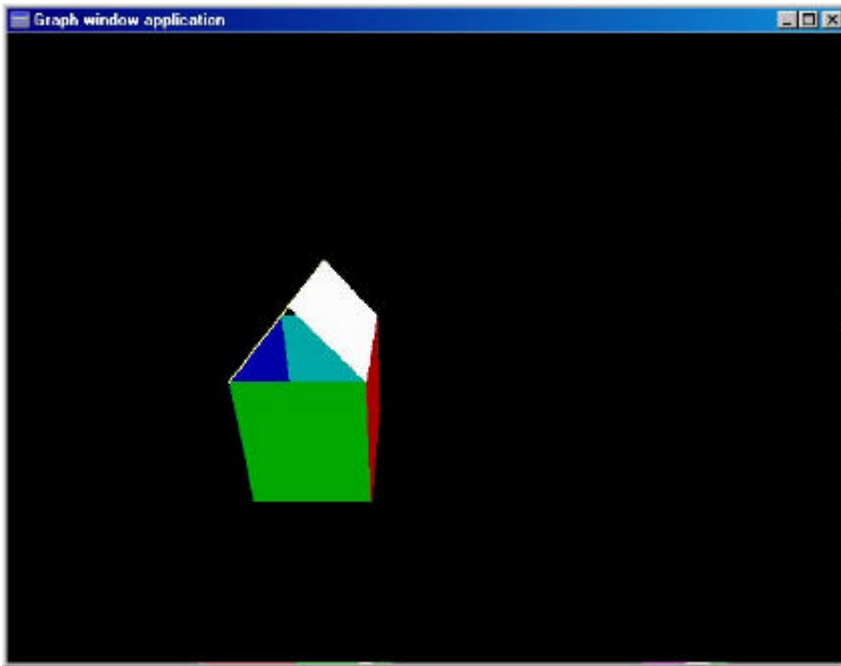


Abb. 4.2.2:

Entfernung: 375
 Augpunkt = (-375|100|300)
 Trägerpunkt der Proj.-Ebene
 = (-750|0|600)
 Normalenvektor der Proj.-
 Ebene = (-500,0,600)

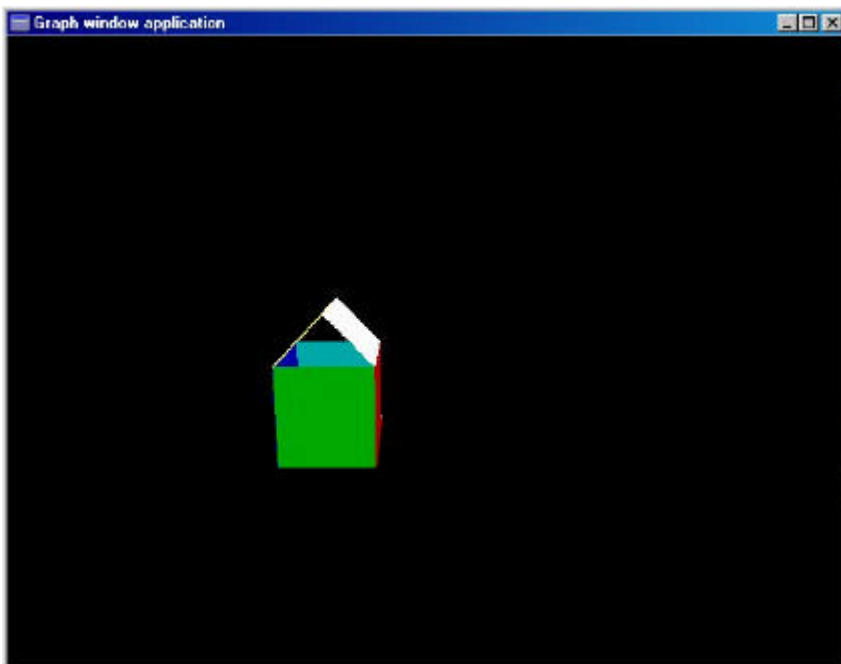


Abb. 4.2.3:

Entfernung: 750
 Augpunkt = (-750|100|300)
 Trägerpunkt der Proj.-Ebene
 = (-1500|0|600)
 Normalenvektor der Proj.-
 Ebene = (-1500,0,600)

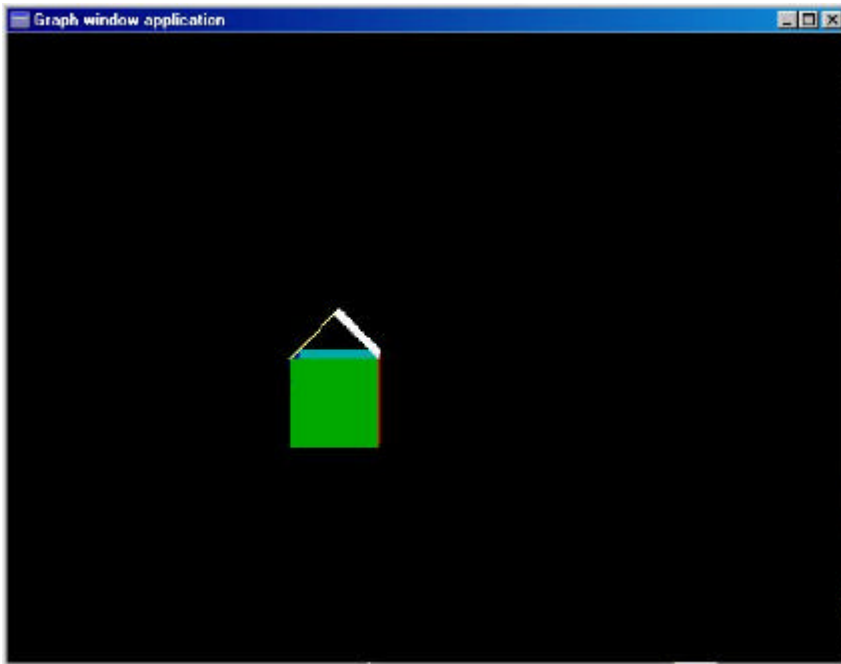


Abb. 4.2.4:

Entfernung: 1000
 Augpunkt = (-1000|100|300)
 Trägerpunkt der Proj.-Ebene
 = (-2000|0|600)
 Normalenvektor der Proj.-
 Ebene = (-500,0,600)

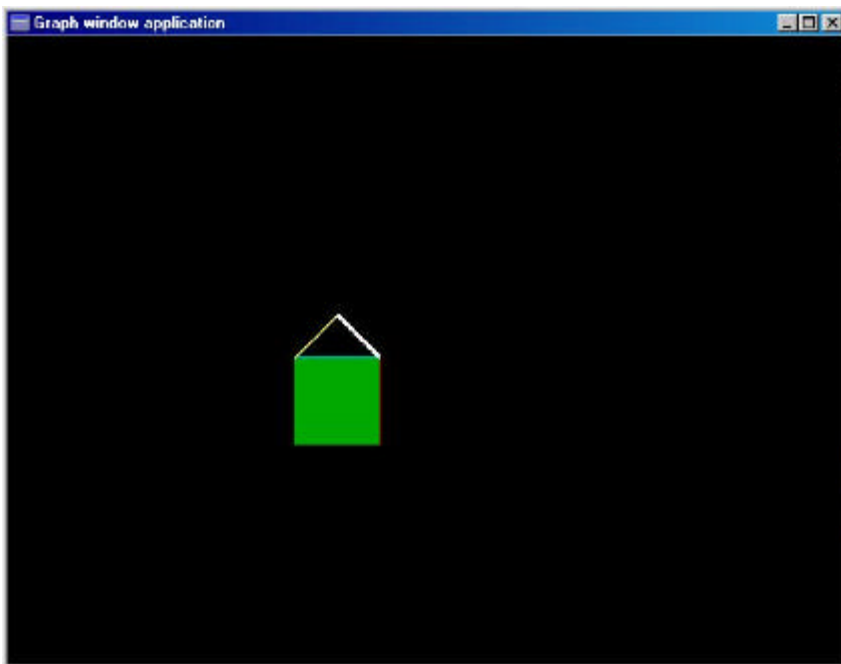


Abb. 4.2.5:

Entfernung: 5000
 Augpunkt = (-5000|100|300)
 Trägerpunkt der Proj.-Ebene
 = (-10000|0|600)
 Normalenvektor der Proj.-
 Ebene=(-10000,0,600)

4.2.3 Kreisbewegung

Die folgende Serie zeigt eine Kreisbewegung, wie man sie sich vorstellen kann, wenn der Augpunkt auf konstanter Höhe um den Ursprung kreist.

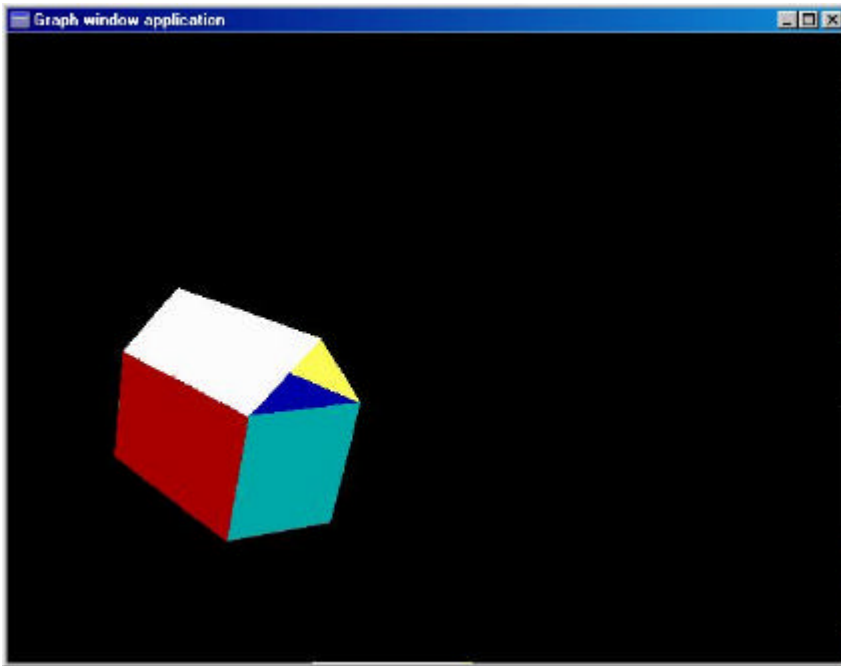


Abb. 4.3.1:

Abweichung von der
x-Achse: -20°
 Augpunkt = (469|-171|300)
 Trägerpunkt der Proj.-Ebene
 = (939|-342|600)
 Normalenvektor der Proj.-
 Ebene = (939|-342|600)

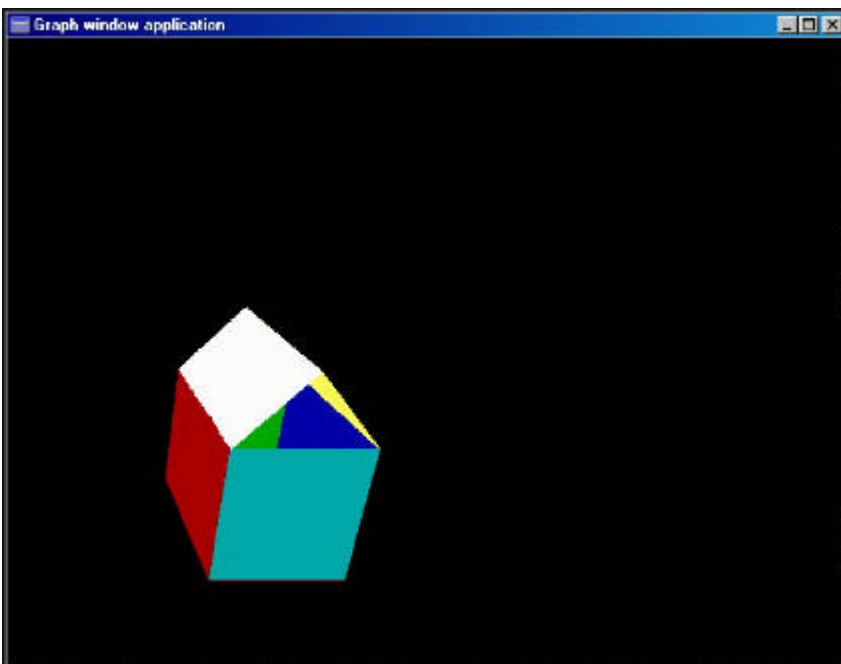


Abb. 4.3.2:

Abweichung von der
x-Achse: 0°
 Augpunkt = (500|0|300)
 Trägerpunkt der Proj.-Ebene
 = (1000|0|600)
 Normalenvektor der Proj.-
 Ebene = (1000|0|600)

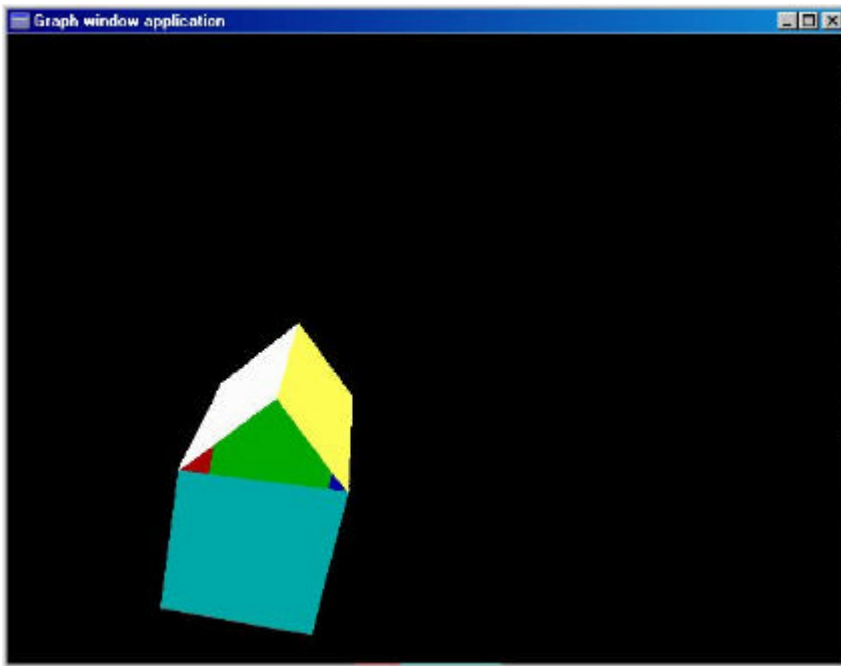


Abb. 4.3.3:

Abweichung von der
x-Achse: 20°
 Augpunkt = (469|171|300)
 Trägerpunkt der Proj.-Ebene
 = (939|342|600)
 Normalenvektor der Proj.-
 Ebene=(939|342|600)

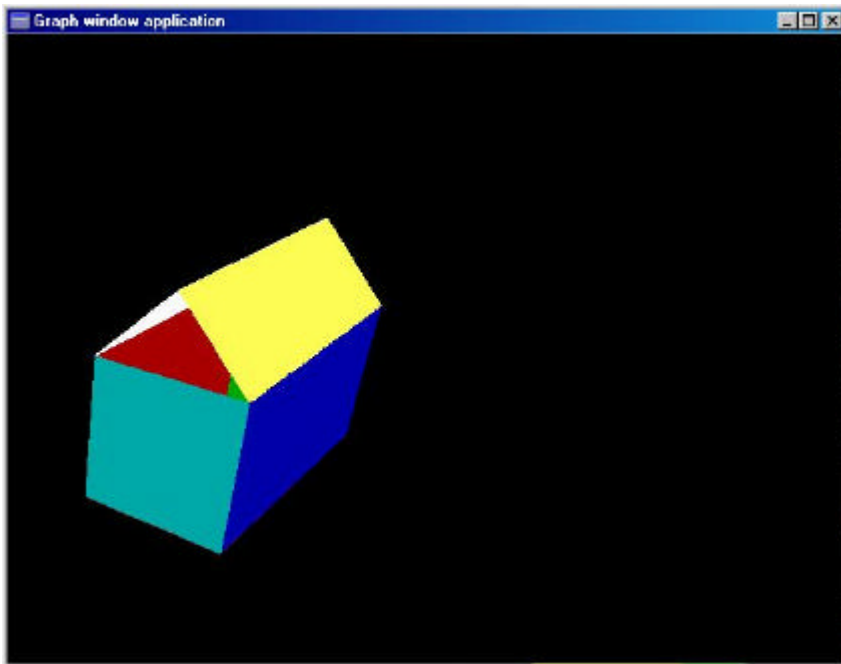


Abb. 4.3.4:

Abweichung von der
x-Achse: 40°
 Augpunkt = (383|321|300)
 Trägerpunkt der Proj.-Ebene
 = (766|642|600)
 Normalenvektor der Proj.-
 Ebene = (766|642|600)

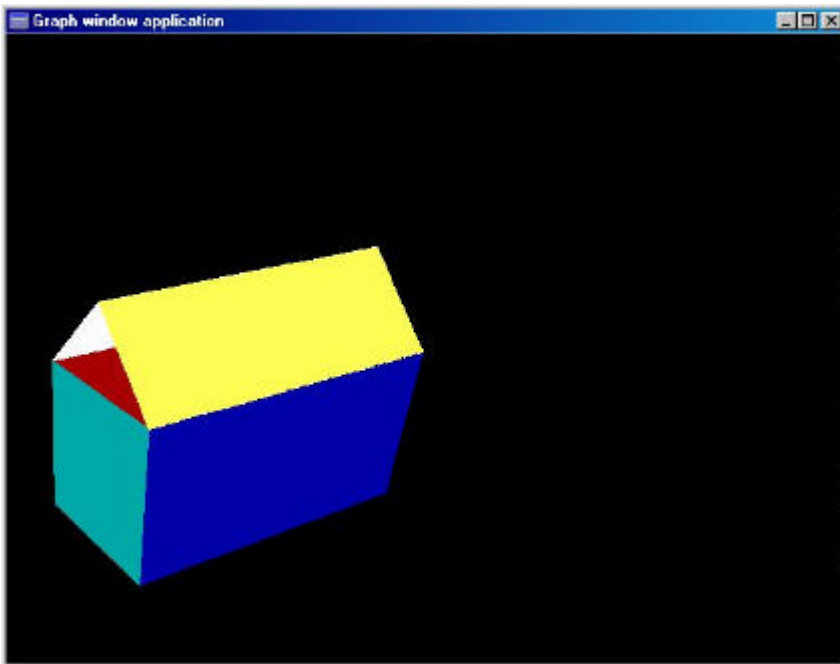


Abb. 4.3.5:
*Abweichung von der
 x-Achse: 60°*
 Augpunkt = (250|433|300)
 Trägerpunkt der Proj.-Ebene
 = (500|866|600)
 Normalenvektor der Proj.-
 Ebene=(500|866|600)

4.2.4 Transparenzeffekt

Die letzte Serie zeigt Bilder, bei denen die Schrittweite für das Projizieren von Flächen vergrößert wird, indem die bereits optimierte Schrittweite mit einem weiteren Faktor multipliziert wird. Dies erscheint als eine Art Transparenzeffekt. Weil die Dichte der projizierten Punkte nicht mehr so gross ist, steigt die Wahrscheinlichkeit, dass ein weiter entfernter Punkt trotzdem sichtbar wird.

Es fällt auf, dass bei dem Bild, das eine Projektion mit dem Schrittweitefaktor 1 darstellt trotz Optimierung eine Transparenz auftritt. Das ist damit zu erklären, dass die zu zeichnenden Punkte gerundet werden.

Je näher ein Punkt beim Projektionszentrum liegt, desto mehr kommt die Schrittweite zum Tragen.

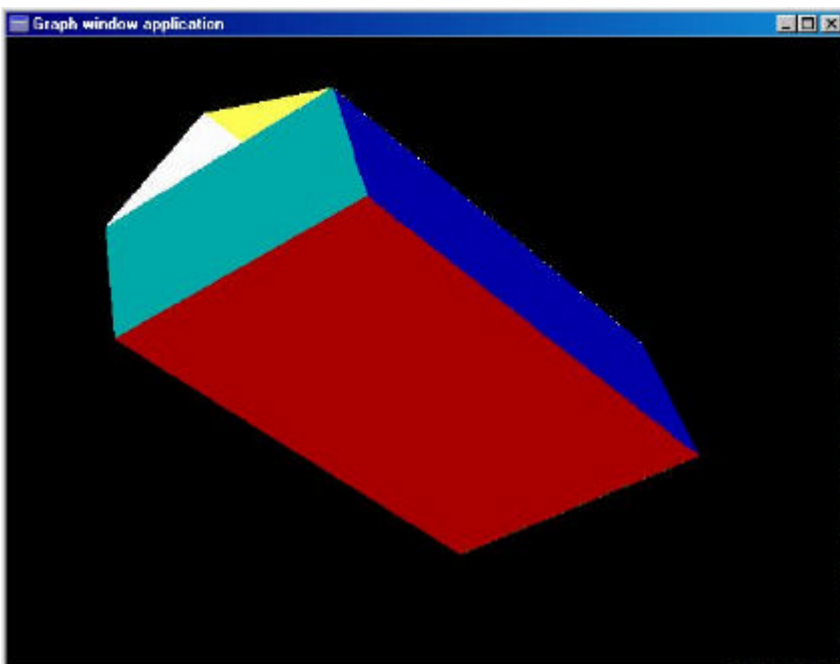


Abb. 4.4.1:
Schrittweite-Faktor: 0.5

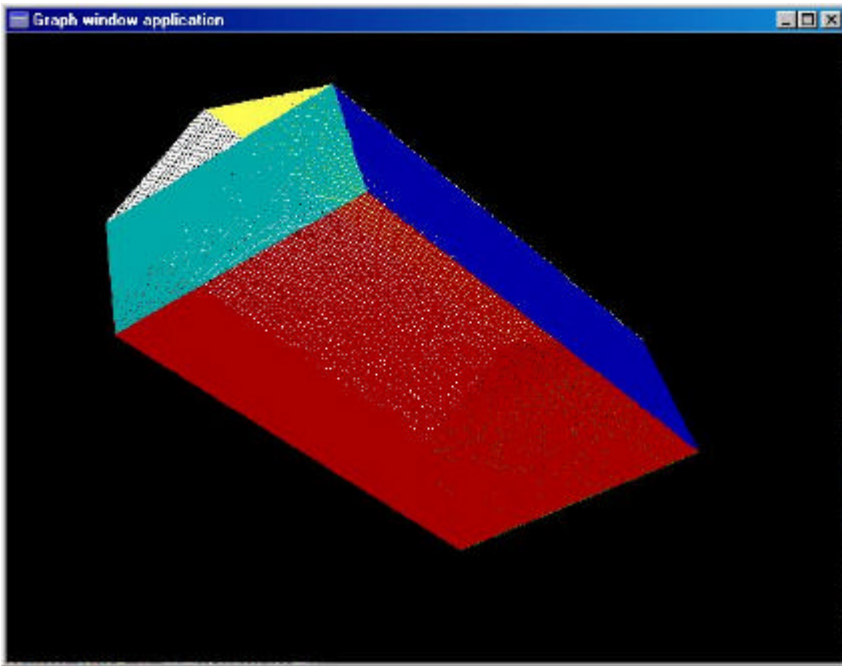


Abb. 4.4.2:
Schrittweite-Faktor: 1

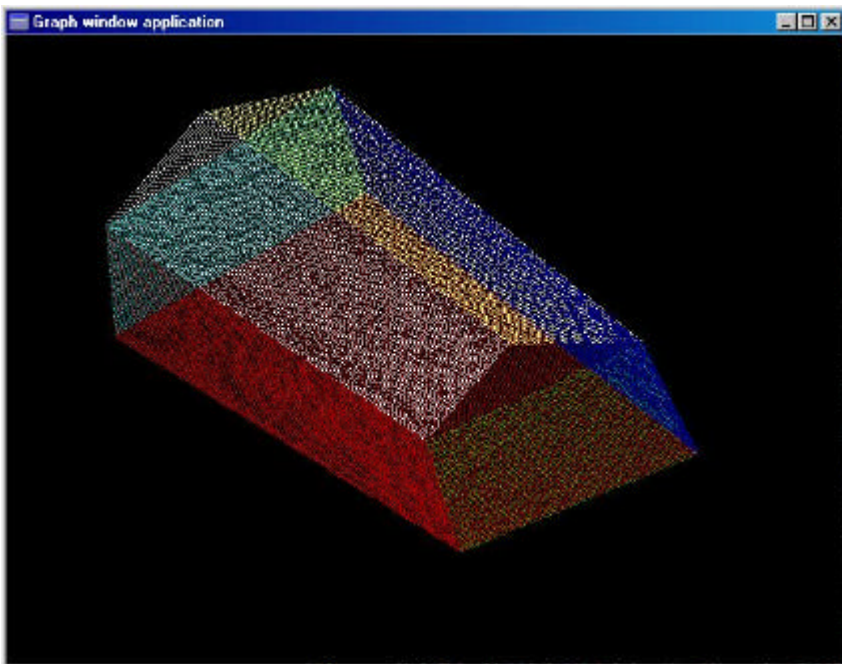


Abb. 4.4.3:
Schrittweite-Faktor: 2

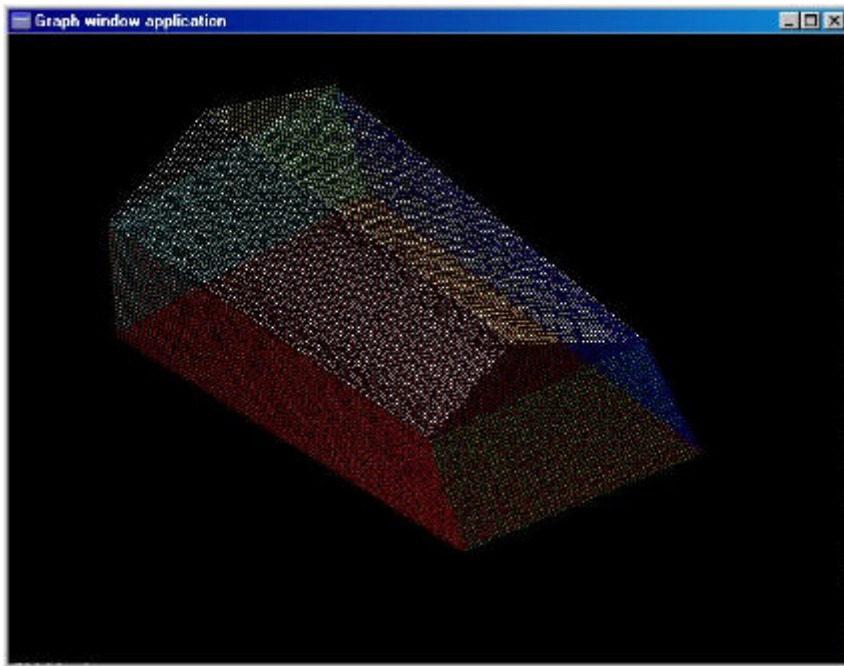


Abb. 4.4.4:
Schrittweite-Faktor: 3

5 Schluss

5.1 Zusammenfassung

Das Programm, das in dieser Arbeit beschrieben wird, benutzt die Zentralprojektion als Mittel zur Darstellung von räumlichen Körpern in einer Ebene. Dieses Prinzip ist auch beim menschlichen Auge und anderen Anwendungen erkennbar.

Zurückgegriffen wird zur Umsetzung auf die Methoden der Vektorgeometrie im Raum. Die zur Zentralprojektion notwendigen Elemente sind ein Objekt, ein Augpunkt und eine Projektionsebene. Das Objekt ist in dieser Arbeit ein „Haus“.

Projiziert werden Punkte, gerade Strecken und Flächen in der Form von Parallelogrammen, wobei die beiden letzteren als Reihe von Punkten bzw. als Punktgitter betrachtet werden. Die Methode, die für die Projektion von Punkten verwendet wird, kann somit für Geraden und Flächen wiederverwendet werden. Folglich ist es möglich, Körper als Gitternetzkörper (Kombination von Geraden) wie auch als Kombination von Flächen darzustellen.

Besonders bei der Projektion von Flächen stellt sich das Problem der Sichtbarkeit: Verschiedene Punkte im Raum können denselben Bildpunkt haben. Das Kriterium für die Lösung des Problems ist der Abstand des zu projizierenden Punktes vom Augpunkt. Für jeden projizierten Punkt wird der Abstand gespeichert, um bei einem identischen Bildpunkt einen Vergleich zu ermöglichen.

Der Abstand der Punkte bei Geraden und Flächen kann verändert werden. Durch die entstehenden Lücken ist ein Transparenzeffekt erzielbar.

Um die Projektion auf dem Bildschirm darzustellen, muss eine Koordinatentransformation vom Koordinatensystem der Projektionsebene in das des Computerbildschirms durchgeführt werden. Dies erfordert die Berechnung der Basisvektoren für das Bildschirmsystem. Zudem kann der Ausschnitt der Projektionsebene, der auf dem Bildschirm dargestellt werden soll, verschoben und skaliert werden.

Die Werte für die Koordinaten des Augpunkts sowie für die Beschreibung der Projektionsebene lassen die Perspektive bestimmen, in welcher der Körper dargestellt wird.

5.2 Ausblick: Alternative Projektionsverfahren

Eine andere Möglichkeit der Projektion ist, den Projektionsvorgang gewissermassen umzukehren. Indem man für jeden Punkt auf dem Bildschirm, der ja ein Ausschnitt der Projektionsebene ist, eine Gerade durch den Augpunkt legt und den Schnittpunkt mit einem Objekt im Raum feststellt, kann man ein Objekt gewissermassen abtasten.

Diese Methode hat durch ihre Anpassung an die Bildschirmauflösung den Vorteil, dass keine Punkte projiziert werden, die schlussendlich sowieso nicht sichtbar sind. Ausserdem sind bei dem sog. *Raytracing* Effekte wie Halbtransparenz und spiegelnde Flächen möglich.

5.3 Literatur- und Quellenverzeichnis

- [1] „Die Camera obscura (Lochkamera)“, <<http://home.mayn.de/ebaumann/fotobuch/node8.html>>
- [2] „Die Hainichener Camera obscura“, <http://www.hainichen.de/stadt/geschichte/camera_i.htm>
- [3] Schmidt, Rudolf: „Perspektive Schritt für Schritt“, Augustus Verlag, Augsburg, 1995
- [4] Stärk, Roland: „Darstellende Geometrie“, Ferdinand Schöningh, Paderborn, 1978
- [5] Rehbock, Fritz: „Geometrische Perspektive“, Springer-Verlag, Berlin, 1980
- [6] Bachmann, Heinz: „Vektorgeometrie“, Sabe, Zürich, 1981
- [7] „Free Pascal – Home Page“, <<http://www.freepascal.org/>>

Anhang A: Vektorgeometrie im Raum

Umgebung

Die Umgebung für die folgenden Berechnungen ist ein dreidimensionales Koordinatensystem im Raum.

Vektor im Raum

Ein Vektor wird durch drei Komponenten a_x , a_y und a_z definiert:

$$\vec{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix}$$

Betrag eines Vektors

Die Länge des Vektors (Betrag) wird folgendermassen berechnet:

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Basisvektoren

Durch Basisvektoren wird ein Koordinatensystem definiert. Basisvektoren sind üblicherweise auf den Betrag 1 normiert. Durch eine Kombination von Einheitsvektoren können die Komponenten eines Vektors angegeben werden.

In einem dreidimensionalen kartesischen System lauten die Basisvektoren üblicherweise

$$\vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \vec{e}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \vec{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Ortsvektoren

Punkte im Raum werden häufig mit Ortsvektoren beschrieben. Der Ortsvektor eines Punktes ist derjenige Vektor, der vom Ursprung zum definierten Punkt zeigt.

$$\text{Ortsvektor } \vec{OP} \text{ für den Punkt } P: \vec{OP} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Punkte im Raum

Punkte werden durch drei Koordinaten x , y und z definiert. Ein Punkt im Raum wird auch folgendermassen notiert:

$$P = (x, y, z)$$

Die Koordinaten x , y und z entsprechen den Komponenten x , y und z des Ortsvektors \vec{OP} .

Basisvektoren

Durch Basisvektoren wird ein Koordinatensystem definiert. Basisvektoren sind üblicherweise auf den Betrag 1 normiert. Durch eine Kombination von Einheitsvektoren können die Komponenten eines Vektors angegeben werden.

In einem dreidimensionalen kartesischen System lauten die Basisvektoren üblicherweise

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, e_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Addition und Subtraktion von Vektoren

Vektoren können komponentenweise addiert oder subtrahiert werden:

$$aA + bB = \begin{pmatrix} aA_x + bB_x \\ aA_y + bB_y \\ aA_z + bB_z \end{pmatrix}$$

Multiplikation und Division von Vektoren mit Skalaren

Ebenso lassen sich Vektoren komponentenweise mit einem Skalar multiplizieren bzw. dividieren.

$$aA = \begin{pmatrix} aA_x \\ aA_y \\ aA_z \end{pmatrix}$$

Das Resultat ist ein Vektor, der zu A kollinear ist.

Verbindungsvektor zwischen zwei Punkten

Der Verbindungsvektor zwischen zwei Punkten kann durch Subtraktion der Ortsvektoren der Punkte berechnet werden. Dabei gilt:

$$\vec{AB} = \vec{OB} - \vec{OA}$$

Skalarprodukt

Eine weitere Operation ist das Skalarprodukt von zwei Vektoren:

$$aA \cdot bB = aA_x bB_x + aA_y bB_y + aA_z bB_z$$

$$aA \cdot bB = |aA| |bB| \cos \gamma$$

Wobei γ denjenigen Winkel angibt, den die beiden Vektoren A und B einschließen.

Vektorprodukt

Das Vektorprodukt (oder Kreuzprodukt) lässt aus zwei gegebenen Vektoren einen dritten berechnen, der mit beiden der gegebenen Vektoren einen rechten Winkel einschliesst. Der dritte Vektor lässt sich mit der Determinanten berechnen:

$$c = a \times b = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$$

Geraden im Raum

Eine Gerade im Raum wird durch den Ortsvektor eines Trägerpunkts r_0 im Raum, einem Richtungsvektor A und dem dazugehörigen Parameter t beschrieben. Es ergibt sich die Parameterdarstellung einer Geraden im Raum:

$$r = r_0 + tA$$

Allgemeine Ebenengleichung

Eine Ebene im Raum lässt sich mit Hilfe der Allgemeinen Ebenengleichung beschreiben. A, B, C und D sind Parameter der Ebene. x, y und z sind die Koordinaten eines Punktes. Alle Punkte, die diese Gleichung erfüllen, liegen in der Ebene.

$$\text{Ebene } E: Ax + By + Cz + D = 0$$

Normalenvektor einer Ebene

Die Parameter A, B und C der Ebene entsprechen den Komponenten des Normalenvektors auf die Ebene:

$$\vec{n} = \begin{pmatrix} A \\ B \\ C \end{pmatrix}$$

Anhang B: Programmcode

Flussdiagramm

Das Diagramm in *Abb. B.1* soll die Struktur des Programmcodes und der Ablauf der Ausführung erläutern. Die Benennung der Funktionen und Prozeduren entsprechen denjenigen im Code.

Compiler

Der Programmcode wurde mit Free Pascal [7] Compiler v1.00 übersetzt. Die Units *Crt* und *Graph* gehören ebenfalls zum Free Pascal Compiler.

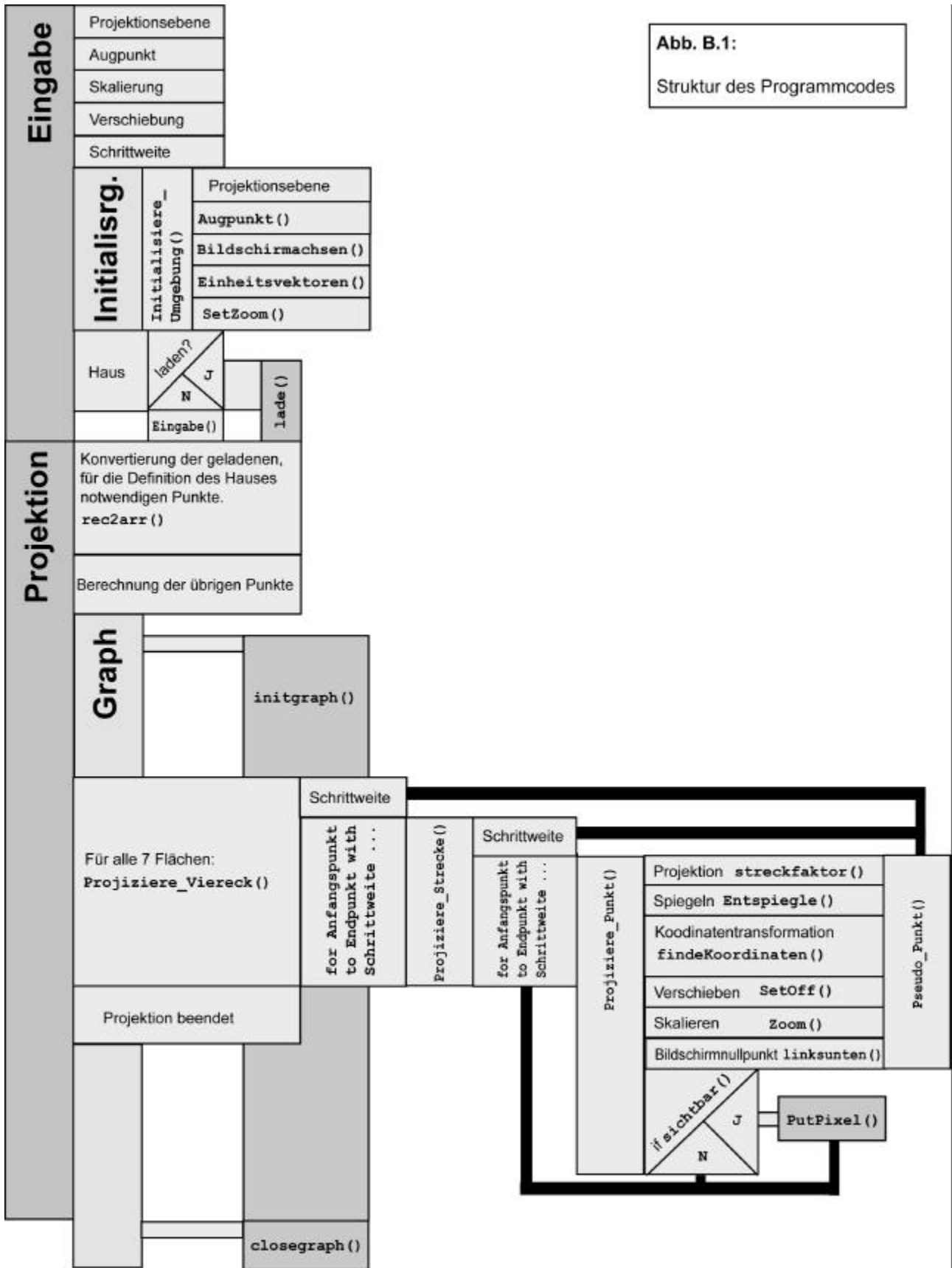


Abb. B.1:
Struktur des Programmcodes

haus.pas: Das Hauptprogramm

```
1  program proj_haus; {13.12.2000}
2
3  {
4
5  *****
6  PROGRAMM ZUR ZENTRALPROJEKTION DES KOERPERS 'HAUS'
7  *****
8
9  Maturarbeit im Fach Mathematik
10 von Ch. Leuzinger <christoph.leuzinger@westworks.ch>
11 an der Kantonsschule Schaffhausen <http://www.kanti.ch/>
12
13 12. 10. 2000 - 12. 12. 2000
14 Sourcecode und Binary unter <http://westworks.ch/~chris/ma/>
15
16 Uebersetzt mit dem Free Pascal Compiler v1.00 <http://www.freepascal.org/>
17 }
18
19 uses
20     Crt,           { Ein-/Ausgabe Unit }
21     Graph,        { Grafikhardware Unit }
22     Vectfunc,     { Vektorgeometriefunktionen }
23     Gedaechnis;   { speichern/laden von Obejktdaten }
24
25 type vektor = array[1..3] of real;           { 3D-Vektor }
26 type vektor2d = array[1..2] of real;        { 2D-Vektor }
27 type parameter = real;                      { allg. Parameter }
28
29 type hausdaten = record                    { Objekt "Haus" }
30     A,B,C,D,E,F,G,H,I,J : vektor;         { Eckpunkte v. "Haus" }
31     end;
32
33
34 const
35     Xres = 640;                             { Aufloesung horizontal }
36     Yres = 480;                             { Aufloesung vertikal }
37
38     mode : smallint = 30009;                { Graphikmodus fuer "InitGraph" }
39     driver : smallint = 15;                 { ----- " ----- }
40
41 var
42     { Projektionsebene, Parallelebenen: }
43     A,B,C,D : parameter;                   { Parameter der Ebenengleichung }
44
45     vn,                                     { Normalenvektor auf die
46                                     { Projektionsebene }
47     vOPn,                                   { Ortsvektor des Tragerpunkts F von vn }
48
49     { Projektionszentrum: }
50     vOP,                                    { Ortsvektor des Projektionszentrums P }
51     vOP2,                                   { Das auf die Projektionsebene
52                                     { gefaellte Lot des Projektionszentrums}
53     : vektor;
54
55
56
57
58     { 3D-System }
59     { Datenstrukturen }
60     daten : eckdaten;                       { Typ aus Gedaechnis-Unit; nur fuer die
61                                     { Definition des Hauses notwendige Punkte }
62     haus : hausdaten;                       { Alle Punkte des Hauses }
63
64
65
66
67     { 2D-System, Bildschirm }
68     vScreenX,                               { Richtungsvektor x-Achse des Bildschirms }
69     vScreenY,                               { Richtungsvektor y-Achse des Bildschirms }
70     ve1,ve2,                                { 3D-Basisvektoren des 2D-Systems }
71     : vektor;
72
73     TheMatrix : array[0..640,0..480] of real; { Sichtbarkeit von Punkten }
74
75     dichte,                                 { Faktor zur Veraenderung der Schrittweite bei der
76                                     { Projektion von Strecken/Parallelogrammen }
77     OffsetX,OffsetY,                       { Parameter, um den auf dem Bildschirm sichtbaren
78                                     { Ausschnitt zu verschieben }
```

```

79     ZoomX,ZoomY           { Zoom- / Skalierungsfaktoren }
80     : parameter;
81
82     Xrange, Yrange : integer;      { Wert fuer die die Breite/Hoehe des Ausschnitts }
83
84
85
86
87     { I/O Variabeln }
88     taste : char;           { Variable fuer Eingabewerte }
89     status : integer;       { Variable fuer den Status der Eingabe }
90
91     { Bildschirmeinstellungen }
92     farbe : word;           { Farbe, mit der gezeichnet wird }
93
94     ausdatei : string;      { Datei, aus der gelesen wird }
95
96
97
98 { ***** Programmfunktionen ***** }
99
100
101 { Zur Berechnung des Streckungsparameters der Projektionsgeraden: }
102 function streckfaktor(
103     A,B,C,D : parameter;      { Projektionsebene }
104     vOX,     { Zu projizierender Punkt }
105     vOP      { Projektionszentrum }
106     : vektor
107
108     ) : parameter;           { Funktion gibt den Streckungsparameter
109                               zurueck }
110
111     var     vXP : vektor;     { Verbindungsvektor XP }
112
113     begin
114         vXP := vsubtr(vOP,vOX);
115         streckfaktor := -(
116             (A * vOX[1] + B * vOX[2] + C * vOX[3] + D)
117             /
118             (A * vXP[1] + B * vXP[2] + C * vXP[3])
119             );
120     end;
121
122
123
124
125 function fvie(
126     A,B,C,D : parameter      { Projektionsbene }
127
128     ) : vektor;              { Funktion gibt Vektor zurueck }
129
130     {
131     Eine Funktion zum Finden eines Vektors im Raum, der in der Ebene liegt
132     kollinear zu dem Vektor ist, der Als horizontale Achse des Bildschirms
133     dient.
134     Er soll ausserdem nach rechts zeigen (von "ausen" gesehen).
135
136     ANMERKUNGEN:
137     o A, B und C entsprechen den Komponenten des Normalenvektors der
138     Projektionsebene.
139     o Die z-Komponente des gesuchten Vektors wird durch den Winkel
140     bestimmt; nur waagrecht => v[3] := 0.
141     }
142
143     var
144         v : vektor;          { Der gesuchte Vektor }
145
146     begin
147         v[3] := 0;
148
149         if (A <> 0) and (B <> 0) then
150             begin
151                 v[1] := -B;
152                 v[2] := A;
153             end
154
155         else if (A <> 0) and (B = 0) then
156             begin
157                 v[1] := 0;
158                 v[2] := A;
159             end

```

```

160
161     else if (A = 0) and (B <> 0) then
162     begin
163         v[1] := -B;
164         v[2] := 0;
165     end
166
167     else if (A = 0) and (B = 0) and (C > 0) then
168     begin
169         v[1] := 0;
170         v[2] := C;
171     end
172
173     else if (A = 0) and (B = 0) and (C < 0) then
174     begin
175         v[1] := 0;
176         v[2] := -C;
177     end
178
179     else writeln('Unerwarteter Normalenvektor (' ,A,'|',B,'|',C,')');
180
181     fvie := v;
182
183 end;
184
185
186 function findeKoordinaten(
187     ve1,ve2,           { Basisvektoren }
188     vOX                { in die Projektionsebene projizierter Punkt }
189     : vektor
190
191     ) : vektor2d;
192
193 var
194     Koordinaten        { die gesuchten Koordinaten des Punktes in der
195                       : vektor2d;
196                       Ebene }
197
198     vP2X                { Verbindungsvektor Ursprung 2D - Punkt }
199     : vektor;
200
201 begin
202     vP2X := vsubtr(vOX,vOP2);
203
204     Koordinaten[1] := vscal(ve1,vP2X);
205     Koordinaten[2] := vscal(vmult(ve2,-1),vmult(vP2X,-1));
206
207     findeKoordinaten := Koordinaten;
208 end;
209 { ***** }
210
211
212
213
214 { ***** Verschieben / Skalieren / Sichtbarkeit ***** }
215
216 function SetOff(vOX : vektor2D) : vektor2D;
217
218     { Verschieben des Bildschirmausschnitts }
219
220 begin
221     vOX[1] := vOX[1] - OffsetX / ZoomX;
222     vOX[2] := vOX[2] - OffsetY / ZoomY;
223     SetOff := vOX;
224 end;
225
226
227 function Zoom(vOX : vektor2D) : vektor2D;
228
229     { Multiplizieren der Koordinaten mit den Zoomfaktoren }
230
231 begin
232     vOX[1] := vOX[1] * ZoomX;
233     vOX[2] := vOX[2] * ZoomY;
234     Zoom := vOX;
235 end;
236
237 function linksunten(vOX : vektor2D) : vektor2D;
238
239     { Beim Bildschirm liegt der Nullpunkt links oben. Ich moechte ihn links unten.}
240

```

```

241     begin
242         vOX[2] := Yres - vOX[2];
243         linksunten := vOX;
244     end;
245
246
247 function sichtbar(vOX : vektor2D; vXP : vektor) : boolean;
248
249     { Prueft ob ein Punkt sichtbar ist.
250     Kriterium:
251     o Punkt liegt innerhalb des Ausschnitts =>
252     o Punkt auf Bild noch nicht besetzt
253     sonst
254     o der neue Punkt liegt naeher beim Projektionszentrum => true
255     sonst => false
256     }
257
258     var    x,y : integer;
259
260     begin
261         x := round(vOX[1]);
262         y := round(vOX[2]);
263
264         sichtbar := false;
265
266         if
267             (x >= 0)
268             and (x <= Xres)
269             and (y >= 0)
270             and (y <= Yres)
271         then
272
273             begin
274                 if
275                     (TheMatrix[x,y] = 0)
276                     or (betr3d(vXP) < TheMatrix[x,y])
277                 then
278                     begin
279                         TheMatrix[x,y] := betr3d(vXP);
280                         sichtbar := true;
281                     end
282                 end
283
284             else writeln('Out of range: ',x,', ',y);
285         end;
286
287
288 function Entspiegle(vOX : vektor) : vektor;
289
290     { Punkt wird am in die Projektionsebene projizierten Punkt
291     gespiegelt. Damit wird die Spiegelung, die beim Projektionsvorgang
292     entstanden ist rueckgaengig gemacht. }
293
294     begin
295         Entspiegle := vadd(vOP2,vmult(vsubtr(vOX,vOP2),-1));
296     end;
297
298
299 { ***** }
300
301
302
303 { ***** Projektionsfunktionen ***** }
304
305 procedure Projiziere_Punkt(vOX : vektor);
306
307     { Projiziert einen einzelnen Punkt in die Projektionsebene
308     und zeichnet ihn. }
309
310     var
311         t : parameter;
312         vOX2,
313         vXP : vektor;
314         vOX_2D : vektor2D;
315
316     begin
317         t := streckfaktor(A,B,C,D,vOX,vOP);
318         vXP := vsubtr(vOP,vOX);
319         vOX2 := vadd(vOX,vmult(vXP,t));
320
321         vOX2 := Entspiegle(vOX2);

```

```

322
323     vOX_2D := findeKoordinaten(ve1,ve2,vOX2);
324
325     vOX_2D := SetOff(vOX_2D);
326
327     vOX_2D := Zoom(vOX_2D);
328
329     vOX_2D := linksunten(vOX_2D);
330
331
332     if sichtbar(vOX_2D,vXP) then
333         PutPixel(round(vOX_2D[1]),round(vOX_2D[2]),farbe);
334
335     end;
336
337
338 function Pseudo_Punkt(vOX : vektor) : vektor2d;
339
340     { Projiziert einen einzelnen Punkt in die Projektionsebene
341       und gibt die Koordinaten des Punktes in der Ebene zurueck. }
342
343     var
344         t : parameter;
345         vOX2,
346         vXP : vektor;
347         vOX_2D : vektor2D;
348
349     begin
350         t := streckfaktor(A,B,C,D,vOX,vOP);
351         vXP := vsubtr(vOP,vOX);
352         vOX2 := vadd(vOX,vmult(vXP,t));
353
354         vOX2 := Entspiegle(vOX2);
355
356         vOX_2D := findeKoordinaten(ve1,ve2,vOX2);
357
358         vOX_2D := SetOff(vOX_2D);
359
360         vOX_2D := Zoom(vOX_2D);
361
362         vOX_2D := linksunten(vOX_2D);
363
364         Pseudo_Punkt[1] := round(vOX_2D[1]);
365         Pseudo_Punkt[2] := round(vOX_2D[2]);
366     end;
367
368
369 procedure Projiziere_Strecke(vOA,vOB : vektor);
370
371     { Projiziert eine Strecke. Prinzip: Parameterdarstellung einer Geraden. }
372
373     var
374         vS,      { Vektor der Strecke }
375         vSr      { Richtungsvektor d. Gerade (AB) mit Laenge 1 }
376         : vektor;
377         vOA2d,vOB2d,vS2d : vektor2d; { 2d-Strecke }
378         i,betr,
379         schrittweite : parameter;
380
381     begin
382
383         vS := vsubtr(vOB,vOA);
384         vSr := vdiv(vS,betr3d(vS));
385
386         vOA2d := Pseudo_Punkt(vOA);
387         vOB2d := Pseudo_Punkt(vOB);
388         vS2d[1] := (vOB2d[1] - vOA2d[1]);
389         vS2d[2] := (vOB2d[2] - vOA2d[2]);
390
391         betr := betr2d(vS2d) + 0.000001; { Div by zero vermeiden }
392
393         schrittweite := betr3d(vS) / betr * dichte;
394         i := 0;
395         while i <= betr3d(vS) do
396             begin
397                 Projiziere_Punkt(vadd(vOA,vmult(vSr,i)));
398                 i += schrittweite;
399             end
400         end;
401
402

```

```

403 procedure Projiziere_Viereck(vOA,vOB,vOC : vektor);
404
405     { Projiziert eine Flaechе. Prinzip: Parameterdarstellung einer Ebene. }
406
407     var
408         vBA,      { Vektor der Strecke }
409         vBar      { Richtungsvektor d. Gerade (AB) mit Laenge 1 }
410         : vektor;
411         vOA2d,vOB2d,vBA2d : vektor2d; { 2d-Strecke }
412         i,betr,
413         schrittweite : parameter;
414
415     begin
416
417         vBA := vsubtr(vOA,vOB);
418         vBar := vdiv(vBA,betr3d(vBA));
419
420         vOA2d := Pseudo_Punkt(vOA);
421         vOB2d := Pseudo_Punkt(vOB);
422         vBA2d[1] := (vOA2d[1] - vOB2d[1]);
423         vBA2d[2] := (vOA2d[2] - vOB2d[2]);
424
425         betr := betr2d(vBA2d) + 0.000001;           { Div by zero vermeiden }
426         schrittweite := betr3d(vBA) / betr * dichte;
427
428         i := 0;
429         while i <= betr3d(vBA) do
430
431             begin
432                 Projiziere_Strecke(
433                     vadd(vOB,vmult(vBar,i)), { Anfangspunkt A' der auf der ersten Gerade }
434
435                     vadd(
436                         vadd(vOB,vmult(vBar,i)), { Gleicher Punkt wie oben (A') ... }
437                         vsubtr(vOC,vOB)         { ... plus Strecke A'B' ... }
438                     )                          { ... = B' . }
439                 );
440
441                 i += schrittweite;
442             end
443
444         end;
445
446
447
448
449 { ***** }
450
451
452
453
454 {***** Initialisierung der Umgebung *****}
455
456 procedure Projektionsebene;
457     begin
458         A := vn[1]; { > }
459         B := vn[2]; { > Normalenvektor }
460         C := vn[3]; { > }
461         D := -A*vOfn[1] - B*vOfn[2] - C*vOfn[3];
462     end;
463
464
465
466 procedure Augpunkt;
467
468     var
469         vPPx,      { Vektor auf der Projektionsgeraden des Projektionszentrums }
470         vPP2      { Verbindungsvektor zw. Projektionszentrum und
471                   projiziertem Projektionszentrum }
472         : vektor;
473
474     begin
475         { Das in die Projektionsebene projizierte Projektionszentrum vOP2: }
476         vPPX[1] := vOP[1] + vn[1]; { Ein Punkt auf derselben Geraden wie }
477         vPPX[2] := vOP[2] + vn[2]; { derjenigen, auf der vOP und das projizierte }
478         vPPX[3] := vOP[3] + vn[3]; { Projektionszentrum liegen => "raten" }
479
480         vPP2 := vmult(vn,streckfaktor(A,B,C,D,vOP,vPPX));
481         vOP2 := vadd(vOP,vPP2);      { Projektion in die P'ebene. }
482     end;
483

```

```

484
485 procedure Bildschirmachsen;
486
487     begin
488         vScreenX := fvie(A,B,C,D);      { Findet einen geeigneten Vektor in der
489                                         Projektionsebene }
490         vScreenY := vprod(vn,vScreenX); { Zweiter Basisvektor durch Vektorprodukt
491                                         aus Normalenvektor und dem waagrechten
492                                         Bildschirmachsenrichtungsvektor vScreenX
493                                         => vScreenY zeigt senkrecht zu ScreenX nach oben }
494     end;
495
496
497
498 procedure Einheitsvektoren(
499     v1,v2          { Bildschirmachsen-Richtungsvektoren }
500     : vektor
501 );
502
503     { ANMERKUNGEN:
504     o Die Richtungsvektoren der Bildschirmachsen werden
505     durch ihren Betrag dividiert. }
506
507     begin
508         ve1 := vdiv(v1,betr3d(v1));
509         ve2 := vdiv(v2,betr3d(v2));
510     end;
511
512
513 procedure SetZoom;
514
515     begin
516         { Berechne Zoomfaktor: }
517         { Bildschirmaufloesung / Breite/Höhe des Ausschnitts }
518         ZoomX := Xres / Xrange;
519         ZoomY := Yres / Yrange;
520     end;
521
522
523 procedure Initialisiere_Umgebung;
524
525     begin
526         Projektionsebene;
527         Augpunkt;
528         Bildschirmachsen;
529         Einheitsvektoren(vScreenX,vScreenY);
530         SetZoom;
531     end;
532
533 { ***** }
534
535
536 { ***** Ein- / Ausgabe ***** }
537
538
539 procedure Eingabe;
540
541     var status : integer; { Lokaler Staus }
542         taste : char; { Lokale Tasteneingabe }
543         indatEI : string;
544
545     begin { der Eingabe }
546
547
548
549         writeln;
550         writeln('Haus: Punkte A,B,D,E,I');
551         writeln('=====');
552
553         write('Punkt A      x: ');
554         readln(daten.A.x);
555         write('              y: ');
556         readln(daten.A.y);
557         write('              z: ');
558         readln(daten.A.z);
559
560         write('Punkt B      x: ');
561         readln(daten.B.x);
562         write('              y: ');
563         readln(daten.B.y);
564         write('              z: ');

```

```

565     readln(daten.B.z);
566
567     write('Punkt D      x: ');
568     readln(daten.D.x);
569     write('           y: ');
570     readln(daten.D.y);
571     write('           z: ');
572     readln(daten.D.z);
573
574     write('Punkt E      x: ');
575     readln(daten.E.x);
576     write('           y: ');
577     readln(daten.E.y);
578     write('           z: ');
579     readln(daten.E.z);
580
581     write('Punkt I      x: ');
582     readln(daten.I.x);
583     write('           y: ');
584     readln(daten.I.y);
585     write('           z: ');
586     readln(daten.I.z);
587
588
589     writeln;
590     writeln('OK, das reicht. ');
591     writeln;
592
593     status := 0;
594     repeat
595     write('Speichern (j/n)? ');
596     readln(taste);
597     case taste of
598         'j', 'J': begin
599             write('Dateiname: ');
600             readln(indatei);
601             speichere(indatei, daten);
602             status := 1;
603             end;
604         'n', 'N': status := 1;
605         else      status := 0;
606     end;
607     until(status = 1);
608
609     end; { der Eingabe. }
610
611
612 { ***** }
613
614
615
616 begin
617
618     clrscr;
619     writeln('Eingabe der Umgebung: ');
620     writeln('=====');
621
622     writeln;
623     writeln('Normalenvektor n der Projektionsebene: ');
624     write('      x: '); Readln(vn[1]);
625     write('      y: '); Readln(vn[2]);
626     write('      z: '); Readln(vn[3]);
627
628
629     writeln;
630     writeln('Ortsvektor des Traegerpunkts F fuer den: ');
631     writeln('Normalenvektor n der Projektionsebene: ');
632     write('      x: '); Readln(vOFn[1]);
633     write('      y: '); Readln(vOFn[2]);
634     write('      z: '); Readln(vOFn[3]);
635
636
637     writeln;
638     writeln('Ortsvektors Projektionszentrums P: ');
639     write('      x: '); Readln(vOP[1]);
640     write('      y: '); Readln(vOP[2]);
641     write('      z: '); Readln(vOP[3]);
642
643     writeln;
644     writeln('Zoom: ');
645     write('      x: '); Readln(Xrange);

```

```

646     write('    y: '); Readln(Yrange);
647
648     writeln;
649     writeln('Offset: ');
650     write('    x: '); readln(OffsetX);
651     write('    x: '); readln(OffsetY);
652
653     writeln;
654     writeln('Dichte: ');
655     write('    x: '); readln(dichte);
656
657     Initialisiere_Umgebung;
658
659     status := 0;
660     repeat
661         writeln;
662         write('Lade Daten (j/n)? ');
663         readln(taste);
664         case taste of
665             'j','J': begin
666                 write('Dateiname: ');
667                 readln(ausdatei);
668                 daten := lade(ausdatei);
669                 status := 1;
670             end;
671             'n','N': begin
672                 Eingabe;
673                 status := 1;
674             end;
675             else      status := 0;
676         end;
677     until(status = 1);
678
679     haus.A := rec2arr(daten.A);
680     haus.B := rec2arr(daten.B);
681     haus.D := rec2arr(daten.D);
682     haus.E := rec2arr(daten.E);
683     haus.I := rec2arr(daten.I);
684
685     haus.C := vadd(haus.B,vsubtr(haus.D,haus.A));
686     haus.F := vadd(haus.B,vsubtr(haus.E,haus.A));
687     haus.G := vadd(haus.C,vsubtr(haus.E,haus.A));
688     haus.H := vadd(haus.D,vsubtr(haus.E,haus.A));
689     haus.J := vadd(haus.I,vsubtr(haus.D,haus.A));
690
691
692
693     { ***** Switche in den Graphikmodus ***** }
694     initgraph(driver, mode, '');
695
696     writeln; write('Zeichne');
697     write(' ABCD'); farbe := red; Projiziere_Viereck(haus.A,haus.B,haus.C);
698     write(' ABFE'); farbe := cyan; Projiziere_Viereck(haus.A,haus.B,haus.F);
699     write(' EIJH'); farbe := white; Projiziere_Viereck(haus.E,haus.I,haus.J);
700     write(' BCGF'); farbe := blue; Projiziere_Viereck(haus.B,haus.C,haus.G);
701     write(' ADHE'); farbe := red; Projiziere_Viereck(haus.A,haus.D,haus.H);
702     write(' DCGH'); farbe := green; Projiziere_Viereck(haus.D,haus.C,haus.G);
703     write(' FGJI'); farbe := yellow; Projiziere_Viereck(haus.F,haus.G,haus.J);
704
705     writeln('.'); writeln;
706     writeln('Beendet. Beliebige Taste druecken ...');
707     readkey;
708     closegraph;
709
710 end.

```

vectfunc.pas: Vektorfunktionen

```

1  unit Vectfunc; { Letzte Aenderung: 29.10.2000 }
2
3  { Funktionen fuer die Vektorrechnung (Addition, Subtraktion, Multiplikation und Division mit
4   Skalaren, etc. }
5
6  interface
7  type vektor = array[1..3] of real;
8  type vektor2d = array[1..2] of real;
9  type parameter = real;
10
11 function vadd(va,vb : vektor) : vektor; {Addition von Vektoren}
12 function vsubtr(va,vb : vektor) : vektor;

```

```

13 function vmult(va : vektor; t : parameter) : vektor; {Multiplikation von Vektoren mit Skalaren}
14 function vdiv(va : vektor; t : parameter) : vektor; {Komponentenweise dividieren}
15 function vschal(va,vb : vektor) : parameter; { Skalarprodukt }
16 function vprod(va,vb : vektor) : vektor; {Vektorprodukt}
17 function betr2d(v2d : vektor2d) : parameter; { Betrag eines 2d-Vektors}
18 function betr3d(v3d : vektor) : parameter; { Betrag eines 3d-Vektors}
19
20
21
22 implementation
23
24 { ***** Vektorfunktionen ***** }
25
26
27 function vadd(va,vb : vektor) : vektor; {Addition von Vektoren}
28     var vc : vektor;
29
30     begin
31         vc[1] := va[1] + vb[1];
32         vc[2] := va[2] + vb[2];
33         vc[3] := va[3] + vb[3];
34
35         vadd := vc;
36     end;
37
38
39 function vsubtr(va,vb : vektor) : vektor;
40 {Subtraktion von Vektoren}
41
42     var vc : vektor;
43
44     begin
45         vc[1] := va[1] - vb[1];
46         vc[2] := va[2] - vb[2];
47         vc[3] := va[3] - vb[3];
48         vsubtr := vc;
49     end;
50
51
52 function vmult(va : vektor; t : parameter) : vektor; {Multiplikation von Vektoren mit Skalaren}
53     var vc : vektor;
54
55     begin
56         vc[1] := va[1] * t;
57         vc[2] := va[2] * t;
58         vc[3] := va[3] * t;
59
60         vmult := vc;
61     end;
62
63
64
65 function vdiv(va : vektor; t : parameter) : vektor; {Komponentenweise dividieren}
66     var vc : vektor;
67
68     begin
69         if(t <> 0) then
70             begin
71                 vc[1] := va[1] / t;
72                 vc[2] := va[2] / t;
73                 vc[3] := va[3] / t;
74
75                 vdiv := vc;
76             end
77         else
78             writeln('Div by zero, func vdiv.');
```

```

94         vc[2] := va[3] * vb[1] - va[1] * vb[3];
95         vc[3] := va[1] * vb[2] - va[2] * vb[1];
96
97         vprod := vc;
98     end;
99
100
101
102
103 function betr3d(v3d : vektor) : parameter;    { Betrag eines 3d-Vektors}
104     begin
105         betr3d := sqrt(v3d[1] * v3d[1] + v3d[2] * v3d[2] + v3d[3] * v3d[3]);
106         if(betr3d = 0) then writeln('Warning: function betr3d returns zero.');

```

gedaechtnis.pas: Speichern und laden von Objekten

```

1  unit gedaechtnis;    { 12.11.2000}
2  { Dient dazu, die Daten des Objekts (Haus) zu speichern und lesen}
3  { Erweitert am 5. 12. 2000 um Konvertierungsfunktionen. }
4
5  interface
6      type vektor = array[1..3] of real;
7      type punkt = record
8          x,y,z : real;
9      end;
10     type eckdaten = record
11         A,B,D,E,I : punkt;
12     end;
13
14     procedure speichere(dateiname : string; daten : eckdaten);
15     function lade(dateiname : string) : eckdaten;
16     function rec2arr(strukt : punkt) : vektor;
17     function arr2rec(strukt : vektor) : punkt;
18
19
20 implementation
21
22 procedure speichere(dateiname : string; daten : eckdaten);
23     var
24         datei : file of punkt;
25
26     begin
27         assign(datei, dateiname);
28         rewrite(datei);
29         write(datei, daten.A);
30         write(datei, daten.B);
31         write(datei, daten.D);
32         write(datei, daten.E);
33         write(datei, daten.I);
34         close(datei);
35     end;
36
37
38
39 function lade(dateiname : string) : eckdaten;
40     var
41         datei : file of punkt;
42
43     begin
44         assign(datei, dateiname);
45         reset(datei);
46         read(datei, lade.A);
47         read(datei, lade.B);
48         read(datei, lade.D);
49         read(datei, lade.E);
50         read(datei, lade.I);
51         close(datei);
52     end;
53

```

```
54 function rec2arr(strukt : punkt) : vektor; { Wandelt Datensrukturen in Records zu Arrays um }
55     begin
56         rec2arr[1] := strukt.x;
57         rec2arr[2] := strukt.y;
58         rec2arr[3] := strukt.z;
59     end;
60
61 function arr2rec(strukt : vektor) : punkt; { wandelt Datensrukturen in Arrays zu Records um }
62
63     var temp : punkt;
64
65     begin
66         temp.x := strukt[1];
67         temp.y := strukt[2];
68         temp.z := strukt[3];
69
70         arr2rec := temp;
71     end;
72
73 end.
```