Technische Hochschule Mittelhessen

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

22. März 2013

Klausur
Computergrafik
für Bachelor-Studierende
WS 2012 / 13

– Lösungshilfe –

Personalien:

Name, Vorname: .....

Matrikelnummer: .....

Hinweise:

- **Die Bearbeitungszeit beträgt 90 Minuten.**
- **Alle schriftlichen Hilfsmittel sind zugelassen; andere Hilfsmittel, insb. elektr. Rechen- und Kommunikationsapparate, dürfen nicht verwendet werden.**
- **Ausgesprochene Folgefehler (durch Übertragung falscher Zwischenergebnisse) werden in Folgerechnungen als richtig gewertet.**
- **Die Aufgaben sollen nur auf diesen Blättern (inkl. Rückseite) bearbeitet werden. Bei Bedarf kann zusätzliches Papier zur Verfügung gestellt werden.**
- **Zur sicheren Zuordnung aller Lösungen wird um eine persönliche Kennung (Name u./o. Matrikelnr.) auf allen Blättern gebeten.**
- **Auf Wunsch darf auch Bleistift verwendet werden.**

Zur leichteren Lesbarkeit werden Substantive nur in einem Geschlecht („Nutzerin“) verwendet.

1. Aufgabe (10 Punkte)

a) Nach Abschluß Ihres Studiums bekommen Sie ein extrem lukratives Stellenangebot von einem früheren Kommilitonen, der als Plattenleger reich geworden ist und nun dringend einen zuverlässigen Geschäftspartner sucht. Selbstverständlich sagen Sie sofort zu.

Ihr erster Auftrag besteht im Neu-Kacheln einer großen weißen Wand im alten Stadtbad. Sie messen die Wand aus und stellen fest, daß sie 56 Kacheln hoch und 158 Kacheln breit ist. Während Sie noch die Bestellungen für die benötigten Materialien vorbereiten, teilt Ihnen Ihr Partner telefonisch mit, daß sich der Auftraggeber eine Diagonale dieser Wand in Rot wünscht.

Wie viele Kacheln werden in roter Farbe gebraucht, wenn Sie die Diagonale nach dem Bresenham-Algorithmus (inkl. Endpunkte) bilden?

158.

b) Ein Freund zeigt Ihnen mit Stolz seine fehlerfreie Implementierung des Bresenham-Algorithmus, die ihm als Anfänger viel Mühe gemacht hatte.

Damit plant er ein eigenes Zeichenprogramm für die Werbebranche: Es soll zum Fahrradkauf anregen mit Fahrradzeichnungen, in denen die Speichen radial abwechselnd von der Achse weg und zur Achse hin in unterschiedlichen Farben gezeichnet werden und sich immer pixelgenau in der jeweils neuen Farbe überdecken.

Sie erklären ihm, daß er dabei mit Bresenham auf Schwierigkeiten stoßen könnte.

Was für ein Problem könnte dabei auftreten?

(Kurze, allgemeine Schilderung genügt.)

Abb. 1.1

Je nach Neigung jeder Speiche trifft der Bresenham-Algorithmus nicht genau dieselben Pixel, wenn die Speiche in zwei entgegengesetzten Richtungen gezeichnet wird.

c) Sie lernen einen älteren Ingenieur kennen, der bei Nutzern virtueller Welten (Stadtplanern, Simulator-Herstellern, Anbietern von Navigationssystemen) als unverzichtbarer Experte gilt: Sein Wissen und seine Algorithmen schaffen es, auch aus nicht fachkundig aufgenommenen Stereo-Bildern die Gelände-Koordinaten so zu extrahieren, daß damit eine optimale Modellierung des gewünschten Geländes gelingt.

Handelt es sich bei diesem großen Fachmann um einen Experten für Bildbearbeitung, für Bildverarbeitung, für Computergrafik, für eine Kombination daraus, oder für keine von ihnen?

Bitte kreuzen Sie die richtige(n) Antwort(en) an!

	Experte für Bildbearbeitung
X	Experte für Bildverarbeitung
	Experte für Computergrafik
	Experte für keines der vorgenannten Gebiete

2. **Aufgabe** (40 Punkte)

Sie führen Vorberechnungen für die Grafik eines Navigationssystems durch.

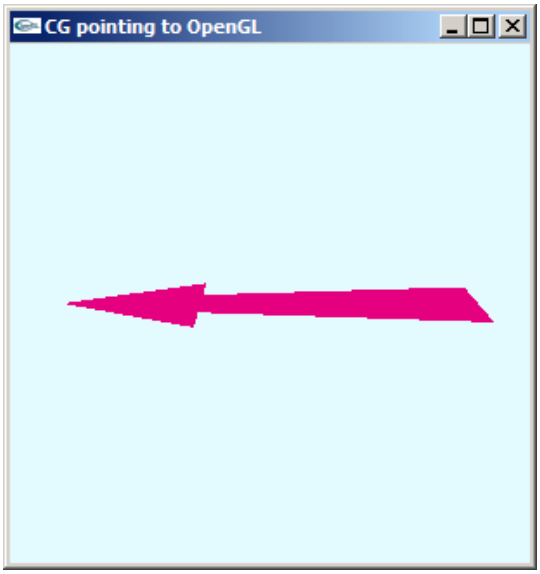


Abb. 2.1

Beim Einschalten soll, während die Navigation Kontakt zu einem Satelliten aufnimmt, anfänglich ein liegender, zur negativen x-Richtung zeigender Pfeil dargestellt werden (Abb. 2.1), der sich zunächst vom Betrachter wegdreht, um sich dann aufzurichten.

Zur exakten Überprüfung der Ergebnisse Ihres Codes modellieren Sie um den Koordinatenursprung einen Pfeil, welcher der x-Achse entgegengerichtet ist (Abb. 2.2); dann drehen Sie ihn um den rechten Winkel α um die y-Achse bis zur negativen z-Achse (Transformation $\underline{\mathbf{T}}_{(i)}$) und stellen ihn aufrecht durch Drehung um einen weiteren rechten Winkel β um die x-Achse (Transformation $\underline{\mathbf{T}}_{(ii)}$). Am Ende führen Sie die perspektivische Transformation ($\underline{\mathbf{T}}_{(iii)}$) und, nach Anwendung auf die Pfeilkoordinaten, die perspektivische Division durch, für eine Projektion auf die x-y-Ebene. Der Augenpunkt Ihrer Grafik befindet sich auf der z-Achse bei z=10.

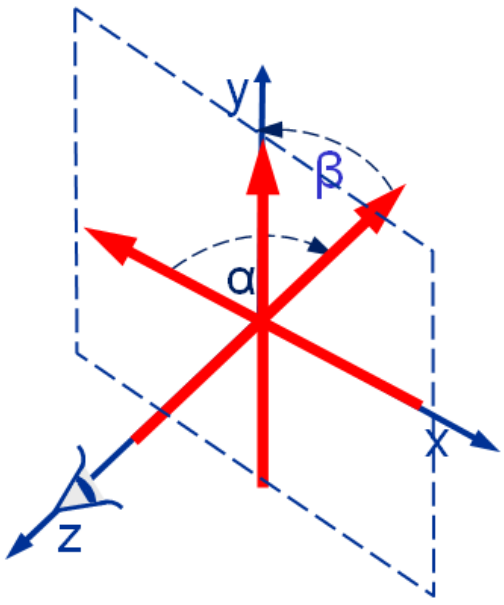


Abb. 2.2

Berechnen Sie bitte die Koordinaten-Transformation und die Projektion des Navigationspfeils, indem Sie folgende Fragen behandeln:

a) Geben Sie bitte (in Grad, unter Berücksichtigung des Drehsinns) den Winkel **α** an, um welchen der Pfeil gedreht werden muß, damit er nicht zur negativen x-, sondern zur negativen z-Achse weist. Wie groß sind dann **sin α** und **cos α**?

α = ***– 90°*** **sin α =** ***–1*** **cos α =** ***0***

b) Wie lautet die Transformationsmatrix $\underline{T}_{(i)}$, die den Pfeil um **α** bis zur z-Achse dreht? Geben Sie sie bitte sowohl in symbolischer (sin α, ...) als auch in arithmetischer (zahlenmäßiger) Form an!

$$\underline{T}_{(i)} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

c) Geben Sie bitte (in Grad, unter Berücksichtigung des Drehsinns) den Winkel **β** an, um welchen der Pfeil als nächstes gedreht werden muß, damit er in positiver y-Richtung zeigt. Wie groß sind **sin β** und **cos β**?

β = ***+ 90°*** **sin β =** ***1*** **cos β =** ***0***

d) Wie lautet die Transformationsmatrix $\underline{T}_{(ii)}$, die den Pfeil in die aufrechte y-Richtung dreht? Geben Sie sie bitte ebenfalls in symbolischer und in arithmetischer Form an!

$$\underline{T}_{(ii)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

e) Geben Sie nun bitte (symbolisch und arithmetisch) eine mögliche Form der Matrix $\underline{T}_{(iii)}$ an, die zur perspektivischen Projektion dient, wenn eine Projektion auf die x-y-Ebene angenommen wird, die ihr Projektionszentrum auf der z-Achse im Abstand N=10 von der Projektionsebene hat:

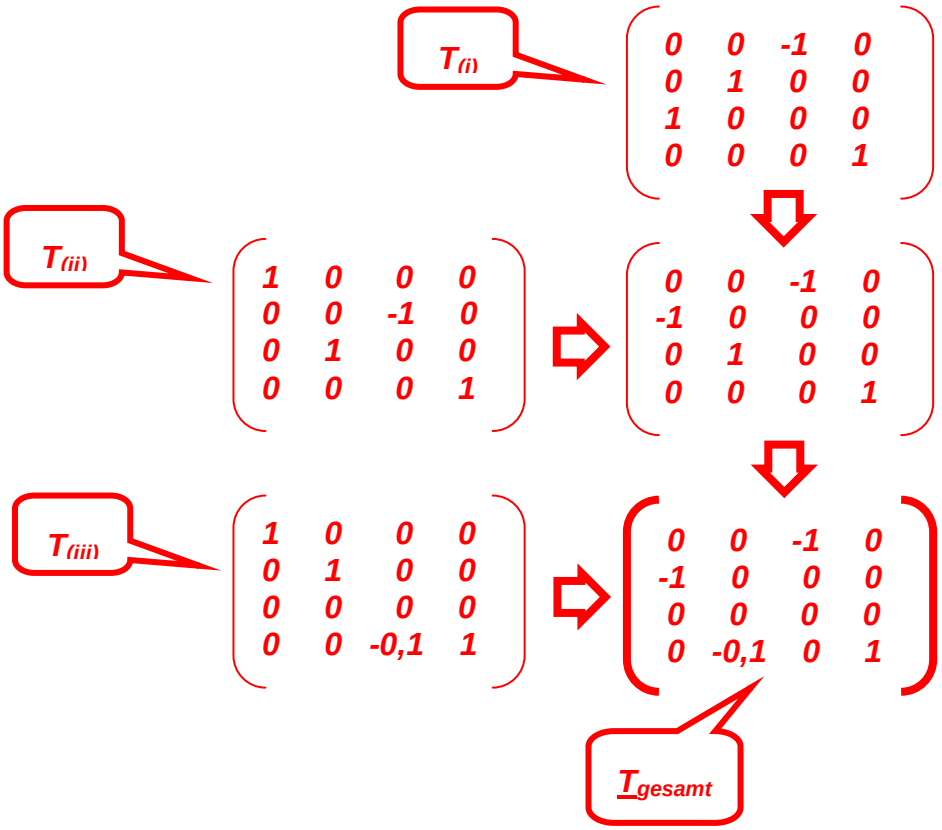
$$\underline{T}_{(iii)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/N & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -0{,}1 & 1 \end{pmatrix}$$

f) Wie läßt sich nun eine Transformationsmatrix $\underline{T}_{gesamt}$ berechnen, welche die vorausgegangenen geometrischen mit der abschließenden perspektivischen Transformation zu einer Operation vereinigt?

$$\underline{T}_{gesamt} = \underline{T}_{(iii)} \bullet \underline{T}_{(ii)} \bullet \underline{T}_{(i)}$$

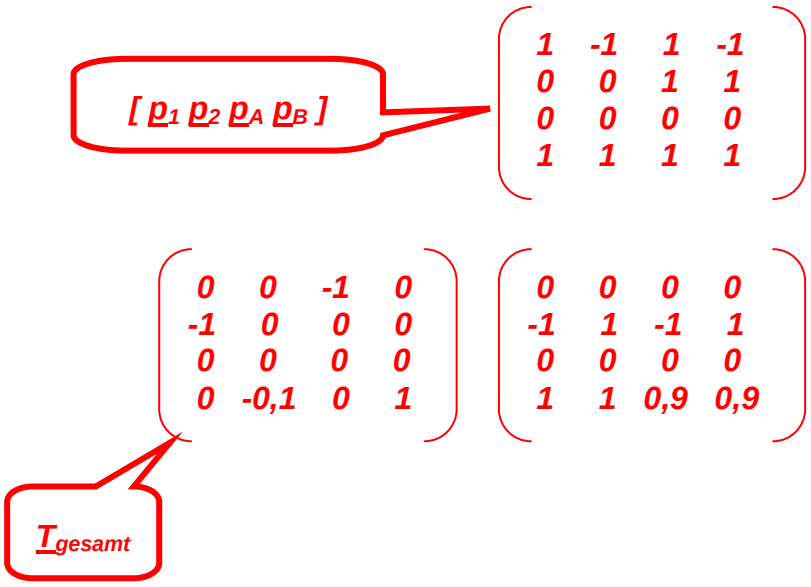
g) Berechnen Sie jetzt bitte die o.a. Transformationsmatrix $\underline{T}_{gesamt}$ in arithmetischer Form:

$\underline{T}_{gesamt} =$



h) Rechnen Sie nun bitte vor, wie nach den vorgenannten Transformationen die Lage $\underline{p}_i^* = (x_i^*, y_i^*)$ der projizierten Endpunkte des o.a. Navigationspfeils zu ermitteln ist, wenn sie anfänglich bei $\underline{p}_1 = (1, 0, 0)$ und $\underline{p}_2 = (-1, 0, 0)$ gelegen haben.

Bestimmen Sie bitte zudem die Projektion eines Pfeiles mit den (von Abb. 2.2 abweichenden) Endpunkten $\underline{p}_A = (1, 1, 0)$ und $\underline{p}_B = (-1, 1, 0)$ und erörtern Sie kurz den Grund für die unterschiedliche Anzahl benötigter Rechenoperationen und die unterschiedliche Größe des projizierten Bildes.



Daraus nach perspektivischer Division (Homogenisierung):

$$[\,\underline{p}_1^*\ \underline{p}_2^*\ \underline{p}_A^*\ \underline{p}_B^*\,] = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 1 & -1{,}1 & 1{,}1 \end{pmatrix}$$

D.h.: Der ursprüngliche Pfeil [$\underline{p}_1^*$ $\underline{p}_2^*$] braucht keine Homogenisierung, weil er in der Projektionsebene liegt; der Pfeil [$\underline{p}_A^*$ $\underline{p}_B^*$] erhält ein vergrößertes Bild, weil er (nach den geometrischen Projektionen) zwischen Augenpunkt und Projektionsebene liegt.

3. **Aufgabe** (50 Punkte)

Ihre Bekanntheit nach dieser Klausur bewirkt, daß sich ein Unternehmen an Sie wendet mit der Bitte, seine Software-Entwickler zu schulen, Pfeile und andere Requisiten für Navigationssysteme mit OpenGL zu codieren.

Sie nehmen die Herausforderung an, bereiten das Programm `NaviPfeil.c` (am Ende dieser Aufgabe) vor und erklären Ihren neuen Schülern, wie sie z.B. einen Pfeil erzeugen, der eine Linkskurve am Ende einer Steigung darstellt (Abb. 3.1).

Dabei gehen Sie auf die Fragen Ihrer Teilnehmer ein:

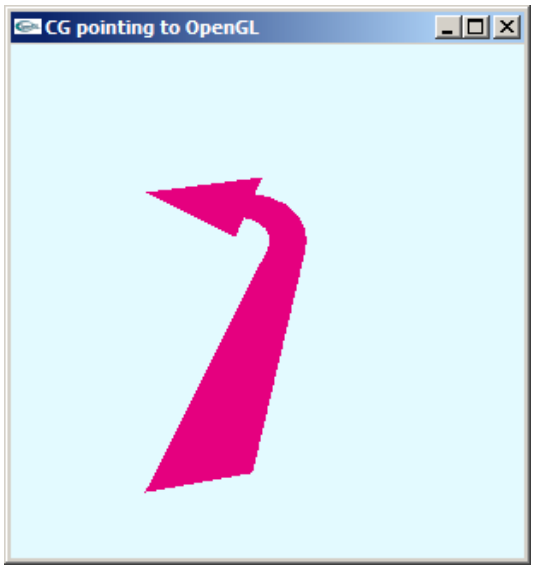


Abb. 3.1

a) Sie werden gefragt, (i) wie groß das Fenster von `NaviPfeil.c` ist, (ii) ob das schon in `main()` festgelegt wird, und (iii) woher man das erfährt.

Wie lautet Ihre Antwort?

(i) Das Fenster ist 300 x 300 Pixel groß.

(ii) Das wird in `main()` mit den anderen Voreinstellungen in `glutInit()` übernommen.

(iii) Das erfährt man in der GLUT-Spezifikation.

b) Auf Frage eines Interessenten antworten Sie: Der Bezeichner `key` in `main()` steht für (Zutreffendes bitte ankreuzen)

	eine Variable vom Typ `char`
	eine Funktion mit Rückgabewert vom Typ `char`
X	eine Callback-Funktion ohne Rückgabewert
	einen Makro-Befehl

c) Am Code-Anfang von `NaviPfeil.c` werden die Farben `about[]` und `arrow[]` definiert. Kreuzen Sie bitte die Beschreibung an, die am ehesten zu jeder Farbe paßt:

about[]	arrow[]	**Beschreibung**
X		Stark ungesättigtes („weißliches“) Cyan (Türkis)
		Sehr helles Grau, fast Weiß
		Ungesättigtes, helles Braun, fast Gelb
	X	Rot mit etwas Blau, fast Magenta (viel dunkler als Pink)
		Stark ungesättigtes Rot, wie Erdbeermilch
		Grünliches Braun, fast Oliv
		Mischung aus Gelb und Blau, leicht grün

d) Die Funktion `init()` modelliert den Navigationspfeil in der ersten Position, in der er dargestellt wird. Das Feld `shaft[][]` enthält die Koordinaten des Pfeilschafts (d.h. des Pfeilkörpers ohne die Spitze). Tragen Sie bitte die Indizes ein, die zu den unten beschriebenen Eckpunkten des Schafts gehören:

Index	**Beschreibung**
3	obere linke Ecke
2	obere rechte Ecke
0	untere linke Ecke
1	untere rechte Ecke

e) Der o.a. Pfeilschaft ist flächig; er ist modelliert in der (Zutreffendes bitte ankreuzen)

X	x-y-Ebene
	y-z-ebene
	x-z-ebene

f) In der Funktion `draw()` wird zunächst das Sichtfeld bestimmt und dann das Objekt positioniert, indem die Funktion `move()` aufgerufen wird, die ihrerseits geometrische Transformationen durchführt.

Welche Anweisung sorgt dafür, daß die geometrischen Transformationen in `move()` sich auf das Modell (d.h.: auf den Pfeil) und nicht etwa auf die Blickrichtung auswirken?

Die Anweisung: `glMatrixMode(GL_MODELVIEW);`

g) `NaviPfeil.c` erzeugt nach dem Start in der Funktion `model()` (bei `layer=1`, `angle[X]=0` und `Phi=0`) einen einfachen, nach oben gerichteten Pfeil (Abb. 3.2).

Wird in `model()` bei dieser Programmausführung die Technik des Stippling eingesetzt?

Erklären Sie kurz, wie Sie zu Ihrer Antwort kommen!

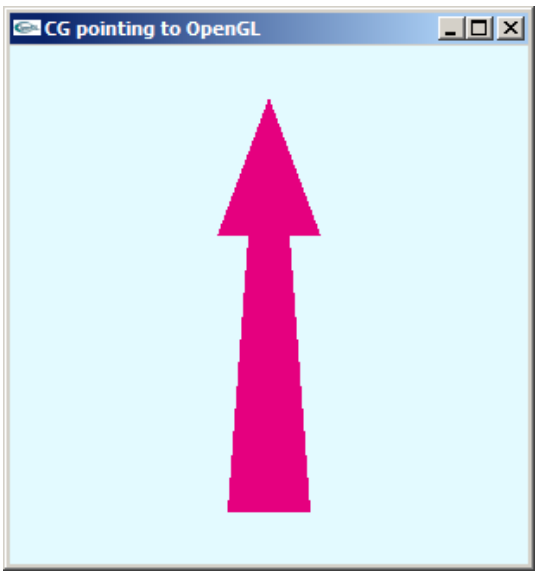


Abb. 3.2

Nein. Die `for`-Schleife wird nur einmal (`layer=1`) durchlaufen, `j1` hat dann den Wert `0`, und aufgrund der `if`-Anweisung wird Stippling abgestellt (`glDisable()`).

h) Wiederholtes Drücken der Taste 'a' verändert in der Funktion `key()` den Wert der Variablen `Phi`. Das führt in `anim()` zur Darstellung eines sich (bis 90°) nach links biegenden und sich wieder aufrichtenden Pfeils (Abb. 3.3). Die Koordinaten des Krümmungsmittelpunktes sind in der indizierten Variablen `center[]` abgelegt. Die zu ihrer Ermittlung verwendete Variable `edge` wurde in `init()` bestimmt.

Ihre Schüler wollen mehr über den Krümmungsmittelpunkt wissen:

Welchen Abstand von der Ebene des Pfeils hat der Krümmungsmittelpunkt, und woran erkennen Sie das?

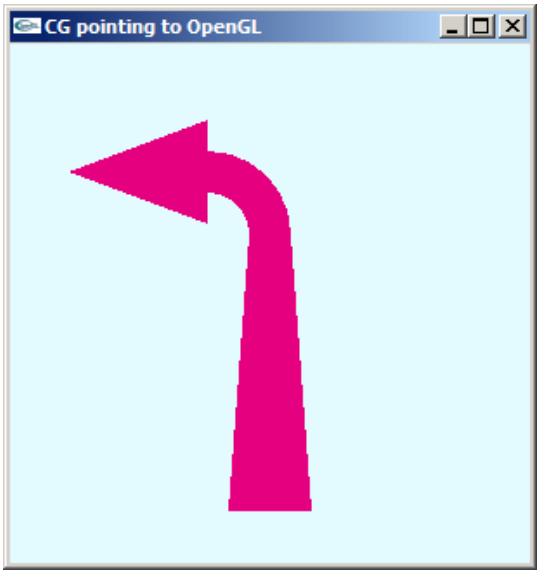


Abb. 3.3

Durch die Anweisungen (in `anim()`)

```
center[Z] = shaft[3][Z];
```

wird deutlich, daß `center[]` in derselben Ebene wie der Pfeil liegt.

(Das ist die x-y-Ebene, da in `init()` `shaft[i][Z] = 0.f; i=0,...,3` gesetzt wird.)

i) Markieren Sie bitte seine ungefähre Lage mit einem Kreuzchen neben der vergrößerten Drahtmodell-Zeichnung des Schafts in Abb. 3.4. Nutzen Sie auch die Größe `edge`, um seine Abstände von den bestimmenden Eckpunkten zu kennzeichnen.

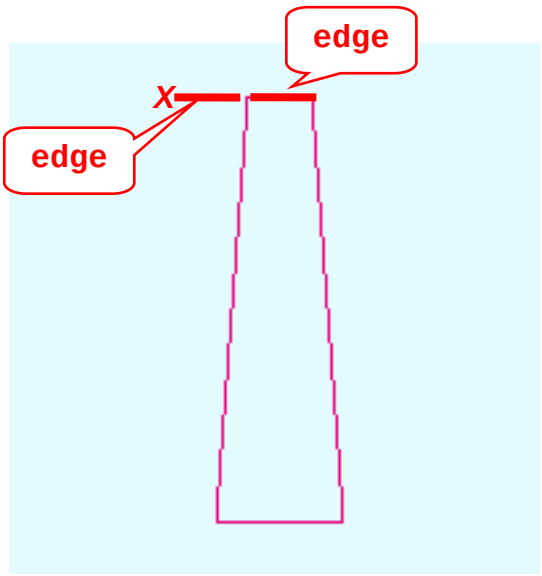


Abb. 3.4

j) Die Biegung des Pfeils erfolgt in der `for`-Schleife der Funktion `anim()`, wo nur die vier Koordinaten-Tripel der indizierten Variablen `bend[]` verwendet werden.

Sie erklären Ihren Schülern, daß hier ein Bogen beschrieben wird (Zutreffendes bitte ankreuzen),

X	indem, durch wiederholte Darstellung eines kleinen Vierecks in unterschiedlichen Positionen, nacheinander ein N-Eck entsteht, das wie ein Ringsegment aussieht, ohne ein Ringsegment zu sein.
	indem, durch mehrfaches Einsetzen der vier Eckpunkte von `bend[]`, Stützstellen markiert werden, durch die dann OpenGL eine „weiche“ Kurve legt.
	indem durch ständiges Überschreiben der Werte von `bend[]` schließlich eine Spur erzeugt wird, wie von einem Stock, der über einen Sandkasten geführt wird.

k) Die `for`-Schleife zur Darstellung der Biegung in `anim()` führt nacheinander eine Translation, eine Rotation und dann eine zweite Translation, welche der ersteren entgegengerichtet ist.

Diese Reihe von Transformationen ist offenbar notwendig; denn Auskommentierung der beiden Translationen erzeugt die Abb. 3.5 (statt Abb. 3.3)

Was bewirken die beiden Translationen, und wie erklärt sich Abb. 3.5? (Kurze Erläuterung)

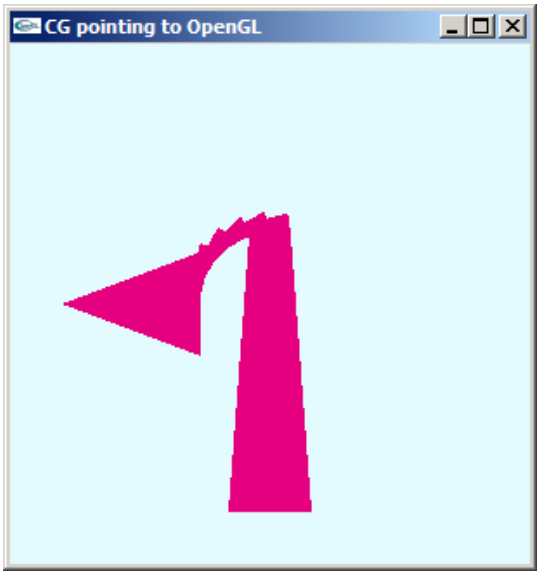


Abb. 3.5

Die Translationen verschieben das Rotationszentrum in den Koordinatenursprung vor der Biegung des Pfeils. Nach der Drehung um den Koordinatenursprung werden die gedrehten Pfeilteile an ihre anfängliche Position zurückverlegt. Unterbleibt dies, so dreht sich die Pfeilspitze um den Koordinatenursprung (der in der Fenstermitte liegt).

l) Bei genauerem Hinsehen fällt auf, daß von den zwei Translationen in `anim()` die erste (obere) eine Verschiebung um `center[i], i=X,Y,Z`, also vom Koordinatenursprung um einen Krümmungsradius weiter weg, die letzte dagegen um `-center[i], i=X,Y,Z` (also zum Koordinatenursprung hin) bewirkt.

Ist die Reihenfolge hier gleichgültig, oder muß sie so sein, und was ist die Erklärung dafür?

Die Reihenfolge der Translationen muß so sein: Da die Transformationen von OpenGL durch Matrizenmultiplikation von rechts erfolgen, müssen sie programmtechnisch in der umgekehrten Reihenfolge ihrer mathematisch-logischen Anwendung erfolgen.

m) In der Funktion `anim()` werden die geometrischen Transformationen zur Biegung mit dem Aufruf `glPushMatrix()` eingeleitet und mit `glPopMatrix()` abgeschlossen. Aus der OpenGL-Spezifikation ist bekannt, daß damit ein bestimmter Teil der gesamten Berechnung bewahrt, ein anderer verworfen werden soll.

Erläutern Sie bitte mit wenigen Worten, welche Transformationen an dieser Stelle erhalten, welche aufgegeben werden sollen.

Die Transformationen zur Positionierung des ganzen Pfeils sind allen Segmenten der Biegung gemeinsam; sie werden für die ganze `for`-Schleife benötigt und werden deshalb mit `glPushMatrix()` festgehalten. Die Transformationen zur Plazierung der Kurven-Teile werden dagegen nur für das jeweils aktuelle Segment benötigt und deshalb nach ihrem Einsatz verworfen.

n) Innerhalb der Anweisungsfolge für die Biegung (Funktion `anim()`) wird der letzte Schleifendurchlauf nicht zu Ende geführt: Die Anweisung `break` beendet die `for`-Schleife vorzeitig und führt über `return` zurück zur aufrufenden Funktion `model()`.

Welchen Sinn hat diese Aktion, und kann sie nicht zu einem Stack-Überlauf durch eine Überzahl von `glPushMatrix()`-Aufrufen führen?

Die `for`-Schleife in `anim()` enthält jeweils einen Durchlauf mehr, als zur Berechnung der Biegung notwendig wäre. Die letzte Transformation dient der Pfeilspitze; nach deren Darstellung wird in `model()` die Funktion `glPopMatrix()` aufgerufen, wodurch ein Überlauf unmöglich wird.

o) Ein „Feature“ dieses Navigationspfeils besteht darin, daß durch Musterung (Stippling) in flachen Positionen eine Art Nebel oder „atmosphärischer Dunst“ imitiert wird (Abb. 3.6), womit dem Bild ein stärkerer Eindruck von Tiefe verliehen wird.

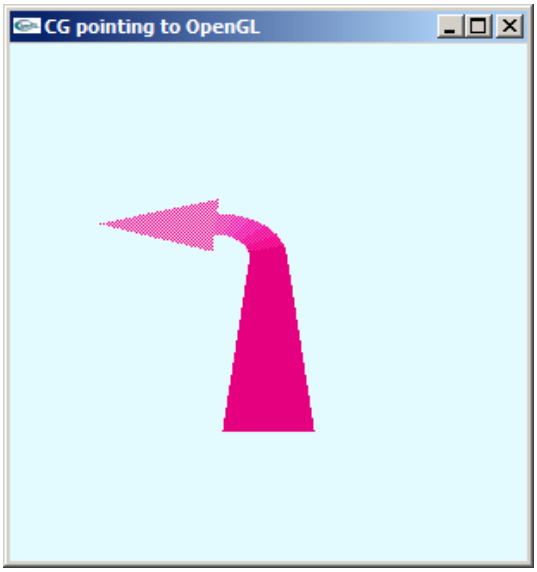


Abb. 3.6

Wie im Code erkennbar, wird die Musterung in `model()` ein- oder ausgeschaltet und in `anim()` ausgeführt. Eine Besonderheit dabei ist, daß die graduelle Abschwächung mit nur einem Muster (`stipX[]`, in `NaviPfeil.h`) erreicht wird.

Erläutern Sie bitte mit wenigen Worten,

- wodurch im Code bei unverändertem Muster eine schrittweise Veränderung des Pfeils (hin zur Spitze) erreicht wird,
- was bewirkt, daß diese Veränderung sich als schwächere Erkennbarkeit des Pfeils bemerkbar macht.

Die Veränderung erfolgt durch Änderung seiner Farbe mittels Aufruf von `glColor3f()`.

Die Abschwächung entsteht durch die gleichmäßige Erhöhung aller Farbkomponenten; damit wird die Pfeilfarbe zunehmend ungesättigter, d.h., sie geht allmählich in Weiß über.

```
/* NaviPfeil.h */
#ifndef NAVIPFEIL_H
  #define NAVIPFEIL_H

  #include <stdio.h> //wg. printf()
  #include <math.h>  //wg. sin(), cos()
  #include <GL/glut.h>
  #define CON_CLS  "cls"
  #define ESC       27
  #define VIEW_X    30
  #define VIEW_Y    30
  #define ARR_X     VIEW_X/40.f
  #define ARR_Y     VIEW_Y/6.f
  #define PHIMAX    75

  enum {X=0, Y=1, Z=2, W=3};

  /*Prototypen:*/
  void anim(void);
  void draw(void);
  void init(void);
  void key (unsigned char key, int x, int y);
  void model(void);
  void move(void);

  /*Globale Variable: */
  GLubyte stipX[] = {
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55};

  /* Makro: */
  #define MIN(x,y) (( (x) < (y) ) ? (x) : (y))

#endif //NAVIPFEIL_H
```

```
/* NaviPfeil.c */
/*Darstellung eines Navigationspfeils mit OpenGL*/
#include "NaviPfeil.h"

/*Globale Variablen: */
float    arrow[3]={.9f,0.f,.5f}, about[3]= {.89f, .98f, 1.f},
         angle[3]={0.,0.,0.}, center[3], edge=0.f;
int      anima=1, Phi=0, dPhi=15;
GLdouble nah=VIEW_Y, fern=2*VIEW_Y;
GLfloat  shaft[4][3], head[3][3], bend[4][3];

/*****************************************************************************/
  void move(void)
/*****************************************************************************/
{ /*Alles ins Sichtvolumen verschieben:*/
  glTranslatef(0., 0., -(nah+VIEW_X/4.));

  /*Positionierung:*/
  glRotatef(angle[X], 1., 0., 0.);
  glRotatef(angle[Y], 0., 1., 0.);
  glRotatef(angle[Z], 0., 0., 1.);
  return;
}

/*****************************************************************************/
   void anim(void)
/*****************************************************************************/
/*Animation:*/
{ int j1=0, dj1=0;
  const  double  GRAD  = atan(1.) / 45.;

  dj1 = dPhi;  center[X] = shaft[3][X]-edge;
  center[Y] = shaft[3][Y];  center[Z] = shaft[3][Z];
  bend[2][X] = center[X] + 2*edge * cos(dPhi*GRAD) ;
  bend[2][Y] = center[Y] + 2*edge * sin(dPhi*GRAD);
  bend[3][X] = center[X] + edge * cos(dPhi*GRAD);
  bend[3][Y] = center[Y] + edge * sin(dPhi*GRAD);

  /*Biegung:*/
  for (j1=0; j1!=Phi+dj1; j1+=dj1)
  { if (glIsEnabled(GL_POLYGON_STIPPLE))
      glColor3f (MIN(arrow[0]+j1/100., 1.),
                 MIN(arrow[1]+j1/100., 1.),
                 MIN(arrow[2]+j1/100., 1.));

    glPushMatrix();

    glTranslatef(center[X], center[Y], center[Z]);
    glRotatef(j1, 0., 0., 1.);
    glTranslatef(-center[X], -center[Y], -center[Z]);

    if (j1==Phi) break;
    glBegin(GL_POLYGON);//
      glVertex3fv(bend[0]);   glVertex3fv(bend[1]);
      glVertex3fv(bend[2]);   glVertex3fv(bend[3]);
    glEnd();

    glPopMatrix();
  }
  return;
}
```

```
/*************************************************************************/
   void model(void)
/*************************************************************************/
/*Modell zeichnen:*/
{ int j1=0, layer=1;

  /*Schaft:*/
  glDisable (GL_POLYGON_STIPPLE);
  glColor3fv (arrow);

  glBegin(GL_POLYGON);
    glVertex3fv(shaft[0]);   glVertex3fv(shaft[1]);
    glVertex3fv(shaft[2]);   glVertex3fv(shaft[3]);
  glEnd();

  if (angle[X] > -120 && angle[X] < -40) layer=2;
  for (j1=0; j1<layer; j1++)
  { if (!j1) { glDisable (GL_POLYGON_STIPPLE); glColor3fv (arrow); }
    else     { glEnable (GL_POLYGON_STIPPLE);
               glPolygonStipple (stipX); }
    if (Phi) anim();

    /*Spitze:*/
    glBegin(GL_POLYGON);
      glVertex3fv(head[0]);   glVertex3fv(head[1]);
      glVertex3fv(head[2]);
    glEnd();
    if (Phi) glPopMatrix();
  }
  return;
}

/*************************************************************************/
  void draw(void)
/*************************************************************************/
/*Grafik erstellen:*/
{ glClear(GL_COLOR_BUFFER_BIT);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  /*Sicht:*/
  glFrustum (-VIEW_X/2., VIEW_X/2., -VIEW_Y/2., VIEW_Y/2., nah, fern);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  /*Positionierung:*/
  move();
  /*Zeichnung:*/
  model();
  /*Darstellung:*/
  glFlush();
  return;
}

/*************************************************************************/
  int main(int argc, char **argv)
/*************************************************************************/
{ glutInit(&argc, argv);
  glutCreateWindow("CG pointing to OpenGL");
  glutDisplayFunc(draw);
  glutKeyboardFunc(key);
  init();
  glutMainLoop();
  return 0;
}
```

```
/*********************************************************************/
  void init(void)
/*********************************************************************/
/*Eckpunkt-Koordinaten Pfeil:*/
{ /*Pfeil-Spitze:*/
  head[0][X] =-5*ARR_X;  head[0][Y] =   ARR_Y;  head[0][Z] = 0;
  head[1][X] = 5*ARR_X;  head[1][Y] =   ARR_Y;  head[1][Z] = 0;
  head[2][X] = 0;        head[2][Y] = 3*ARR_Y;  head[2][Z] = 0;

  /*Pfeil-Schaft:*/
  shaft[0][X] = -4*ARR_X;   shaft[1][X] =  4*ARR_X;
  shaft[2][X] =  2*ARR_X;   shaft[3][X] = -2*ARR_X;
  shaft[0][Y] = shaft[1][Y] = -3*ARR_Y;
  shaft[2][Y] = shaft[3][Y] =  ARR_Y;
  shaft[0][Z] = shaft[1][Z] = shaft[2][Z] = shaft[3][Z] = 0.f;

  /*Biegung:*/
  bend[0][X] = shaft[3][X];
  bend[1][X] = shaft[2][X];
  bend[0][Y] = shaft[3][Y];
  bend[1][Y] = shaft[2][Y];
  bend[0][Z] = bend[1][Z] = bend[2][Z] = bend[3][Z] = 0.f;

  edge = shaft[2][X]-shaft[3][X];

  /*Loeschfarbe:*/
  glClearColor (about[0], about[1], about[2], 1.);

  /*Tastendruck simulieren, um Menue auszugeben:*/
  key(' ', 0, 0);
  return;
}

/*********************************************************************/
  void key(unsigned char key, int x, int y)
/*********************************************************************/
/*Menue und Eingabe-Behandlung:*/
{ switch (key)
  { case ESC: exit(0);
    case 'a': if (Phi + anima*dPhi < 0 || Phi > PHIMAX) anima = -anima;
              if (anima == 1) Phi+=dPhi; else  Phi-=dPhi;
              break;
    case 'r': angle[X]=angle[Y]=angle[Z]=0.;
              anima=1; Phi=0; dPhi=15;
              break;
    case 'x': angle[X] -= 5.; if (angle[X] <=-360) angle[X]+=360; break;
    case 'X': angle[X] += 5.; if (angle[X] >= 360) angle[X]-=360; break;
    case 'y': angle[Y] -= 5.; if (angle[Y] <=-360) angle[Y]+=360; break;
    case 'Y': angle[Y] += 5.; if (angle[Y] >= 360) angle[Y]-=360; break;
    case 'z': angle[Z] -= 5.; if (angle[Z] <=-360) angle[Z]+=360; break;
    case 'Z': angle[Z] += 5.; if (angle[Z] >= 360) angle[Z]-=360; break;
  }
  system (CON_CLS);
  printf ("\n\r Press (keeping the GLUT window activated):");
  printf ("\n\n\r <ESC>  to quit");
  printf ("\n\r a         animate");
  printf ("\n\r r         reload");
  printf ("\n\r x / X     decrease/increase X-rotation angle");
  printf ("\n\r y / Y     decrease/increase Y-rotation angle");
  printf ("\n\r z / Z     decrease/increase Z-rotation angle");
  printf ("\n\n\r Current values:");
  printf ("\n\r           view angle[X]=%7.2f", angle[X]);
  printf ("\n\r           view angle[Y]=%7.2f", angle[Y]);
  printf ("\n\r           view angle[Z]=%7.2f", angle[Z]);

  draw();
  return;
}
```

Platz für Notizen: