Fachhochschule Gießen-Friedberg

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

27. März 2009

Klausur
Computergrafik
WS 2008 / 09

– 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 wird zusätzliches Papier zur Verfügung gestellt.**
- **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 (25 Punkte)

a) Sie wollen Ihre digitalen Paßfotos retuschieren und die roten Pupillen pixelweise schwärzen. Benötigen Sie dazu ein Programm für Bildverarbeitung, für Bildbearbeitung, für Computergrafik, eine Kombination daraus, oder nichts davon? (Nennung genügt.)

Benötigt wird ein Programm für Bildbearbeitung.

b) Wozu dient der sog. Midpoint-Algorithmus in der Computergrafik, und warum hat er diesen Namen bekommen? Bitte kreuzen Sie die richtige(n) Antwort(en) an!

	Der Algorithmus wird zum Zeichnen von Linien, Kreisen und Ellipsen genutzt. Er prüft zuerst, wo der Verlauf der zu ziehenden Linie die Pixel-Mittelpunkte exakt trifft (daher der Name) und setzt zunächst diese Pixel. Weitere Berechnungen entscheiden, was zu tun ist, wenn die Linie nicht durch die mathematisch ermittelten Mittelpunkte verläuft.
	Im Lehrstoff dieses Faches gab es keinen Algorithmus mit diesem Namen.
X	Mit dem Midpoint-Algorithmus werden gerade Linien im ersten Oktanten gezogen. Sie wachsen von Spalte zu Spalte einer Zeile und wechseln die Zeile, wenn feststeht, daß in der Mitte (Midpoint) der nächsten Spalte die gedachte (unendlich schmale) Linie die Grenze zur nächsten Zeile überschritten haben wird.
	Mit dem Algorithmus lassen sich gerade Linien beliebiger Stärke ziehen. Mathematisch läßt sich das Vorgehen so deuten, daß der nach Länge und Breite mittlere Punkt (Midpoint) der gesamten Linie als erstes gezeichnet wird. Dann werden abwechselnd die Punkte bis zu den beiden Linien-Enden gesetzt.
	Dieser Algorithmus zeichnet Kurven mit intuitiv weichen Krümmungen. Nach Setzen des Anfangs- und des Endpixels der Kurvenlinie wird mit einem Polynom 3. Grades der Mittelpunkt der Linie gesetzt, dann die Mittelpunkte (Midpoints) der beiden dadurch definierten Segmente u.s.w.. Das hat bei zeitkritischen Animationen den großen Vorteil, daß bei Überschreiten der verfügbaren Rechenzeit wenigstens eine gestrichelte Linie zu sehen ist.

c) Auf welchen Oktanten bezieht sich eine Linie, die nach dem Bresenham-Algorithmus vom Punkt (1; –5) zum Punkt (–1; 5) gezogen wird, und auf welchen, wenn sie, umgekehrt, von (–5; 1) nach (5; –1) gezogen wird? (Angaben genügen.)

Auf den 3. bzw. auf den 8. Oktanten.

d) Sie lernen einen Informatiker der ersten Generation kennen, der als Datenbank-Spezialist jahrzehntelang sehr erfolgreich im Dienst der internationalen Flugsicherheit gearbeitet hat. Sie erfahren, daß er berühmt wurde, weil er ein Verfahren entwickelte, das nach Eintippen einer Paßnummer sofort ein Referenzbild des Reisepaß-Inhabers mit zusätzlich gespeicherten Koordinaten abruft und dreidimensional von mehreren Seiten darstellt.

Handelt es sich bei diesem großen Datenbank-Fachmann gleichzeitig um einen Experten für Bildverarbeitung, für Computergrafik, für beide oder weder für die eine, noch für die andere? Bitte begründen kurz Sie Ihre Antwort!

Er ist gleichzeitig Experte für Computergrafik (wegen der dreidimensionalen Darstellung).

e) Ihre Arbeitsgruppe entwickelt einen neuartigen Automaten, und Sie sollen dafür ein rechteckiges Display aussuchen, von dem bereits feststeht, daß die eine Kantenlänge 150 mm betragen muß. Bekannt ist, daß die in Frage kommenden Displays kreisrunde Pixel mit einem Radius von 0,1 mm enthalten; diese sind matrixartig unter einem dunklen Abdeck-Glas, auf engstem Raum (aneinander angrenzend) angeordnet.

Die Wahl des Displays soll vom Stromverbrauch abhängen. Als Kenngröße hierfür verwenden Sie innerhalb Ihres Projekts den Stromverbrauch, der beim Aufleuchten der (nach Bresenham gezogenen) Display-Diagonalen entsteht. Der Verbrauch eines einzelnen Pixels liegt bei 1 µW (10^{-6} Watt).

i) Wieviele Millimeter lang kann die noch zu bestimmende Kante werden, wenn die o.a. Kenngröße nicht größer als 1 mW (10^{-3} W) werden darf?

ii) Lassen sich allgemeine Aussagen darüber machen, wie klein der Betrag der o.a. Kenngröße noch gehalten werden kann? Gibt es nach der obigen Beschreibung ein Minimum und/oder ein Maximum?

Die matrixartige Anordnung nimmt je Pixel 0,2 x 0,2 mm² (Durchmesser) in Anspruch.

Die eine Kante enthält also 150 mm / 0,2 mm = 750 Pixel.

i) Die Diagonale darf max. 10^{-3} W = 1.000•10^{-6} W verbrauchen, d.h., sie darf bis zu 1.000 Pixeln enthalten. Daraus ergibt sich die Länge der noch zu bestimmenden Kante: 1.000 • 0,2 mm = 200 mm

ii) Wenn die eine Ausdehnung mit 150 mm vorgegeben ist, kann die Diagonale nicht unter die 750-Pixel-Grenze fallen. Damit beträgt der niedrigste Verbrauch, der erreicht werden kann, 750•10^{-6} W (750 µW). Einen max. Verbrauch kann man nicht angeben, er hängt von der Länge der noch zu bestimmenden Kantenlänge ab.

2. **Aufgabe** (35 Punkte)

Sie erstellen Illustrationen für einen Omnibus-Hersteller und wollen mit zwei Grafiken den Einsatz der Sonnenblende bei Sonneneinstrahlung von der Fahrerseite zeigen:

Die (gemäß Abb. 2.1) an der Höhe **H** über dem Koordinaten-Ursprung und der x-Achse angebrachte Blende der Länge **L** und der Breite **B** (bei vernachlässigbarer Tiefe) wird erst heruntergeklappt und dann zur Seite gedreht (Abb. 2.2).

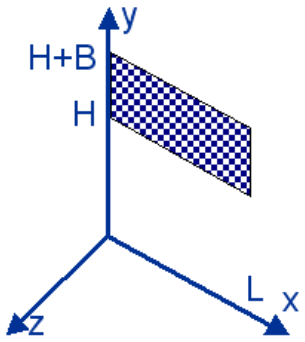


Abb. 2.1

Sie wollen die dazugehörige Koordinaten-Transformation in folgenden vier Schritten berechnen:

i) Die Blende wird entlang der y-Achse an den Koordinaten-Ursprung verschoben, so daß ihre untere Kante an der x-Achse liegt.

ii) Anschließend wird sie um die x-Achse (Winkel α) geklappt.

iii) Dann wird sie um die y-Achse (Winkel β) gedreht.

iv) Schließlich wird sie, wieder entlang der y-Achse, an die korrekte Höhe verschoben.

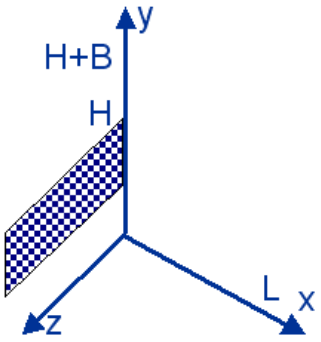


Abb. 2.2

Lösen Sie bitte diese Aufgabe, indem Sie nacheinander folgende Fragen bearbeiten:

a) Wie lautet die Transformationsmatrix $\underline{\mathbf{T}}_{(i)}$ zum o.a. Schritt (i), bei dem die untere Kante der Sonnenblende an den Koordinaten-Ursprung verschoben wird?

$$\underline{\mathbf{T}}_{(i)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -H \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

b) Geben Sie (in Grad, unter Berücksichtigung des Drehsinns) den Winkel **α** an, um den in Schritt (ii) die Sonnenblende um die x-Achse gedreht wird:

α = ***±180°*** **sin α =** ***0*** **cos α =** ***–1***

c) Wie lautet nun die Transformationsmatrix $\underline{T}_{(ii)}$ zu Schritt (ii), nach welchem die Blende umgeklappt ist (in symbolischer und arithmetischer Form)?

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

d) Geben Sie (entsprechend Frage (b), in Grad, unter Berücksichtigung des Drehsinns) den Winkel **β** an, um den in Schritt (iii) die Blende um die y-Achse gedreht wird! Wie groß sind **sin β** und **cos β**?

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

e) Wie lautet nun die Transformationsmatrix $\underline{T}_{(iii)}$ zum o.a. Schritt (iii)? Geben Sie sie bitte wieder in symbolischer und in arithmetischer Form an!

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

f) Wie lautet schließlich die Transformationsmatrix $\underline{T}_{(iv)}$ zum o.a. Schritt (iv), mit dem die Sonnenblende entlang der y-Achse an ihre Position verschoben wird?

$$\underline{T}_{(iv)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & H \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

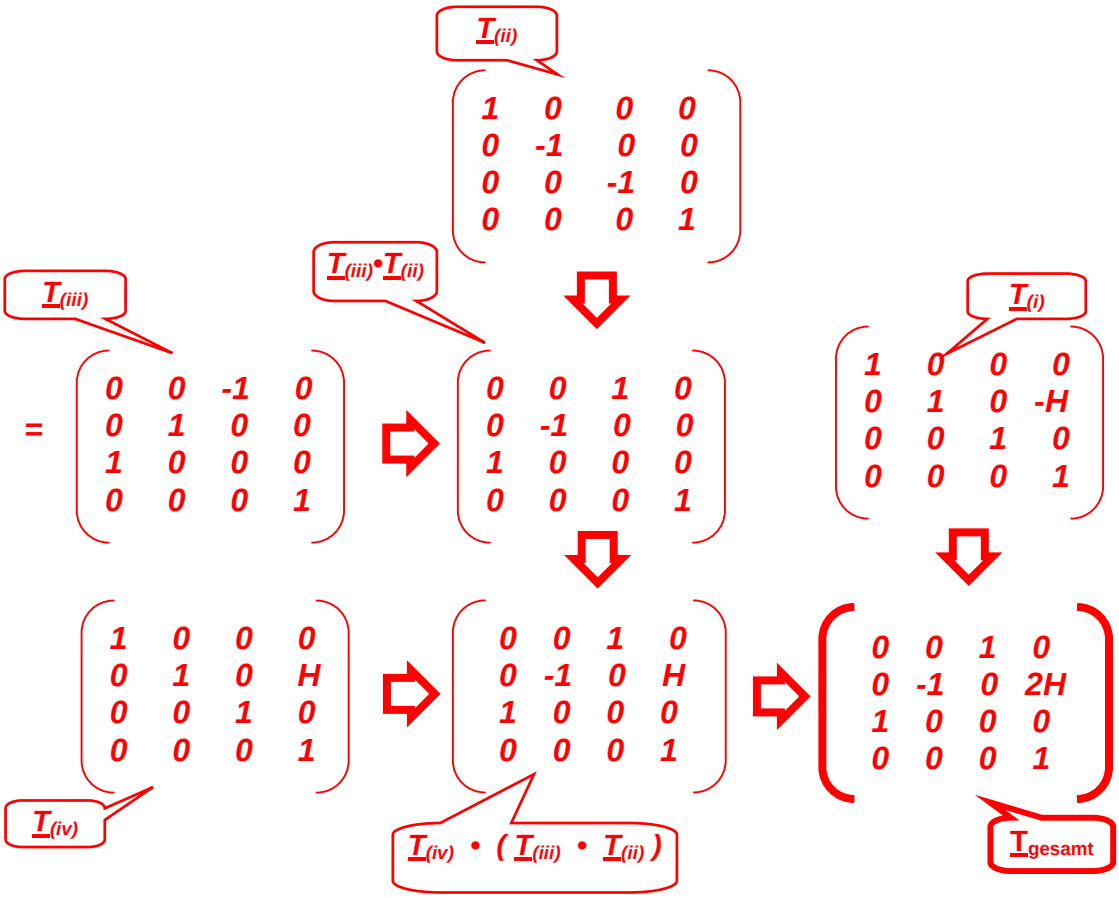
g) Wie berechnet sich die gesuchte Transformationsmatrix für die Koordinaten der Buch-Eckpunkte $\underline{T}_{gesamt}$ aus den bisher besprochenen Transformationen $\underline{T}_{(i)}$ bis $\underline{T}_{(iv)}$?

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

h) Berechnen Sie jetzt bitte die gesuchte Transformationsmatrix $\underline{T}_{gesamt}$ nach den obigen Angaben.

(Tip: Es ist meist einfacher, zunächst die zahlenmäßig vorliegenden Matrizen zusammenzufassen und danach dieses Zwischenergebnis erst mit den schwächer, dann mit den stärker besetzten Symbol-Matrizen zu verknüpfen.)

$$\underline{T}_{gesamt} = [\, \underline{T}_{(iv)} \bullet (\underline{T}_{(iii)} \bullet \underline{T}_{(ii)}) \,] \bullet \underline{T}_{(i)}$$



i) Welche Koordinaten x_{iStart}, y_{iStart}, z_{iStart} (i=1,…4) hatten die vier Eckpunkte der Blende vor der Transformation, welche danach?

Lesen Sie bitte die entsprechenden Angaben aus der Aufgabenstellung und der Abb.2.1 ab, und weisen Sie durch Anwendung von $\underline{T}_{gesamt}$ nach, daß die von Ihnen ermittelte Matrix die vier Ecken entsprechend Abb.2.2 transformiert.

(Sie können die Punkte zu einer 4x4-Matrix zusammenfassen.)

$$\begin{pmatrix} x_{iStart} \\ y_{iStart} \\ z_{iStart} \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & L & L & 0 \\ H & H & H+B & H+B \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 2H \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 0 & 0 & 0 \\ H & H & H-B & H-B \\ 0 & L & L & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} x_{iEnde} \\ y_{iEnde} \\ z_{iEnde} \\ 1 \end{pmatrix}$$

3. **Aufgabe** (40 Punkte)

Sie wollen am PC eine Animation mit OpenGL und GLUT entwickeln, um die Nutzung der Sonnenblende eines Lkws zu veranschaulichen. Hierzu haben Sie ein kurzes Programm geschrieben, das eine einfache Objekt-Darstellung und (zur Orientierung) ein kleines Achsenkreuz zeichnet. Das Listing am Ende dieser Aufgabe enthält die wichtigsten Funktionen. Das Programm wollen Sie bitte anhand folgender Fragen erklären und verbessern.

a) Sofort nach dem Programmstart ist das Fenster nach Abb. 3.1 zu sehen. Kann man am Code erkennen, wie groß es genau ist?

Wenn ja: (i) Wie groß ist das Fenster? (ii) Woran erkennen Sie das? (iii) Können Sie ausschließen, daß die Fenstergröße in einer der hier nicht vorgestellten Funktionen geändert wurde?

Wenn nein: (i) Welchen Teil des Codes vermissen Sie? (ii) Wo vermuten Sie ihn? (iii) Wie könnte er lauten (z.B. Funktionsaufruf mit Platzhaltern für fehlende Größen)? ⇒

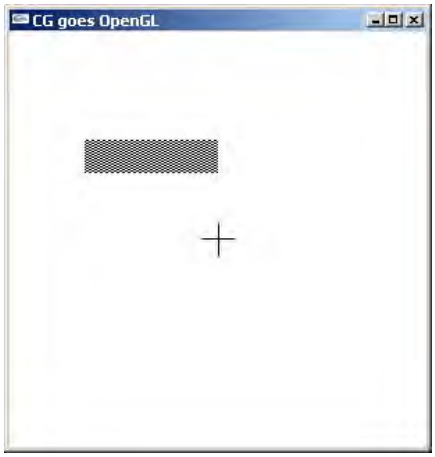


Abb. 3.1

Ja:

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

(ii) Es gibt keinen Aufruf zur Festlegung der Fenstergröße; also muß das Fenster die o.a. Voreinstellung haben.

(iii) Ja (ausgeschlossen): Im `main()` und in `init()` gibt es keine Anweisung zur Fenstergröße, und mit dem Aufruf von `glutMainLoop()` erscheint schon das abgebildete erste Fenster.

b) Das Programm soll später das ganze Lkw-Führerhaus aus der Sicht des Fahrers darstellen können (auch mit heruntergeklappter Blende, Abb. 3.2). Das hat dann eine viermal so große Front wie das Rechteck, das die hochgeklappte Sonnenblende und den Koordinaten-Ursprung enthält (Spiegelung an beiden Achsen). Diese Gesamtfläche soll auch in diagonaler Stellung im selben Fenster Platz finden.

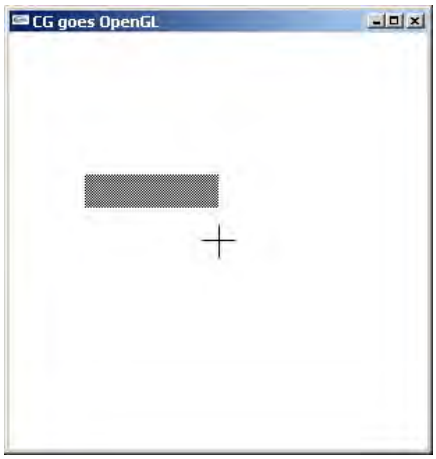


Abb. 3.2

An welcher OpenGL-Anweisung kann man feststellen, daß das Programm für solche Darstellungen vorbereitet ist? Hat die eingerichtete Projektionsfläche dafür ungefähr die passende Größe? (Berechnung u./o. Begründung!)

Der Aufruf `glFrustum()` dient der Vorbereitung der o.a. Sicht. Die eingesetzte Größe der Diagonalen (`diag`) ist genau:

$(H+B)^2+L^2 = 3^2 + 4^2 = 25 = 5^2$

c) Zur Unterscheidung zwischen der Vorder- und der Rückseite der Sonnenblende werden zwei Bitmuster eingesetzt, `pattern1` und `pattern2`, die im Code global verfügbar sind. Klären Sie bitte anhand des Programm-Codes:

(i) Welches der beiden Muster ist dichter, welches ist lichter belegt? (Begründung!)

(ii) Welches Muster wird jeder Seite (Objekt-Fläche) der Blende zugewiesen? (Begründung!) ⇒

(i) pattern1 enthält $6_{16}=0110_2$ und $9_{16}=1001_2$, pattern2 dagegen $A_{16}=1010_2$ und $5_{16}=0101_2$, d.h., der Wechsel zwischen gesetzten und nicht gesetzten Pixeln ist bei pattern1 lichter, bei pattern2 dichter.

(ii) Die Anweisung: `if (!jj) glPolygonStipple(pattern1);` bewirkt, daß der Fläche mit dem Index 0 (!0==true) pattern1, Index 1 pattern2 zugewiesen wird.

d) Die obigen Abbildungen zeigen die Sonnenblende als rechteckigen Rahmen mit einem gitterartigen Muster. Handelt es sich dabei um das Muster einer Seite, oder um die Überlagerung von `pattern1` und `pattern2`? Welche Anweisung / Variable entscheidet darüber?

Es handelt sich um das Muster der einen Fläche (Index 0). Das wird mit der Anweisung:

`if (backF) glEnable (GL_CULL_FACE);`

entschieden, und `backF` ist mit `1` initialisiert.

e) Sie experimentieren mit dem Augenpunkt und stellen fest, daß z.B. eine Halbierung seiner Entfernung zur Projektionsebene die Objekt-Abbildung deutlich verkleinert (Abb. 3.3).

Erläutern Sie bitte kurz (ggf. mit einer Skizze), wie dieser Effekt zu erklären ist:

(i) Welche OpenGL-Anweisung und welcher Parameter darin sind dafür verantwortlich?

(ii) Wie läßt sich dieser Effekt anhand des Sichtvolumens erklären?

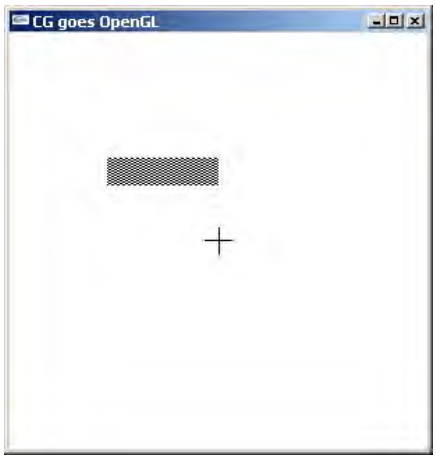


Abb. 3.3

Das Sichtvolumen vergrößert sich durch die Annäherung des Augenpunkts an die Projektionsebene (Verkleinerung von eyez).

(i) Verantwortlich dafür ist die `frustum()`-Anweisung und darin der Parameter nah (wegen nah=eyez;)

(ii) Das vergrößerte Volumen muß im selben Fenster dargestellt werden. Dafür verkleinert OpenGL die Darstellung.

f) Wenn Sie dem Parameter `angle[X]` den Wert 180 geben, erhalten Sie nach dem vorliegenden Code das nebenstehende Bild (Abb. 3.4), was nicht der Bewegung einer Sonnenblende und der o.a. Beschreibung entspricht.

(i) Welche Anweisungen werden benötigt, damit sich die Sonnenblende wie erwartet (Abb. 3.2) herunterklappen läßt?

(ii) Wo müssen diese Anweisungen eingefügt werden?

Abb. 3.4

Benötigt werden Anweisungen, welche die Drehachse an die (x-) Koordinatenachse verschieben und zurück:

(i) (a) `glTranslatef(0, -H, 0.0);` und

(b) `glTranslatef(0, H, 0.0);;`

(ii) Sie müssen unterhalb (a) und oberhalb (b) der Anweisung `glRotatef(angle[X], 1.0, 0.0, 0.0);` eingesetzt werden:

```
glTranslatef(0, H, 0.0);
glRotatef(angle[X], 1.0, 0.0, 0.0);
glTranslatef(0, -H, 0.0);
```

g) Welche Code-Änderung ist zusätzlich notwendig, damit sich die Sonnenblende auch zur Seite drehen läßt (Abb. 3.5)?

(Sie können die Code-Änderung als zusätzliche Anweisungen oder als Anpassung bestehender einbringen.)

Abb. 3.5

Benötigt werden Anweisungen, welche die Drehachse an die (y-) Koordinatenachse verschieben und zurück:

⇒

(i) (a) `glTranslatef(L, 0, 0.0);` und

(b) `glTranslatef(-L, 0, 0.0);`

(iii) Sie müssen unterhalb (a) und oberhalb (b) der Anweisung `glRotatef(angle[Y], 0.0, 1.0, 0.0);;` gesetzt werden:

```
glTranslatef(-L, 0, 0.0);
glRotatef(angle[Y], 0.0, 1.0, 0.0);
glTranslatef(L, 0, 0.0);
```

(Die vier `glTranslatef()`-Anweisungen lassen sich natürlich auch zu einer vor und einer nach den 3 `glRotatef()`-Anweisungen zusammenfassen – s.u..)

h) Ein Freund fragt Sie nach dem Nutzen des letzten Aufrufs in der Funktion `draw()`: `glFlush();`

Zur Veranschaulichung kommentieren Sie diese Zeile aus und führen ihm vor, wie die Grafik-Ausgabe ohne diese Zeile aussieht (Abb. 3.6).

Was sagen Sie ihm dazu über die Wirkung dieses OpenGL-Aufrufs? (Kurze Beschreibung!)

Abb. 3.6

Mit `glFlush()` wird die Ausführung der vorausgegangenen OpenGL-Anweisungen erzwungen; nur dann ist (anders als hier) gesichert, daß sie definitiv ausgelöst wird.

```
/*Darstellung einer Sonnenblende mit OpenGL*/
#include <conio.h> //wg. _getch()
#include <stdio.h> //wg. printf()
#include <GL/glut.h>

#define B           1
#define H           2
#define L           4

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

/*Globale Variablen: */
float    eyez=10., diag=0., angle[3]={0.,0.,0.};
int      backF=1, stipple=1;
GLdouble nah=10., fern=10.;
GLfloat  Vrtx[4][3];
GLint    faces[2][4] = { {0, 1, 2, 3}, {3, 2, 1, 0}};

GLubyte pattern2[] = {
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55,
      /*... insg. 16malige Wiederholung ...*/
      0xAA, 0xAA, 0xAA, 0xAA, 0x55, 0x55, 0x55, 0x55 };

GLubyte pattern1[] = {
      0x66, 0x66, 0x66, 0x66, 0x99, 0x99, 0x99, 0x99,
      /*... insg. 16malige Wiederholung ...*/
      0x66, 0x66, 0x66, 0x66, 0x99, 0x99, 0x99, 0x99};

/*************************************************************************/
  void draw(void)
/*************************************************************************/
{ int jj;
  nah=eyez; fern=nah+2*diag;

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum (-diag, diag, -diag, diag, nah, fern);

  /*BackFaceCulling, Stippling:*/
  if (backF) glEnable  (GL_CULL_FACE);
  else       glDisable (GL_CULL_FACE);
  if (stipple) glEnable  (GL_POLYGON_STIPPLE);
  else         glDisable (GL_POLYGON_STIPPLE);

  glClear(GL_COLOR_BUFFER_BIT);

  /*Bisherige Bewegung neu aufbauen (ohne Fehlerfortpflanzung):*/
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();

  /*Alles ins Sichtvolumen verschieben:*/
  glTranslatef(0.0, 0.0, -(nah+diag/2));

  /*Achsenkreuz (unbeweglich):*/
  glBegin(GL_LINES);
    glVertex3f(-.5, 0., 0.);  glVertex3f( .5, 0., 0.);
    glVertex3f( 0.,-.5, 0.);  glVertex3f( 0., .5, 0.);
    glVertex3f( 0., 0.,-.5);  glVertex3f( 0., 0., .5);
  glEnd();
```

```
  /*Blenden-Bewegung:*/
  glTranslatef(-L, H, 0.);
  glRotatef(angle[X], 1., 0., 0.);

  glRotatef(angle[Y], 0., 1., 0.);

  glRotatef(angle[Z], 0., 0., 1.);
  glTranslatef(L, -H, 0.);
  /*Zeichnung:*/
  for (jj = 0; jj < 2; jj++)
  { if (!jj) glPolygonStipple (pattern1);
    else     glPolygonStipple (pattern2);
    glBegin(GL_POLYGON);
      glVertex3fv(Vrtx[faces[jj][0]]);
      glVertex3fv(Vrtx[faces[jj][1]]);
      glVertex3fv(Vrtx[faces[jj][2]]);
      glVertex3fv(Vrtx[faces[jj][3]]);
    glEnd();
  }
  glFlush();
}

/*****************************************************************************/
  void init(void)
/*****************************************************************************/
{ /*Eckpunkt-Koordinaten:*/
  Vrtx[0][X] = Vrtx[3][X] = -L;   // L=4
  Vrtx[1][X] = Vrtx[2][X] =  0;
  Vrtx[0][Y] = Vrtx[1][Y] =  H;   // H=2
  Vrtx[2][Y] = Vrtx[3][Y] =  H+B; // B=1
  Vrtx[0][Z] = Vrtx[1][Z] = Vrtx[2][Z] = Vrtx[3][Z] =  0;

  /*Diagonale:*/
  diag = 5;

  /*Farben:*/
  glClearColor (1.0, 1.0, 1.0, 1.0);
  glColor3f (0.0, 0.0, 0.0);
  return;
}

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

Platz für Notizen: