**Fachhochschule Gießen-Friedberg**

**Prof. Dr.-Ing. A. Christidis**

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

**18. März 2008**

# Klausur
# Computergrafik
# WS 2007 / 08

## *– Lösungshilfe –*

## Personalien:

**Name, Vorname:** .....

**Matrikelnummer:** .....

## Hinweise:

- **Die Bearbeitungszeit beträgt 90 Minuten.**
- **Alle schriftlichen Hilfsmittel sind zugelassen; andere Hilfsmittel, insb. elektronische Rechen- und Kommunikationsapparate dürfen nicht verwendet werden.**
- **Die Aufgaben sollen nur auf diesen Aufgabenblättern 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** (25 Punkte)

**a)** Sie arbeiten mit an einem Projekt zur Weiterentwicklung von Anzeigegeräten (Joystick, Maus, Knob Box, ...). Zu Ihren Aufgaben gehören vor allem der Entwurf und die Echtzeit-Darstellung neuer Cursor-Grafiken (Pfeile, Fadenkreuze, Sanduhren etc.). Damit sich Ihre Software selbst testen kann, haben Sie Routinen implementiert, welche die Cursor-Bewegungen dokumentieren und immer wieder Screenshots (Abbilder der Bildschirm-Anzeige) erstellen, die sie zum Abgleich der Cursor-Position untersuchen.

Verwenden Sie in Ihrer Arbeit Verfahren der Computergrafik, der Bildverarbeitung, beider oder weder der einen noch der anderen? Begründen Sie Ihre Antwort!

***Beider: Entwurf und Darstellung von Anzeigen gehört zur Computergrafik, Suche und Ortung von Mustern innerhalb eines Pixelbildes (Screenshot) gehören zur Bildverarbeitung.***

**b)** Der Bresenham-Algorithmus ist für den ersten Oktanten entworfen worden und muß zur Anwendung auf andere Oktanten angepaßt werden. An welchem Fehler (kurze Beschreibung) läßt es sich erkennen, wenn der Algorithmus ohne Anpassung auf den 2. Oktanten angewandt wird? Erklären Sie kurz anhand der `MidpointLine()`-Routine, wie dieser Fehler entsteht.

***Es gibt Lücken in der Linie. Der Grund ist, daß jedem x-Wert genau ein y-Wert zugeordnet wird, obwohl mehrere zu ihm gehören. Das erkennt man auch an der Code-Zeile:***

```
for (x=x0; x < xn; x++)
```

***Hier müßten in diesem Fall y-Werte (y, y0, yn) statt x-Werten stehen.***

**c)** Es sollen mit Hilfe des Bresenham-Algorithmus die Koordinaten ermittelt werden, die eine gerasterte Linie mit den Anfangskoordinaten (-15,-17) und den Endkoordinaten (-5,-10) durchläuft.

Auf welchen Oktanten bezieht sich diese Anwendung des Linien-Algorithmus? (Nennung genügt.)

***Auf den 1.***

**d)** Sie zeigen einer Freundin mit berechtigtem Stolz, daß Ihre selbstentwickelte Grafik-Software dank Bresenham-Implementierung das Drahtmodell einer Kugel mit 320 Dreiecken (s. Abb.), auch ohne Backface Culling, bildfüllend, in Echtzeit, wie einen Kinofilm darstellen kann.

Ihre Freundin behauptet, Ihre Software wäre auch ohne die Implementierung des Bresenham-Algorithmus echtzeitfähig gewesen: Die Einsparung von einer Floatingpoint-Operation je Pixel würde, bei geschätzten 10 Nanosekunden pro Floatingpoint-Operation, auch noch bei einem durchschnittlichen Dreiecksumriß von 1.000 Pixeln die Frequenz Ihrer Software auch nicht wie einen frühen Chaplin-Film aussehen lassen.

Angenommen, Ihre Freundin hat die richtigen Zahlen im Kopf: Hat sie recht mit ihrer Behauptung, wenn Sie genau nachrechnen?

***Differenz Δt der Rendering-Zeiten pro Bild, wenn je Pixel eine Floatingpoint-Operation einzuplanen ist:***

***Δt = 10 nsec/ Floatingpoint-Operation * 1 Floatingpoint-Operation/Pixel * 1.000 Pixel/Dreieck * 320 Dreiecke***

***= 3.200.000 nsec = 3.200 µsec = 3,2 msec.***

***Die Auffrischungszeit pro Bild beträgt im modernen Kino 41,66... ms (24 Hz), bei Chaplin waren es noch 62,5 ms (16 Hz). Die Differenz von 20,833... ms ist deutlich höher als die o.a. 3,2 ms.***

***Die Freundin hat recht!***

**e)** Über ein Bild mit den Maßen Δx=300 Pixel und Δy=400 Pixel zeichnen Sie nach dem Bresenham-Algorithmus eine Diagonale. Wieviele Pixel genau werden dabei gesetzt, wenn sowohl Anfangs- als auch Endpixel gesetzt werden?

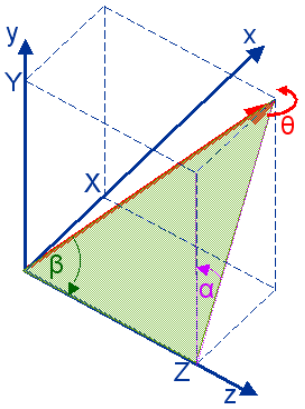
Erläutern Sie bitte kurz (Skizze, Rechnung oder Beschreibung), wie Sie zu Ihrem Ergebnis kommen!

***400 Pixel werden gesetzt:***
***Der Bresenham-Algorithmus nimmt immer die längste Ausdehnung (hier: Δy=400 Pixel) als Bezugsrichtung und ermittelt lediglich zu jedem Pixel in dieser Richtung die passende Koordinate in der anderen Richtung, um ein Pixel zu setzen.***

**2. Aufgabe** (40 Punkte)

Zwei Fliegen (im folgenden betrachtet als Punkte im Raum) begegnen sich in einem Zimmer, an einer Hängelampe, die nachfolgend den Koordinaten-Ursprung markieren soll. Als sie sich das erste Mal erblicken, hat die eine die kartesischen Koordinaten $\underline{f}_1$=(0, 6, 8), während die andere bei $\underline{f}_2$=(10, 6, 8) fliegt.

Die letztere, gesellige Fliege dreht einen Dreiviertel-Kreis (θ=270°) um eine Achse, die durch den Ursprung (Lampe) und ihre Artgenossin führt. Berechnen Sie ihren Flug und die anschließende Annäherung, indem Sie folgende Teilaufgaben wie in der Vorlesung besprochen lösen.




(Die Werte der trigonometrischen Winkelfunktionen zu den hier vorkommenden Winkeln werden als bekannt vorausgesetzt; Folgefehler werden nicht angerechnet.)

Zunächst soll die Drehachse um eine der Hauptachsen (x, y, z) gedreht werden, bis sie mit einer der Hauptebenen (x-y-, y-z-, x-z-Ebene) zusammenfällt.

**a)** Wenn Sie hier die Drehachse um den Winkel α um die z-Achse bis zur y-z-Ebene rotieren lassen wollen – wie berechnen sich der Sinus und der Cosinus des zugehörigen Winkels?

**sin α =** $X / (X^2+Y^2)^{½} = 0 / (0^2+6^2)^{½} = 0 / 6 = 0$

**cos α =** $Y / (X^2+Y^2)^{½} = 6 / 6 = 6 / (0^2+6^2)^{½} = 1$

**b)** Wie lautet die Transformationsmatrix, die diese freie Drehachse um die z-Achse dreht, bis sie mit der y-z-Ebene zusammenfällt? Schreiben Sie sie bitte in symbolischer und arithmetischer Form. Wie erklären Sie sich die Zahlenwerte in der Matrix aus der Anschauung in der Aufgabenstellung?

$$\underline{R}_z(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

***Die Rotationsmatrix ist eine Einheitsmatrix: Die ruhige Fliege liegt bereits in der y-z-Ebene (x-Koordinate=0).***

**c)** Als Nächstes wollen Sie die Drehachse um den Winkel β um die x-Achse rotieren lassen, bis sie mit der z-Achse zusammenfällt. Wie berechnen sich Sinus und Cosinus des Winkels β (symbolisch und arithmetisch)?

**sin β =** $(X^2+Y^2)^{1/2} / (X^2+Y^2+Z^2)^{1/2} = 6 / (36+64)^{1/2} = 0,6$

**cos β =** $Z / (X^2+Y^2+Z^2)^{1/2} = 8 / (36+64)^{1/2} = 0,8$

**d)** Wie lautet die Transformationsmatrix $\underline{R}_X(\beta)$, die diese freie Drehachse um die x-Achse dreht, bis sie mit der z-Achse zusammenfällt? Schreiben Sie sie bitte in symbolischer und arithmetischer Form.

$$\underline{R}_x(\beta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta & 0 \\ 0 & \sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0,8 & -0,6 & 0 \\ 0 & 0,6 & 0,8 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**e)** Wie lautet nun die Matrix für die Rotation der Fliege um die so transformierte z-Achse? Geben Sie bitte die symbolische und die zahlenmäßige Form an:

$$\underline{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**f)** Stellen Sie nun die Rotation der Fliege um die freie Drehachse in symbolischer Form zusammen und vereinfachen Sie sie in dieser Form soweit wie möglich:

$\underline{R}(\theta) = \underline{R}_Z(-\alpha) \bullet \underline{R}_X(-\beta) \bullet \underline{R}_Z(\theta) \bullet \underline{R}_X(\beta) \bullet \underline{R}_Z(\alpha)$ *[ $\underline{R}_Z(\alpha)$: Einheitsmatrix, $\underline{R}_X(\beta)$: orthogonal ]*

$= \underline{R}_X(\beta)^T \bullet \underline{R}_Z(\theta) \bullet \underline{R}_X(\beta)$

**g)** Berechnen Sie jetzt bitte die Transformationsmatrix für die Rotation unserer Fliege auch in arithmetischer Form

$$\underline{R}(\theta) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0{,}8 & -0{,}6 & 0 \\ 0 & 0{,}6 & 0{,}8 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0{,}8 & 0{,}6 & 0 \\ 0 & -0{,}6 & 0{,}8 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \\ -0{,}8 & 0 & 0{,}6 & 0 \\ 0{,}6 & 0 & 0{,}8 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 0{,}8 & -0{,}6 & 0 \\ -0{,}8 & 0{,}36 & 0{,}48 & 0 \\ 0{,}6 & 0{,}48 & 0{,}64 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\underline{R}(\theta)$

**h)** Welches sind nun die Koordinaten der kontaktfreudigen Fliege nach ihrem Rundflug um das Objekt ihrer Begierde? Und, da die umworbene Fliege auch die Drehung um dieselbe Achse mitgemacht hat: Ermitteln Sie die Endposition beider Fliegen!

$$\underline{R}(\theta) \cdot [\, \underline{f}_1 \; \underline{f}_2 \,] = \begin{pmatrix} 0 & 10 \\ 6 & 6 \\ 8 & 8 \\ 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0{,}8 & -0{,}6 & 0 \\ -0{,}8 & 0{,}36 & 0{,}48 & 0 \\ 0{,}6 & 0{,}48 & 0{,}64 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 4{,}8 - 4{,}8 & 4{,}8 - 4{,}8 \\ 2{,}16 + 3{,}84 & -8+2{,}16+3{,}84 \\ 2{,}88 + 5{,}12 & 6+2{,}88+5{,}12 \\ 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 \\ 6 & -2 \\ 8 & 14 \\ 1 & 1 \end{pmatrix}$$

**i)** Nun wollen Sie bitte der mobilen Fliege eine Translationsmatrix entwerfen, mit der sie ihrer Artgenossin bis zur Verschmelzung nahekommt. Führen Sie bitte als Nachweis auch die Matrizenmultiplikation durch, die die Koordinaten der geselligen in jene der ruhig gebliebenen überführt.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & -6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 6 \\ 8 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ -2 \\ 14 \\ 1 \end{pmatrix}$$

**3. Aufgabe** (20 Punkte)

Sie helfen einem Freund, ein Grafik-Programm zu entwickeln, und klären zunächst einige Fragen bezüglich der GLUT-Bibliothek, die er in `main()` einsetzt:

```
int main(int argc, char **argv)
{ glutInit(&argc, argv);
  glutInitWindowSize(300,300);
  glutCreateWindow("CG goes OpenGL");
  glutDisplayFunc(draw);
  glutKeyboardFunc(key);
  init();
  glutMainLoop();
  return 0;
}
```

Beantworten Sie bitte folgende Fragen:

**a)** Das Akronym „GLUT“ steht bekanntlich für „OpenGL Utility Toolkit“. Wofür steht darin das Akronym „OpenGL“ (ausgeschrieben), und was bedeutet inhaltlich das vorgesetzte Wort „Open“ (stichwortartig)?

***„OpenGL” steht für „Open Graphics Library”; „Open” bedeutet hierbei etwa: „Erweiterbar“.***

**b)** Die Anweisung `glutCreateWindow()` weist darauf hin, daß GLUT Fenster einrichtet. Klären Sie bitte folgende Fragen:

Wie sehen GLUT-Fenster aus? Ändern sie ihr Aussehen,

- wenn man denselben Source-Code von einem Betriebs-/Fenstersystem auf ein anderes überträgt, bzw.,
- wenn man innerhalb desselben Fenstersystems die Fensterdarstellung (das sog. Fensterdesign) ändert?

***GLUT richtet native Fenster ein; d.h., die Fenster sehen aus wie alle übrigen Fenster desselben Systems. Deswegen ändert sich das Aussehen sowohl beim Wechsel als auch bei Umstellung des Systems.***

**c)** Beim ersten Entwurf des Grafik-Programms wurde versehentlich die Anweisung `glutCreateWindow()` ausgelassen. Das Programm wurde erfolgreich compiliert, und nach dem Start gab es einen Programm-Absturz mit dem Hinweis auf eine Zugriffsverletzung. Kann dieses Verhalten mit dem vergessenen Statement zusammenhängen?

Wenn ja: Welches Wissen über diese Anweisung würde das erklären?

Wenn nein: Was für ein Programm-Verhalten wäre eher zu erwarten? Warum?

***Ja; das ist sogar die Ursache: `glutCreateWindow()` richtet das GLUT-Fenster ein, d.h., es bereitet das System (Speicher etc.) darauf vor, ein (weiteres) Fenster zu verwalten. Ist der Speicher nicht reserviert, kommt es zwangsläufig zu Zugriffsverletzungen.***

**d)** Was bedeutet die Anweisung `glutInitWindowSize()` im allgemeinen, was im vorliegenden Fall? Was ändert sich, wenn sie ausgelassen wird?

***Sie setzt die Größe des neuen Fensters fest. Ihr Aufruf ist optional (zustandshaftes API). Da die Voreinstellung (300, 300) ist, würde sich hier nichts ändern, wenn sie ausgelassen würde.***

**e)** Darf die o.a. Anweisung `glutInitWindowSize()` auch vor dem Aufruf `glutInit()` stehen?

Wenn ja: Ändert sich dadurch etwas für das aktuelle oder für andere GLUT-Fenster?

Wenn nein: Gegen welche Regel der „Design-Philosophie“ von GLUT verstößt solche Programmierung?

***Ja. Da dies der Festlegung neuer Standards dient und die hier gewählte Fenstergröße gleich dem Default-Wert ist, ändert dies weder am aktuellen noch an zukünftigen Fenstern etwas.***

**f)** Das Programm verwendet offenbar kein Double Buffering. Was ist das, und für welche Beziehung zwischen Bildrate und Bildauffrischungsrate eines Animationssystems ist es besonders interessant?

***Double Buffering ist die Technik, bei der die Grafik erst in einem (nicht sichtbaren) Hintergrund-Puffer entsteht, bevor sie in den (sichtbaren) Bildspeicher übertragen wird; das ist immer interessant für Anwendungen, bei denen die Bildrate unter die Bildauffrischungsrate fallen kann (genau genommen – auch wenn sie ihr „gefährlich nahe“ kommt).***

**g)** Sie beabsichtigen, in spätere Versionen dieses Grafik-Programms eine Hilfe-Funktion zu integrieren, die über die Taste <F1> aufzurufen sein wird. Ist das `main()` dafür vorbereitet?

Wenn ja: Woran erkennen Sie das?

Wenn nein: Ist der Einbau einer solchen Funktionalität mit GLUT überhaupt codierbar, und, falls ja, wie?

***Nein, das Programm ist nicht dafür vorbereitet. Die C-Funktion, über welche die Funktionstasten ansprechbar sind, muß mit einem Aufruf von `glutSpecialFunc()` angemeldet werden.***

**h)** Bei näherer Befassung mit dem gesamten Programm stellen Sie fest, daß der Prototyp der Funktion zur Fenster-Auffrischung `void draw(void)` lautet. Raten Sie Ihrem Freund, einen anderen Prototyp zu wählen, der mit seiner Parameterliste und seinem Rückgabewert den Umgang etwas intuitiver gestaltet?

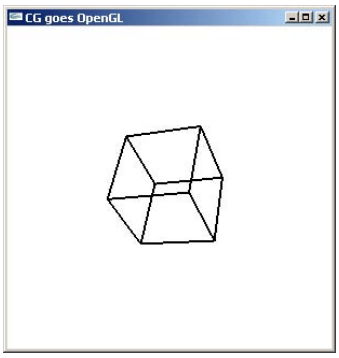
Wenn ja: Was wären interessante Kriterien zur sinnvollen Gestaltung eines GLUT-Programms?

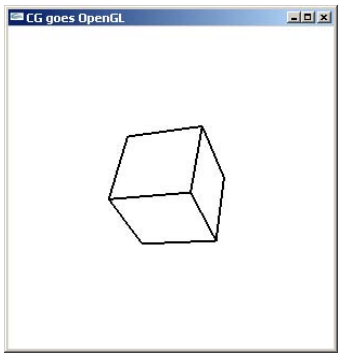
Wenn nein: Was macht einen solchen Prototypen erforderlich?

***Nein. Der Prototyp ist durch GLUT vorgegeben.***

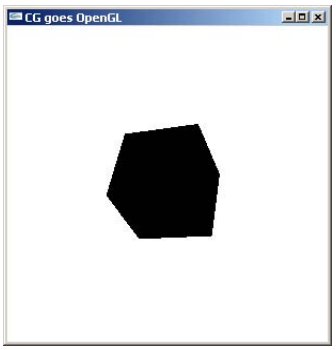
## 4. **Aufgabe** (15 Punkte)

Zum o.a. `main()` schreiben Sie nun die Funktion `draw()` und wollen damit schwarz-weiße Drahtmodelle und Flächenmodelle (s. Abbn.) erzeugen.




Abbn.: Gewünschte Drahtmodell- und Flächenmodell-Darstellungen (v.l.n.r.)




Sie experimentieren mit `draw()` und stellen fest, daß die u.a. Programm-Version (bei Drahtmodellen ebenso wie bei Flächenmodellen) Bilder nach nebenstehender Abb. erzeugt.
Sie erkennen schnell, daß dies am Aufruf:

```
glPolygonMode(GL_BACK, GL_LINE);
```

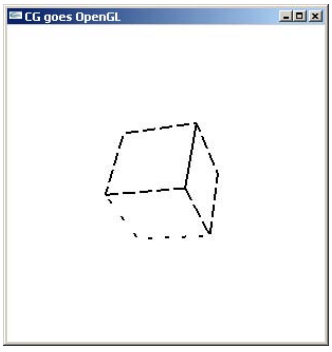
liegt, der richtig lauten sollte:

```
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
```

**a)** Erklären Sie kurz das Programm-Verhalten, das zum o.a falschen Ergebnis führte. Wieso ist die Darstellung sowohl des Flächenmodells als auch des Drahtmodells falsch? Welche Rolle spielt dabei die Realisierung von OpenGL als Zustandsautomat mit Voreinstellungen?

***Der Aufruf `glPolygonMode(GL_BACK,, GL_LINE)` stellt die Rückseiten auf Liniendarstellung ein. Damit ist nichts über die Vorderflächen ausgesagt, und somit gilt die Voreinstellung des Zustandsautomaten; diese ist `GL_FILL` (bei Zeichenfarbe Schwarz). Da die Vorderflächen immer dargestellt werden, sind immer schwarze Flächen zu sehen.***

**b)** Nun wenden Sie sich mit Erfolg der gestrichelten Darstellung zu:




Der abgebildete Würfel (Abb. li.) enthält zwei Linien-Muster, die einzelnen Flächen zugewiesen werden.

Erklären Sie zunächst anhand des u.a. Code-Fragments, welchen Würfelflächen-Indizes welche der beiden Hexadezimal-Kennungen zugewiesen wird. Woran ist die Zuordnung zu erkennen?

***Die bitweise UND-Verknüpfung (`if (jj & 1)`) mit der Eins gewährleistet, daß ungeradezahligen Flächen-Indizes das Muster `0x000F`, den übrigen (inkl. Index 0) `0x0FFF` zugeordnet wird.***

**c)** Beschreiben Sie am Beispiel einer Würfelkante, die von links nach rechts gezeichnet wird, wie die einzelnen Pixel gemäß dem übergebenen Bitmuster gesetzt werden (wieviele Pixel setzen, auslassen etc.):

***`0x000F` = 0000 0000 0000 1111: 4 Pixel setzen, 12 Pixel auslassen, ...***
***`0x0FFF` = 0000 1111 1111 1111: 12 Pixel setzen, 4 Pixel auslassen, ...***

```
void draw(void)
{ int jj;
  nah=eyez; fern=nah+2*diag;
  glClearColor (1.0, 1.0, 1.0, 1.0);
  glColor3f(0.0, 0.0, 0.0);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum (-diag, diag, -diag, diag, nah, fern);

  if (backF)   glEnable  (GL_CULL_FACE);
  else         glDisable (GL_CULL_FACE);
  if (stipple) glEnable  (GL_LINE_STIPPLE);
  else         glDisable (GL_LINE_STIPPLE);

  /*Vorerst nur als Drahtmodell:*/
  glPolygonMode(GL_BACK, GL_LINE);
  glClear(GL_COLOR_BUFFER_BIT);
  /*Ins Sichtvolumen verschieben:*/
  /* .. */
  /*Wuerfel zeichnen:*/
  for (jj = 0; jj < 6; jj++)
  { if (jj & 1) glLineStipple(1, 0x000F);
    else        glLineStipple(1, 0x0FFF);
    glBegin(GL_POLYGON);
    glVertex3fv(&Vrtx[faces[jj][0]][0]);
    glVertex3fv(&Vrtx[faces[jj][1]][0]);
    glVertex3fv(&Vrtx[faces[jj][2]][0]);
    glVertex3fv(&Vrtx[faces[jj][3]][0]);
    glEnd();
  }
/* .. */
}
```

**Platz für Notizen:**