Fachhochschule Gießen-Friedberg

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

11. Februar 2010

Klausur
Computergrafik
für Bachelor-Studierende
WS 2009 / 10

– 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 arbeiten mit mehreren Kollegen an einem neuen Programm, mit dem in digitalen Paßfotos die roten Pupillen geschwärzt werden. Die Software, deren Entwicklung Sie leiten, soll das Original darauf prüfen, ob es überhaupt rote Augen enthält, damit es evtl. den nachgelagerten Programmteilen erst gar nicht zugeführt wird. (Ihr Programm soll also den gesamten Vorgang beschleunigen.)

Fällt Ihre o.a. Arbeit in den Bereich der Bildverarbeitung, der Bildbearbeitung, der Computergrafik, ist sie eine Kombination daraus, oder nichts davon? (Nennung genügt.)

Diese Arbeit fällt in den Bereich der Bildverarbeitung.

b) Welches Kriterium ist für die korrekte Beantwortung der Frage a) maßgeblich? Bitte kreuzen Sie die (eine) richtige Antwort an!

X	Es sollen nach Untersuchung von Bildern Aussagen („rote Augen: ja/nein“) getroffen werden.
	Es wird Software entwickelt, die Dateien (z.B. Bilddateien) bearbeitet.
	Ziel ist die synthetische Erzeugung eines natürlich wirkenden (photo-realistischen) Bildes.
	Es handelt sich um eine anspruchsvolle Software-Entwicklung.
	Es geht um die Behandlung natürlicher / reeller Bilder.
	Es entstehen grundlegende Funktionalitäten, die auch in biometrischen Verfahren verwendet werden können.

c) Sie implementieren den Linien-Algorithmus nach J.E. Bresenham und stellen bei seiner Testung fest, daß eine Linie, die vom Punkt (5;5) zum Punkt (17;17) gezogen wird, einen Fehler aufweist.

Für welchen Oktanten ist der Algorithmus fehlerhaft implementiert worden? (Nennung genügt)

Für den 2. Oktanten. (Von Spezialanwendungen abgesehen, zählt man die Grenze zum jeweils nächsten Oktanten.)

d) Bei einer anderen Implementierung erkennen Sie einen Fehler bei einer Linie vom Punkt (4; 4) zum Punkt (16; 18).

Auf welchen Oktanten weist dieser Fehler hin? (Nennung genügt)

Auf den 2. Oktanten.

e) Warum dürfte jede/r gewissenhafte Informatiker/in zur Beantwortung der Frage d) weniger Zeit brauchen als zur Beantwortung der Frage c) ?

Weil die Linie nach Frage c) die Grenze zwischen zwei (dem 1. und dem 2.) Oktanten bildet; gewissenhafte Informatiker/innen wollen sich erst anhand des Codes vergewissern, welchem Oktanten die Grenze zugeordnet wird.

f) Sie arbeiten als Designer/in und wollen an einem Haushaltsgerät ein rahmenloses Display für die Darstellung von Bedienungshilfen anbringen. Es ist beschlossen, daß der kleine Bildschirm ein Seitenverhältnis von 4:3 (Breite zu Höhe) haben soll, und die Konstrukteure erwarten von Ihnen die Angabe der Diagonalen für die rechteckige Öffnung, die ihn aufnehmen soll.

Sie entscheiden sich für die Verwendung eines Bildschirms mit nichtquadratischen Pixeln, die 0,25 mm breit und 0,15 mm hoch sind und setzen die Display-Breite auf 160 Pixel fest. Beantworten Sie bitte folgende Fragen:

i) Wie viele Pixel hoch sollen die verwendeten Displays sein?

ii) Wie viele cm lang ist die Diagonale der o.a. rechteckigen Öffnung für das Display?

iii) Wie viele Pixel enthält diese Diagonale, wenn sie nach dem Midpoint-Algorithmus gezogen wird?

(Ergebnis nur mit Berechnung bzw. Begründung gültig!)

i) Display-Höhe:

Entscheidend ist die geometrische Ausdehnung;

Display-Breite: 160 Pixel x 0,25 mm / Pixel = 40 mm (= 4 cm)

B / H = 4 / 3 $\Rightarrow$ H = B x 3 / 4 = 4 cm x 3 / 4 = 3 cm (=30 mm)

30 mm / (0,15 mm / Pixel) = <u>200 Pixel</u>

ii) Display-Diagonale:

$d^2 = 4\ cm^2 + 3\ cm^2$

$= (200 \times 0{,}2\ mm)^2 + (100 \times 0{,}3\ mm)^2 = (40\ mm)^2 + (30\ mm)^2$

$= 16cm^2 + 9\ cm^2 = 25\ cm^2$

$\Rightarrow \underline{d = 5\ cm}$

iii) Pixel in der Diagonalen:

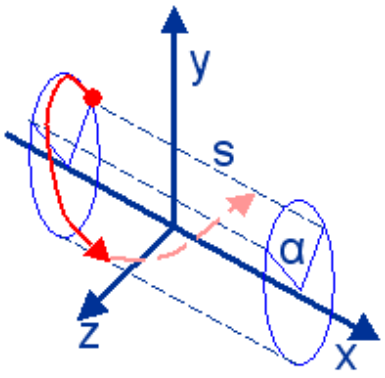
Maßgeblich ist die Dimension, die mehr Pixel enthält:

Display-Breite: 160 Pixel vs. Display-Höhe: 200 Pixel

⇒ 200 Pixel in der Diagonalen

2. Aufgabe (30 Punkte)

Sie wollen in einer Animation einen Leuchtpunkt entlang einer Schraubenlinie bewegen. Dazu lassen Sie den Punkt gleichzeitig um die x-Achse des eingesetzten Koordinatensystems um den Winkel α rotieren und sich um die Strecke s parallel zur x-Achse verschieben. Beide Größen lassen Sie in Abhängigkeit von der Zeit variieren als α(t) bzw. s(t).



Bitte behandeln Sie folgende Fragen:

a) Es seien: $\underline{T}_{Rx}$ die Transformationsmatrix zur Berechnung der Rotation des Punktes um die x-Achse, $\underline{T}_{Tx}$ jene zur Translation entlang derselben Achse.

Wie werden die beiden Transformationsmatrizen zu einer Gesamt-transformation $\underline{T}_{gesamt}$ verknüpft, wenn der Punkt (i) erst um die x-Achse gedreht und dann entlang derselben Achse verschoben wird bzw. (ii) erst verschoben und dann gedreht wird?

(i) $\underline{\mathbf{T}}_{\mathbf{gesamt(i)}} = \underline{T}_{Tx} \bullet \underline{T}_{Rx}$

(ii) $\underline{\mathbf{T}}_{\mathbf{gesamt(ii)}} = \underline{T}_{Rx} \bullet \underline{T}_{Tx}$

b) Welche Regel der Matrizenrechnung bringt zum Ausdruck, daß das Ergebnis mehrerer nacheinander ausgeführter Koordinatentransformationen von der Reihenfolge ihrer Ausführung abhängt?

Die Tatsache, daß das Matrizenprodukt nicht kommutativ ist.
($\underline{A} \bullet \underline{B} \neq \underline{B} \bullet \underline{A}$)

c) Wie lautet die Transformationsmatrix $\underline{T}_{Rx}$ zur Berechnung der Rotation des Punktes um die x-Achse in Abhängigkeit vom Winkel α?

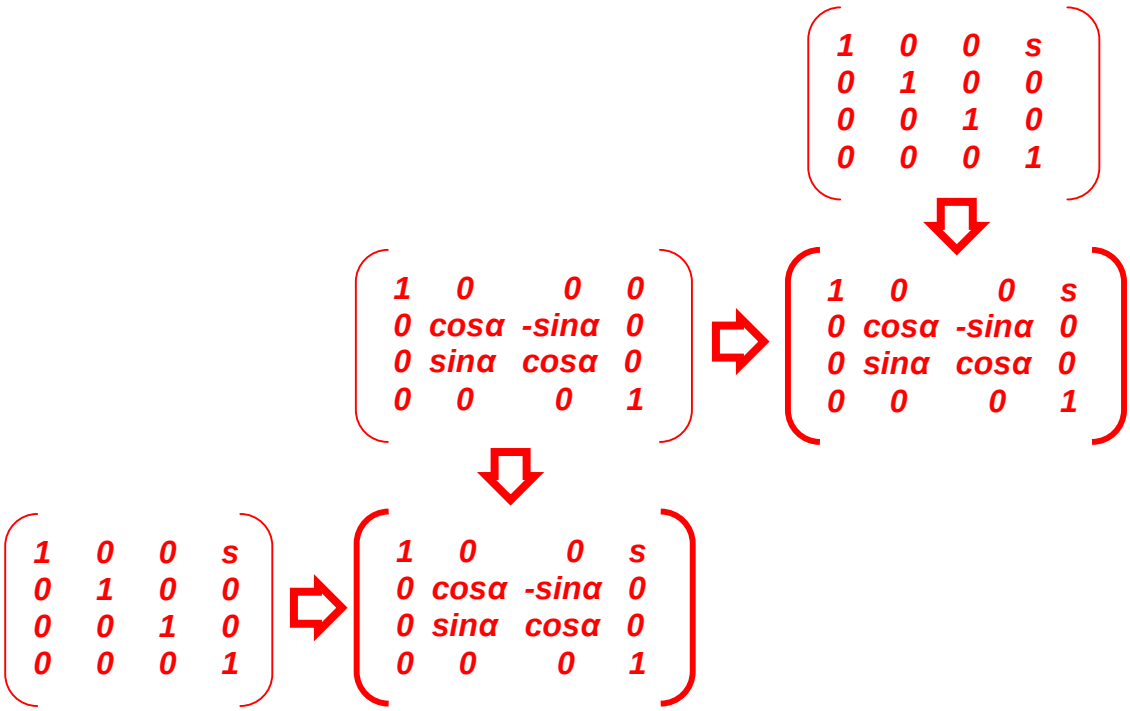
(Im folgenden können Sie die Ausdrücke sinα und cosα auch als sA und cA entsprechend abkürzen.)

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

d) Wie lautet die Transformationsmatrix $\underline{T}_{Tx}$ zur Berechnung der Punktkoordinaten des Punktes nach der Translation entlang der x-Achse in Abhängigkeit von der Strecke s?

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

e) Rechnen Sie bitte beide unter a) vorgestellten Varianten aus:



f) Erklären Sie bitte (möglichst kurz), anhand welcher Regel / Beobachtung der Koordinatentransformationen die Beziehung zwischen den beiden Ergebnissen unter e) hätten vorausgesagt werden können:

Rotation von Punkten um eine (hier: x-)Achse verändert nicht die entsprechende (x-) Koordinate der rotierenden Punkte; Translation entlang dieser (x-)Achse verändert nur diese (x-) Koordinate. Folglich bleibt die eine Transformation unabhängig von der anderen; deshalb hat die Reihenfolge keinen Einfluß auf das Ergebnis.

3. Aufgabe (45 Punkte)

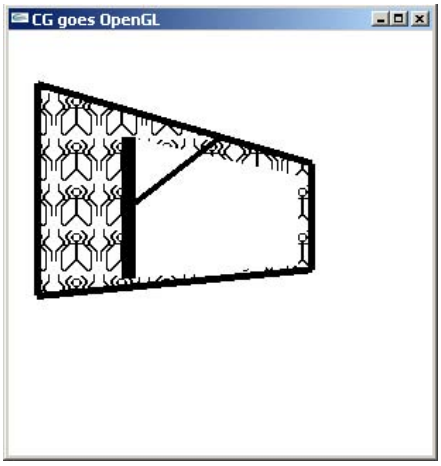


Abb. 3.1

Für einen Hersteller von Scheibenwischern für die Automobil-Industrie haben Sie mit OpenGL eine kleine Animation vorbereitet, die auf lustige Weise (Abb. 3.1) die Qualität seiner Produkte zeigt. Der dazugehörige C-Code ist am Ende dieser Aufgabe wiedergegeben.

Anhand der nachfolgenden Fragen wollen Sie bitte Aufbau und Funktionsweise des Programms analysieren und erläutern:

a) Das Programm funktioniert als Endlos-Schleife, die über die Tastatur gesteuert wird. An welcher Stelle im Code wird diese Schleife gestartet? (Nennung der entsprechenden Zeile/n genügt.)

Das passiert mit der Anweisung `glutMainLoop()` (in `main()`)

b) Die Funktion `init()` initialisiert das Programm, indem sie u.a. (im Feld `screen[][]`) die Eckpunkt-Koordinaten der darzustellenden Windschutzscheibe setzt. Diese Werte werden in der Funktion `draw()` benötigt.

Welcher Teil der Software ruft zu welchem Anlaß `draw()` auf? Woran erkennt man das? (Kurze Erläuterung)

Die Funktion `draw()` wird von GLUT bei jeder Fenster-Auffrischung aufgerufen; sie wurde über den Aufruf `glutDisplayFunc(draw)` als Fenster-Callback an GLUT gemeldet.

Die Funktion `draw()` ist gebrauchsfertig. Behandeln Sie bitte die folgenden Fragen zu ihrer Arbeitsweise:

c) Entgegen dem allgemeinen Wissen, daß Fenster für OpenGL mit Koordinaten $-1. \leq x,y \leq +1.$ arbeiten, werden hier offenbar Koordinaten-Werte (Windschutzscheibe) dargestellt, die deutlich außerhalb dieses Zahlenintervalls liegen. Mit welcher Anweisung (Nennung genügt) wurde dem System mitgeteilt, für welche Dimensionen das Fenster ausreichen muß?

(Antwort ⇒)

```
glFrustum (-WID, WID, -WID, WID, nah, fern);
```

Noch vor Vereinbarung der genutzten Farben finden in `draw()` vier Koordinaten-Transformationen statt. Beantworten Sie dazu bitte folgende Fragen:

d) Wie wirkt die Transformation `glTranslatef(0.,0.,-(nah+WID/2))`? Verlegt sie das Objekt (vom Augenpunkt aus betrachtet) weiter nach links, nach rechts, nach oben, nach unten, näher, weiter weg, oder um eine (welche?) Kombination aus einzelnen dieser Richtungen? (Angabe genügt.)

Sie positioniert das Objekt weiter vom Augenpunkt entfernt.

e) Die Funktion `draw()` enthält drei Rotationen. Erklären Sie bitte am Beispiel des Aufrufs `glRotatef(angle[Z],1.,0.,1.)` , um welche Achse das Objekt gedreht wird:

Um eine Achse, die durch den Koordinaten-Ursprung (0,0,0) und den Punkt (1,0,1) führt.

Wann ist die Drehrichtung dieser Transformation positiv?
(Im oder gegen den Uhrzeigersinn, wenn man von wo aus wohin schaut?)

Die Drehrichtung ist positiv gegen den Uhrzeigersinn, wenn man vom Punkt (1,0,1) auf den Ursprung schaut.

f) Sie wollen überprüfen, ob `draw()` so funktioniert, wie Sie es gedacht haben, und wollen von Hand die transformierten Objekt-Koordinaten ausrechnen. Dazu fassen Sie zunächst, für eine ausgewählte Positionierung des Objekts, die Translationsmatrix (die Sie $\underline{\mathbf{T}}_T$(WID) nennen) und drei Rotationsmatrizen (denen Sie die Namen $\underline{\mathbf{T}}_R$(X), $\underline{\mathbf{T}}_R$(Y), $\underline{\mathbf{T}}_R$(Z) geben) zur Gesamt-Transformationsmatrix $\underline{\mathbf{T}}_{gesamt}$ zusammen. Wie berechnen Sie $\underline{\mathbf{T}}_{gesamt}$ aus den o.a. einzelnen Transformationsmatrizen?

$\underline{\mathbf{T}}_{gesamt} = \underline{T}_T(WID) \bullet \underline{T}_R(X) \bullet \underline{T}_R(Y) \bullet \underline{T}_R(Z)$

In `draw()` werden mehrmals Farben vereinbart. Beantworten Sie bitte dazu folgendes:

g) OpenGL verwendet stets eine aktuelle (Fenster-) Löschfarbe und eine Zeichenfarbe. Mit welchen Anweisungen werden diese Farben jeweils eingestellt? (Namen der Anweisungen genügen.)

Löschfarbe: ***glClearColor();***

Zeichenfarbe: ***glColor3f();***

h) Welche der beiden aktuellen (Lösch-, Zeichen-) Farben wird verwendet, wenn eine einfarbige Fläche gezeichnet wird?

Die Zeichenfarbe

i) Wie werden die beiden eingestellten (Lösch-, Zeichen-) Farben verwendet, wenn eine einfarbige Fläche mit der Stipple-Technik gemustert wird?

Farbe des Musters: ***Zeichenfarbe***

Farbe der restlichen Fläche: ***Löschfarbe***

j) Bei einem Programmablauf ohne Stippling erhält man das Bild nach Abb. 3.2; erwartet wäre dagegen ein Aussehen nach Abb. 3.3. Wo und wie kann man den vorliegenden Code abändern, um das gewünschte Erscheinungsbild zu erreichen?

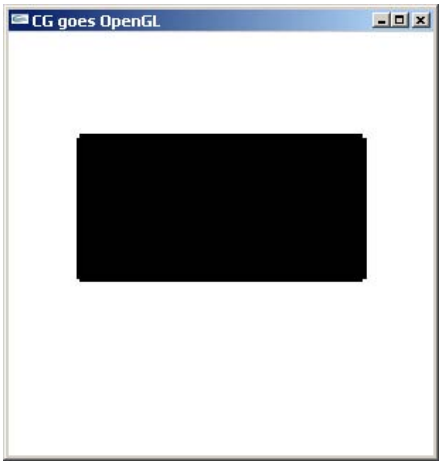


Abb.3.2

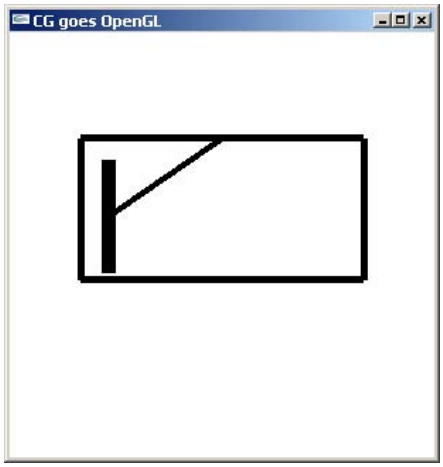


Abb.3.3

(Geben Sie bitte die benötigte/n Anweisung/en an und die Stelle, an der die Änderung vorzunehmen wäre – wenn Sie wollen, im Code.) (Antwort ⇒)

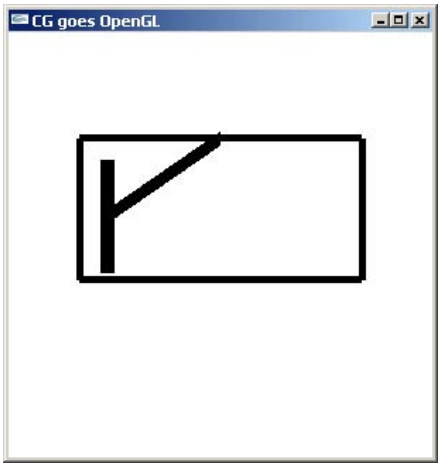
```
glDisable(GL_POLYGON_STIPPLE); glColor3f(1.,1.,1.);
```

k) Der Funktion `wiper()` (engl. für Scheibenwischer) wird am Ende von `draw()` aufgerufen; sie bestimmt die aktuell gewischte Fläche als Teil der Windschutzscheibe mit fester (Teil-) Höhe und variablen Grenzen in der Breite; diese werden in `init()` initialisiert. Wie wird die Wirkung des Scheibenwischers in `wiper()` erzeugt?

	Als flächige Löschung eines Teils der Musterung auf der Windschutzscheibe (durch Anwendung der Löschfarbe).
X	Als darüber gezeichnete, eigenständige Fläche.
	Als Kennzeichnung eines Fensterteils, in dem kein Fliegen-Muster gezeichnet wird.
	Als Kennzeichnung einer Fläche, die zwar Polygonzüge, aber keine Muster enthalten darf.

l) Der Wischer und der dazugehörige Hebel werden mit Linien unterschiedlicher Breite gezeichnet. Faßt man aber den Code für beide Linien zusammen (durch Auslassung des ersten schließenden `glEnd()` und des darauffolgenden `glBegin()`), werden beide Linien mit gleicher Strichstärke gezeichnet (Abb. 3.4): Die Umstellung der Strichstärke wird ignoriert.

Mit welchem Nutzungshinweis (OpenGL-Regel) werden OpenGL-Programmiererinnen vor solchen Fehlern gewarnt? Abb.3.4



„Nur wenige Befehle sind zwischen `glBegin()` und `glEnd()` wirksam“; `glLineWidth()` gehört nicht zu ihnen.

m) Für ein „futuristisches“ Modell wollen Sie den Hebel nicht mehr mittig, sondern am oberen Ende des Scheibenwischers ansetzen lassen (Abb. 3.5).

Welche Code-Stelle muß dazu in welcher Weise geändert werden?

(Antwort: bitte nächstes Blatt)

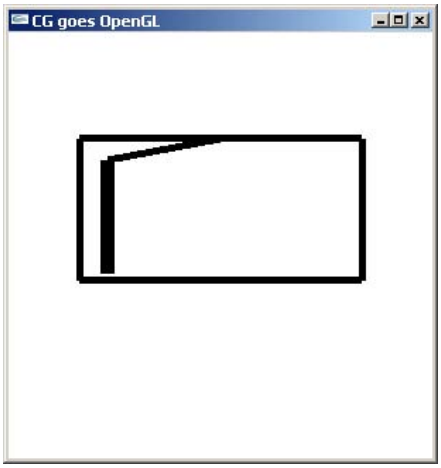


Abb. 3.5

Lösung (z.B.):

```
glVertex3f(wiped[1][X], wiped[2][Y], wiped[1][Z]);
```

anstelle von :

```
glVertex3f(wiped[1][X], (wiped[1][Y]+wiped[2][Y])/2,
wiped[1][Z]);
```

n) Sie wollen sich vergewissern, daß das Programm das leistet, was Sie von ihm erwarten, und daß keine unerwarteten Effekte zu befürchten sind. Nach kurzer Testung stellen Sie fest, daß bei Eintritt in die Funktion `wiper()` das Stippling noch aktiv ist und als aktuelle Belegung das Fliegen-Muster hat.

Erklären Sie bitte mit wenigen Worten,

(i) wieso von diesem (nicht gezeichneten? nicht sichtbaren?) Muster nichts zu erkennen ist

(ii) wieso nicht wenigstens das alte (durch die Funktion `draw()` gezeichnete) Muster auf der Fläche geblieben war.

i) Die am Anfang der Funktion `wiper()` eingestellte Farbe ist Weiß (`glColor3f(1.,1.,1.);`); das bewirkt, daß das Fliegen-Muster mit weißer Farbe vor weißem Hintergrund gezeichnet wird und somit nicht erkennbar ist.

ii) Aufgrund der bitgenauen Alignierung wird das neue Muster pixelgenau über das schwarze Muster gelegt, das von `draw()` gezeichnet wurde. Deshalb wird das alte Muster durch das neue („unsichtbare“) vollkommen gelöscht.

```
/*Darstellung eines Scheibenwischers mit OpenGL*/
#include <stdio.h> //wg. printf()
#include <GL/glut.h>

#define CON_CLS  "cls"
#define ESC        27
#define CR         13
#define PART         .8f
#define WID         4
#define HIG         3
#define WIP_0     -PART*WID
#define WIP_N      PART*WID

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

/*Globale Variablen: */
float    wip0, wipN, dWip=WID*.1f, angle[3]={0.,0.,0.};
int      stipple=1;
GLdouble nah=WID, fern=10.;
GLfloat  screen[4][3], wiped[4][3];
GLubyte fly[] = { /*... von OpenGL vorgestellt ...*/
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      0x03, 0x80, 0x01, 0xC0, 0x06, 0xC0, 0x03, 0x60,
      0x04, 0x60, 0x06, 0x20, 0x04, 0x30, 0x0C, 0x20,
      0x04, 0x18, 0x18, 0x20, 0x04, 0x0C, 0x30, 0x20,
                  /*... hier verkuerzt ...*/
      0x06, 0x64, 0x26, 0x60, 0x0c, 0xcc, 0x33, 0x30,
      0x18, 0xcc, 0x33, 0x18, 0x10, 0xc4, 0x23, 0x08,
      0x10, 0x63, 0xC6, 0x08, 0x10, 0x30, 0x0c, 0x08,
      0x10, 0x18, 0x18, 0x08, 0x10, 0x00, 0x00, 0x08};

/*************************************************************************/
  void wiper(void)
/*************************************************************************/
{ /*Zu wischende Flaeche:*/
  wiped[0][X] = wiped[3][X] = wip0;
  wiped[1][X] = wiped[2][X] = wipN;
  wiped[0][Y] = wiped[1][Y] = PART*screen[0][Y];
  wiped[2][Y] = wiped[3][Y] = PART*screen[3][Y];
  wiped[0][Z] = wiped[1][Z] = wiped[2][Z] = wiped[3][Z] = screen[0][Z];

  glColor3f (1., 1., 1.);
  glBegin(GL_POLYGON);
    glVertex3fv(wiped[0]); glVertex3fv(wiped[1]);
    glVertex3fv(wiped[2]); glVertex3fv(wiped[3]);
  glEnd();

  /*Wischer und Hebel:*/
  glColor3f (0., 0., 0.);
  glLineWidth (15.);
  glBegin(GL_LINES);
    glVertex3fv(wiped[1]); glVertex3fv(wiped[2]);
  glEnd();

  glLineWidth (5.);
  glBegin(GL_LINES);
    glVertex3f((screen[2][X]+screen[3][X])/2, screen[2][Y], screen[2][Z]);
    glVertex3f(wiped[1][X], wiped[2][Y], wiped[1][Z]);
  //glVertex3f(wiped[1][X], (wiped[1][Y]+wiped[2][Y])/2, wiped[1][Z]);
  glEnd();
}
```

```
/*****************************************************************************/
  void draw(void)
/*****************************************************************************/
{ fern=nah+2.5*WID;

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

  glClear(GL_COLOR_BUFFER_BIT);

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

  /*Alles ins Sichtvolumen verschieben:*/
  glTranslatef(0., 0., -(nah+WID/2));

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

  /*Farben:*/
  glClearColor (1., 1., 1., 1.);
  glColor3f (0., 0., 0.);

  /*Windschutzscheibe:*/
  if (stipple)
  { glEnable  (GL_POLYGON_STIPPLE);  glPolygonStipple (fly);
  }  else
  { glDisable (GL_POLYGON_STIPPLE);  glColor3f (1., 1., 1.);
  }
  glBegin(GL_POLYGON);
    glVertex3fv(screen[0]); glVertex3fv(screen[1]);
    glVertex3fv(screen[2]); glVertex3fv(screen[3]);
  glEnd();

  /*Rahmen Windschutzscheibe:*/
  glColor3f (0., 0., 0.);
  glLineWidth (5.);
  glBegin(GL_LINE_LOOP);
    glVertex3fv(screen[0]); glVertex3fv(screen[1]);
    glVertex3fv(screen[2]); glVertex3fv(screen[3]);
  glEnd();

  /*Scheibenwischer:*/
  wiper();

  glFlush();
}
```

```
/*************************************************************************/
  void key(unsigned char key, int x, int y)
/*************************************************************************/
/*Menue und Eingabe-Behandlung:*/
{ switch (key)
  { case ESC: exit(0);
    case CR : wipN += dWip;
              if (dWip > 0. && wipN >= WIP_N)
              { wip0 = wipN = WIP_N;  dWip = -dWip; }
              if (dWip < 0. && wipN <= WIP_0)
              { wip0 = wipN = WIP_0;  dWip = -dWip; }             break;
    case 't': stipple = 1-stipple;                                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 <CR>    wipe windscreen");
  printf ("\n\r t         toggle Stippling ");
  if (stipple) printf ("(ON)"); else printf ("(OFF)");
  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                angle[X]=%7.2f", angle[X]);
  printf ("\n\r                angle[Y]=%7.2f", angle[Y]);
  printf ("\n\r                angle[Z]=%7.2f", angle[Z]);
  draw();
  return;
}
/*************************************************************************/
  void init(void)
/*************************************************************************/
{ /*Eckpunkt-Koordinaten:*/
  screen[0][X] = screen[3][X] = -WID;   //WID=4
  screen[1][X] = screen[2][X] =  WID;
  screen[0][Y] = screen[1][Y] = -HIG/2; //HIG=3
  screen[2][Y] = screen[3][Y] =  HIG;
  screen[0][Z] = screen[1][Z] = screen[2][Z] = screen[3][Z] = 0.f;

  wip0 = wipN = PART*screen[0][X];

  /*Tastendruck simulieren, um Menue auszugeben:*/
  key(' ', 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: