Technische Hochschule Mittelhessen

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

2. Oktober 2013

Klausur
Computergrafik
für Bachelor-Studierende
SS 2013

– 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 (15 Punkte)

a) Sie lernen jemanden kennen, der als Informatiker mit Erfahrung auf dem Gebiet der Datenbanken jahrzehntelang als Entwickler im Dienst der internationalen Flugsicherheit gearbeitet hat. Sie erfahren, daß er berühmt geworden ist, weil er ein Verfahren entwickelt hat, mit dem durch Nutzung von Symmetrien Phantombilder auch aus halbverdeckten Bildern von Überwachungskameras automatisch angefertigt werden können.

Handelt es sich bei diesem großen Datenbank-Fachmann gleichzeitig um einen Experten für Bildbearbeitung, für Bildverarbeitung, für Computergrafik, für alle drei oder für keine von ihnen?

Bitte begründen kurz Sie Ihre Antwort!

Er ist Experte für Bildverarbeitung: Ausgehend von einem Bild gelangt er zu seiner Beschreibung / seinen Merkmalen. (Ggf. auch: Experte für Computergrafik, weil er aus Beschreibungen Bilder erzeugt.)

b) J. E. Bresenham leitet seinen Linienalgorithmus ab, indem er die Pixel einer digitalisierten Darstellung auf mathematische Punkte zurückführt. Als den „Stellvertreter" jedes (zweidimensionalen) Bildpunktes betrachtet er hierbei den (dimensionslosen) Punkt

	… an der oberen linken Ecke …
	… am Mittelpunkt der oberen Kante …
X	… im Schnittpunkt der Diagonalen …
	… an einer beliebigen Stelle innerhalb …

…jedes Pixels. (Zutreffendes bitte ankreuzen!)

c) Welche der folgenden Aussagen über Augenpunkt und Projektionszentrum sind richtig:

X	Sie sind identisch.
	Die geradlinige Verbindung zwischen ihnen steht immer senkrecht zur Blickrichtung.
	Die geradlinige Verbindung zwischen ihnen liegt parallel zur Blickrichtung.
X	Die Verbindungsgerade von dort zu einem Objektpunkt schneidet die Projektionsebene in der Abbildung des Objektpunktes.

d) Viele mathematische Verfahren unterscheiden nicht zwischen Vektoren und Punkten. Welcher grundsätzliche Unterschied (bzw.: Widerspruch) zwischen den beiden Begriffen bleibt dabei unberücksichtigt, dem die Einführung von „Coordinate Frames" Rechnung trägt?

⇒

Vektoren haben Betrag und Richtung, aber keine Position. Punkte haben Position, aber weder Betrag, noch Richtung.

2. Aufgabe (35 Punkte)

Sie wollen einen virtuellen Billardtisch implementieren. Bei einem Koordinatensystem, das in der Mitte der Spielfläche installiert sein soll mit den x- und y-Achsen parallel zu den Tischkanten, wollen Sie vorrechnen, wie eine Kugel, deren Mittelpunkt die Koordinaten (-a, -b, 0) hat, die Bande am Punkt (c, d, 0) trifft, dort abprallt und eine ebensolange Strecke zurücklegt, bevor sie zum Stehen kommt (a, b, c, d > 0).

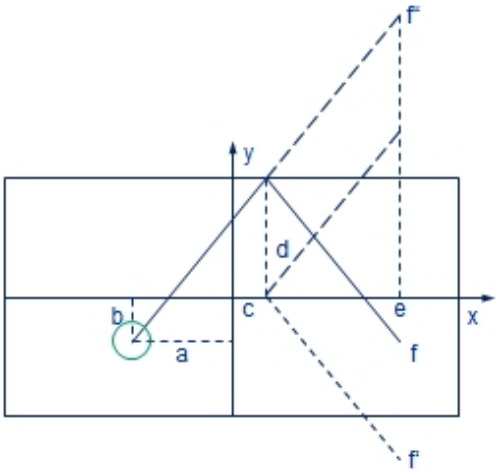


Abb. 2.1

Sie beabsichtigen, folgenden Lösungsansatz zu realisieren (vgl. Abb. 2.1):

Zunächst wird die Kugel mit einer Translation (Transformation $\underline{T_1}$) an einen Punkt bei (e, f‘, 0) verschoben, der von ihrer Ausgangslage doppelt so weit entfernt ist wie der Punkt bei (c, d, 0).

Dann wird mit einer weiteren Translation (Transformation $\underline{T_2}$) der Punkt bei (c, d, 0) auf die x-Achse verlegt.

Anschließend werden die Koordinaten mit dem Skalierungsfaktor -1 an der x-Achse gespiegelt (Transformation $\underline{T_3}$), so daß danach der Haltepunkt der Kugel bei den Koordinaten (e, f‘, 0) liegt.

Am Ende werden mit einer letzten Translation (Transformation $\underline{T_4}$) die Koordinaten der Kugel an den gesuchten Haltepunkt bei (e, f, 0) verschoben.

Lösen Sie bitte diese Aufgabe unter Verwendung räumlicher homogener Koordinaten, indem Sie nacheinander folgende Fragen behandeln:

a) Welche Transformationsmatrix $\underline{T_1}$ verschiebt den Kugelmittelpunkt von (-a, -b, 0) zum Punkt bei (e, f‘, 0), der doppelt so weit entfernt liegt wie der Punkt bei (c, d, 0)?

$$\underline{T_1} = \begin{pmatrix} 1 & 0 & 0 & 2(a+c) \\ 0 & 1 & 0 & 2(b+d) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

b) Wie lautet die Transformationsmatrix $\underline{T}_2$, die den Punkt bei (c, d, 0) auf die x-Achse verschiebt?

$$\underline{T}_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

c) Geben Sie bitte die Skalierungsmatrix $\underline{T}_3$ an, mit welcher der Haltepunkt der Kugel an die Koordinaten (e, f', 0) gespiegelt wird.

$$\underline{T}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

d) Welche Translationsmatrix $\underline{T}_4$ führt zum gesuchten Haltepunkt bei (e, f, 0)?

$$\underline{T}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

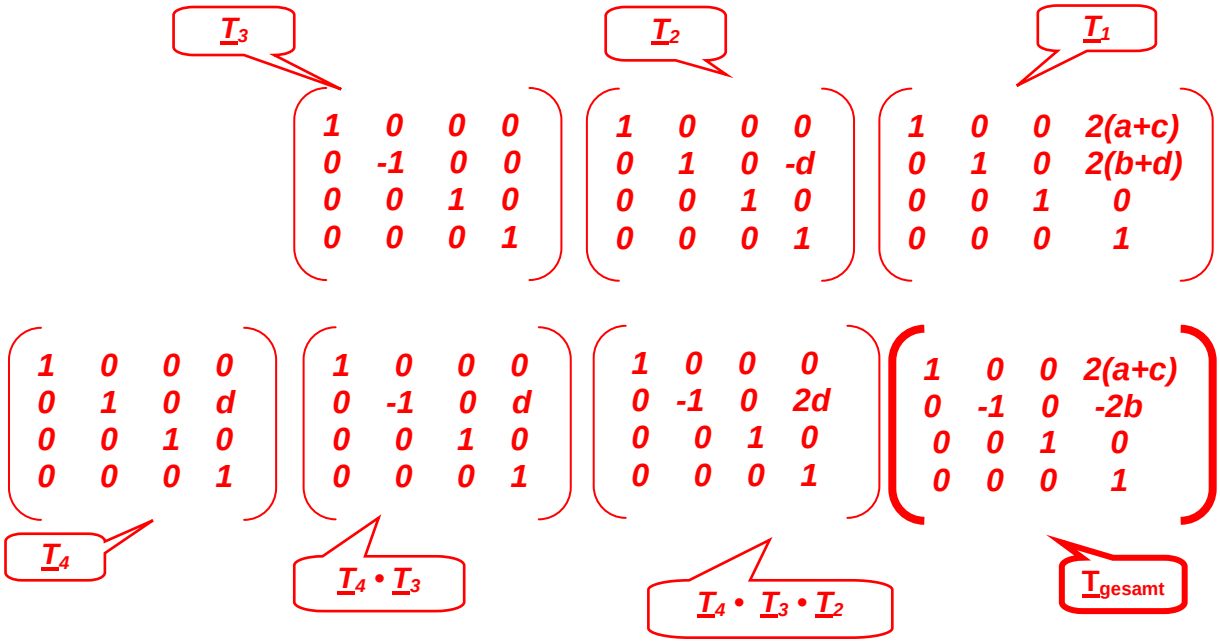
e) Wie berechnet sich die gesuchte Transformationsmatrix $\underline{T}_{gesamt}$ für die Koordinaten des Kugelmittelpunkts am Haltepunkt aus den oben besprochenen Transformationen $\underline{T}_1$ bis $\underline{T}_4$?

$$\underline{T}_{gesamt} = \underline{T}_4 \bullet \underline{T}_3 \bullet \underline{T}_2 \bullet \underline{T}_1$$

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

⇒

$$T_{gesamt} = T_4 \cdot T_3 \cdot T_2 \cdot T_1$$



g) Berechnen Sie nun bitte auch die Koordinaten des Kugelmittelpunkts in der Endstellung.

$$\begin{pmatrix} 1 & 0 & 0 & 2(a+c) \\ 0 & -1 & 0 & -2b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -a \\ -b \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a + 2c \\ -b \\ 0 \\ 1 \end{pmatrix}$$

3. **Aufgabe** (50 Punkte)

In der Planungsphase einer Automobilausstellung beraten Sie Lkw-Hersteller und haben dazu ein provisorisches Visualisierungsprogramm entwickelt (s. Abb. 3.1), das ein rudimentäres Modell zeigt. Der Code hierzu (`ShinyGround.h`, `ShinyGround.h`) ist am Ende dieser Aufgabenblätter enthalten.

Für interessierte Teilnehmer wollen Sie im folgenden die Programmeigenschaften und Ihre Vorgehensweise erklären:

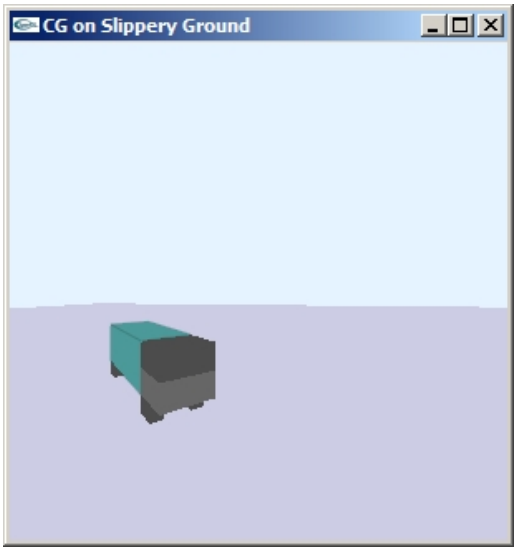


Abb. 3.1

a) Das GLUT-Fenster wird in `main()` eingerichtet. Beim Programm-Start erscheint es in der oberen linken Bildschirm-Ecke. Geben Sie bitte eine Anweisung an, mit welcher es jeweils um 200 Pixel nach rechts und nach unten versetzt dargestellt würde:

glutPositionWindow(200,200); – oder
glutInitWindowPosition(200,200);

b) In `main()` wird u.a. die Funktion `init()` aufgerufen; darin werden die Dimensionen zweier doppelt indizierter globaler Variablen initialisiert. Bei näherem Hinsehen erkennt man, daß sich der Fußboden (Variable `field[][]`) in der …

X	… x-y-Ebene …
	… y-z-Ebene …
	… x-z-Ebene …

… aufspannt (Zutreffendes bitte ankreuzen).

Das später öfter genutzte Modul (Variable `box[][]`) dagegen stellt einen Würfel dar (Zutreffendes bitte ankreuzen), …

	… der symmetrisch um den Koordinatenursprung liegt und seine Kanten parallel zu den Hauptachsen hat.
X	… der einen Eckpunkt am Koordinatenursprung und seine Kanten an den positiven x-, y- und z-Achsen hat.
	… der einen Eckpunkt am Koordinatenursprung und seine Kanten an den negativen x-, y- und z-Achsen hat.
	… der einen Eckpunkt am Koordinatenursprung, je eine Kante an den positiven x- u. z-Achsen sowie eine Kante an der negativen y-Achse hat.

c) Die Funktion `draw()` steuert gewissermaßen die gesamte grafische Darstellung. Wie kann man als Nutzer(in) das ausführbare Programm so bedienen, daß man den planmäßigen Aufruf von `draw()` überprüfen (d.h.: auslösen) kann?

Verdeckung und erneute Freigabe des GLUT-Ausführungsfensters muß einen `draw()`–Aufruf auslösen.

d) Ebenfalls in der Funktion `draw()` wird unmittelbar nach der Positionierung des Augenpunktes (Aufruf von `positEye()`) die Positionierung des Bodens unter dem Lkw vorgenommen. In der dafür zuständigen Funktion `positField()` finden mehrere geometrische Transformationen (`glTranslatef()`, `glRotatef()`) statt.

Wodurch wird sichergestellt, daß diese Transformationen die nachfolgenden grafischen Objekte und nicht etwa das Projektionszentrum betreffen? (Nennung der Maßnahme genügt.)

Durch den Aufruf `glMatrixMode(GL_MODELVIEW)`.

e) Weiter unten in der Funktion `draw()` erfolgt, durch Aufruf von `positVan()`, die Positionierung des Lkw. Gelten in `positVan()` die vorausgegangenen Transformationen für den Boden immer noch, oder handelt es sich um die Berechnung einer neuen, unabhängigen Lage im Raum?

Falls die Transformationen von `positField()` weiter gelten: Wie kann man sie ungültig machen?

Falls sie in `positVan()` nicht gelten: Wie kann man die Transformationen in `positVan()` zur Fortsetzung jener in `positField()` machen?

Die Transformationen von `positField()` gelten weiter. Um dies aufzuheben, muß man den Aufruf `glLoadIdentity()` einsetzen.

f) Als besonderes „Feature“ bietet Ihr Programm die Möglichkeit, den glänzenden Boden der geplanten Ausstellung so zu simulieren, daß die Spiegelung des Lkw darin dargestellt wird (Abb. 3.2). Dies geschieht ebenfalls In der Funktion `draw()`.

Kreuzen Sie bitte unten die Ebene an, an der die Spiegelung stattfindet:

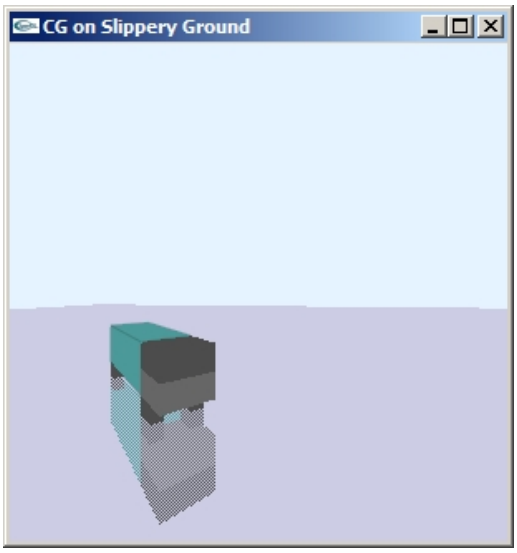


Abb. 3.2

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

Welche Anweisung bekräftigt Ihre Aussage über die Spiegelebene?

```
glScalef (1., 1., -1.);
```

g) Wie ebenfalls in der Funktion `draw()` festzustellen ist, wird die Spiegelung durch die Technik des Stippling realisiert. Die dazu genutzten Daten sind in der indizierten Variablen `stipple[]` einzusehen (s. Datei `ShinyGround.h`).

Wieviel Prozent der ursprünglichen Objektfarbe werden bei der angegebenen Variablen verwendet? (Angabe genügt.)

___50___ %

h) Die eingesetzte Musterung (Stippling) ist auf dem Fußboden nicht bzw. kaum sichtbar. Nach eingehender Untersuchung der Funktion `draw()` erkennt man, woran das liegt, nämlich (Zutreffendes bitte ankreuzen):

	Der vorliegende Ausdruck ist nicht fein genug; ein Ausdruck mit mehr Grautönen wäre nötig.
	Die verwendeten Farben für den Fußboden und das Muster sind zwar verschieden; aber ihre Schwarz-Weiß-Wiedergabe führt zu nahezu demselben Grauton.
	Die Farben von Boden und Muster wären auch bei Farbdarstellung nur schwer zu unterscheiden.
X	Der Fußboden ist außerhalb der Spiegelung nicht gemustert.

Können Sie Ihre Antwort zur Bodenmusterung durch Anweisung(en) im Quellcode belegen?

Wenn ja: Durch welche?

Wenn nein: Welches Vorwissen führt Sie zu dieser Aussage?

⇒

Ja. In der Funktion wird die Musterung nach der Zeichnung der Spiegelung mit der Anweisung `glDisable (GL_POLYGON_STIPPLE)` abgestellt.

i) Dem Quellcode entnimmt man unschwer, daß das Lkw-Modell und sein Spiegelbild durch zweimaligen Aufruf von `drawVan()` gezeichnet werden; dort werden zuerst die vier Räder an die entsprechenden Stellen kopiert. Der Inhalt der `for`-Schleife ist zwischen den Anweisungen `glPushMatrix()` und `glPopMatrix()` eingeschlossen. Dies geschieht (zutreffende Aussagen bitte ankreuzen),

X	damit jedes Rad individuell positioniert wird.
	damit nicht durch unnötige Transformationen der Heap überläuft.
	damit die Zeichnung des Lkw-Spiegelbilds vorbereitet wird.
X	damit sich die Transformation jedes einzelnen Rades nicht auf die übrigen überträgt.

j) Unmittelbar nach der Verschiebungsanweisung (`glTranslatef()`) für die Räder wird in `drawVan()` eine Skalierung ausgeführt (`glScalef()`). Das hat zur Wirkung (Zutreffendes bitte ankreuzen),

	daß die Räder erst passend positioniert werden, bevor skaliert wird.
X	daß die Räder erst passend skaliert werden, bevor positioniert wird.

Begründen Sie bitte kurz Ihre Aussage:

In OpenGL werden Transformationen durch Matrizenmultiplikation von rechts ausgeführt. Deswegen stehen die logisch (physisch) zuerst ausgeführten Transformationen unten im Quellcode und umgekehrt.

k) Die Kontur des Lkw-Laderaum (Lkw-Containers) wird mit einer Liniengrafik nachgezeichnet. Welche Strichstärke wird dabei angewandt, und welche Technik wird durch die Aufrufe `glEnable(GL_LINE_SMOOTH)` und `glEnable(GL_BLEND)` aktiviert?

Die gewählte Strichstärke ist 0,5.

Die o.a. Aufrufe aktivieren das (Linien-) Antialiasing.

l) Die Farbe des Lkw-Containers ist durch die globale Variable `color[]` festgelegt. Kreuzen Sie bitte jene Beschreibung an, die am ehesten zu dieser Farbe paßt:

`color[]`	**Beschreibung**
	Sehr helles Cyan (Türkis)
	Sehr helles Grau, fast Weiß
	Ungesättigtes, helles Braun, fast Gelb
X	Dunkles Cyan (Türkis) mit leichter Neigung zu Oliv
	Rot mit etwas Blau (Magenta)
	Stark ungesättigtes Rot, wie Erdbeermilch

m) Die Linienfarbe, mit welcher der Umriß des Lkw-Containers nachgezeichnet wird, ist durch die globale Variable `linecolor[]` festgelegt. Was können Sie anhand der eingesetzten Zahlenwerte über die Helligkeit und den Ton dieser Farbe im Vergleich zu `color[]` sagen? (Kurze Begründung!)

Die niedrigeren Werte weisen im Vergleich zu `color[]` auf eine dunklere Farbe hin.

Der Farbton ist etwa der gleiche, weil die Zahlenwerte im selben Verhältnis zueinander stehen.

n) In `main()` und in den später aufgerufenen Funktionen wird der Tiefenpuffer (…_DEPTH_…) aktiviert, was zusätzliche Rechenzeit kostet. In OpenGL-Programmen wird das vermieden, wenn nur zwei Objekte darzustellen sind, deren Reihenfolge man im Quellcode bestimmen kann.
Würde sich an der Abbildung etwas ändern, wenn man den Tiefenpuffer deaktivieren würde? (Wenn ja: Was?)

Es werden hier weit mehr als zwei Objekte dargestellt – nicht nur, weil die Lkw-Spiegelung getrennt zu zählen ist: Der Lkw (wie auch seine Spiegelung) besteht aus mehreren Modulen, die sich gegenseitig verdecken. Deaktivierung des Tiefenpuffers würde zu unkontrollierten Verdeckungen unter den einzelnen Modul-Instanzen führen.

```
/* ShinyGround.h */

#ifndef RAINROAD_H
  #define RAINROAD_H
  #include <stdio.h> //wg. printf()
  #include <math.h>  //wg. sin()
  #include <GL/glut.h>

  #define CON_CLS  "cls"
  #define ESC        27
  #define FLD_X      50
  #define FLD_Y      50
  #define VAN_X    1/3.f
  #define VAN_Y    1
  #define VAN_Z    1/3.f

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

  /*Globale Variable:*/
  GLubyte stipple[] = {
      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};

  /*Prototypen:*/
  void draw (void);
  void drawField (void);
  void drawModule (void);
  void drawVan (int linFlag);
  void init (void);
  void key (unsigned char key, int x, int y);
  void positEye (void);
  void positField (void);
  void positVan (void);
#endif //RAINROAD_H
```

```
/* ShinyGround.c */
/*Darstellung einer verregneten Strasse mit OpenGL*/
#include "ShinyGround.h"

/*Globale Variablen: */
int      mirror=1;
GLint    faces[6][4] = { {0, 1, 2, 3}, {1, 0, 4, 5}, {1, 5, 6, 2},
                         {4, 0, 3, 7}, {3, 2, 6, 7}, {5, 4, 7, 6} };
float    tx=0., ty=0., rz=0., angle[3]={0.,0.,0.},
         box[8][3], Rz=0., Tx=0., Ty=2*VAN_Y, Tz=0.,
         about[3]= {.9f, .95f, 1.f}, black[3]={0.3f,0.3f,0.3f},
         ground[3]={0.8f,.8f,0.9f}, steel[3]={.4f, .4f, .4f},
         color[3]={.3f,.6f,.6f}, linecolor[3]={.2f,.4f,.4f};
GLfloat  field[4][3];
GLdouble nah=VAN_Y/4, fern=2*FLD_Y;
```

```
/*************************************************************************/
  void positEye (void)
/*************************************************************************/
/*Augenpunkt positionieren:*/
{ glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  /*Festlegung des Sichtvolumens:*/
  glFrustum (-VAN_Y, VAN_Y, -VAN_Y, VAN_Y, VAN_Y, FLD_Y);
  /*Aktualisierung der Blickrichtung:*/
  glRotatef(-Rz, 0., 1., 0.);
  /*Konstanter (y-)Tiefstand, aktualisierte (x-,z-)Position:*/
  glTranslatef(-Tx, -Ty, -Tz);
  return;
}

/*************************************************************************/
  void positField (void)
/*************************************************************************/
/*Gelaende positionieren:*/
{ glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  /*Alles ins Sichtvolumen verschieben:*/
  glTranslatef(0., 0., -(2*VAN_Y));
  /*Positionierung:*/
  glRotatef(angle[X], 1., 0., 0.);
  glRotatef(angle[Y], 0., 1., 0.);
  glRotatef(angle[Z], 0., 0., 1.);
  return;
}

/*************************************************************************/
  void drawField (void)
/*************************************************************************/
/*Boden zeichnen:*/
{ glDisable (GL_POLYGON_STIPPLE); glColor3fv (ground);
  glBegin(GL_POLYGON);
    glVertex3fv(field[0]);   glVertex3fv(field[1]);
    glVertex3fv(field[2]);   glVertex3fv(field[3]);
  glEnd();
  return;
}

/*************************************************************************/
  void positVan (void)
/*************************************************************************/
/*Objekt positionieren:*/
{ glTranslatef((field[0][X]+field[3][X])/2.,
               (field[1][Y]+field[0][Y])/2., 0.);
  /*Nach dem Start erreichte Position:*/
  glTranslatef(tx, ty, 0.);
  glRotatef(rz, 0., 0., 1.);
  return;
}

/*************************************************************************/
  void drawModule (void)
/*************************************************************************/
/*Modul zeichnen:*/
{ int jj=0;
  for (jj = 0; jj < 6; jj++)
  { glBegin(GL_POLYGON);
    glVertex3fv(box[faces[jj][0]]);  glVertex3fv(box[faces[jj][1]]);
    glVertex3fv(box[faces[jj][2]]);  glVertex3fv(box[faces[jj][3]]);
    glEnd();
  }
  return;
}
```

```
/*************************************************************************/
  void drawVan (int linFlag)
/*************************************************************************/
/*Auto aus Modulen zusammenstellen:*/
{ int jj=0;

  glMatrixMode(GL_MODELVIEW);
  glEnable(GL_DEPTH_TEST);

  /*Raeder (ab v.li., gegen den UZS):*/
  glColor3fv (black);
  for (jj=0; jj<4; jj++)
  { glPushMatrix();
    if (jj==0) glTranslatef(0.f,         VAN_Y, 0.f); else
    if (jj==1) glTranslatef(0.f,         0.f,   0.f); else
    if (jj==2) glTranslatef(VAN_X*3/4.f, VAN_Y, 0.f); else
    if (jj==3) glTranslatef(VAN_X*3/4.f, 0.f,   0.f);
    glScalef (VAN_X/4.f, VAN_X/2.f, VAN_X/3.f);
    drawModule();
    glPopMatrix();
  }

  glPushMatrix(); //Karosserie auf die Raeder heben
    glTranslatef (0., 0., VAN_X/3.f);

    /*Lade-Container:*/
    glColor3fv (color);
    glPushMatrix();
      glScalef (VAN_X, VAN_Y, VAN_Z);
      drawModule();
      if (linFlag)
      { glColor3fv (linecolor);
        glLineWidth (.5);
        glEnable (GL_LINE_SMOOTH);
        glEnable (GL_BLEND);
        glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        drawModule();
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        glDisable (GL_LINE_SMOOTH);  glDisable (GL_BLEND);
      }
      glPopMatrix();

      /*Fuehrerhaus:*/
      glColor3fv (steel);
      glPushMatrix();
        glTranslatef(0, VAN_Y, 0.);
        glScalef (VAN_X, VAN_Y/3.f, VAN_Z/2.f);
        drawModule();
      glPopMatrix();

      glColor3fv (black);
      glPushMatrix();
        glTranslatef(0, VAN_Y, VAN_Z/2.f);
        glScalef (VAN_X, VAN_Y/3.f, VAN_Z/2.f);
        drawModule();
      glPopMatrix();

    glPopMatrix(); //Karosserie auf die Raeder heben
  glDisable(GL_DEPTH_TEST);
  return;
}
```

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

  /*Positionierung Augenpunkt:*/
  positEye();

  /*Positionierung Gelaende:*/
  positField();
  /*Zeichnung Gelaende:*/
  drawField();

  /*Positionierung Fahrzeug:*/
  positVan();
  /*Zeichnung Fahrzeug:*/
  glScalef (2., 2., 2.);
  drawVan(1);
  /*Spiegelung:*/
  if (mirror)
  { glEnable (GL_POLYGON_STIPPLE);
    glPolygonStipple (stipple);
    glPushMatrix();
      glScalef (1., 1., -1.);
      drawVan(0);
    glPopMatrix();
    glDisable (GL_POLYGON_STIPPLE);
  }

  /*Darstellung ausloesen:*/
  glFlush();
  return;
}

/*************************************************************************/
  void init (void)
/*************************************************************************/
/*Eckpunkt-Koordinaten Gelaende (gegen den UZS, beginnend o.li.):*/
{ field[0][X] = field[1][X] = -FLD_X/2.; //FLD_X=50
  field[2][X] = field[3][X] =  FLD_X/2.;
  field[0][Y] = field[3][Y] =  FLD_Y/2.; //FLD_Y=50
  field[1][Y] = field[2][Y] = -FLD_Y/2.;
  field[0][Z] = field[1][Z] = field[2][Z] = field[3][Z] = 0.f;

  /*Modul-Abmessungen:*/
  box[0][X] = box[3][X] = box[4][X] = box[7][X] = 0;
  box[1][X] = box[2][X] = box[5][X] = box[6][X] = 1;
  box[0][Y] = box[1][Y] = box[4][Y] = box[5][Y] = 0;
  box[2][Y] = box[3][Y] = box[6][Y] = box[7][Y] = 1;
  box[0][Z] = box[1][Z] = box[2][Z] = box[3][Z] = 1;
  box[4][Z] = box[5][Z] = box[6][Z] = box[7][Z] = 0;

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

  /*Backface-Culling fuer Drahtmodell (nicht-default):*/
  glEnable (GL_CULL_FACE);

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

```
/*********************************************************************/
  void key (unsigned char key, int x, int y)
/*********************************************************************/
/*Menue und Eingabe-Behandlung:*/
{ const  double  GRAD  = atan(1.) / 45.;
  switch (key)
  { case ESC: exit(0);
    case 'a': rz += 10.;  if (rz >= 360) rz-=360;                break;
    case 'd': rz -= 10.;  if (rz <=-360) rz+=360;                break;
    case 'm': mirror = 1-mirror;                                 break;
    case 's': ty -= VAN_Y*cos(rz*GRAD); tx += VAN_Y*sin(rz*GRAD); break;
    case 'w': ty += VAN_Y*cos(rz*GRAD); tx -= VAN_Y*sin(rz*GRAD); break;
    case 'A': Rz += 5.;  if (Rz >= 360) Rz-=360;                 break;
    case 'D': Rz -= 5.;  if (Rz <=-360) Rz+=360;                 break;
    case 'S': Tz += VAN_Y*cos(Rz*GRAD); Tx += VAN_Y*sin(Rz*GRAD); break;
    case 'W': Tz -= VAN_Y*cos(Rz*GRAD); Tx -= VAN_Y*sin(Rz*GRAD); break;
    case '+': Ty *= 1.1f;                                        break;
    case '-': Ty /= 1.1f;                                        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 / d     turn van left / right (10 degrees)");
  printf ("\n\r m         toggle mirroring ");
  if (mirror) printf ("(ON)"); else printf ("(OFF)");
  printf ("\n\r w         drive on");
  printf ("\n\r A / D     turn eye 5 deg. l./r. (curr.%7.2f)", Rz);
  printf ("\n\r S / W     go back / forth (curr.%5.1f units)", -2*VAN_Y*Tz);
  printf ("\n\r + / -     increase/decrease eye height (curr.%5.1f units)", Ty);
  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]);
  printf ("\n\r           driving angle=%7.2f", rz);

  glutPostRedisplay();
  return;
}

/*********************************************************************/
  int main (int argc, char **argv)
/*********************************************************************/
{ glutInitDisplayMode (GLUT_DEPTH);
  glutInit(&argc, argv);
  glutCreateWindow("CG on Slippery Ground");
  glutDisplayFunc(draw);
  glutKeyboardFunc(key);
  init();
  glutMainLoop();
  return 0;
}
```

Platz für Notizen: