Fachhochschule Gießen-Friedberg **8. Februar 2011**

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

Klausur
Computergrafik
für Bachelor-Studierende
WS 2010 / 11

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 lernen jemanden kennen, der eine neue Hilfe für Piloten entwickelt: Im Bild einer Kamera, die die Flugroute festhält, erkennt sein Algorithmus besonders schnell den Verlauf des Horizonts, approximiert ihn mit einer Geraden und ermittelt jede Abweichung von der Horizontalen, die ein ziviles Flugzeug und seine Fracht in Gefahr bringen könnte. Sein Konzept funktioniert sowohl mit natürlichen Bildern aus Flugzeugen als auch mit synthetischen Bildern aus Simulatoren.

Ist dieser Sicherheitsexperte ein Fachmann für Computergrafik, für Bildverarbeitung, für Bildbearbeitung, für keine von ihnen, oder für eine Kombination aus einzelnen (dann: welchen) unter ihnen?

Begründen Sie bitte Ihre Antwort!

b) Ihr Bildschirm ist auf eine Auflösung von 1920 x 1200 Pixel eingestellt. Wie viele Pixel leuchten auf, wenn man eine helle Bildschirmdiagonale nach dem Bresenham-Algorithmus zeichnet? (Zahl genügt)

c) Eine Freundin bittet Sie um Rat, weil ihre Bresenham-Implementierung offensichtliche Fehler aufweist, wenn man eine Linie von (-1, -1) nach (0, 1) oder von (0, 1) nach (-1, -1) zieht. Sie schauen sich ihr Programm an und entdecken tatsächlich in der Codierung des 2. Oktanten einen Fehler, den Sie beheben. Was tun Sie dann (mehrere Antworten möglich)?

	Sie suchen weiter, weil erst einer von zwei Fehlern behoben ist.
	Sie betrachten den Fall als erledigt und empfehlen Ihrer Freundin eine hervorragende Vorlesung in Gießen, damit es in Zukunft besser klappt.
	Sie erklären der Freundin, daß der Bresenham-Algorithmus aufgrund seiner Ganzzahl-Arithmetik immer wieder Probleme macht, wenn eine der Startpunkt-Koordinaten null ist; sie braucht aber nichts weiter zu unternehmen.
	Sie erklären ihr, daß der Fall gelöst wäre, wenn sie sich dafür entscheiden sollte, alle Linien programm-intern (notfalls durch Spiegelung) nur noch von links nach rechts ziehen zu lassen.

d) Für grafische Darstellungen auf einem Schuhputz-Automaten sollen Sie ein passendes kleines Display besorgen. Die in Frage kommenden Bildschirme (mit starkem Staubschutz aber wenigen Farben) gibt es in vielen Varianten mit unterschiedlich großen Pixeln. Bekannt ist vorerst nur, daß alle Display-Modelle quadratische Pixel haben, und daß eine Auflösung von 200 x 150 Pixeln gewählt wurde.

Wie viele Pixel wird ein Display insgesamt enthalten?

Als Projekt-Verantwortliche werden Sie ständig von Technikerinnen gefragt, wie groß die erwartete Bildschirmdiagonale sein wird, damit sie die entsprechende Öffnung am Gehäuse einplanen können. Deshalb beschließen Sie, die Abmessungen der (rahmenlosen) Displays in den (noch unbekannten) Pixelmaßen auszurechnen, um nach getroffener Wahl die Diagonale sofort in mm angeben zu können.

Wie lang ist die Diagonale der Displays, gemessen in „display-eigenen" Längeneinheiten?

[Tip: Soweit hilfreich, können folgende Berechnungen genutzt werden:

$15^2=225$; $17^2=289$; $23^2= 529$; $25^2=625$; $33^2=1089$; $35^2=1.225$; $38^2=1.444$; $45^2=2.025$]

Geben Sie mit einem kurzen Ausdruck an, was unter den o.a. Längen-einheiten zu verstehen ist (Länge der Pixeldiagonalen, Flächeninhalt eines Pixels, Display-Kantenlänge o.a.):

e) Sie erstellen ein Animationsprogramm, das ein grafisches Modell um die y-Achse rotieren läßt, und verfolgen die Koordinaten seiner Eckpunkte im Debugger. Welche der Koordinaten jedes Punktes (x_i, y_i, z_i) verändern sich dabei, welche nicht?

Sich verändernde Koordinate(n): .....

Unverändert bleibende Koordinate(n): .....

f) Bei welcher Blickrichtung und welchem Drehsinn sprechen Sie von einer positiven Drehung um eine Achse?

2. **Aufgabe** (45 Punkte)

Mit Ihren Kenntnissen in Computergrafik unterstützen Sie Architekten, die in einem Park eine Skateboard-Rampe planen.

Sie bereiten hierzu eine Animation vor (Abb. 2.1) und wollen darin eine Fahrt darstellen, bei der das Skateboard erst die Rampe hinauffährt, um dann, nahe dem Koordinaten-Ursprung, zu wenden und wieder abwärts zu rollen.

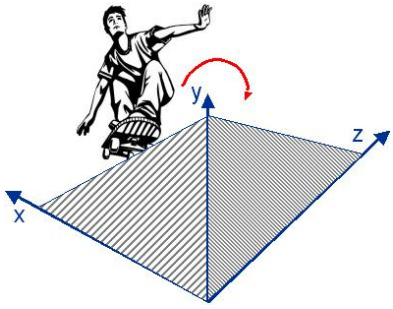


Abb. 2.1

Zur Überprüfung von Zwischenwerten im Debugger berechnen Sie von Hand die Rotation um eine Achse durch den Koordinaten-Ursprung und den Punkt (X, Y, Z) = (10, 6, 8).

Sie stützen sich hierbei auf eine Darstellung nach Abb. 2.2 und führen die folgenden Transformationen aus: Sie drehen die Rotationsachse erst um den Winkel α um die x-Achse, bis sie in der x-y-Ebene liegt, dann um den Winkel β um die z-Achse, bis sie mit der y-Achse zusammenfällt, dann um θ=180° um die transformierte y-Achse; schließlich machen Sie die zwei ersten Transformationen rückgängig.

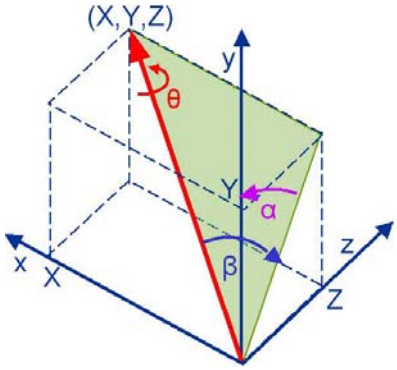


Abb. 2.2

Behandeln Sie bitte die nachstehenden Fragen:

a) Wie berechnen sich (symbolisch, unter Verwendung der Größen X, Y, Z) die Absolutbeträge des Sinus und des Kosinus des Winkels α, um den Sie den Punkt (X, Y, Z) um die x-Achse bis zur x-y-Ebene rotieren lassen?

|sin α| =

|cos α| =

b) Wie berechnen sich (arithmetisch, mit den o.a. Zahlenangaben) und unter Berücksichtigung des Drehsinns die exakten Werte für sin α und cos α?

sin α =

cos α =

c) Wie lautet die Transformationsmatrix, die diese freie Drehachse um die x-Achse dreht, bis sie mit der x-y-Ebene zusammenfällt? Schreiben Sie sie bitte in symbolischer und arithmetischer Form.

$\underline{R}_X(\alpha)$ =

d) Berechnen Sie nun bitte symbolisch die Absolutbeträge des Sinus und des Kosinus des Winkels β, um den Sie den Punkt (X, Y, Z) um die z-Achse rotieren lassen, bis er auf der y-Achse liegt:

|sin β| =

|cos β| =

e) Geben Sie nun bitte, unter Berücksichtigung des Drehsinns, auch die arithmetischen Ausdrücke für die exakten Werte für sin β und cos β an:

[Tip: Wurzeln einstelliger Zahlen, die nicht ganzzahlig sind, brauchen nicht ermittelt zu werden; sie können als Wurzelzeichen weitergeführt werden.]

sin β =

cos β =

f) Wie lautet nun die Transformationsmatrix, die diese freie Drehachse um die z-Achse dreht, bis er auf der y-Achse liegt? Schreiben Sie sie bitte in symbolischer und arithmetischer Form.

$\underline{R}_Z(\beta)$ =

g) Wie lautet die Transformationsmatrix für die Rotation der Drehachse um die transformierte y-Achse gemäß der Aufgabenbeschreibung? Geben Sie bitte ihre symbolische und zahlenmäßige Form an:

$\underline{R}_Y(\theta)$ =

h) Stellen Sie nun bitte die Skateboard-Rotation um die freie Drehachse, $\underline{R}(\theta)$, in symbolischer Form (aus den Symbolen $\underline{R}_X(\alpha)$, $\underline{R}_Z(\beta)$, $\underline{R}_Y(\theta)$) zusammen und vereinfachen Sie sie in dieser Form soweit wie möglich unter Berücksichtigung der Eigenschaften der beteiligten Matrizen. Wählen Sie dabei Umformungen, mit denen sich die Anzahl von Operationen minimieren läßt :

$\underline{R}(\theta)$ =

i) Bonusfrage (3 Sonderpunkte):

Sie codieren die oben zusammengestellte Transformation, lassen damit nacheinander erst ein Auto-, dann ein Skateboard-Modell die Rampe befahren, und bekommen die Bestätigung von kundigen Freunden, daß es sehr realitätsnah wirke. Dann setzen Sie eine menschenähnliche Figur darauf und führen das gleiche Programm vor. Nun meinen dieselben Betrachter, ein Mensch würde nie so stehen.

Was ist damit gemeint, und wie läßt sich der Mangel beheben?

j) Nutzen Sie nun bitte diese Seite, um die Matrix der Gesamttransformation, $\underline{R}(\theta)$, arithmetisch zu ermitteln:

[Tip: Für Matrizenprodukte von mehreren Matrizen nach dem Falkschen Schema kann es platzsparend sein, oben rechts auf dem Blatt zu beginnen und immer weiter von links zu multiplizieren.]

$\underline{R}(\theta)$ =

3. **Aufgabe** (30 Punkte)

Für einen Hersteller von Wandfarben haben Sie mit OpenGL eine kleine Animation vorbereitet, die auf lustige Weise (Abb. 3.1) zeigt, wie gut seine Farben decken, wenn ein Farbroller nur zweimal über eine Fläche fährt (Abb. 3.2).

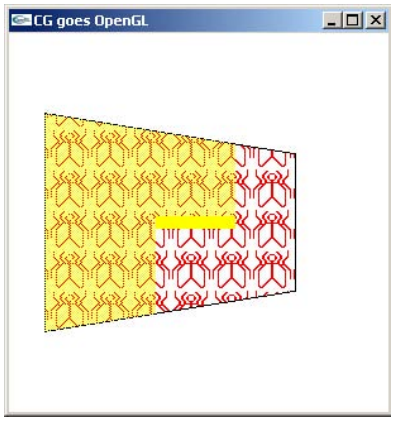


Abb. 3.1

Das Programm ist „gebrauchsfertig“, d.h., es erfüllt die in es gesetzten Erwartungen und braucht nicht weiter verändert zu werden. Es besteht aus der Code-Datei `Roller.c` und der Header-Datei (mit den verwendeten Konstanten etc.) `Roller.h`. Beide sind am Ende dieser Aufgabe ausgedruckt.

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

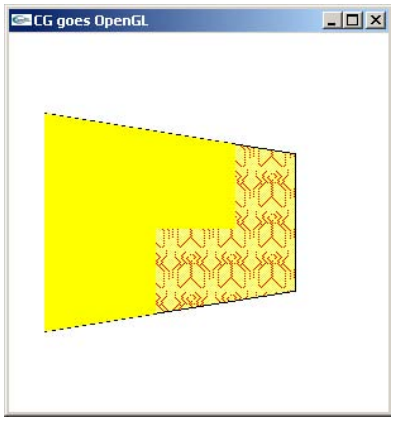


Abb. 3.2

`Roller.c` benötigt und verwendet ein GLUT-Fenster, in dem die Grafik dargestellt wird. Dieses Fenster wird in der Hauptroutine (`main()`) eingerichtet.

Beantworten Sie bitte hierzu die nachstehenden Fragen:

a) Verwendet das Programm eine Callback-Funktion?

Wenn ja: Welche? (Name genügt)

Wenn nein: Was macht hier eine solche Funktion überflüssig?

b) Die Hauptroutine (`main()`) schließt ordnungsgemäß mit einer `return`-Anweisung ab. Wird diese Anweisung immer, manchmal oder nie erreicht?

Wenn immer: Warum ist sie notwendig?

Wenn manchmal: Wann wird sie z.B. nicht erreicht?

Wenn nie: Wieso nicht? Kann sie nicht weggelassen werden?

c) In `key()` wird u.a. die Funktion `init()` aufgerufen; diese setzt zunächst die Koordinaten der Wand-Ecken (globale Variable `wall[][]`) gleichmäßig um den Koordinaten-Ursprung in der x-y-Ebene und die erste Position des Farbrollers (genauer: seines rechten Endes).

In welcher Reihenfolge sind die Wand-Ecken im Speicher abgelegt? (Bitte Zahlen 1 bis 4 verteilen für eine Betrachtung von der positiven z-Achse aus!)

obere linke Ecke	
obere rechte Ecke	
untere linke Ecke	
untere rechte Ecke	

d) Ebenfalls in der Funktion `init()` wird eine Farbe gesetzt. Was für eine Farbe ist dies, und wozu nützt sie?

Die globale Variable `paintNr` soll festhalten, zum wievielten Mal die Wand nun angestrichen wird. `paintNr` wird erstmalig am Anfang der Datei initialisiert und wird dann nur noch in `init()` verändert. (Es sei darauf hingewiesen, daß `init()`nicht nur in `main()`, sondern auch in `key()` aufgerufen wird, wenn <CR> gedrückt wird und das Ende der Wand inzwischen erreicht wurde.) Welche Werte nimmt die Variable `paintNr` während eines Programmlaufs an, und woran erkennen Sie das?

Die Funktion `draw()` enthält genau eine Schleife, die offenbar mehrmals die gezeichnete Wand überstreicht. Bitte kreuzen Sie die richtigen unter den u.a. Aussagen an: Die Schleife in `draw()`...

	... wird beim ersten Aufruf von `draw()`genau einmal durchlaufen.
	... zeichnet zuerst (beim ersten Schleifen- Durchlauf) einen schwarzen Linien-Umriß der Wand.
	... verwendet zuletzt die Eckpunkt-Koordinaten der Wand, um die Wand-Umrandung als geschlossenen Linienzug zu zeichnen.
	... füllt bei jedem Aufruf von `draw()` als erstes das Wand-Polygon mit einem Fliegenmuster und zeichnet erst dann evtl. weiter.
	... verwendet zu jeder Flächenfüllung eines der verfügbaren Muster (d.h., es gibt immer Stippling).
	... verwendet je nach Aufruf auch gemusterte (gestrichelte) Linienzüge.
	... schließt den Linien-Umriß genau dann, wenn der erste Punkt auch als letzter angegeben wird.

e) Das Programm verwendet zwei Farben, die über die globalen indizierten Variablen `before[]` und `after[]` eingesetzt werden. Welche Farbe repräsentiert jede von ihnen?

`before[]`: .....

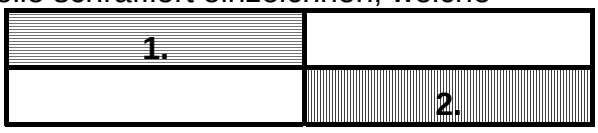
`after []`: .....

f) Die globale Variable `rightEnd` enthält die Raum-Koordinaten des rechten Endes des Farbrollers; sie wird entsprechend in `init()` initialisiert, in `key()` (bei <CR>) gesetzt und in der Funktion `roller()` zur Modellierung der gestrichenen Teilfläche der Wand verwendet. Beantworten Sie dazu bitte folgende Fragen:

Entlang welcher Achse (in positiver oder negativer Richtung) und um welche Länge (Zahlenwert) wird der Farbroller nach dem ersten <CR> bewegt? Woran erkennen Sie dies?

g) Die Funktion `roller()` überstreicht die Wand neu; die Bewegung des Farbrollers ist dabei jedes Mal die gleiche. Dabei wird offenbar die neu gestrichene Fläche in zwei Teilen gezeichnet.

Für den Fall, daß gerade die Hälfte der Wandfläche überstrichen wurde, wollen Sie bitte in der folgenden Tabelle schraffiert einzeichnen, welche Flächenteile in welcher Reihenfolge überstrichen wurden – etwa:



```
/* Roller.c */
/*Darstellung eines Farbrollers mit OpenGL*/
#include "Roller.h"

/*Globale Variablen (1): */
float    angle[3]={0.,0.,0.}, before[3]={1.,0.,0.},
         after[3]={1.,1.,0.};
int      paintNr=2;
GLdouble nah=WIN_WID/2., fern=10.;
GLfloat  wall[4][3], rightEnd[3];

/*****************************************************************/
  void roller(void)
/*****************************************************************/
{ if (paintNr==1) glPolygonStipple (halftone);
  else            glDisable (GL_POLYGON_STIPPLE);

  /*Neu gestrichene Flaeche:*/
  glColor3fv (after);
  glBegin(GL_POLYGON);
    glVertex3fv(wall[0]);  glVertex3f(wall[0][X],  rightEnd[Y], 0.);
    glVertex3fv(rightEnd); glVertex3f(rightEnd[X], wall[0][Y],  0.);
  glEnd();

  if (rightEnd[X] > wall[0][X]+ROL_WID && rightEnd[Y] > wall[1][Y])
  { glBegin(GL_POLYGON);
      glVertex3f (wall[0][X], rightEnd[Y], 0.);
      glVertex3fv(wall[1]);
      glVertex3f (rightEnd[X]-ROL_WID, wall[1][Y],  0.);
      glVertex3f (rightEnd[X]-ROL_WID, rightEnd[Y], 0.);
    glEnd();
  }

  /*Farbroller:*/
  glLineWidth (50.);
  glBegin(GL_LINES);
    glVertex3f (rightEnd[X]-ROL_WID, rightEnd[Y], 0.);
    glVertex3fv(rightEnd);
  glEnd();

  return;
}
```

```
/*********************************************************************/
  void draw(void)
/*********************************************************************/
{ int j1=0;
  fern=nah+2.*WIN_WID;

  glClear(GL_COLOR_BUFFER_BIT);

  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum(-WIN_WID/2.,WIN_WID/2.,-WIN_WID/2.,WIN_WID/2.,nah,fern);

  /*Aktuelle Positionierung:*/
  move();

  /*Bisher gestrichene Wand mit Rand:*/
  glEnable (GL_POLYGON_STIPPLE);
  for (j1=0; j1 <= paintNr; j1++)
  { if (j1==paintNr) { glColor3f (0., 0., 0.);  glLineWidth (1.);
                       glBegin(GL_LINE_LOOP); } else
    { if (j1 == 0) { glPolygonStipple(fly);     glColor3fv(before);}
      else         { glPolygonStipple(halftone);glColor3fv (after);}
      glBegin(GL_POLYGON);
    }
    glVertex3fv(wall[0]); glVertex3fv(wall[1]);
    glVertex3fv(wall[2]); glVertex3fv(wall[3]);
    glEnd();
  }

  /*Farbroller:*/
  roller();
  glFlush();
}

/*********************************************************************/
  void init(void)
/*********************************************************************/
{ /*Eckpunkt-Koordinaten (gegen den UZS, beginnend o.li.):*/
  wall[0][X] = wall[1][X] = -WIN_WID/2.; //WIN_WID=9
  wall[2][X] = wall[3][X] =  WIN_WID/2.;
  wall[0][Y] = wall[3][Y] =  WIN_HIG/2.; //WIN_HIG=6
  wall[1][Y] = wall[2][Y] = -WIN_HIG/2.;
  wall[0][Z] = wall[1][Z] = wall[2][Z] = wall[3][Z] = 0.f;

  rightEnd[X] = wall[0][X] + ROL_WID;
  rightEnd[Y] = wall[0][Y];
  rightEnd[Z] = wall[0][Z];

  paintNr = 3 - paintNr;

  glClearColor (1., 1., 1., 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 CR :rightEnd[Y] -= WIN_HIG*.1f;
             if (rightEnd[Y] <= wall[1][Y])
             {if (rightEnd[X] >= wall[2][X]) init();
              else {rightEnd[X]=MIN(rightEnd[X]+ROL_WID,wall[2][X]);
                    rightEnd[Y] = wall[0][Y]; }
              }                                                  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>      paint wall");
  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;
}
/*****************************************************************/
  int main(int argc, char **argv)
/*****************************************************************/
{ glutInit(&argc, argv);
  glutCreateWindow("CG goes OpenGL");
  glutDisplayFunc(draw);
  glutKeyboardFunc(key);
  init();
  glutMainLoop();
  return 0;
}
/*****************************************************************/
  void move(void)
/*****************************************************************/
{ /*Positionierung neu berechnen:*/
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  /*Alles ins Sichtvolumen verschieben:*/
  glTranslatef(0., 0., -(nah+WIN_WID/4.));
  /*Positionierung:*/
  glRotatef(angle[X], 1., 0., 0.);
  glRotatef(angle[Y], 0., 1., 0.);
  glRotatef(angle[Z], 0., 0., 1.);
}
```

```
/* Roller.h */
#include <stdio.h> //wg. printf()
#include <GL/glut.h>
#define CON_CLS  "cls"
#define ESC        27
#define CR         13
#define ROL_WID     3
#define WIN_WID     3*ROL_WID
#define WIN_HIG     6
#ifndef MIN
  #define MIN(x,y) (( (x) < (y) ) ? (x) : (y))
#endif  MIN

enum {X=0, Y=1, Z=2, W=3};
/*Prototypen:*/
  void draw  (void);
  void init  (void);
  void key   (unsigned char key, int x, int y);
  void move  (void);
  void roller(void);

/*Globale Variablen (2): */
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,
      0x04, 0x06, 0x60, 0x20, 0x44, 0x03, 0xC0, 0x22,
      0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
      0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
      0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
      0x66, 0x01, 0x80, 0x66, 0x33, 0x01, 0x80, 0xCC,
      0x19, 0x81, 0x81, 0x98, 0x0C, 0xC1, 0x83, 0x30,
      0x07, 0xe1, 0x87, 0xe0, 0x03, 0x3f, 0xfc, 0xc0,
      0x03, 0x31, 0x8c, 0xc0, 0x03, 0x33, 0xcc, 0xc0,
      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};
GLubyte halftone[] = {
      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};
```

Platz für Notizen: