Technische Hochschule Mittelhessen **27. Juli 2015**

Prof. Dr.-Ing. A. Christidis

**Fachbereich Mathematik,
Naturwissenschaften und Informatik**

Klausur
Computergrafik
für Bachelor-Studierende
SS 2015

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 besuchen mit einem Freund einen neu organisierten Informatik-Kongreß und hören über jemanden, den die tagenden Experten als den „Pixel-Guru“ bezeichnen. Sie fragen eine Kongreßteilnehmerin, was dieser Mann Besonderes geleistet hat, und sie antwortet überrascht, daß jeder diesen Mann kenne: Er habe die wichtigsten Top-Down-Algorithmen entwickelt, vor allem für problematische Lichtverhältnisse.

Handelt es sich bei dem „Pixel-Guru“ um einen Spezialisten für Computergrafik, für Bildverarbeitung, für Bildbearbeitung, für alles oder für keines davon? (Nennung genügt.)

Ihr Freund hat nichts verstanden und fragt Sie, was Top-Down-Algorithmen mit Pixeln zu tun haben – vor allem aber, ob dann Pixel zum „Top“- oder zum „Down“-Teil des Algorithmus gehören. Was erklären Sie ihm?

b) Kann die Einheitsmatrix eine Rotationsmatrix um die x-Achse darstellen?

Wenn ja: Welche Rotation beschreibt sie dann?

Wenn nein: Woran ist das zu erkennen?

c) Sie haben ein Programm geschrieben, bei dem man mit der Maus eine beliebig lange, waagerechte Linie ziehen und dann im Menü wählen kann, mit dieser Linie als Radius einen Kreis zu zeichnen. Jeder Klick auf ein Pixel innerhalb des Kreises bewirkt, daß mit dem Bresenham-Algorithmus ein weiterer Radius eingezeichnet wird, der dieses Pixel enthält (Abb. 1.1).

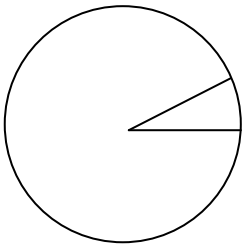

Abb. 1.1

Enthalten so eingezeichnete Radien maximal, minimal oder stets dieselbe Anzahl von Pixeln, wie der waagerecht eingezeichnete Radius?

Warum? (Kurze Erklärung)

d) Für welche (englischen) Wörter stehen die Akronyme „OpenGL“ und „GLUT“, und was bedeuten sie (nur sprachlich) ins Deutsche übersetzt?

e) Was ist öffentlich zugänglich und rechtfertigt so die Bezeichnung „open“ in OpenGL? (Nennung genügt)

2. **Aufgabe** (55 Punkte)

Ein Hersteller von Kippfenstern (Abb. 2.1) bittet Sie um Unterstützung: Er plant einen Zeichentrickfilm über seine Produkte, und einer seiner Ingenieure hat schon im Selbststudium eine erste einfache Grafik mit einem solchen Fenster vorbereitet (Abb. 2.2a). Die Berechnung des gedrehten und gekippten Fenstergriffs für die Animation erweist sich aber für den unerfahrenen Grafiker als zu komplex: Abb. 2.2b existiert nur als Beschreibung.

Abb. 2.1

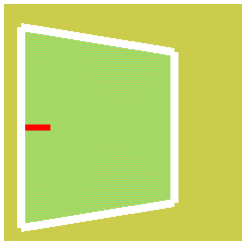

Abb. 2.2a

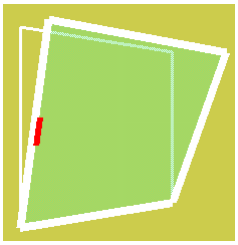

Abb. 2.2b

Das Fenster ist als Liniengrafik (Abb. 2.3) in der x-y-Ebene eingebettet. Seine Breite liegt von **–w** bis **+w** symmetrisch um die y-Achse, aber seine Höhe ist zu ¼ unterhalb, zu ¾ oberhalb der x-Achse, von **–h** bis **+3h** modelliert. Der faßbare Teil des Griffs, um den es hier geht, ist auf halber Höhe, am linken Fensterrand, im Abstand **D** von der Fensterebene angebracht; er hat die Anfangskoordinaten $\mathbf{[-w, +h, +D]}^T$, und die Länge **L**. Zum Kippen um den Winkel **α** wird der Griff um 90° nach oben gedreht.

Sie erklären den staunenden Kollegen, daß die Ermittlung der Endlage des Fenstergriffs mit fünf affinen Transformationen möglich ist:

Zuerst wird der Griff in die Fensterebene, und zwar an den Koordinatenursprung, verschoben (Transformationsmatrix $\underline{\mathbf{T}}_{(i)}$).

Dann wird er um den rechten Winkel **φ** (nicht eingezeichnet) um die z-Achse nach oben gedreht (Transformationsmatrix $\underline{\mathbf{T}}_{(ii)}$).

Danach wird er zwar wieder an den Fensterrand verschoben, aber an eine um **h** erhöhte Position (Transformationsmatrix $\underline{\mathbf{T}}_{(iii)}$); dort wäre er, wenn die Fenster-Unterkante mit der x-Achse zusammenfiele. Sodann wird er (als Teil eines gedachten Fensters über der x-Achse) um den Winkel **α** um die x-Achse gedreht (Transformationsmatrix $\underline{\mathbf{T}}_{(iv)}$).

Schließlich wird er wieder um **h** auf seine korrekte (niedrigere) Höhe verschoben (Transformationsmatrix $\underline{\mathbf{T}}_{(v)}$).

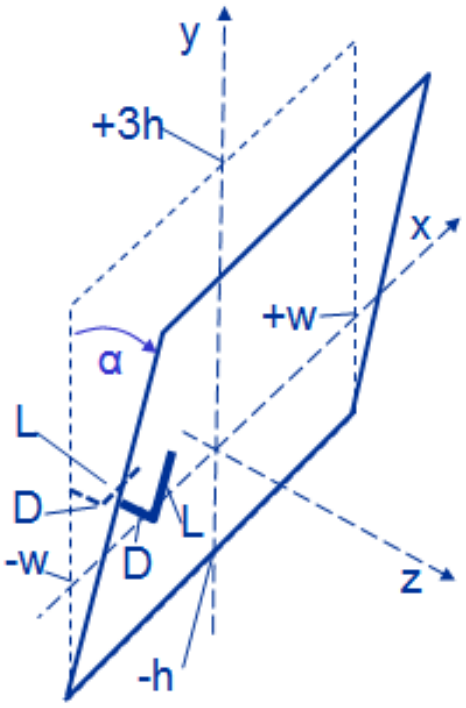

Abb. 2.3

Beantworten Sie bitte folgende Fragen:

a) Wie lautet die Transformationsmatrix $\underline{\mathbf{T}}_{(i)}$, die eine Translation der Griffbefestigung an den Koordinatenursprung ermöglicht?

Geben Sie sie bitte mit den hier verfügbaren Größen an!

$\underline{\mathbf{T}}_{(i)}$ =

b) Geben Sie bitte (in Grad, unter Berücksichtigung des Drehsinns) den rechten Winkel **φ** an, um welchen der Fenstergriff nach oben gedreht wird. Wie groß sind **sin φ** und **cos φ**?

φ = **sin φ =** **cos φ =**

c) Wie lautet die Transformationsmatrix $\underline{\mathbf{T}}_{(ii)}$, die den Griff um den rechten Winkel **φ** bis zur y-Achse dreht? Geben Sie sie bitte sowohl in symbolischer (sin φ, ...) als auch in arithmetischer (zahlenmäßiger) Form an!

$\underline{\mathbf{T}}_{(ii)}$ =

d) Wie lautet die Transformationsmatrix $\underline{\mathbf{T}}_{(iii)}$, die den Fenstergriff an die Position verschiebt, die er innehätte, wenn die Fenster-Unterkante mit der x-Achse zusammenfiele?

Geben Sie sie bitte direkt mit den verfügbaren Größen an!

$\underline{\mathbf{T}}_{(iii)}$ =

e) Zur Veranschaulichung Ihres Vorgehens rechnen Sie mit einem unrealistischen Kippwinkel **α** von 90° (absolut). Geben Sie bitte (in Grad, unter Berücksichtigung des Drehsinns) den Winkel **α** an, um welchen das Fenster (und mit ihm der Griff) um die x-Achse gedreht wird. Wie groß sind **sin α** und **cos α**?

α = **sin α =** **cos α =**

f) Welche Transformationsmatrix $\underline{\mathbf{T}}_{(iv)}$ kippt, wie oben beschrieben, das Fenster um die x-Achse um den o.a. Winkel **α**? Geben Sie sie bitte sowohl in symbolischer als auch in arithmetischer Form an!

$\underline{\mathbf{T}}_{(iv)}$ =

g) Mit welcher Transformationsmatrix $\underline{\mathbf{T}}_{(v)}$ wird schließlich der Fenstergriff auf die korrekte Höhe verschoben? Geben Sie sie bitte direkt mit den verfügbaren Größen an!

$\underline{\mathbf{T}}_{(v)}$ =

h) Wie berechnet sich nun die Transformationsmatrix **T**, mit welcher alle vorausgegangenen Transformationen zu einer Operation vereinigt werden?

T =

i) Berechnen Sie nun bitte die Transformationsmatrix **T** anhand der bisher gemachten Angaben.

T =

j) Berechnen Sie jetzt bitte die Anfangs- und Endkoordinaten des Fenstergriffs am gekippten Fenster:

k) Einer Ihrer Bewunderer im Betrieb des Fenster-Herstellers fragt, Sie, ob die Transformationsmatrix $\underline{\mathbf{T}}_{(iii)}$, nicht unnötig kompliziert sei; ob z.B. die Verschiebungen in x- und in z-Richtung nicht ebensogut Teil von $\underline{\mathbf{T}}_{(v)}$ mit demselben Rechenergebnis sein könnten. Sie antworten:

	Nein, weil das Matrizenprodukt nicht assoziativ ist.
	Teils-teils: Die z-Koordinaten des Objekts verändern sich durch die Rotation um die x-Achse; d.h., die Translation in z-Richtung muß noch Teil von $\underline{\mathbf{T}}_{(iii)}$ sein. Bei den x-Koordinaten verhält es sich umgekehrt, die x-Translation könnte also ebenso in $\underline{\mathbf{T}}_{(v)}$ stecken.
	Nein, weil das Matrizenprodukt nicht kommutativ ist.
	Ja, denn der Radius der Rotation beim Kippen mißt sich entlang der y-Achse und ist somit unabhängig von Translationen in x- und in z-Richtung.
	Teils-teils: Die x-Koordinaten (w) haben hier hohe Beträge; deswegen dürfen sie nicht aus den Zwischenrechnungen fehlen. Die z-Ausdehnungen (D) sind dagegen nahezu vernachlässigbar, sie können erst in $\underline{\mathbf{T}}_{(v)}$ eingesetzt werden.

l) Dankbar für den Erkenntnisgewinn besprechen die Mitarbeiter Ihre Berechnung. Jemand betrachtet die Größen **D**, **h**, **L** und **w**, welche die Koordinaten des Fenstergriff-Endpunkts bestimmen, und konzentriert sich auf die Frage, in welche der Koordinaten dieses Endpunkts sie bei Wahl eines beliebigen Winkels α (0° < | α | < 90°) eingehen würden. Klar ist dabei zunächst, daß w weiterhin nur die x-Koordinate des Endpunkts beeinflussen würde.

Ergänzen Sie bitte die u.a. Tabelle:

Einfluß auf Griff-Koordinate	**D**	**h**	**L**	**w**
x				**X**
y				**–**
z				**–**

3. Aufgabe (30 Punkte)

a) Die Grafik-unerfahrenen Ingenieure des örtlichen Kippfenster-Herstellers bitten Sie um Hilfe: Sie haben im Internet Bilder und ein OpenGL-Programm gefunden, das ein Kippfenster darstellt. Aber sie wissen nicht, ob sie zusammengehören.

Nach einem kurzen Blick auf den Source-Code von `WinTilt.c` (s. Ende dieser Aufgabe) können Sie sagen, daß mindestens zwei Zeilen im `main()` nicht zu Abb. 3.1 passen.

Welche Zeilen sind damit gemeint und warum?

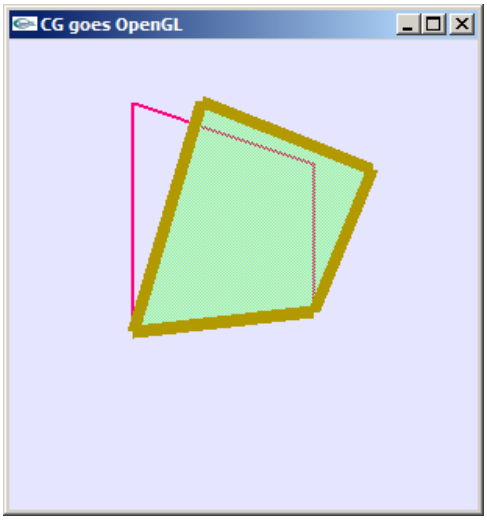

Abb. 3.1

b) Mit Ihrer Hilfe wird `WinTilt.c` zum Laufen gebracht. Es zeigt ein Kippfenster ohne Fenstergriff (Abb. 3.2).

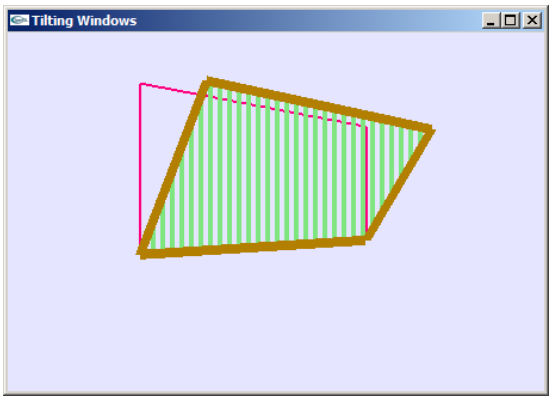

Abb. 3.2

Ihre Kollegen wollen das Programm debuggen; dabei setzen sie u.a. einen Haltepunkt (breakpoint) an die letzte Zeile von `main()` (bei `return 0;`), um einer vorzeitigen Beendigung des Programms vorgreifen zu können.

Davon raten Sie ab und sagen, das sei wirkungslos.

Warum?

c) Der Code von `WinTilt.c` enthält genau einen Aufruf der Funktion `key()`: am Ende der Funktion `init()`, zudem mit Übergabe eines Leerzeichens (`' '`), was keiner Menü-Position entspricht. Der Aufruf von `key()` stellt nur sicher, daß das Menü auf der Konsole ausgegeben und die Grafik im GLUT-Fenster (mit dem `draw()`-Aufruf) gezeichnet wird.

Muß `WinTilt.c` noch vervollständigt werden, wenn das dort codierte Menü wirksam werden soll?

Wenn ja: Welche Funktionalitäten fehlen noch im Programm?

Wenn nein: Wie kann man bewirken, daß `key()` aufgerufen wird?

d) In der Funktion `init()` werden in der indizierten Variablen `fenster[][]` die Koordinaten für den Fensterflügel gespeichert. Ihre Werte werden in die Variable `rahmen[][]` übernommen; sie steht für den (unbeweglichen) Fensterrahmen.

In `draw()` wird erst die gesamte Szene (Fensterflügel und -rahmen) positioniert, dann der Rahmen gezeichnet und schließlich (in Abhängigkeit vom Wert der Variablen `kipp`) die Funktion `tilt()` aufgerufen; sie berechnet die Koordinaten des gekippten Fensters mit zwei Translationen und einer Rotation.

Sie wollen die Richtigkeit von `tilt()` überprüfen, indem Sie die entsprechenden Matrizenprodukte ausführen.

Geben Sie bitte unten mit den Zahlen 1-3 an, welche der dort aufgelisteten Operationen Sie in welcher Reihenfolge durchführen müssen, und kreuzen Sie bitte an, welche Raumachse und welche Richtung dabei relevant wäre (in den schattierten Feldern).

Nr.		***X***		***X***	
	Rotation um 25° um die		x-		positiver
			y- Achse in		Richtung
			z-		negativer
	Verschiebung entlang der		x-		positiver
			y- Achse in		Richtung
			z-		negativer
	Verschiebung entlang der		x-		positiver
			y- Achse in		Richtung
			z-		negativer

e) Die Funktion `draw()` ruft `glFrustum()` auf. Entlang welcher der drei Koordinatenachsen (x, y, z) gelten die Werte für die Parameter `nah` und `fern`, und welche Rolle spielen sie beim Zeichnen („Rendern“) einer Szene?

(Kurze Schilderung)

f) Das „virtuelle Fensterglas“ (Abb. 3.2) weist eine Streifenstruktur auf. Beantworten Sie dazu bitte (mit wenigen Angaben) die Fragen:

- Welche Stelle im Code bewirkt die Strukturierung der Fensterfläche?
- Wie viele Pixel breit sind die Streifen im Fensterglas, und woran erkennen Sie das?

g) Die vorliegende Programm-Version verwendet vier unterschiedliche Farben. Ihre Komponenten (eingestellt in `draw()`) sind unten aufgeführt. Kreuzen Sie bitte die Beschreibung an, die jeweils am ehesten zu den Farbkomponenten paßt:

	Farbbeschreibung	Löschen	Rahmen	Glas	Einfassung
	Farbkomponenten (R, G, B)	**.9, .9, 1.**	**1., 0., .5**	**.5, 1., .5**	**.7, .6, 0.**
	Rot mit etwas Blau, fast Magenta				
	Grelles Weiß				
	Braun, Richtung oliv				
	Leichtes Blau, fast Weiß				
	Hellste Gelb-Stufe				
	Ungesättigtes („weißliches“) Grün				
	Tiefes Schwarz				

h) Verwendet `WinTilt.c` Double Buffering? Woran erkennen Sie dies?

(Kurze Erläuterung)

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

#define CON_CLS  "cls"
#define ESC        27
#define UNIT        4
#define TOP       UNIT
#define RIGHT     UNIT
#define BOTTOM   -UNIT/3.f
#define LEFT     -UNIT

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

/*Globale Variablen: */
float    angle[3]={0.,0.,0.};
int      kipp=0;
GLdouble nah=UNIT*2, fern=10*UNIT;
GLfloat  fenster[4][3], rahmen[4][3];

GLubyte halftone[] = {
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
      0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0};

/*************************************************************************/
  void tilt(void)
/*************************************************************************/
{ /*Kippen des Fensters:*/
  glTranslatef(0.f, BOTTOM, 0.f);
  glRotatef(25., 1., 0., 0.);
  glTranslatef(0., -BOTTOM, 0.);
  return;
}

/*************************************************************************/
  void draw(void)
/*************************************************************************/
{ /*Zeichnen der Szene:*/
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum (-UNIT, UNIT, -UNIT, UNIT, nah, fern);

  /*Alles loeschen:*/
  glClear(GL_COLOR_BUFFER_BIT);

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

```
  /*Alles ins Sichtvolumen verschieben:*/
  glTranslatef(0., 0., -(nah+1.5*UNIT));

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

  /*Loeschfarbe:*/
  glClearColor (0.9f, .9f, 1.f, 1.f);

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

  /*Kippen:*/
  if (kipp) tilt();

  /*Fenster-Glas*/
  glColor3f (.5f, 1.f, .5f);

  glEnable  (GL_POLYGON_STIPPLE);  glPolygonStipple (halftone);
  glBegin(GL_POLYGON);
    glVertex3fv(fenster[0]); glVertex3fv(fenster[1]);
    glVertex3fv(fenster[2]); glVertex3fv(fenster[3]);
  glEnd();
  glDisable (GL_POLYGON_STIPPLE);

  /*Fenster-Einfassung:*/
  glLineWidth (8.);
  glColor3f (.7f, .6f, 0.f);
  glBegin(GL_LINE_LOOP);
    glVertex3fv(fenster[0]); glVertex3fv(fenster[1]);
    glVertex3fv(fenster[2]); glVertex3fv(fenster[3]);
  glEnd();

  glFlush();
  return;
}

/*****************************************************************************/
  void key(unsigned char key, int x, int y)
/*****************************************************************************/
/*Menue und Eingabe-Behandlung:*/
{ switch (key)
  { case ESC: exit(0);
    case 't': kipp = 1-kipp;                                  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 t         toggle tilting window");
  if (kipp) 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)
/*************************************************************************/
{ int j1=0, j2=0;
  /*Eckpunkt-Koordinaten Fenster:*/
  fenster[0][X] = fenster[3][X] = LEFT;
  fenster[1][X] = fenster[2][X] = RIGHT;
  fenster[0][Y] = fenster[1][Y] = BOTTOM;
  fenster[2][Y] = fenster[3][Y] = TOP;
  fenster[0][Z] = fenster[1][Z] = fenster[2][Z] = fenster[3][Z] = 0.f;

  /*Rahmen- wie Fenster-Koordinaten:*/
  for (j1=0; j1<4; j1++)
    for (j2=0; j2<3; j2++)
      rahmen[j1][j2] = fenster[j1][j2];

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

/*************************************************************************/
  int main(int argc, char **argv)
/*************************************************************************/
{ glutInit(&argc, argv);
  glutInitWindowSize(450, 300);
  glutCreateWindow("Tilting Windows");
  glutDisplayFunc(draw);
  glutKeyboardFunc(key);
  init();
  glutMainLoop();
  return 0;
}
```

Platz für Notizen: