Technische Hochschule Mittelhessen **17. Juli 2017**

Prof. Dr.-Ing. A. Christidis

Fachbereich Mathematik,
Naturwissenschaften und Informatik

Klausur
Computergrafik
für Bachelor-Studierende
SS 2017

Personalien:

Name, Vorname: .....

Matrikelnummer: .....

☐ **Ich möchte mein Ergebnis anhand meiner Matrikelnr. auf der Homepage zu diesem Fach für begrenzte Zeit abrufen können. (Bitte ggf. ankreuzen und unterschreiben)**

(Unterschrift) .....

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, die als Informatikerin mit Erfahrung auf dem Gebiet der Datenbanken jahrzehntelang als Entwicklerin im Dienst der internationalen Flugsicherheit gearbeitet hat. Sie erfahren, daß sie berühmt geworden ist, weil sie ein Verfahren entwickelt hat, mit dem, nach Einscannen des Paßbildes und entsprechender Auswertung, eine 3D-Maske des Paßinhabers erstellt wird, die sein Gesicht in beliebigen Positionen zeigt (Abb. 1.1).

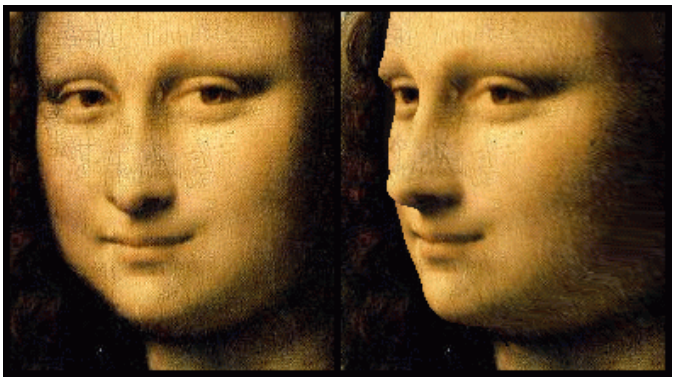

Abb. 1.1

Handelt es sich bei dieser großen Datenbank-Expertin gleichzeitig um eine Fachfrau für Bildverarbeitung, für Computergrafik, für beide, oder weder für die eine, noch für die andere?

Bitte begründen Sie kurz Ihre Antwort!

b) Nach Ihrem Erfolg in der Klausur Computergrafik wendet sich ein Hamburger Straßenmaler an Sie mit der Bitte um Hilfe: Er will seiner Stadt ein Denkmal setzen mit einem „Pixelbild“ aus 1-cm²-kleinen Metall-Plättchen, die er selbst in passenden Farben emailliert.

Das Bild (seine persönliche Abstraktion eines bekannten Hamburger Motivs) soll einen rechteckigen Schriftzug aus 80 x 60 Plättchen in einem Kreis zeigen; die Linienstärke des Kreises soll (anders als in Abb. 1.2) vernachlässigbar sein.

Abb. 1.2

Seine Frage an Sie lautet:

Kann er sein Projekt realisieren, wenn er von einem Sponsor die Finanzierung von max. 7.500 Plättchen und von der Stadt die Auflage bekommt, nicht mehr als 1,2 Quadratmeter auf dem Vorplatz der Elbphilharmonie mit seinem Kunstwerk zu belegen?

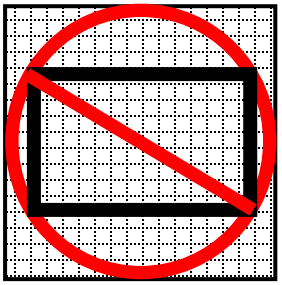

Abb. 1.3

Sie wollen sein Anliegen prüfen.

Hierzu fertigen Sie zunächst eine Skizze nach seiner Beschreibung an (Abb. 1.3) und berechnen für ihn die benötigten Größen, indem Sie mit Hilfe des Algorithmus von J.E.Bresenham und des Lehrsatzes von Pythagoras ($a^2+b^2=c^2$) folgende Fragen beantworten:

- Wieviele „Pixel“ (Plättchen) werden die Diagonale des Rechtecks bilden, und nach wessen Lehre erkennen Sie das? (Kurze Begründung!)

- Welche Länge (in cm) hat der Durchmesser des Umkreises des o.a. Rechtecks, und wonach berechnen Sie das?

- Wie groß ist die Fläche (in m^2), die das Künstlerprojekt benötigt?

- Wieviele Pixel werden die Kante des umschließenden Quadrats bilden, und wieviele Plättchen werden insgesamt für das Kunstwerk benötigt?

- Kann das Kunstprojekt nach den obigen Vorgaben zustande kommen – falls nicht: Woran scheitert das Vorhaben?

2. **Aufgabe** (40 Punkte)

Sie planen ein Grafik-Programm, das animiert darstellt, wie sich Windräder mit ihren Schaltkästen (Abb. 2.1) optimal in den Wind stellen.

Abb. 2.1

Zur Überprüfung wollen Sie für ein stark abstrahiertes Modell eine Position für die Spitze eines seiner Rotor-Blätter von Hand vorausberechnen. Das Rotor-Blatt soll zu Beginn der Berechnung aufrecht stehen (in Abb. 2.2 dunkel dargestellt).

Seine Endstellung (Abb. 2.5), zunächst ohne Berücksichtigung seiner Höhe, wollen Sie wie folgt berechnen:

Sie modellieren das Windrad in der x-y-Ebene mit seinem Mittelpunkt am Koordinatenursprung, drehen es um $|\boldsymbol{\alpha}|=90°$ um die z-Achse, bis das anfänglich nach oben gerichtete Rotor-Blatt auf der negativen x-Achse liegt (Transformationsmatrix $\underline{\mathbf{T}}_{(i)}$, Abb. 2.3), neigen dann das Rad um $|\boldsymbol{\beta}|=30°$ um die x-Achse, so, daß der obere Teil des Windrads sich dem Betrachter etwas nähert (Transformationsmatrix $\underline{\mathbf{T}}_{(ii)}$, Abb. 2.4) und wenden es anschließend um $|\boldsymbol{\gamma}|=60°$ um die y-Achse zur linken Seite des Sichtfeldes (Transformationsmatrix $\underline{\mathbf{T}}_{(iii)}$, Abb. 2.5).

Abb. 2.2

Abb. 2.3

Abb.2.4

Abb. 2.5

Berechnen Sie bitte die Koordinaten-Transformation für das Windrad, indem Sie die nachfolgenden Fragen behandeln. Sie können dabei die gerundeten Werte verwenden: sin 60° = cos 30° ≈ 0,9; sin 30° = cos 60° = 0,5.

a) Geben Sie bitte (in Grad, unter Berücksichtigung des Drehsinns) den Winkel **α** an, um welchen das Windrad gedreht wird, bis das ursprünglich aufrechte Rotor-Blatt auf der negativen x-Achse liegt.

Wie groß sind dann **sin α** und **cos α**?

α = **sin α =** **cos α =**

b) Wie lautet die Transformationsmatrix $\underline{\mathbf{T}}_{(i)}$, die das Rotor-Blatt um **α** bis zur negativen x-Achse dreht? Geben Sie sie bitte sowohl in symbolischer (sin α, ...) als auch in arithmetischer (zahlenmäßiger) Form an!

(b.w.)

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

c) Geben Sie nun bitte (in Grad, unter Berücksichtigung des Drehsinns) den Winkel **β** an, um welchen das Windrad um die x-Achse hin zum Betrachter gekippt wird und beziffern Sie (ggf. gerundet) **sin β** und **cos β**:

β = **sin β =** **cos β =**

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

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

e) Wie groß ist (in Grad, unter Berücksichtigung des Drehsinns) der Winkel **γ**, um welchen das Windrad um die y-Achse nach links gedreht wird? Geben Sie bitte auch die (ggf. gerundeten) Werte für **sin γ** und **cos γ** an.

γ = **sin γ =** **cos γ =**

f) Wie lautet die dazugehörige Transformationsmatrix $\underline{\mathbf{T}}_{(iii)}$, die das Windrad um die y-Achse dreht? Geben Sie sie bitte sowohl in symbolischer (sin γ, ...) als auch in arithmetischer Form an!

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

g) Wie berechnet sich nun die Gesamt-Rotationsmatrix $\underline{\mathbf{T}}_{\mathbf{RG}}$, die alle vorausgegangenen Transformationen enthält?

$\underline{\mathbf{T}}_{\mathbf{RG}}$ =

h) Ermitteln Sie nun bitte die Gesamt-Rotationsmatrix $\underline{\mathbf{T}}_{\mathbf{RG}}$ anhand der bisher gemachten Angaben.

$\underline{\mathbf{T}}_{\mathbf{RG}}$ =

i) Bei dem Wunsch, die realen Koordinaten der Rotor-Spitze zu berechnen, schauen Sie sich die technischen Details des Windrads an. Sie erfahren, daß dieses Exemplar einen Durchmesser von 4 Metern hat und sein Mittelpunkt in einer (Mast-)Höhe von 10 Metern montiert ist. Da Sie die Koordinaten ohnehin in Metern verstehen, können Sie die dazugehörige Translationsmatrix $\underline{\mathbf{T}}_{\mathbf{T}}$ ohne metrische Einheiten (m) hinschreiben.

Wie lautet sie? (b.w.)

$\underline{\mathbf{T}}_T$ =

j) Berechnen Sie nun bitte die Gesamt-Transformationsmatrix $\underline{\mathbf{T}}_{ges}$ mit allen Raumtransformationen anhand der bisherigen Erkenntnisse.

$\underline{\mathbf{T}}_{ges}$ =

k) Ermitteln Sie nun bitte die Raum-Koordinaten der (anfänglich obersten) Spitze des Windrad-Blattes an der Höhe, in der es montiert ist, nach den vorausgegangenen Transformationen:

3. Aufgabe (45 Punkte)

Ihre Arbeitsgruppe hat mit OpenGL ein Animationsprogramm für den Hersteller eines neuartigen Windrads erstellt. Das innovative Gerät kann als kleiner Stromgenerator auf jeder Dachterrasse angebracht werden; auf Wunsch kann es aber auch im Gebäude aufgestellt und als Ventilator genutzt werden. In der letzteren Betriebsart schwenkt der Schaltkasten mit den Rotor-Blättern horizontal und vertikal (Abb. 3.1). Ihre Kollegen schicken Sie vor, um die gesamte Arbeit einem Grafik-unerfahrenen Vertreter des Auftraggebers vorzustellen.

Abb. 3.1

Sie besprechen das Programm vor allem anhand des ausgedruckten C-Codes (s. Ende dieser Aufgabe) und der hier wiedergegebenen Abbildungen.

a) Ihr Gegenüber sagt Ihnen, er beabsichtige, die letzte Zeile von `main()` (`return 0`) auf `return 1` zu ändern, wie er das in allen seinen Programmen macht, damit er nach Beendigung den letzten Rückgabewert vom Betriebssystem abrufen kann. Er fragt Sie, ob das etwas ändern würde.

Was würde eine solche Änderung tatsächlich bewirken? Warum?

b) Sie erklären dem Vertreter, daß die `init()`-Funktion als allererste aufgerufen wird, noch vor Einrichtung (und später bei jeder Größenänderung) des Fensters. `draw()` wird erst danach (und später bei jeder Offenlegung) zum Füllen des Fensters aufgerufen. Dieser fragt Sie, ob deswegen die Aufrufe `glutReshapeFunc(init)` und `glutDisplayFunc(draw)` in dieser Reihenfolge in `main()` stehen.

Was antworten Sie ihm?

c) Das Windrad besteht aus einem unbeweglichen Mast, welcher (als bewegliche Teile) einen Schaltkasten trägt, an dem die Rotorblätter angebracht sind. Alle Bildteile werden (gemäß `WindWheel.c`, `WindWheel.h`) erst als Flächenmodelle gezeichnet und dann als Drahtmodelle nachgezogen. (b.w.)

Welche Anweisungen lassen, z.B. an der Funktion `mast()`, erkennen,

1. daß zweimal gezeichnet wird und
2. daß einmal eine Flächen-, einmal eine Strichzeichnung entsteht?

(Kurzer Hinweis jeweils genügt.)

d) Was für eine geometrische Figur zeichnet die Funktion mast()?

Bitte Zutreffendes ankreuzen!

☐	regelmäßiges Polygon	☐	Rechteck
☐	Quader	☐	Trapez
☐	Dreieck	☐	Zylinder
☐	Pyramidenstumpf	☐	Kegelstumpf

e) Die Funktion `box()` zeichnet den Schaltkasten des Windrads. Dabei belegt sie nicht alle Flächen mit einem Muster. Welche Flächen-Indizes läßt sie aus (und zeichnet sie ohne Muster)? (Index-Angaben genügen.)

f) Die Funktionen `mast()` und `box()` belegen die dargestellten Flächen mit unterschiedlichen Mustern (festgelegt in `WindWheel.h`). Ihr Gesprächspartner fragt, wieviel mehr Lichtenergie das hellere gegenüber dem dunkleren Muster ans Auge des Betrachters senden würde, wenn die dunklen Pixel in beiden Mustern überhaupt keine (null) Lichtenergie entsenden würden (absolutes Schwarz). Wie kann man diese Frage beantworten? (Kurze Begründung!)

g) Die Funktion `blade()` erzeugt die Rotor-Blätter. Werden diese nur schwach, oder gar nicht gemustert? Woran ist es im Quellencode zu erkennen?

h) Nach kurzer Durchsicht des Codes erkennt man, daß die Rotor-Blätter in der Funktion `blade()` ihre Farbe über den globalen Bezeichner `myColor` erhalten. Dieser bekommt wiederum seine Werte in der Funktion `drawSettings()` in Abhängigkeit von seinem Schwenk-Winkel zugewiesen.

Beantworten Sie bitte dazu die Fragen:

- Ist für die Farbgebung die horizontale oder die vertikale Drehung des Windrads maßgeblich? (Nennung genügt.)

- An welchem Ende (Oben / Unten / Links / Rechts) der entsprechenden Drehung wird den Rotor-Blättern die hellere Farbe zugewiesen?

  Welche (R-G-B-)Parameter enthält diese Farbe?

i) In der Code-Besprechung erklären Sie, daß beim Programm-Start (Abb. 3.2) das obere Rotor-Blatt des Windrads als erstes gezeichnet wird. (Das ergibt sich aus den positiven y-Koordinaten und den x-Koordinaten beiderlei Vorzeichens, die von `blade()` an `mill()` übergeben werden.)

Welches Blatt wird als zweites gezeichnet: das unten links, oder jenes unten rechts im Bild?

An welchem Aufruf in `mill()` erkennt man das?

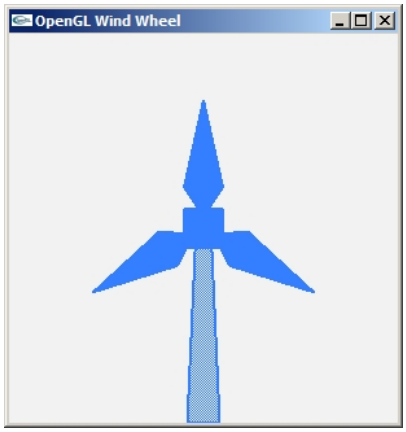

Abb. 3.2

j) Ihr unerfahrener Gesprächspartner bemerkt, daß in der Funktion `mill()` die drei Rotor-Blätter mit voneinander unabhängigen `blade()`-Aufrufen gezeichnet werden. Da er mit der Verwendung des Matrizen-Stapels von OpenGL noch nicht vertraut ist, erklären Sie ihm, daß die beiden Aufrufe (`glPushMatrix()` / `glPopMatrix()`) hier entfallen können. Man muß dann nur die zwei Rotationsbefehle entsprechend anpassen. Welche Parameter erhalten dann die beiden Aufrufe?

Ergänzen Sie bitte die Parameter in der u.a. Sequenz.

```
blade();
glRotatef(_____________________);
blade();
glRotatef(_____________________);
blade();
```

k) Während des Gesprächs fällt Ihnen auf, daß im Papier-Ausdruck der Funktion `draw()`, die das animierte Windrad zusammenstellt, jemand die wichtigsten Rotationsaufrufe gelöscht hat und nur noch die Aufrufe der Teilzeichnungen übriggelassen hat, in ihrer aufeinander aufbauenden (und insofern zwangsläufigen) Reihenfolge.

Sie wissen, daß der Mast nicht bewegt wird, der Schaltkasten horizontal und vertikal geschwenkt wird, während das daran montierte Windrad (neben der Bewegung des Kastens) auch um die eigene Achse rotieren soll.

Wo sollten die fehlenden Rotationsbefehle (**`glRotatef(angle[ ], ...)`**) um die drei Raumachsen stehen, und wie sollten sie lauten?

```
mast();




box();




mill();
```

l) Der Interessent sagt Ihnen, daß es beim Auftraggeber Ihres Programms noch nicht entschieden ist, ob bei einer Änderung der Fenstergröße die dargestellte Windrad-Grafik mitwachsen oder -schrumpfen soll.

Welche Funktion des vorliegenden Programms müßte ggf. angepaßt werden? (Nennung genügt.) (b.w.)

m) Kurz vor Beendigung des Gesprächs kommt ein weiterer Kollege dazu, der über Grafik-Erfahrung verfügt. Er fragt, wozu das Programm-Menü (Funktionen `key()` und `drawSettings()`) Anti-Aliasing für Linien anbietet, obwohl es sich hier um ein Flächenmodell handelt. Er probiert die Wirkung aus (Abbn. 3.3 mit ein-, Abb. 3.4 mit ausgeschaltetem Anti-Aliasing) und fordert Sie auf, Ihren Kommentar dazu abzugeben, ob hier Anti-Aliasing vorliegt oder nicht.

Was antworten Sie, und wie begründen Sie es?

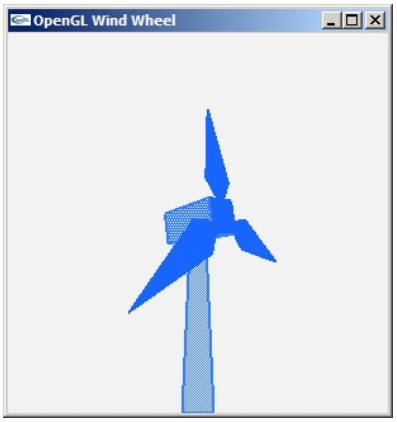

Abb. 3.3

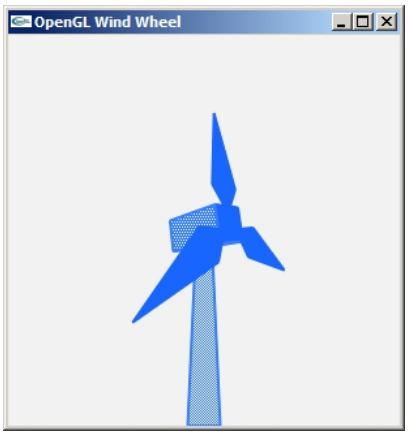

Abb. 3.4

n) Sie werden schließlich gefragt, wie es kommt, daß in den Bildern eine korrekte Verdeckung zwischen den einzelnen Windrad-Bauteilen (Mast, Kasten, Blättern) zu sehen ist, obwohl kein GLUT-Aufruf zur Einrichtung eines Tiefen-Puffers (Depth Buffering) erfolgt. Was antworten Sie?

```
/* WindWheel.c */
/*OpenGL-Darstellung eines Windrads*/
#include "WindWheel.h"

/*Globale Variablen: */
GLdouble nah=0., fern=0.;
float    bladeW=0., bladeH=0., boxX=0., boxY=0, boxZ=0.;
float    deltaX=10., deltaY=15., deltaZ=30., borderX=20., borderY=45.;
float    millCol[]={0.2f, 0.5f, 1.0f}, *myColor,
         millCol01[]={0.4f, 0.6f, 1.f}, millCol02[]={0.1f, 0.4f, 1.f};
float    boxVrtx[8][3], angle[3]={0.,0.,0.};
int      aa=0, winW=0, winH=0, win=0, swingX=1, swingY=1;
int      boxFace[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} };
/*************************************************************************/
   void mast(void)
/*************************************************************************/
/*Windrad-Mast zeichnen:*/
{ int j1=0;

  for (j1=0; j1<=OUTLINE; j1++)
  { if (j1==0)
    { glPolygonMode(GL_FRONT, GL_FILL);
      glEnable (GL_POLYGON_STIPPLE);
      glPolygonStipple (halftone);
    } else glPolygonMode(GL_FRONT, GL_LINE);

    glBegin(GL_POLYGON);
      glVertex3f(-boxX/2., -winH, 0.);    //u.l.
      glVertex3f( boxX/2., -winH, 0.);    //u.r.
      glVertex3f( boxX/4., -boxY/2., 0.); //o.r.
      glVertex3f(-boxX/4., -boxY/2., 0.); //o.l.
    glEnd();
  }
  return;
}

/*************************************************************************/
   void box(void)
/*************************************************************************/
/*Schaltkasten zeichnen:*/
{ int j1=0, j2=0;

  for (j1=0; j1<=OUTLINE; j1++)
  { if (j1==0)
    { glPolygonMode(GL_FRONT, GL_FILL);
      glEnable (GL_POLYGON_STIPPLE);
    } else  glPolygonMode(GL_FRONT, GL_LINE);

    for (j2=0; j2<6; j2++)
    { glPolygonStipple (quart3tone);
      if (j2==0 || j2==5) { glColor3fv (myColor);
                            glDisable (GL_POLYGON_STIPPLE); }

      glBegin(GL_POLYGON);
        glVertex3fv(boxVrtx[boxFace[j2][0]]);
        glVertex3fv(boxVrtx[boxFace[j2][1]]);
        glVertex3fv(boxVrtx[boxFace[j2][2]]);
        glVertex3fv(boxVrtx[boxFace[j2][3]]);
      glEnd();

      /*Standard wieder herstellen:*/
      glColor3fv (millCol);
      glEnable (GL_POLYGON_STIPPLE);
    }
  }
  return;
}
```

```
/*************************************************************************/
   void blade(void)
/*************************************************************************/
/*Rotor-Blatt zeichnen:*/
{ float bias=bladeH*.1, factor=1.;
  int j1=0;

  /*Auch Rueckseite erwuenscht:*/
  glDisable  (GL_CULL_FACE);
  glDisable (GL_POLYGON_STIPPLE);

  for (j1=0; j1<=OUTLINE; j1++)
  { if (j1==0)
    { glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
      glColor3fv (myColor);
    } else glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

    glBegin(GL_POLYGON);
      glVertex3f(0., bias, boxX);
      glVertex3f(bladeW/2., bias+bladeH/4., boxX);
      glVertex3f(0., bias+bladeH, boxX);
      glVertex3f(-bladeW/2., bias+bladeH/4., boxX);
    glEnd();
  }
  /*Standard wiederherstellen:*/
  glEnable  (GL_CULL_FACE);
  glColor3fv (millCol);

  return;
}

/*************************************************************************/
   void mill(void)
/*************************************************************************/
/*Windrad aus Blaettern zusammenstellen:*/
{
  blade();
glPushMatrix();
  glRotatef(120., 0., 0., 1.);
  blade();
glPopMatrix();
  glRotatef(240., 0., 0., 1.);
  blade();
  return;
}

/*************************************************************************/
  void draw (void)
/*************************************************************************/
/*Grafik erstellen, positioinieren, animieren:*/
{ glClear(GL_COLOR_BUFFER_BIT);
  /*Allgemeine Einstellungen:*/
  drawSettings();
  /*Transformationen betreffen Modell:*/
  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  /*Positionierung ins Sichtvolumen:*/
  glTranslatef(0., 0., -(nah+bladeH));

  /*Modell zusammenstellen und animieren:*/
  mast();

  box();

  mill();
  /*Darstellung:*/
  glFlush();
  return;
}
```

```
/*************************************************************************/
   void drawSettings(void)
/*************************************************************************/
/*Aktuelle Einstellungen in der Darstellung:*/
{
  /*Anti-Aliasing:*/
  if (aa)
  { glEnable (GL_LINE_SMOOTH);   glEnable (GL_BLEND);
    glBlendFunc (GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
  } else
  { glDisable (GL_LINE_SMOOTH); glDisable (GL_BLEND); }

  /*Coloring:*/
  if (angle[Y] <= -20.) myColor = millCol01; else
  if (angle[Y] >=  20.) myColor = millCol02; else
  myColor = millCol;

  return;
}

/*************************************************************************/
  void init (int w, int h)//(void)
/*************************************************************************/
/*Initialisierung - einmalige und fensterabhaengige Einstellungen - wird
  wg. glutReshapeFunc() auch vor Einrichten des Fensters aufgerufen:*/
{ winW = w;  winH = h;
  /*Manches soll sich nach der kleineren Fensterseite richten:*/
  win = MIN(w, h);

  /*Aktuelle Groesse der Zeichenflaeche:*/
  glViewport(0, 0, winW, winH);

  /*Loeschfarbe:*/
  glClearColor (.95f, .95f, .95f, 1.f);

  /*Farbe Windrad:*/
  glColor3fv (millCol);

  /*Strichstaerke:*/
  glLineWidth(2.f);

  /*Kein Drahtmodell:*/
  glEnable  (GL_CULL_FACE);

  /*Groesse Rotor-Blatt*/
  bladeW=win/6.;  bladeH=win/2.;

  /*Koordinaten Kasten-Eckpunkte:*/
  boxX=bladeW;  boxY=boxX;  boxZ=2*boxX;
  boxVrtx[0][X] = boxVrtx[3][X] = boxVrtx[4][X] = boxVrtx[7][X] = -boxX/2.;
  boxVrtx[1][X] = boxVrtx[2][X] = boxVrtx[5][X] = boxVrtx[6][X] =  boxX/2.;
  boxVrtx[0][Y] = boxVrtx[1][Y] = boxVrtx[4][Y] = boxVrtx[5][Y] = -boxY/2.;
  boxVrtx[2][Y] = boxVrtx[3][Y] = boxVrtx[6][Y] = boxVrtx[7][Y] =  boxY/2.;
  boxVrtx[0][Z] = boxVrtx[1][Z] = boxVrtx[2][Z] = boxVrtx[3][Z] =  boxZ/2.;
  boxVrtx[4][Z] = boxVrtx[5][Z] = boxVrtx[6][Z] = boxVrtx[7][Z] = -boxZ/2.;

  /*Sichtfeld-Einstellung auf 90 Grad (bzgl. kleiner Fensterseite):*/
  nah=win/2.; fern=9.5*winW;
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glFrustum (-winW/2., winW/2.,-winH/2., winH/2., nah, fern);

  /*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:  /*Windrad rotieren lassen:*/
              angle[Z] += deltaZ;
              if (angle[Z] >= 360.) angle[Z] -= 360.; else
              if (angle[Z] <=-360.) angle[Z] += 360.;
              /*Nach voller Umdrehung Windrad schwenken lassen:*/
              if (angle[Z] >= 0. && angle[Z] < deltaZ ||
                  angle[Z] <= 0. && angle[Z] > -deltaZ)
              { if (swingY > 0) angle[Y] += deltaY;
                else            angle[Y] -= deltaY;
                if (angle[Y] <= -borderY)
                { angle[Y] = -borderY;  swingY =  1;
                  if (swingX > 0) angle[X] += deltaX;
                  else            angle[X] -= deltaX;
                  if (angle[X] <= -borderX)
                  { angle[X] = -borderX;  swingX =  1; } else
                  if (angle[X] >=  borderX)
                  { angle[X] =  borderX;  swingX = -1; }
                } else
                if (angle[Y] >=  borderY)
                { angle[Y] =  borderY;  swingY = -1;
                  if (swingX > 0) angle[X] += deltaX;
                  else            angle[X] -= deltaX;
                  if (angle[X] <= -borderX)
                  { angle[X] = -borderX;  swingX =  1; } else
                  if (angle[X] >=  borderX)
                  { angle[X] =  borderX;  swingX = -1; }
                }   }                                                 break;
    case 'a': aa = 1 - aa;                                            break;
    case 'r': angle[X]=angle[Y]=angle[Z]=0.;                          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>     animate wind wheel");
  printf ("\n\r a        toggle wire frame AntiAliasing ");
  if (aa) printf ("(ON)"); else printf ("(OFF)");
  printf ("\n\r r        Reload");
  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]);
  draw();
  return;
}
/*********************************************************************/
  int main (int argc, char **argv)
/*********************************************************************/
{ glutInit(&argc, argv);
  glutCreateWindow("OpenGL Wind Wheel");
  glutReshapeFunc(init);
  glutDisplayFunc(draw);
  glutKeyboardFunc(key);
  glutMainLoop();
  return 0; }
```

```
/* WindWheel.h */
#ifndef WINDWHEEL_H
  #define WINDWHEEL_H
  #include <stdio.h> //wg. printf()
  #include <GL/glut.h>

  #define CON_CLS  "cls"
  #define CR       13
  #define ESC      27
  #define OUTLINE   1
  #ifndef MIN
    #define MIN(x,y) (( (x) < (y) ) ? (x) : (y))
  #endif  MIN
  #ifndef MAX
    #define MAX(x,y) (( (x) > (y) ) ? (x) : (y))
  #endif  MAX

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

  /*Prototypen:*/
  void blade(void);
  void box  (void);
  void draw (void);
  void drawSettings(void);
  void init (int w, int h);
  void key  (unsigned char key, int x, int y);
  void mast (void);
  void mill (void);

  /*Stippling:*/
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};

GLubyte quart3tone[] = {
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF,
       0xAA, 0xAA, 0xAA, 0xAA, 0xFF, 0xFF, 0xFF, 0xFF,
       0x55, 0x55, 0x55, 0x55, 0xFF, 0xFF, 0xFF, 0xFF};
#endif //WINDWHEEL_H
```

Platz für Notizen: