Kapitel 21

OpenGL

21.1 Grundlagen

OpenGL bietet eine Schnittstelle zwischen dem Anwenderprogramm und der Grafikhardware eines Computers zum Modellieren und Projizieren von dreidimensionalen Objekten.

Diese Schnittstelle enthält

- 200 Befehle in der *OpenGL Library* (beginnen mit `gl`) zum Verwalten von elementaren geometrischen Primiviten wie Punkte, Linien, Polygone, Bezierkurven und ihrer Attribute wie Farben und Normalenvektoren.
- 50 Befehle in der *OpenGL Utility Library* (beginnen mit `glu`) zum Verwalten von *NURBS* (non uniform rational b-splines) und *quadrics* (Körper mit durch quadratische Gleichungen beschreibbaren Oberflächen, z.B. Kugel, Zylinder) sowie zum vereinfachten Manipulieren von Projektionsmatrizen.

OpenGl enthält keine Routinen zur Benutzerein- und -ausgabe und zur Kommunikation mit dem (plattformspezifischen) Fenstersystem. Hierfür sind zuständig

- 30 Befehle im *OpenGL Utility Toolkit* (beginnen mit `glut`) zur Anbindung der von OpenGl gerenderten Ausgabe an das jeweilige Fenstersystem und zum Verwalten von höheren geometrischen Objekten wie Kugel, Kegel, Torus und Teapot.

Bei manchen OpenGl-Implementationen (z.B. X Window System) kann die Berechnung der Grafik und ihre Ausgabe auf verschiedenen Rechnern stattfinden. Der *Klient* ruft hierbei ein OpenGl-Kommando auf, mit Hilfe eines *Protokolls* wird es übertragen und der *Server* setzt es in ein Bild um.

Um die Kommunikationsbandbreite zu minimieren, arbeitet OpenGl als Zustandsmaschine. Dies bedeutet, daß einmal gesetzte Zustände (z.B. die Vordergrundfarbe) bis zu ihrem Widerruf beibehalten werden. Auch können komplexe geometrische Figuren unter einem Namen in einer *Display-Liste* eingetragen werden, so daß sie im Falle einer Wiederverwendung nicht erneut übertragen werden müssen sondern unter Angabe ihres Namens ausgewertet werden können.

Die Bestandteile der OpenGl Rendering Pipeline lauten:

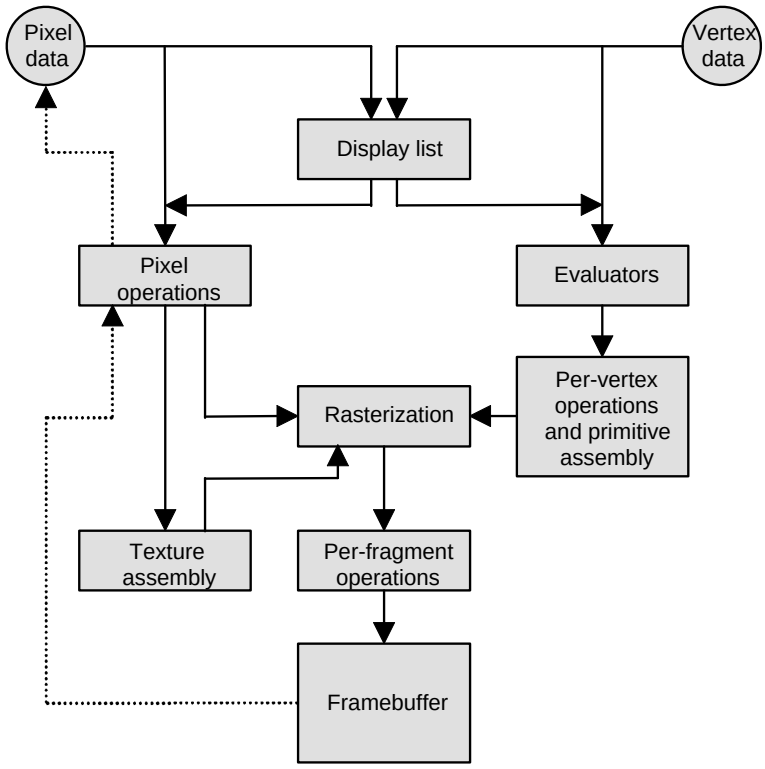


Abbildung 21.1: OpenGl Rendering Pipeline

- *Vertex data* (Knoten) oder *Pixel data* (Pixel) sind definierende Bestandteile eines zu berechnenden Bildes.
- *display lists* referenzieren Vertex und Pixel data.
- *Pixel operations* regeln das Pre- und Postprocessing von Pixeldaten.
- *evalutators* approximieren die durch Kontrollpunkte definierten parametrischen Kurven und Oberflächen zu Polygonzügen und Flächen samt Normalenvektoren.
- *Per Vertex operations* transformieren 3D-Welt-Koordinaten nach Vorgabe der syntetischen Kamera.
- *Primitive Assembly* erledigt clipping und backface removal.
- *Texture Assembly* bildet benutzerdefinierte Texturen auf Objekte ab.
- *Rasterization* überführt geometrische Daten und Pixelmengen zu *fragments*, welche die für eine Bildschirmkoordinate relevanten Informationen aufsammeln.
- *per fragment operations* bilden den Inhalt eines Fragments unter Berücksichtigung des Tiefenpuffers und unter Anwendung von Nebel- und Alpha-Masken auf eine Bildschirmkoordinate im *Frame buffer* ab.

21.2 Syntax

Durch Aufruf von Prozeduren manipuliert der Benutzer seine Daten. OpenGl verwendet dabei acht Datentypen. Der in Spalte 1 von Tabelle 21.1 genannte Suffix kündigt als letzter Buchstabe eines Kommandos den erwarteten Datentyp an.

Suffix	Datentyp	C-Korrespondenz	OpenGl Name
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	int	GLint
f	32-bit floating point	float	GLfloat, GLclampf
d	64-bit floating point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned int	GLuint, GLenum, GLbitfield

Tabelle 21.1: OpenGL Datentypen

Da OpenGl-Kommandos keine Überladung kennen, müssen für alle sinnvollen Parameterdatentypen unterschiedliche Prozedurnamen vorhanden sein.

Zum Beispiel erzeugt jeder der beiden Befehle

```
glVertex2i(5,3);
glVertex2f(5.0, 3.0);
```

einen zweidimensionalen Punkt mit x-Koordinate 5 und y-Koordinate 3.

Manche OpenGl-Kommandos enden mit dem Buchstaben **v** und signalisieren dadurch, daß sie einen Vektor als Übergabeparameter erwarten. Zum Beipiel kann das Setzen eines RGB-Farbwerts durch Angabe von drei Zahlen oder durch Angabe eines drei-elementigen Vektors geschehen:

```
glColor3f(1.0, 0.8, 0.8);
GLfloat farbvektor[] = {1.0, 0.8, 0.8};
glColor3fv(farbvektor);
```

Elementare geometrischen Figuren (wie z.B. Linien, Dreiecke, Vierecke, Polygone, etc.) werden mittels `glVertex*` durch eine Folge von Punkten $p_0, p_1, p_2, ..., p_{n-1}$ definiert, welche durch zwei Kommandos geklammert wird:

```
glBegin(MODUS);
...
glEnd();
```

Hierbei ist `MODUS` eine der folgenden vordefinierten OpenGl-Konstanten:

GL_POINTS	Punkte $p_0, p_1, p_2, ...$
GL_LINES	Gradenstücke $(p_0, p_1), (p_2, p_3), ...$
GL_LINE_STRIP	Linienzug $(p_0, p_1, p_2, p_3, ..., p_{n-1})$
GL_LINE_LOOP	geschlossener Linienzug $(p_0, p_1, p_2, p_3, ..., p_{n-1}, p_0)$
GL_TRIANGLES	Dreiecke $(p_0, p_1, p_2), (p_3, p_4, p_5), (p_6, p_7, p_8), ...$
GL_TRIANGLE_STRIP	Streifen von Dreiecken $(p_0, p_1, p_2), (p_2, p_1, p3), (p_2, p_3, p_4), ...$
GL_TRIANGLE_FAN	Fächer von Dreiecken $(p_0, p_1, p_2), (p_0, p_2, p_3), (p_0, p_3, p_4), ...$
GL_QUADS	Vierecke $(p_0, p_1, p_2, p_3), (p_4, p_5, p_6, p_7), (p_8, p_9, p_10, p_11), ...$
GL_QUAD_STRIP	Streifen von Vierecken $(p_0, p_1, p_3, p_2), (p_2, p_3, p_5, p_4), (p_4, p_5, p_7, p_6), ...$
GL_POLYGON	konvexes (!) Polygon $(p_0, p_1, p_2, ..., p_0)$

Es existieren Stacks von 4×4 Matrizen zur Transformation des Modells und zur Transformation der Projektion.

Die Auswahl geschieht durch `glMatrixMode()`. Durch die Anweisung

```
glMatrixMode(GL_PROJECTION);
```

bzw. durch die Anweisung

```
glMatrixMode(GL_MODEL_VIEW);
```

beziehen sich bis auf weiteres alle Matrix-Operationen auf den Projektions-Stack bzw. auf den Modell-Stack.

Zum Sichern und späteren Wiederherstellen der aktuellen Transformationsmatrix wird diese durch den Befehl `glPushMatrix()` kopiert und oben auf den Stack gelegt. Nun kann sie beliebig manipuliert werden. Ihr Originalzustand kann dann später durch `glPopMatrix()` wiederhergestellt werden.

Multiplikation der obersten Stackmatrix mit einer beliebigen Matrix M geschieht durch Definition der Matrix und durch Aufruf der Multiplikation:

```
GLfloat M[][] = {{2.0,0.0,0.0,0.0},
                 {0.0,4.0,0.0,0.0},
                 {0.0,0.0,3.0,0.0},
                 {0.0,0.0,0.0,2.0}};
glMultMatrix(M);
```

21.3 Programmbeispiele

Auf den folgenden Seiten werden einige durch deutsche Kommentare erweiterte OpenGl-Beispiele vorgestellt, die alle dem bei *Addison-Wesley* erschienenen Buch *OpenGL Programming Guide* entnommen wurden. Das Copyright für diese Programme liegt bei der Firma *Silicon Graphics*. Aus Platzgründen wurde auf eine explizite Wiederholung der Copyright-Klausel im jeweiligen Quelltext verzichtet. Eine komplette Sammlung aller Beispiele findet sich unter `http://trant.sgi.com/opengl/examples/redbook/redbook.html`.

hello.c zeigt in Orthogonalprojektion ein rotes Rechteck auf blauem Hintergrund. Es wird zunächst um eine Einheit in x-Richtung verschoben und danach um 45 Grad gedreht.

wire-cube.c zeigt in perspektivischer Projektion einen gelben Drahtgitterwürfel vor blauem Hintergrund. Der Würfel ist um 5 Einheiten nach hinten verschoben, in y-Richtung um den Faktor 1.5 skaliert und um 30 Grad bzgl. der y-Achse gedreht.

wire-prop-cube.c zeigt in perspektivischer Projektion einen gelben Drahtgittelwürfel vor blauem Hintergrund. Die Kamera wurde um 5 Einheiten nach hinten verschoben. Bei Verändern des Ausgabefensters durch den Benutzer wird die Funktion `reshape` aufgerufen, die eine Neuberechnung des Viewports durchführt.

click.c zeigt die Interaktion durch die Maus für den Farbwechsel eines Dreiecks. Die globale Variablen g wird durch den linken Maus-Button auf 1, durch den rechten Mausbutton auf 0 gesetzt. In der Funktion *display* wird dadurch entweder ein gelb oder ein rot erzeugt.

key.c zeigt die Interaktion durch die Tastatur für den Farbwechsel eines Dreiecks. Die globalen Variablen *r,g* und *b* werden durch Betätigen der gleichnamigen Tasten entsprechend gesetzt. In der Funktion *display* wird dadurch das RGB-Tripel auf *rot, grün* oder *blau* gesetzt.

planet.c zeigt die Interaktion durch die Tastatur für das Fortschalten einer Rotationsbewegung. Durch Drücken der *d*-Taste wird die Rotation der Erde um sich selbst um 10 Grad angestoßen, durch Drücken der *y*-Taste wird die Rotation der Erde um die Sonne um 5 Grad angestoßen.

smooth.c zeigt ein Dreieck mit Farbverlauf. Erreicht wird dies im Smooth-Shading-Modus durch Interpolation von RGB-Werten, welche an den Ecken des Dreiecks vorgegeben sind.

list.c zeigt eine Anwendung von Display-Listen. Unter dem Namen *listName* wird ein rotes, gefülltes Dreieck zusammen mit einer Translation hinterlegt. Ein wiederholter Aufruf dieses Namens erzeugt eine Sequenz von nebeneinander platzierten Dreiecken.

bezier-curve.c zeigt eine durch vier zwei-dimensionale Kontrollpunkte definierte Bezier-Kurve. Zur besseren Sichtbarkeit werden die Kontrollpunkte mit 5-facher Stärke platziert. Die Kurve entsteht durch Auswertung eines Evaluators längs eines von 0 nach 1 in 30 Schritten wandernden Parameters.

bezier-surface.c zeigt ein durch 16 drei-dimensionale Kontrollpunkte definiertes Bezier-Gitter. Das Gitter entsteht durch Auswertung eines Evaluators längs eines von 0 nach 1 in 8 Schritten und von 0 nach 1 in 30 Schritten wandernden Parameters.

teapot.c zeigt den klassischen Utah-Teapot in photo-realistischer Projektion. Berücksichtigt werden Reflektionskoeffizienten für diffuses und spekulares Licht, erzeugt von einer Lichtquelle.

teapot-rotate.c zeigt den klassischen Utah-Teapot in Bewegung. Erreicht wird dies durch eine 1-Grad-Drehung in der Funktion display, welche immer dann aufgerufen wird, wenn keine anderen Events zur Abarbeitung anstehen (Registrierung durch `glutIdleFunc`).

```
#include <GL/glut.h>                          /* Header fuer OpenGl utility toolkit */
#include <stdlib.h>                           /* Header fuer C-Library              */

void init(void) {                             /* Initialisierung                    */
  glClearColor(0.0, 0.0, 1.0, 0.0);           /* setze Blau als Hintergrundfarbe    */
  glMatrixMode(GL_PROJECTION);                /* betrifft Projektionsmatrix         */
  glLoadIdentity();                           /* beginne mit Einheitsmatrix         */
  gluOrtho2D(0.0, 4.0, 0.0, 3.0);             /* Orthogonalprojektions-Clip-Ebenen  */
}                                             /* in 2D: left,right,bottom,top       */

void display(void){                           /* Prozedur zum Zeichnen              */
  glClear(GL_COLOR_BUFFER_BIT);               /* alle Pixel zurecksetzen            */
  glMatrixMode(GL_MODELVIEW);                 /* betrifft Modelview-Matrix          */
  glLoadIdentity();                           /* beginne mit Einheitsmatrix         */
  glRotatef(45,0.0,0.0,1.0);                  /* drehe 45 Grad bzgl. der z-Achse    */
  glTranslatef(1.0, 0.0, 0.0);                /* verschiebe eine Einheit nach rechts*/
  glColor3f(1.0, 0.0, 0.0);                   /* Farbe rot                          */
  glBegin(GL_POLYGON);                        /* Beginn eines Polygons              */
    glVertex2f(0.25, 0.25);                   /* links unten                        */
    glVertex2f(0.75, 0.25);                   /* rechts unten                       */
    glVertex2f(0.75, 0.75);                   /* rechts oben                        */
    glVertex2f(0.25, 0.75);                   /* links oben                         */
  glEnd();                                    /* Ende des Polygons                  */
  glFlush();                                  /* direkt ausgeben                    */
}

int main (int argc, char ** argv) {           /* Hauptprogramm                      */
  glutInit(&argc, argv);                      /* initialisiere GLUT                 */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); /* single buffer, true color          */
  glutInitWindowSize(800, 600);               /* initiale Fenstergroesse            */
  glutInitWindowPosition(0,0);                /* initiale Fensterposition           */
  glutCreateWindow("hello");                  /* Fenster mit Aufschrift             */
  init();                                     /* rufe init auf                      */
  glutDisplayFunc(display);                   /* registriere display                */
  glutMainLoop();                             /* beginne Event-Schleife             */
  return 0;                                   /* ISO C verlangt Rueckgabe           */
}
```

hello.c: Rotes Quadrat, gedreht und verschoben, auf blauem Hintergrund

```
#include <GL/glut.h>                          /* Header fuer OpenGl utility toolkit */
#include <stdlib.h>                           /* Header fuer C-Library              */

void init(void) {                             /* Initialisierung                    */
  glClearColor(0.0, 0.0, 1.0, 0.0);           /* setze Blau als Hintergrundfarbe    */
  glMatrixMode(GL_PROJECTION);                /* ab jetzt: Projektionsmatrix        */
  glLoadIdentity();                           /* lade Einheitsmatrix                */
  glFrustum(-2,2,-1.5,1.5,2,20);              /* left,right,bottom,top,near,far     */
}

void display(void){                           /* Prozedur zum Zeichnen              */
  glClear(GL_COLOR_BUFFER_BIT);               /* alle Pixel zurecksetzen            */
  glMatrixMode(GL_MODELVIEW);                 /* ab jetzt:  Modellierungsmatrix     */
  glColor3f(1.0, 1.0, 0.0);                   /* Farbe gelb                         */
  glLoadIdentity();                           /* Einheitsmatrix                     */

  glTranslatef(0.0,0.0,-5.0);                 /* verschiebe 5 Einheiten nach hinten */
  glScalef(1.0, 1.5, 1.0);                    /* Skalierung in 3 Dimensionen        */
  glRotatef(30.0, 0.0, 1.0, 0.0);             /* rotiere 30 Grad um y-Achse         */
  glutWireCube(2.0);                          /* Drahtgitterwuerfel                 */
  glFlush();                                  /* direkt ausgeben                    */
}

int main (int argc, char ** argv) {           /* Hauptprogramm                      */
  glutInit(&argc, argv);                      /* initialisiere GLUT                 */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); /* single buffer, true color          */
  glutInitWindowSize(800, 600);               /* initiale Fenstergroesse            */
  glutInitWindowPosition(0,0);                /* initiale Fensterposition           */
  glutCreateWindow("wire-cube");              /* Fenster mit Aufschrift             */
  init();                                     /* rufe init auf                      */
  glutDisplayFunc(display);                   /* registriere display                */
  glutMainLoop();                             /* beginne Event-Schleife             */
  return 0;                                   /* ISO C verlangt Rueckgabe           */
}
```

wire-cube.c: Perspektivische Projektion eines Drahtgitterquaders, ohne Reshape

```
#include <GL/glut.h>                          /* Header f. OpenGl utility toolkit  */
#include <stdlib.h>                           /* Header fuer C-Library             */

void init(void) {                             /* Initialisierung                   */
  glClearColor(0.0, 0.0, 1.0, 0.0);           /* setze Blau als Hintergrundfarbe   */
}

void reshape(int w, int h) {
  GLfloat p = (GLfloat) w / (GLfloat) h;      /* Proportionalitaetsfaktor          */
  glViewport(0, 0, (GLsizei)w, (GLsizei)h);   /* setze Viewport                    */
  glMatrixMode(GL_PROJECTION);                /* betrifft Projektionsmatrix        */
  glLoadIdentity();                           /* lade Einheitsmatrix               */
  if (p > 1.0)                                /* falls breiter als hoch            */
       glFrustum(  -p,  p,-1.0,1.0,1.5,20.0); /* left,right,bottom,top,near,far    */
  else glFrustum(-1.0,1.0,-1/p,1/p,1.5,20.0); /* left,right,bottom,top,near,far    */
}

void display(void){                           /* Prozedur zum Zeichnen             */
  glClear(GL_COLOR_BUFFER_BIT);               /* alle Pixel zurecksetzen           */
  glMatrixMode(GL_MODELVIEW);                 /* ab jetzt:  Modellierungsmatrix    */
  glColor3f(1.0, 1.0, 0.0);                   /* Farbe gelb                        */
  glLoadIdentity();                           /* Einheitsmatrix                    */
  gluLookAt(0.0, 0.0, 5.0,                    /* Kamera-Standpunkt                 */
            0.0, 0.0, 0.0,                    /* Kamera-Fokussierpunkt             */
            0.0, 1.0, 0.0);                   /* Up-Vektor                         */
  glScalef(1.0, 1.5, 1.0);                    /* Skalierung in 3 Dimensionen       */
  glRotatef(30.0, 0.0, 1.0, 0.0);             /* rotiere 30 Grad um y-Achse        */
  glutWireCube(2.0);                          /* Drahtgitterwuerfel                */
  glFlush();                                  /* direkt ausgeben                   */
}

int main (int argc, char ** argv) {           /* Hauptprogramm                     */
  glutInit(&argc, argv);                      /* initialisiere GLUT                */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);  /* single buffer, true color         */
  glutInitWindowSize(800, 600);               /* initiale Fenstergroesse           */
  glutInitWindowPosition(0,0);                /* initiale Fensterposition          */
  glutCreateWindow("wire-cube-prop");         /* Fenster mit Aufschrift            */
  init();                                     /* rufe init auf                     */
  glutDisplayFunc(display);                   /* registriere display               */
  glutReshapeFunc(reshape);                   /* registriere reshape               */
  glutMainLoop();                             /* beginne Event-Schleife            */
  return 0;                                   /* ISO C verlangt Rueckgabe          */
}
```

wire-prop-cube.c: Perspektivische Projektion eines Drahtgittelquaders, mit Reshape

```
#include <GL/glut.h>                          /* Header fuer OpenGl utility toolkit */
#include <stdlib.h>                           /* Header fuer C-Library              */

GLfloat g=0.0;                                /* globale Variable                   */

void init(void) {                             /* Initialisierung                    */
  glClearColor(0.5, 0.5, 0.5, 0.0);           /* setze Grau als Hintergrundfarbe    */
  glMatrixMode(GL_PROJECTION);                /* betrifft Projektionsmatrix         */
  glLoadIdentity();                           /* beginne mit Einheitsmatrix         */
  gluOrtho2D(0.0, 1.5, 0.0, 1.0);             /* Orthogonalprojektions-Clip-Ebenen  */
}                                             /* in 2D: left,right, bottom, top     */

void display(void){                           /* Prozedur zum Zeichnen              */
  glClear(GL_COLOR_BUFFER_BIT);               /* alle Pixel zurecksetzen            */
  glColor3f(1.0,g,0.0);                       /* Farbe gemaess RGB-Tripel           */
  glBegin(GL_TRIANGLES);                      /* Beginn eines Dreiecks              */
    glVertex2f(0.25, 0.25);                   /* links unten                        */
    glVertex2f(0.75, 0.25);                   /* rechts unten                       */
    glVertex2f(0.25, 0.75);                   /* links oben                         */
  glEnd();                                    /* Ende des Dreiecks                  */
  glFlush();                                  /* direkt ausgeben                    */
}

void mouse(int button,int state,int x,int y){/* Maus-Klick bei Koordinate x,y      */
  switch(button) {                            /* analysiere den Mausevent           */
    case GLUT_LEFT_BUTTON:                    /* falls linke Taste gedrueckt        */
      if (state==GLUT_DOWN) g=1.0;            /* setzte Gruenanteil auf 1           */
      break;
    case GLUT_RIGHT_BUTTON:                   /* falls rechte Taste gedrueckt       */
      if (state==GLUT_DOWN) g=0.0;            /* setzte Gruenanteil auf 0           */
      break;
  } glutPostRedisplay();                      /* durchlaufe display erneut          */
}

int main (int argc, char ** argv) {           /* Hauptprogramm                      */
  glutInit(&argc, argv);                      /* initialisiere GLUT                 */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); /* single buffer, true color          */
  glutInitWindowSize(800, 600);               /* initiale Fenstergroesse            */
  glutInitWindowPosition(0,0);                /* initiale Fensterposition           */
  glutCreateWindow("click");                  /* Fenster mit Aufschrift             */
  init();                                     /* rufe init auf                      */
  glutDisplayFunc(display);                   /* registriere display                */
  glutMouseFunc(mouse);                       /* registriere mouse                  */
  glutMainLoop();                             /* beginne Event-Schleife             */
  return 0;                                   /* ISO C verlangt Rueckgabe           */
}
```

click.c: Interaktion durch Maus zur Farbwahl

```
#include <GL/glut.h>                          /* Header fuer OpenGl utility toolkit*/
#include <stdlib.h>                           /* Header fuer C-Library             */

GLfloat r=1.0, g=0.0, b=0.0;                  /* globales RGB-Tripel               */

void init(void) {                             /* Initialisierung                   */
  glClearColor(0.5, 0.5, 0.5, 0.0);           /* setze Grau als Hintergrundfarbe   */
  glMatrixMode(GL_PROJECTION);                /* betrifft Projektionsmatrix        */
  glLoadIdentity();                           /* beginne mit Einheitsmatrix        */
  gluOrtho2D(0.0, 1.5, 0.0, 1.0);             /* Orthogonalprojektions-Clip-Ebenen */
}                                             /* in 2D: left,right,bottom, top     */

void display(void){                           /* Prozedur zum Zeichnen             */
  glClear(GL_COLOR_BUFFER_BIT);               /* alle Pixel zurecksetzen           */
  glColor3f(r,g,b);                           /* Farbe gemaess RGB-Tripel          */
  glBegin(GL_TRIANGLES);                      /* Beginn eines Dreiecks             */
    glVertex2f(0.25, 0.25);                   /* links unten                       */
    glVertex2f(0.75, 0.25);                   /* rechts unten                      */
    glVertex2f(0.25, 0.75);                   /* links oben                        */
  glEnd();                                    /* Ende des Dreiecks                 */
  glFlush();                                  /* direkt ausgeben                   */
}

void key(unsigned char key, int x, int y){    /* Bei Tastendruck                   */
  switch(key) {                               /* analysiere den Tastendruck        */
    case 'r': r=1.0; g=0.0; b=0.0; break;     /* falls 'r': setze Tripel auf Rot   */
    case 'g': r=0.0; g=1.0; b=0.0; break;     /* falls 'g': setze Tripel auf Gruen */
    case 'b': r=0.0; g=0.0; b=1.0; break;     /* falls 'b': setze Tripel auf Blau  */
  }
  glutPostRedisplay();
}

int main (int argc, char ** argv) {           /* Hauptprogramm                     */
  glutInit(&argc, argv);                      /* initialisiere GLUT                */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB); /* single buffer, true color         */
  glutInitWindowSize(800, 600);               /* initiale Fenstergroesse           */
  glutInitWindowPosition(0,0);                /* initiale Fensterposition          */
  glutCreateWindow("key");                    /* Fenster mit Aufschrift            */
  init();                                     /* rufe init auf                     */
  glutDisplayFunc(display);                   /* registriere display               */
  glutKeyboardFunc(key);                      /* registriere key                   */
  glutMainLoop();                             /* beginne Event-Schleife            */
  return 0;                                   /* ISO C verlangt Rueckgabe          */
}
```

key.c: Interaktion durch Tastatur zur Farbwahl

```
#include <GL/glut.h>                                  /* OpenGl Utility Toolkit     */
#include <stdlib.h>                                   /* C-library                  */

int year = 0, day = 0;                                /* Variablen fuer Drehungen   */

void init(void) {
   glClearColor (0.0, 0.0, 1.0, 0.0);                 /* blauer Hintergrund         */
   glShadeModel (GL_FLAT);                            /* Flatshading                */
   glMatrixMode (GL_PROJECTION);                      /* ab jetzt: Projektion       */
   glLoadIdentity ();                                 /* lade Einheitsmatrix        */
   gluPerspective(60.0, 1.33, 1.0, 20.0);             /* Blickwinkel, w/h, near,far */
   glMatrixMode(GL_MODELVIEW);                        /* ab jetzt: Modelview        */
   glLoadIdentity();                                  /* lade Einheitsmatrix        */
   gluLookAt(-1.0,1.0,5.0,0.0,0.0,0.0,0.0,1.0,0.0);  /* Kamera,Fokussier,Up-Vektor */

}

void display(void){
   glClear (GL_COLOR_BUFFER_BIT);                        /* reset Pixels           */
   glColor3f (1.0, 1.0, 1.0);                            /* Farbe weiss            */
   glPushMatrix();                                       /* sichere Matrix         */
   glutWireSphere(1.0, 20, 16);                          /* zeichne die Sonne      */
   glRotatef ((GLfloat) year, 0.0, 1.0, 0.0);            /* Drehung um Sonne       */
   glTranslatef (2.0, 0.0, 0.0);                         /* Verschiebung von Sonne */
   glRotatef ((GLfloat) day, 0.0, 1.0, 0.0);             /* Erd-Drehung            */
   glutWireSphere(0.2, 10, 8);                           /* zeichne die Erde       */
   glPopMatrix();                                        /* restauriere Matrix     */
   glutSwapBuffers();                                    /* tausche Puffer         */
}

void keyboard (unsigned char key, int x, int y){
   switch (key) {                                        /* abhaengig von der Taste */
     case 'd': day =(day+10) % 360; break;               /* erhoehe Tag um 10       */
     case 'y': year=(year+5) % 360; break;}              /* erhoehe Jahr um 5       */
   glutPostRedisplay();                                  /* rufe display auf        */
}

int main (int argc, char ** argv) {          /* Hauptprogramm                        */
  glutInit(&argc, argv);                     /* initialisiere GLUT                   */
  glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB); /* double buffer, true color            */
  glutInitWindowSize(800, 600);              /* initiale Fenstergroesse              */
  glutInitWindowPosition(0,0);               /* initiale Fensterposition             */
  glutCreateWindow("planet");                /* Fenster mit Aufschrift               */
  init(); glutDisplayFunc(display);          /* rufe init auf, registriere display   */
  glutKeyboardFunc(keyboard);                /* registriere keyboard                 */
  glutMainLoop();                            /* beginne Event-Schleife               */
  return 0;                                  /* ISO C verlangt Rueckgabe             */
}
```

planet.c: Interaktion durch Tastatur zur Rotationsbewegung

```
#include <GL/glut.h>                          /* Header fuer OpenGl utility toolkit*/
#include <stdlib.h>                           /* header fuer C-library             */

void init(void) {
   glClearColor (0.0, 0.0, 0.0, 0.0);         /* Hintergrundfarbe schwarz          */
   glShadeModel (GL_SMOOTH);                  /* smooth shading                    */
}

void reshape (int w, int h) {
   GLfloat p = (GLfloat) w / (GLfloat) h;     /* berechne Fensterverhaeltnis       */
   glViewport (0,0,(GLsizei)w,(GLsizei)h);    /* lege Viewport fest                */
   glMatrixMode (GL_PROJECTION);              /* ab jetzt: Projektionsmatrix       */
   glLoadIdentity ();                         /* lade Einheitsmatrix               */
   if (p > 1.0)                               /* falls breiter als hoch            */
        gluOrtho2D (0.0,p*30.0,0.0,30.0  );   /* left,right,bottom,top             */
   else gluOrtho2D (0.0,  30.0,0.0,30.0/p);   /* left,right,bottom,top             */
   glMatrixMode(GL_MODELVIEW);                /* ab jetzt: Modelview-Matrix        */
   glLoadIdentity ();                         /* lade Einheitsmatrix               */
}

void display(void) {
   glClear (GL_COLOR_BUFFER_BIT);             /* reset Farbpuffer                  */
   glBegin (GL_TRIANGLES);                    /* Beginn der Dreiecks-Knoten        */
   glColor3f  ( 1.0,  0.0, 0.0);              /* Farbe rot                         */
   glVertex2f (25.0,  5.0);                   /* Knoten unten rechts               */
   glColor3f  ( 0.0,  1.0, 0.0);              /* Farbe gruen                       */
   glVertex2f ( 5.0, 25.0);                   /* Knoten links oben                 */
   glColor3f  ( 0.0,  0.0, 1.0);              /* Farbe blau                        */
   glVertex2f ( 5.0,  5.0);                   /* Knoten unten links                */
   glEnd();                                   /* Ende der Dreiecks-Knoten          */
   glFlush ();                                /* direkt ausgeben                   */
}


int main (int argc, char ** argv) {           /* Hauptprogramm                     */
  glutInit(&argc, argv);                      /* initialisiere GLUT                */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);  /* single buffer, true color         */
  glutInitWindowSize(800, 600);               /* initiale Fenstergroesse           */
  glutInitWindowPosition(0,0);                /* initiale Fensterposition          */
  glutCreateWindow("smooth");                 /* Fenster mit Aufschrift            */
  init();                                     /* rufe init auf                     */
  glutDisplayFunc(display);                   /* registriere display               */
  glutReshapeFunc(reshape);                   /* registriere reshape               */
  glutMainLoop();                             /* beginne Event-Schleife            */
  return 0;                                   /* ISO C verlangt Rueckgabe          */
}
```

smooth.c: Dreieck mit Farbverlauf durch Interpolation

```
#include <GL/glut.h>                                    /* OpenGl utility toolkit    */
#include <stdlib.h>                                     /* fuer C-Library            */

GLuint listName;                                        /* Handle fuer Display-Liste */

void init (void) {
   listName = glGenLists (1);                           /* trage Namen ein           */
   glNewList (listName, GL_COMPILE);                    /* erzeuge Display-Liste     */
      glColor3f (1.0, 0.0, 0.0);                        /* Farbe rot                 */
      glBegin (GL_TRIANGLES);                           /* Beginn Dreiecks-Punkte    */
        glVertex2f (0.0, 0.0);                          /* Knoten links unten        */
        glVertex2f (1.0, 0.0);                          /* Knoten rechts unten       */
        glVertex2f (0.0, 1.0);                          /* Knoten links oben         */
      glEnd ();                                         /* Ende der Dreiecksliste    */
      glTranslatef (1.5, 0.0, 0.0);                     /* Verschiebe nach rechts    */
   glEndList ();                                        /* Ende der Display-Liste    */
   glShadeModel (GL_FLAT);                              /* verwende Flat-Shading     */
}

void reshape(int w, int h) {                            /* Reshape-Routine           */
   GLfloat p = (GLfloat) w / (GLfloat) h;               /* Fensterverhaeltnis        */
   glViewport(0, 0, w, h);                              /* setze Viewport            */
   glMatrixMode(GL_PROJECTION); glLoadIdentity();       /* ab jetzt Projektion       */
   if (p > 1 ) gluOrtho2D (0.0, 2.0*p, -0.5, 1.5);      /* links,rechts, unten, oben */
         else  gluOrtho2D (0.0, 2.0, -0.5/p, 1.5/p);    /* links,rechts, unten, oben */
   glMatrixMode(GL_MODELVIEW); glLoadIdentity();        /* ab jetzt Modelview-Matrix */
}

void display(void) {                                    /* Anzeige-Routine           */
   GLuint i;                                            /* Integer-Variable          */
   glClear (GL_COLOR_BUFFER_BIT);                       /* reset Farb-Puffer         */
   for (i=0; i<10; i++) glCallList(listName);           /* rufe Display-Liste auf    */
   glBegin (GL_LINES);                                  /* Beginn Linien-Punkte      */
     glVertex2f (0.0,0.5);                              /* ein Knoten                */
     glVertex2f(15.0,0.5);                              /* noch ein Knoten           */
   glEnd ();                                            /* Ende Linen-Punkte         */
   glFlush ();                                          /* direkt ausgeben           */
}

int main (int argc, char ** argv) {                     /* Hauptprogramm             */
  glutInit(&argc, argv);                                /* initialisiere GLUT        */
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);            /* single buffer, true color */
  glutInitWindowSize(800, 100);                         /* initiale Fenstergroesse   */
  glutInitWindowPosition(0,0);                          /* initiale Fensterposition  */
  glutCreateWindow("list");                             /* Fenster mit Aufschrift    */
  init();                                               /* rufe init auf             */
  glutDisplayFunc(display);                             /* registriere display       */
  glutReshapeFunc(reshape);                             /* registriere reshape       */
  glutMainLoop(); return 0;                             /* beginne Event-Schleife    */
}
```

list.c : Dreiecke definiert durch Display-Liste

```
#include <GL/glut.h>
#include <stdlib.h>

GLfloat ctrlpoints[4][3] = {
        {-4.0, -4.0, 0.0}, {-2.0, 4.0, 0.0},
        { 2.0, -4.0, 0.0}, { 4.0, 4.0, 0.0}};

void init(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glShadeModel(GL_FLAT);
   glMap1f(GL_MAP1_VERTEX_3, 0.0, 1.0, 3, 4, &ctrlpoints[0][0]);
   glEnable(GL_MAP1_VERTEX_3);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-8.0, 8.0, -6.0, 6.0, -5.0, 5.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void display(void)
{
   int i;
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glBegin(GL_LINE_STRIP);
      for (i=0; i<=30; i++) glEvalCoord1f((GLfloat) i/30.0);
   glEnd();
   glPointSize(5.0);
   glColor3f(1.0, 1.0, 0.0);
   glBegin(GL_POINTS);
      for (i=0; i<4; i++) glVertex3fv(&ctrlpoints[i][0]);
   glEnd();
   glFlush();
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (800, 600);
   glutInitWindowPosition (0,0);
   glutCreateWindow ("bezier-curve");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

bezier-curve.c: zweidimensionale Bezier-Kurve

```
#include <stdlib.h>
#include <GL/glut.h>

GLfloat ctrlpoints[4][4][3] = {
   {{-1.5, -1.5,  4.0}, {-0.5, -1.5,  2.0}, {0.5, -1.5, -1.0}, {1.5, -1.5,  2.0}},
   {{-1.5, -0.5,  1.0}, {-0.5, -0.5,  3.0}, {0.5, -0.5,  0.0}, {1.5, -0.5, -1.0}},
   {{-1.5,  0.5,  4.0}, {-0.5,  0.5,  0.0}, {0.5,  0.5,  3.0}, {1.5,  0.5,  4.0}},
   {{-1.5,  1.5, -2.0}, {-0.5,  1.5, -2.0}, {0.5,  1.5,  0.0}, {1.5,  1.5, -1.0}}
};

void init(void) {
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glMap2f(GL_MAP2_VERTEX_3, 0, 1, 3, 4, 0, 1, 12, 4, &ctrlpoints[0][0][0]);
   glEnable(GL_MAP2_VERTEX_3);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(-8.0, 8.0, -6.0, 6.0, -5.0, 5.0);
   glMatrixMode(GL_MODELVIEW);
}
void display(void) {
   int i, j;
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glColor3f(1.0, 1.0, 1.0);
   glLoadIdentity();
   glRotatef(60.0, 1.0, 1.0, 1.0);
   for (j = 0; j <= 8; j++) {
      glBegin(GL_LINE_STRIP);
        for (i = 0; i <= 30; i++) glEvalCoord2f((GLfloat)i/30.0, (GLfloat)j/8.0);
      glEnd();
      glBegin(GL_LINE_STRIP);
        for (i = 0; i <= 30; i++) glEvalCoord2f((GLfloat)j/8.0, (GLfloat)i/30.0);
      glEnd();
   }
   glFlush();
}


int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (800, 600);
   glutInitWindowPosition (0,0);
   glutCreateWindow ("bezier-surface");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

bezier-surface.c : dreidimensionale Bezierkurve

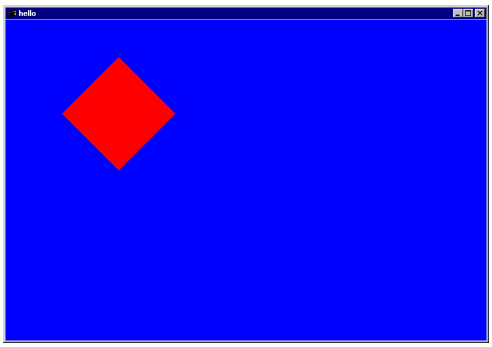
```
#include <GL/glut.h>
#include <stdlib.h>

void init(void) {
   GLfloat mat_diffuse[]      = { 1.0, 0.3, 0.3, 1.0 };
   GLfloat mat_specular[]     = { 1.0, 1.0, 1.0, 1.0 };
   GLfloat mat_shininess[]    = { 50.0 };
   GLfloat light_position[]   = { 1.0, 1.0, 1.0, 0.0 };
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);
   glMaterialfv (GL_FRONT,  GL_DIFFUSE,   mat_diffuse);
   glMaterialfv (GL_FRONT,  GL_SPECULAR,  mat_specular);
   glMaterialfv (GL_FRONT,  GL_SHININESS, mat_shininess);
   glLightfv    (GL_LIGHT0, GL_POSITION,  light_position);
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
   glEnable(GL_DEPTH_TEST);
}

void reshape (int w, int h){
   GLfloat p = (GLfloat) w / (GLfloat) h;
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (p > 1.0) glOrtho(-2*p,2*p,-1.5,1.5,-10.0,10.0);
           else glOrtho(-1.5,1.5,-2/p,2/p,-10.0,10.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void display(void) {
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glRotatef(-40.0, 0.0, 1.0, 0.0);
   glRotatef( 20.0, 1.0, 0.0, 0.0);
   glutSolidTeapot(1.0);
   glFlush();
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
   glutInitWindowSize (800, 600);
   glutInitWindowPosition (0,0);
   glutCreateWindow ("teapot");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop(); return 0;
}
```

teapot.c: Teekanne mit Beleuchtung

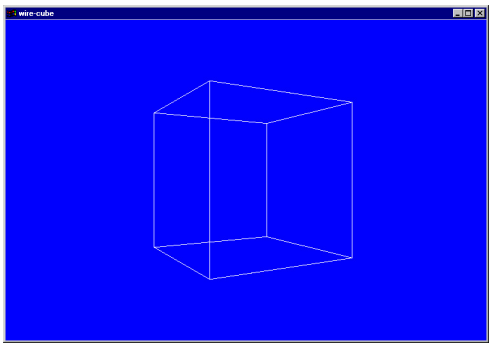
```
#include <GL/glut.h>                                    /* OpenGl Utiltiy Toolkit */
#include <stdlib.h>                                     /* C-library              */


void init(void) {
   GLfloat mat_diffuse[]      = { 1.0, 0.3, 0.3, 1.0 };  /* diffuse Farbe          */
   GLfloat mat_specular[]     = { 1.0, 1.0, 1.0, 1.0 };  /* spekulare Farbe        */
   GLfloat mat_shininess[]    = { 50.0 };                /* Reflexionskoeffizient  */
   GLfloat light_position[]   = { 1.0, 1.0, 1.0, 0.0 };  /* Lichtquellenposition   */
   glClearColor (0.0, 0.0, 0.0, 0.0);                   /* Hintergrundfarbe       */
   glShadeModel (GL_SMOOTH);                            /* smooth shading         */
   glMaterialfv(GL_FRONT, GL_DIFFUSE,  mat_diffuse);    /* diffuse Farbzuweisung  */
   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);   /* spekulare Farbzuweisung*/
   glMaterialfv(GL_FRONT, GL_SHININESS,mat_shininess);  /* Reflexionszuweisung    */
   glLightfv   (GL_LIGHT0,GL_POSITION, light_position); /* Lichtpositionszuweisung*/
   glEnable(GL_LIGHTING); glEnable(GL_LIGHT0);          /* Beleuchtung aktivieren */
   glEnable(GL_DEPTH_TEST);                             /* Tiefentest aktiveren   */
   glMatrixMode (GL_PROJECTION);                        /* ab jetzt Projektion    */
   glLoadIdentity();                                    /* lade Einheitsmatrix    */
   glOrtho(-2.0,2.0,-1.5,1.5,-10.0,10.0);               /* links,rechts,unten,oben*/
   glMatrixMode(GL_MODELVIEW);                          /* ab jetzt Modelview     */
   glLoadIdentity();                                    /* lade Einheitsmatrix    */
   glRotatef( 20.0, 1.0, 0.0, 0.0);                     /* 20 Grad um x-Achse     */
}

void display(void) {
   glClear (GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);   /* reset Buffer           */
   glRotatef((GLfloat) 1, 0.0, 1.0, 0.2);               /* 1 Grad um Drehachse    */
   glutSolidTeapot(1);                                  /* zeichne Teapot         */
   glutSwapBuffers();                                   /* wechsele Puffer        */
}


int main(int argc, char** argv) {
   glutInit(&argc, argv);                               /* initialisiere GLUT     */
   glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH); /* Double buffer, ...     */
   glutInitWindowSize (800, 600);                       /* init. Fenstergroesse   */
   glutInitWindowPosition (0,0);                        /* init. Fensterposition  */
   glutCreateWindow ("teapot-rotate");                  /* Fenster mit Aufschrift */
   init ();                                             /* Initialisierung        */
   glutDisplayFunc(display);                            /* registriere display    */
   glutIdleFunc(display);                               /* bei Leerlauf: display  */
   glutMainLoop();                                      /* starte Hauptschleife   */
   return(0);
}
```

teapot-rotate.c: rotierende Teekanne

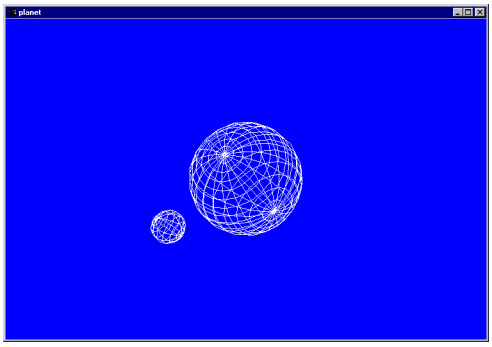
21.4 Screenshots



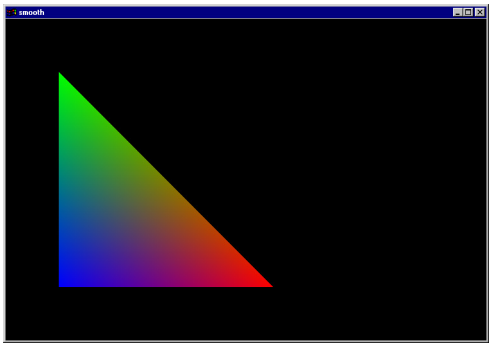
hello.exe



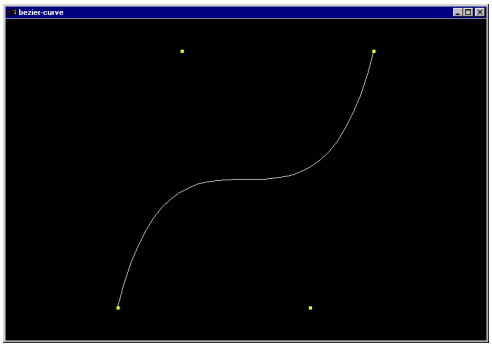
wire-cube.exe



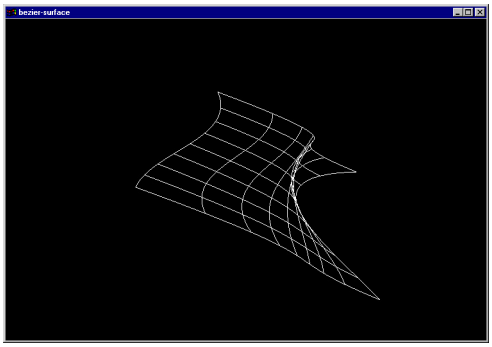
planet.exe



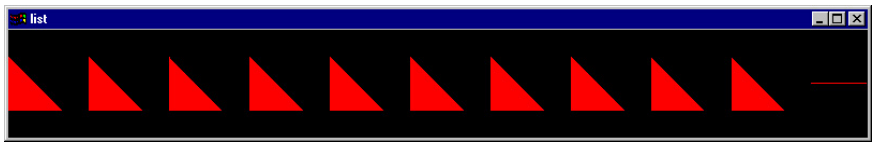
smooth.exe



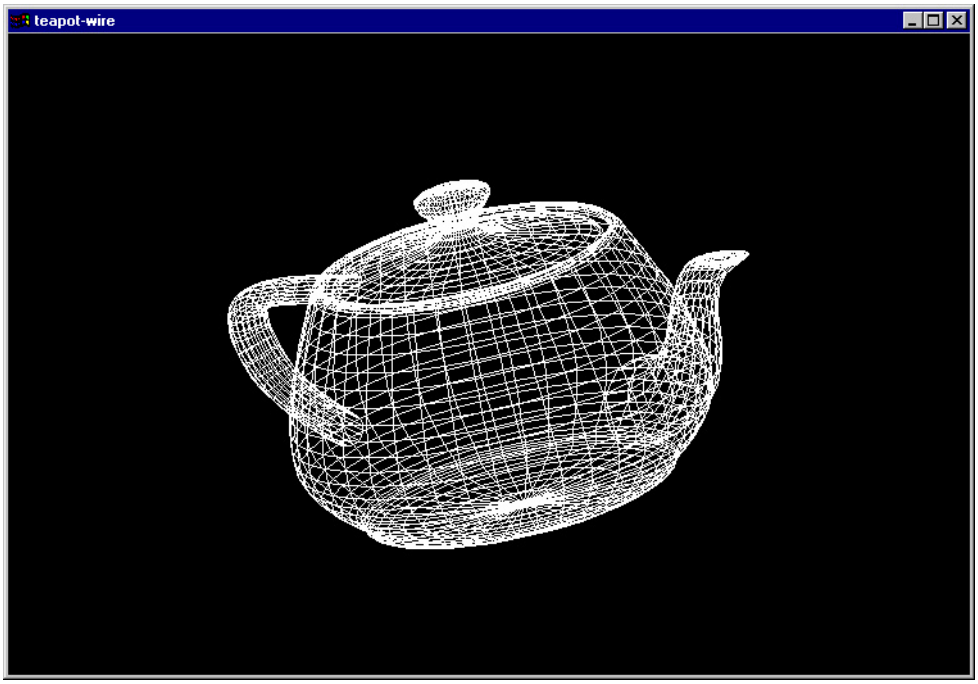
bezier-curve.exe



bezier-surface.exe



list.exe



Drahtgitter für teapot

teapot.exe