D O K U M E N T A T I O N

B e r u f s b e z o g e n e P r o j e k t e 2 0 0 6
I n d u s t r i e s c h u l e C h e m n i t z

Klasse: KM03-2

Gruppe: 2

Mitglieder: Rico Wündisch

Thema: Erstellen Sie eine 3D-Präsentation verschiedener Radbauformen für den Einsatz im lernfeldorientierten Unterricht!

Erarbeiten Sie dazu eine Dokumentation mit erforderlichen Konstruktionszeichnungen!

Inhalt

Anhang

Vorwort

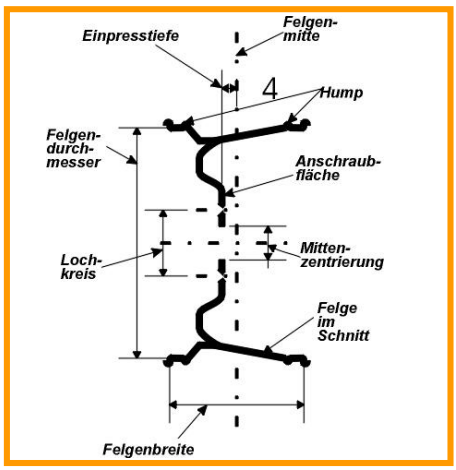
Dieses Projekt ist für den Einsatz im lernfeldorientierten Unterricht entwickelt worden. Es soll Auszubildenden Informationen zum Aufbau verschiedener Radbauformen liefern und diese direkt am Objekt sichtbarmachen. Dadurch soll die Suche nach Unterlagen in verschiedenen Lehr- oder Tabellenbüchern entfallen und es brauchen entsprechende Sachverhalte nicht mehr umständlich an platzraubenden Modellen und Rädern erklärt werden, sofern diese überhaupt zur Verfügung stehen.

Mit dieser Anwendung soll jeder selbst dazu in der Lage sein, sich Informationen zu beschaffen und diese gegebenenfalls in seine Unterlagen übernehmen.

Anregungen

Hintergrund, dieses Projekt zu entwerfen, waren 3D-Darstellungen am PC aus dem Unterricht. Mit ihnen war es für mich persönlich einfacher, gewisse Zusammenhänge von Radstellgrößen oder Fahrwerksbauteilen zu verstehen, da sie die Theorie besser erklären konnten als Lehrbücher. Einen Vergleich bieten die unteren Abbildungen.

2D-Darstellung



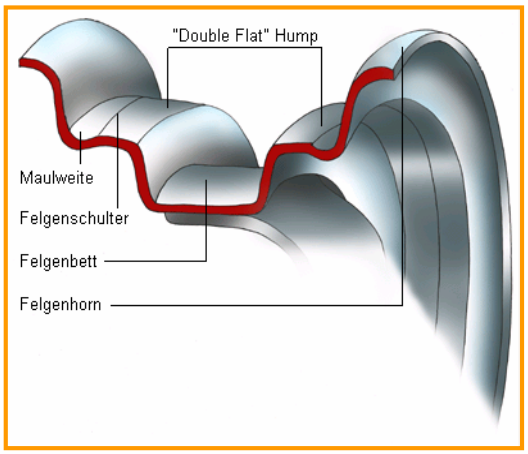
Quelle: www.reifen-heilmaier.de

3D-Darstellung

Quelle: VisualWheel

Da die Darstellung von Radbauformen auch auf diese Weise möglich ist und die Industrieschule aufgrund der jetzigen räumlichen Gegebenheiten eher mit Platzmangel zu kämpfen hat, habe ich mich entschieden, dieses Programm zu entwerfen.

Außerdem ist es meiner Meinung besser, verschiedene Bemaßungen an einen 3D-Modell darzustellen, als diese an unzähligen Felgen teilweise ziemlich umständlich zu erklären. Dabei denke ich unter Anderem an die Einpresstiefe, die ich persönlich besser mit diesem Programm als an einer richtigen Felge erklären kann.



Quelle: Das Auto 4D - glasklar EDITION 2001

Weiterhin gibt es aber auch Medien, in denen Zusammenhänge falsch dargestellt sind, wie im linken Beispiel.
Hier wird die Felgenschulter als Maulweite deklariert.

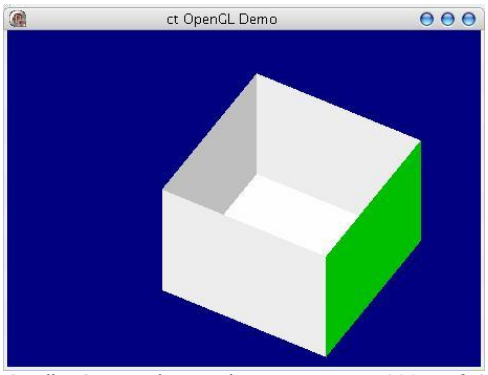
Solche Dinge haben mich veranlasst, eine Anwendung zu entwerfen, die solche Fehler nicht enthält.

Projektablauf

Um die verschiedenen Radbauformen am PC überhaupt zeigen zu können, ist es nötig, zu wissen, in welcher Entwicklungsumgebung das Programm realisiert werden kann. Dazu war zum Einen eine geeignete Möglichkeit erforderlich, mit der man 3D-Modelle anschaulich darstellen kann und zum Anderen eine geeignete Programmiersprache, mit der man diese programmieren kann.

Nach langer Suche kam OpenGL, eine plattform- und programmiersprachenunabhängige Entwicklungsumgebung für 3D-Computergrafikinen, in Betracht, auf die man mittels Delphi zugreifen kann. Bei beiden handelt es sich um voneinander getrennte Programmiersprachen mit eigenen Besonderheiten und Tücken.

Da diese völliges Neuland für mich waren, musste ich mich vorab in beide Programmiersprachen einarbeiten, um mich damit bekannt zu machen und herauszufinden, wie und vor Allem mit welchem Aufwand das Thema des Projektes am Besten umgesetzt werden kann. Mit einigen Beispielprogrammen aus dem Internet und Vorlagen aus der Industrieschule Chemnitz, war ich in der Lage, eine funktionsfähige Programmvorlage nach meinen Vorstellungen zu bearbeiten und zu verändern.



Quelle: Screenshot ct_demo1.exe - c't 1999, Heft 20 Author Ingo Wulf

Dabei handelte es sich um eine Beispielanwendung der Computerzeitschrift c't aus dem Jahr 1999. In dieser Anwendung war ein offener Würfel dargestellt, den man in jede beliebige Richtung drehen und dessen Oberfläche auf verschiedene Arten dargestellt werden konnte.

Diese Grundlage habe ich genutzt, um eine 3D-Präsentation zu entwickeln, die auf dem gleichen Prinzip aufbaut, jedoch um eine Vielzahl an Funktionen und Darstellungsmöglichkeiten erweitert wurde.

Um das Programm jedoch am PC erzeugen zu können, war ich auf der Suche nach geeignetem Material und Unterlagen über Felgen, Räder und Radbauformen, wobei mir die eigenen Aufzeichnungen aus dem Unterricht sowie diversere Darstellungen in verschiedenen Lehr- und Tabellenbüchern und dem Internet sehr hilfreich waren. Da das Material jedoch nicht geeignet war, um es unbearbeitet zu präsentieren bzw. in das Projekt oder das Programm zu übernehmen, musste ich das Material noch überarbeiten und eigene Konstruktionszeichnungen erstellen.

Um die Konstruktionszeichnungen in einer entsprechenden Form zu präsentieren, habe ich mich entschlossen, diese in einer CAD-Anwendung zu erstellen, in die ich mich ebenfalls erst einarbeiten musste. Dadurch hatte ich aber dann die Möglichkeit, Bemaßungen relativ schnell vom Original in die OpenGL-Umgebung zu übertragen und hier und da noch einige Änderungen am Felgenmodell vornehmen zu können, ohne andauernd neue Zeichnungen anfertigen zu müssen. Zu diesen ist Folgendes zu sagen: Bei dem Felgenmodell handelt es sich um die vereinfachte Form einer Schrägschulterfelge (6J x 15 H2 ET 34).

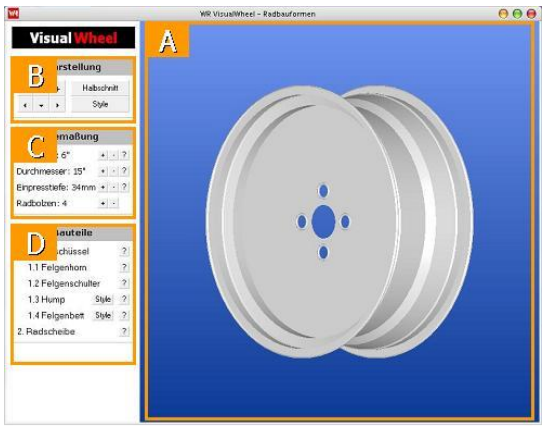
Nachdem die Konstruktionszeichnungen fertiggestellt waren, habe ich diese am PC in ein 3D-Modell umgewandelt und verschiedene Möglichkeiten zur Darstellung und Bemaßung in die Anwendung eingearbeitet. Dabei bin ich aber nur vom grundsätzlichen Aufbau der gewählten Felge ausgegangen, da die Programmierung weiterer Radbauformen wie Leichtmetallrad oder Drahtspeichenrad den zeitlichen Rahmen des Projektes immens gesprengt hätten. Welche Bauteile, Bemaßungen und Funktionen in der Anwendung enthalten sind, kann man in der Programmbeschreibung nachlesen.

Zum Programmieren des 3D-Modells muss man sagen, dass der vorab relativ hoch eingeschätzte Zeitraum dem tatsächlichen Aufwand nicht gerecht wurde und so die eine oder andere Überstunde außerhalb des angegebenen Projektzeitraumes notwendig war, um das Programm fertigzustellen. Es gab immer wieder Rückschläge wegen falsch durchdachter Programmkomponenten. Einige Dinge mussten neu bzw. anders realisiert werden, als sie anfänglich gedacht waren.

Nachdem schließlich das Programm funktionierte und alle Fehler behoben waren, habe ich den Ablauf der Projektarbeit dokumentiert und eine geeignete Präsentation erstellt, um zu zeigen, wie aufwendig eine solche Arbeit sein kann, falls sich ein anderer dazu entscheidet, dieses Projekt fortzuführen bzw. diese Arbeit für weitere Denkansätze als Grundlage zu nutzen. Schließlich kann dieses Projekt bzw. das Programm um einige Funktionen erweitert werden. Näheres dazu kann man unter den Erweiterungsmöglichkeiten nachlesen.

Programmbeschreibung

Die Anwendung trägt den Namen „VisualWheel.exe“. Nach deren Start zeigt sich dem Anwender die rechts dargestellte Oberfläche, die sich in vier Bereiche unterteilen lässt.

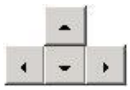


A - Anzeigebereich

In diesem Bereich werden alle Bauteile und Bemaßungen der Felge dargestellt.

B - Darstellung

In diesem Bereich lässt sich die Felge im Anzeigebereich drehen, zoomen und Ihre Darstellung ändern.



Felge drehen

Felge vergrößern oder verkleinern

Felge im Halbschnitt oder in Volldarstellung anzeigen

Felgenoberfläche ändern

C - Bemaßung

In diesem Bereich können Bemaßungen der Felge verändert, ein- und ausgeblendet werden.

Maß oder Anzahl vergrößern oder verkleinern

Bemaßung ein- oder ausblenden

D - Bauteile

In diesem Bereich können Bauteile farbig de- oder markiert oder deren Bauart verändert werden.

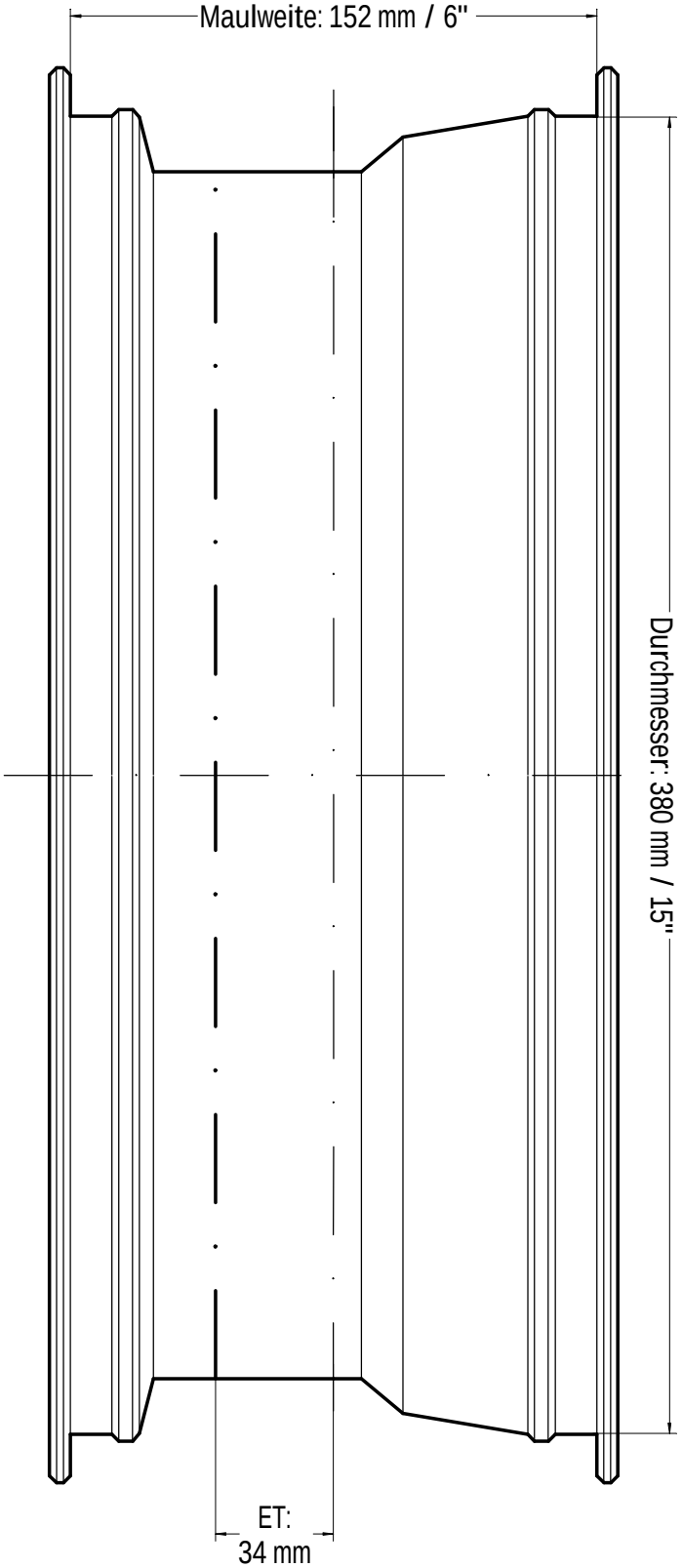
Bauteil de- oder markieren

Bauteil-Bauart verändern

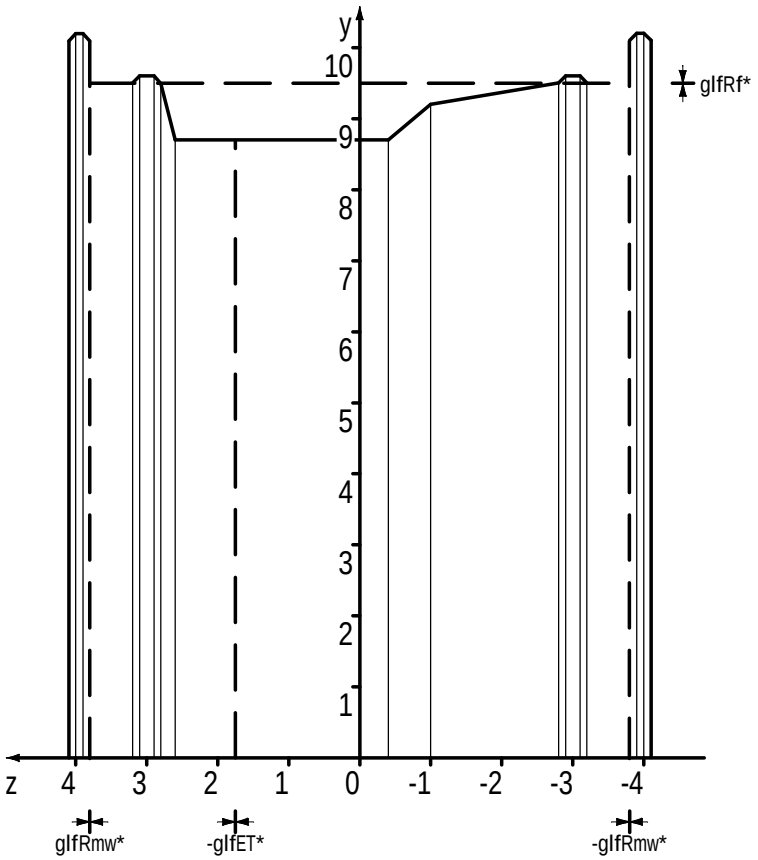
Erweiterungsmöglichkeiten

Aufgrund der geringen Zeitspanne der Projektwoche ist der Umfang des Programms noch nicht ausgereizt und ist somit erweiterbar. Es ist zum Beispiel noch möglich, andere Radbauformen wie Drahtspeichen- oder Leichtmetallrad in die Anwendung einzubinden. Außerdem geht das Programm nicht auf geteilte Felgen ein und veranschaulicht auch keine mehrteilige Felgenschüsseln. Neben diesen Themen kann die Anwendung aber auch mit gewissen Abänderungen dazu genutzt werden, modulhaft andere 3D-Modelle einzubinden.

Aus diesem Grund sind der Dokumentation alle verwendeten Konstruktionszeichnungen sowie der komplette Quelltext des Programms beigefügt. Diese befinden sich im Anhang.

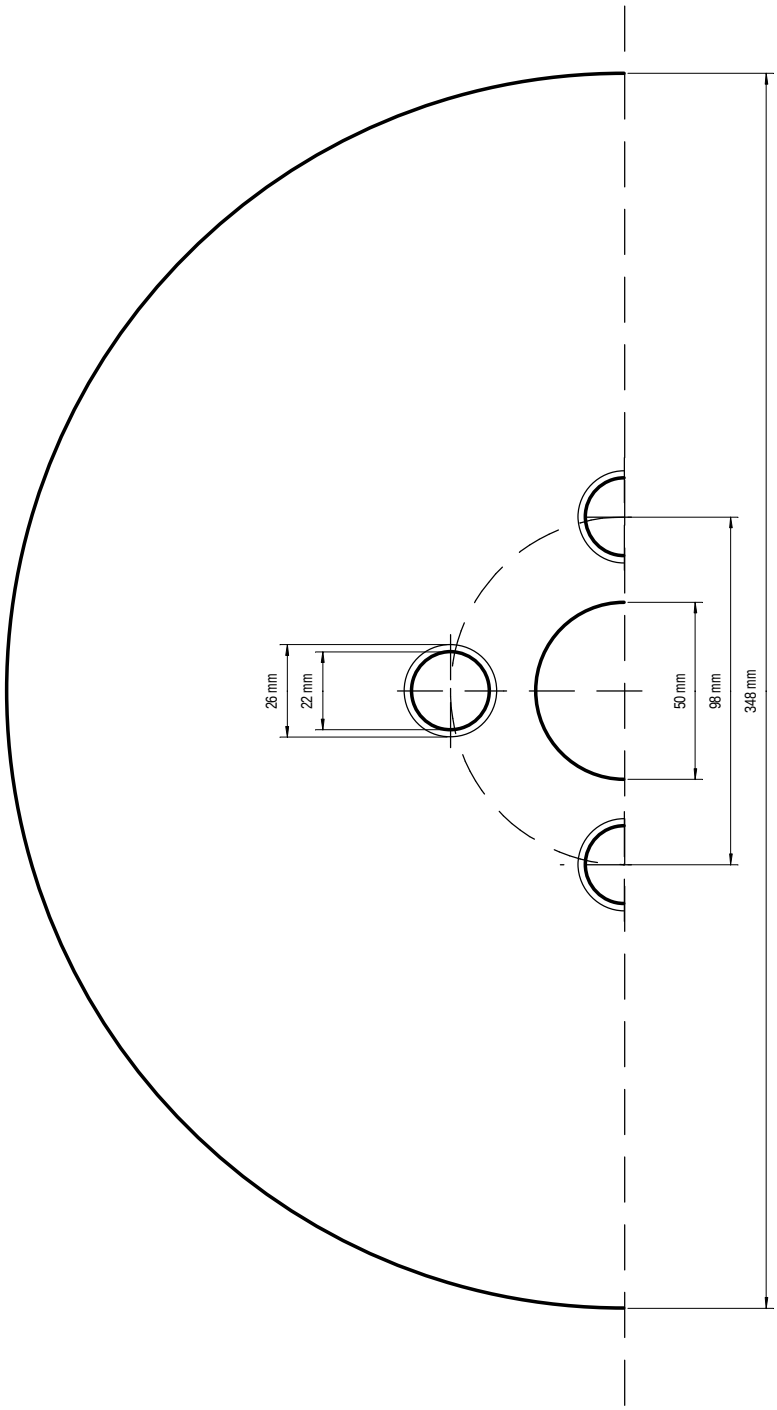


Gezeichnet:	29.05.06	Rico Wündisch	Industrieschule	KM03-2
Geprüft:				
1:2	Felgenschüssel			8/19



Zusätzliche Informationen	
——	Bezugskanten zur Darstellung in OpenGL-Umgebung
*	Bezugsvariablen zur Darstellung in OpenGL-Umgebung
	glfET = Einpresstiefe
	glfRf = Felgenradius
	glfRmw = 1/2 x Maulweite
	Maßstab Original : OpenGL = 1 : 20

Gezeichnet:	29.05.06	Rico Wündisch	Industrieschule	KM03-2
Geprüft:				
1:2	Felgenschüssel - OpenGL-Bemaßung			9/19



Gezeichnet:	29.05.06	Rico Wündisch	Industrieschule	KM03-2
Geprüft:				
1:2	Radscheibe - stark vereinfacht			10/19

Quelltext

```
unit ct_ogl1;


interface
uses
	Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
	Dialogs, OpenGL, Unit1, StdCtrls, ExtCtrls, Buttons;

type
	TOpenGL_box = class(TForm)
	FrameSteuerung1: TFrameSteuerung;

	procedure FormCreate(Sender: TObject);
	procedure FormDestroy(Sender: TObject);
	procedure FormResize(Sender: TObject);

	private { Private-Deklarationen }
		DC: HDC;
		hrc: HGLRC;
		Palette: HPALETTE;
		procedure SetDCPixelFormat;
		procedure InitializeLight;
		procedure InitializeMaterial;

	protected
		procedure WMPaint(var Msg: TWMPaint); message WM_PAINT;

	public
		{ Public-Deklarationen }
		procedure Buildlist_Wuerfel;
		procedure DrawScene;
end;

var
	OpenGL_box: TOpenGL_box;
	glfAspect: glDouble;
	Const_Max_Texture_Size, Wuerfel, Shademode: GLint;
	Zoom,Alpha: GLfloat;
	Const_gl_Version:string;

	Clip_Data: Array[0..3] of glDouble;	// globale Schnittebene
	Matrix: Array[0..15] of GLfloat;
	Axis: Array [0..2] of GLfloat;

	// Felgengrößen
	glfET: glDouble;
	// Einpresstiefe ET [OpenGL-Einheiten]
	glfMW: glDouble;
				// Maulweite MW [Zoll]
	glfRmw:glDouble;		// Hilfsvariable: 1/2 * MW [OpenGL-Einheiten]
	glfDf: glDouble;
				// Felgendurchmesser [Zoll]
	glfRf:glDouble;
		// Felgenradius Rf [Opengl-Einheiten]
	gliRadbolzen: Integer;	// Anzahl an Radbolzen
	gliHumpStyle: Integer;
	// Humps

	gliCounter: Integer;	// globale Zähl-/ Hilfsvariable für Schleifen


	ShowHalbschnitt: Boolean;	// Schalter für Anzeige
	ShowHumpLinks: Boolean;	// Schalter für Anzeige
	ShowHumpRechts: Boolean;	// Schalter für Anzeige
	ShowTiefbett: Boolean;
	// Schalter für Anzeige

	ShowMaulweite: Boolean;
	ShowEinpresstiefe: Boolean;
	ShowDurchmesser: Boolean;

	ColorFelgenschuessel: Boolean;
	ColorFelgenhorn: Boolean;
	ColorFelgenschulter: Boolean;
	ColorHump: Boolean;
	ColorFelgenbett: Boolean;
	ColorRadscheibe: Boolean;

implementation
{$R *.DFM}

procedure TOpenGL_box.FormCreate(Sender: TObject);
const glfLightModelAmbient: Array[0..3] of GLfloat =(0.2,0.2,0.2,1.0);
begin
	OpenGL_Box.FrameSteuerung1.Height := Opengl_Box.ClientHeight;
	Shademode:= 1;
	Zoom:= 20;
	Clip_Data[0]:=0;Clip_Data[1]:=0;Clip_Data[2]:=0; Clip_Data[3]:=0;

	glfET:= 34;					// [mm]
	glfMW:=6;

	// [Zoll]
	glfRmw:=glfMW * 2.54 / 4; // [OpenGL-Einheiten]
	glfDf:=15;
							// [Zoll]
	glfRf:=glfDf * 2.54 / 4;  // [OpenGL-Einheiten]
	gliRadbolzen:=4;
	gliHumpStyle:=0;

	ShowHumpLinks:=True;	// Schalter für Anzeige
	ShowHumpRechts:=True;	// Schalter für Anzeige
	ShowTiefbett:=True;
	// Schalter für Anzeige

	ShowMaulweite:= False;
	ShowEinpresstiefe:= False;
	ShowDurchmesser:= False;

	ColorFelgenschuessel:= False;
	ColorFelgenhorn:= False;
	ColorFelgenschulter:= False;
	ColorHump:= False;
	ColorFelgenbett:= False;
	ColorRadscheibe:=  False;

	// *** Erzeugen des Rendering Kontext (RC)
	DC:= GetDC(Handle);
	SetDCPixelFormat;
	hrc:= wglCreateContext(DC);
	wglMakeCurrent(DC, hrc);
	// *** Abfragen der OpenGL-Impelmentierung
	Const_gl_Version:= glGetString(GL_VERSION);
	glGetIntegerv(GL_MAX_TEXTURE_SIZE, @const_MAX_TEXTURE_SIZE);
	glEnable(GL_DEPTH_TEST);		// Einschalten depth testing
	// *** Initialisieren verschiedener Variablen
	glMatrixMode(GL_MODELVIEW);		// Modelviewmatix
```

```
        glLoadIdentity;

        { Aufbau Koordinatensystem:

                              +Y
                              |
                              |
                              |
                              +- - - - +X
                             /
                  /
                 /
          +Z

        }

        // *** Matrix um -45° um die Y-Achse drehen
        //glRotatef(Winkel in °, X, Y, Z);
        glRotatef(45,0,-1,0);

        glGetFloatv(GL_MODELVIEW_MATRIX,
@Matrix);
        // *** Lichtquelle und Material initialisieren
        glEnable(GL_LIGHTING);
        glEnable(GL_COLOR_MATERIAL);
        glLightMode-
li(GL_LIGHT_MODEL_LOCAL_VIEWER, 0);
        glLightModeli(GL_LIGHT_MODEL_TWO_SIDE,
1);
        glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
@glfLightModelAmbient);
        InitializeLight;
        InitializeMaterial;
        glfAspect := ClientWidth/ClientHeight;
        BuildList_Wuerfel;                    // Displaylist
erzeugen
        DrawScene;                            // zeichnen der
Scene
end;
procedure TOpenGL_box.SetDCPixelFormat;
var   hHeap: THandle;
        nColors, i: Integer;
        lpPalette : PLogPalette;
        byRedMask, byGreenMask, byBlueMask: Byte;
        nPixelFormat: Integer;
        pfd: TPixelFormatDescriptor;
begin
        FillChar(pfd, SizeOf(pfd), 0);
        with pfd do begin
                nSize     := sizeof(pfd);             //
Länge der pfd-Struktur
                nVersion  := 1;                       // Versi-
on
                dwFlags                               :=
PFD_DRAW_TO_WINDOW or PFD_SUPPORT_OPENGL
or
                                PFD_DOUBLEBUFFER;
// Flags
                iPixelType:=        PFD_TYPE_RGBA;
// RGBA Pixel Type
                cColorBits:= 24;                      // 24-bit
color
                cDepthBits:= 32;                      // 32-bit
depth buffer
                iLayerType:=      PFD_MAIN_PLANE;
// Layer Type
        end;
        nPixelFormat:= ChoosePixelFormat(DC, @pfd);
        SetPixelFormat(DC, nPixelFormat, @pfd);
// *** Palettenoptimierung wenn erforderlich
        DescribePixelFormat(DC, nPixelFormat,


        sizeof(TPixelFormatDescriptor),pfd);
        if ((pfd.dwFlags and PFD_NEED_PALETTE) <>
0) then begin
                nColors  := 1 shl pfd.cColorBits;
                hHeap    := GetProcessHeap;
                lpPalette:= HeapAlloc

(hHeap,0,sizeof(TLogPalette)+(nColors*sizeof(TPaletteEnt
ry)));
                lpPalette^.palVersion := $300;
                lpPalette^.palNumEntries := nColors;
                byRedMask  := (1 shl pfd.cRedBits) -
1;
                byGreenMask:= (1 shl pfd.cGreenBits)
- 1;
                byBlueMask := (1 shl pfd.cBlueBits) -
1;
                for i := 0 to nColors - 1 do begin
                        lpPalet-
te^.palPalEntry[i].peRed  :=
                                (((i                shr
pfd.cRedShift)  and byRedMask)  *255)DIV byRedMask;
                        lpPalet-
te^.palPalEntry[i].peGreen:=
                                (((i                shr
pfd.cGreenShift)and    byGreenMask)*255)DIV    byGreen-
Mask;
                        lpPalet-
te^.palPalEntry[i].peBlue :=
                                (((i                shr
pfd.cBlueShift) and byBlueMask) *255)DIV byBlueMask;
                        lpPalet-
te^.palPalEntry[i].peFlags:= 0;
                end;
                Palette:= CreatePalette(lpPalette^);
                HeapFree(hHeap, 0, lpPalette);
                if (Palette <> 0) then begin
                        SelectPalette(DC,     Palette,
False);
                        RealizePalette(DC);
                end;
        end;
end;
procedure TOpenGL_box.InitializeLight;
const
        glfLightAmbient   : Array[0..3] of GLfloat = (0.3,
0.3, 0.3, 1.0);
        glfLightDiffuse  : Array[0..3] of GLfloat = (0.7, 0.7,
0.7, 1.0);
        glfLightSpecular : Array[0..3] of GLfloat = (0.8,
0.8, 0.8, 1.0);
        glfLight0Position: Array[0..3] of GLfloat = (0.0,
0.0, 3.0, 0.0);
begin
 glMatrixMode(GL_MODELVIEW);
 glLoadIdentity;
 glEnable(GL_LIGHT0);
 glLightfv(GL_LIGHT0,                  GL_POSITION
,@glfLight0Position);
 glLightfv(GL_LIGHT0,                  GL_AMBIENT,
@glfLightAmbient);
 glLightfv(GL_LIGHT0, GL_DIFFUSE,  @glfLightDiffuse);
 glLightfv(GL_LIGHT0,                   GL_SPECULAR,
@glfLightSpecular);
end;
procedure TOpenGL_box.InitializeMaterial;

const
 glfMatrialAmbient : Array[0..3] of GLfloat = (0.5, 0.5, 0.5,
1.0);
 glfMatrialDiffuse : Array[0..3] of GLfloat = (0.5, 0.5, 0.5,
1.0);
 glfMatrialSpecular: Array[0..3] of GLfloat = (0.7, 0.7, 0.7,
1.0);
 glfMatrialShine:  GLfloat = 90;
begin
 glMaterialfv(GL_FRONT,                GL_AMBIENT,
@glfMatrialAmbient);
 glMaterialfv(GL_FRONT,                GL_DIFFUSE,
@glfMatrialDiffuse);
```

```
 glMaterialfv(GL_FRONT, GL_SPECULAR, @glfMatrialSpecular);
 glMaterialf (GL_FRONT, GL_SHININESS, glfMatrialShine);
end;
procedure Ring(innerRadius, outerRadius: glDouble; slices, stacks: Integer);
var
 dR, dH, dR2, dH2, dV, dW, dK, dS: glDouble;
 i, m: Integer;

begin
 dK  := (outerRadius+innerRadius)/2;
 dS  := outerRadius-innerRadius;
 dR  := sin(0);
 dH  := cos(0);
 for i := 0 to stacks-1 do begin
  dR2  := sin((i+1)/stacks*2*Pi);
  dH2  := cos((i+1)/stacks*2*Pi);
  glBegin(GL_QUAD_STRIP);
  for m := 0 to slices do begin
   dV := sin(m/slices*2*Pi);
   dW := cos(m/slices*2*Pi);
   glNormal3f(dR2*dW, dH2*dW, dV);
   glVertex3f(dR2*dK+dR2*dW*dS, dH2*dK+dH2*dW*dS, dV*dS);
   glNormal3f(dR*dW, dH*dW, dV);
   glVertex3f(dR*dK+dR*dW*dS, dH*dK+dH*dW*dS, dV*dS);
  end;
  glEnd;
  dR  := dR2;
  dH  := dH2;
 end;
end;

procedure TOpenGl_Box.Buildlist_Wuerfel;
var
        Schnittebene: Array[0..3] of glDouble;
        Koerper: GLUquadricObj;
        intCounter:Integer;
begin
        Wuerfel := glGenLists(1);
        glNewList(Wuerfel, GL_COMPILE_AND_EXECUTE);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
        Koerper := gluNewQuadric;
        gluQuadricDrawStyle(Koerper, GLU_FILL);
        gluQuadricNormals(Koerper, GLU_SMOOTH);
        gluQuadricTexture(Koerper, TRUE);
        gldisable(GL_CULL_FACE);


        //
        // O P E N G L - M A S S S T A B = 1 : 20
        //
        // Bsp.: 10 mm = 0.5 OpenGL-Einheiten
        //


                // linkes Felgenhorn
                Case ColorFelgenhorn of
                        True:  begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                        False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                glPushMatrix;
                        // Abschnitt [ / ]
                        glTranslatef(0, 0, glfRmw + 0.2);
                        gluCylinder(Koerper, glfRf + 0.7, glfRf + 0.6, 0.1, 64, 1);
                glPopMatrix; glPushMatrix;
                        // Abschnitt [ - ]
                        glTranslatef(0, 0, glfRmw + 0.1);
                        gluCylinder(Koerper, glfRf + 0.7, glfRf + 0.7, 0.1, 64, 1);
                glPopMatrix; glPushMatrix;
                        // Abschnitt [ \ ]
                        glTranslatef(0, 0, glfRmw);
                        gluCylinder(Koerper, glfRf + 0.6, glfRf + 0.7, 0.1, 64, 1);
                glPopMatrix; glPushMatrix;
                        // Abschnitt [ | ]
                        glTranslatef(0, 0, glfRmw);
                        gluDisk(Koerper, glfRf, glfRf + 0.6, 64, 1);
                glPopMatrix;



                // linke Felgenschulter
                Case ColorFelgenschulter of
                        True:  begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                        False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                glPushMatrix;
                        // Abschnitt [ - ]
                        glTranslatef(0, 0, glfRmw - 0.6);
                        gluCylinder(Koerper, glfRf, glfRf, 0.6, 64, 1);
                glPopMatrix;



                // linker Hump
                Case ColorHump of
                        True:  begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                        False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                Case ShowHumpLinks of
                True:begin
                        glPushMatrix;
                        // Abschnitt [ / ]
                        glTranslatef(0, 0, glfRmw - 0.7);
                        gluCylinder(Koerper, glfRf + 0.1, glfRf, 0.1, 64, 1);
                        glPopMatrix; glPushMatrix;
                        // Abschnitt [ - ]
                        glTranslatef(0, 0, glfRmw - 0.9);
                        gluCylinder(Koerper, glfRf + 0.1, glfRf + 0.1, 0.2, 64, 1);
                        glPopMatrix;glPushMatrix;
                        // Abschnitt [ \ ]
                        glTranslatef(0, 0, glfRmw - 1.0);
                        gluCylinder(Koerper, glfRf , glfRf + 0.1, 0.1, 64, 1);
                        glPopMatrix;
                        end;
                False: begin
                        glPushMatrix;
                        glTranslatef(0, 0, glfRmw - 1.0);
                        gluCylinder(Koerper, glfRf, glfRf, 0.4, 64, 1);
                        glPopMatrix;
                        end;
                end;



                // Tiefbett
                Case ColorFelgenbett of
                        True:  begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
```

```
                    False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                Case ShowTiefbett of
                True:begin
                    glPushMatrix;
                        // Abschnitt [ \ ]
                        glTranslatef(0, 0, glfRmw - 1.2);
                        gluCylinder(Koerper, glfRf - 0.8, glfRf, 0.2, 64, 1);
                    glPopMatrix; glPushMatrix;
                        // Abschnitt [ - ]
                        glTranslatef(0, 0, -glfRmw + 3.4);
                        gluCylinder(Koerper, glfRf - 0.8, glfRf -0.8, 2 * glfRmw - 4.6, 64, 1);
                    glPopMatrix; glPushMatrix;
                        // Abschnitt [ / ]
                        glTranslatef(0, 0, -glfRmw + 2.8);
                        gluCylinder(Koerper, glfRf - 0.3, glfRf -0.8, 0.6, 64, 1);
                    glPopMatrix; glPushMatrix;
                        // Abschnitt [ / ]
                        glTranslatef(0, 0, -glfRmw + 1.0);
                        gluCylinder(Koerper, glfRf, glfRf - 0.3, 1.8, 64, 1);
                    glPopMatrix;
                    end;
                False: begin
                    glPushMatrix;
                    glTranslatef(0, 0, -glfRmw + 1.0);
                    gluCylinder(Koerper, glfRf , glfRf, 2 * glfRmw - 2.0, 64, 1);
                    glPopMatrix;
                    end;
                end;


                // rechter Hump
                Case ColorHump of
                    True: begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                    False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                Case ShowHumpRechts of
                True:begin
                    glPushMatrix;
                    // Abschnitt [ / ]
                    glTranslatef(0, 0, - glfRmw + 0.9);
                    gluCylinder(Koerper, glfRf + 0.1, glfRf, 0.1, 64, 1);
                    glPopMatrix; glPushMatrix;
                    // Abschnitt [ - ]
                    glTranslatef(0, 0, - glfRmw + 0.7);
                    gluCylinder(Koerper, glfRf + 0.1, glfRf + 0.1, 0.2, 64, 1);
                    glPopMatrix; glPushMatrix;
                    // Abschnitt [ \ ]
                    glTranslatef(0, 0, - glfRmw + 0.6);
                    gluCylinder(Koerper, glfRf, glfRf + 0.1, 0.1, 64, 1);
                    glPopMatrix;
                    end;
                False: begin
                    glPushMatrix;
                    glTranslatef(0, 0, - glfRmw + 0.6);
                    gluCylinder(Koerper, glfRf, glfRf, 0.4, 64, 1);
                    glPopMatrix;
                    end;
                end;


                // rechte Felgenschulter
                Case ColorFelgenschulter of
                    True: begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                    False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                glPushMatrix;
                    // Abschnitt [ - ]
                    glTranslatef(0, 0, - glfRmw);
                    gluCylinder(Koerper, glfRf, glfRf, 0.6, 64, 1);
                glPopMatrix;


                // rechtes Felgenhorn
                Case ColorFelgenhorn of
                    True: begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                    False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
                end;
                glPushMatrix;
                    // Abschnitt [ | ]
                    glTranslatef(0, 0, -glfRmw);
                    gluDisk(Koerper, glfRf, glfRf + 0.6, 64, 1);
                glPopMatrix; glPushMatrix;
                    // Abschnitt [ / ]
                    glTranslatef(0, 0, - glfRmw - 0.1);
                    gluCylinder(Koerper, glfRf + 0.7, glfRf + 0.6, 0.1, 64, 1);
                glPopMatrix; glPushMatrix;
                    // Abschnitt [ - ]
                    glTranslatef(0, 0, - glfRmw - 0.2);
                    gluCylinder(Koerper, glfRf + 0.7, glfRf + 0.7, 0.1, 64, 1);
                glPopMatrix; glPushMatrix;
                    // Abschnitt [ \ ]
                    glTranslatef(0, 0, - glfRmw - 0.3);
                    gluCylinder(Koerper, glfRf + 0.6, glfRf + 0.7, 0.1, 64, 1);
                glPopMatrix;



            // Radscheibe
            Case ColorRadscheibe of
                    True: begin glColor4f(1.0, 0.5, 0.0, 1.0); end;
                    False: begin glColor4f(0.8, 0.8, 0.8, 1.0); end;
            end;
            for intCounter:= 1 to gliRadbolzen do begin
                glPushMatrix;
                // Bohrung Radbolzen Vorderseite
                glTranslatef(0.0, 2.3, glfET/20);
                Ring(0.5,0.6,8,32);
                gluDisk(Koerper, 0.6, 1.0, 32, 1);
                glPopMatrix;
                glRotatef(360/gliRadbolzen,0,0,-1);
            end;

            for intCounter:= 1 to gliRadbolzen do begin
                glPushMatrix;
                // Füllkörper zwischen Bohrungen für Radbolzen
                glTranslatef(0,0,glfET/20);
```

```
                gluPartial-
Disk(Koerper,1.8,3.2,5,1,10+2,(360/gliRadbolzen)-20-4);
                glPopMatrix;
                glRotatef(360/gliRadbolzen,0,0,-1);
        end;

        glPushMatrix;
        glTranslatef(0,0,glfET/20);
        // Innenkreis Radscheibe
        gluDisk(Koerper, 1.25, 1.8, 64, 1);
        // Außenkreis Radscheibe
        case ShowTiefbett of
        True: gluDisk(Koerper, 3.1, glfRf - 0.8, 64, 1);
        false: gluDisk(Koerper, 3.1, glfRf, 64, 1);
        end;
        glPopMatrix;


        case ShowMaulweite of
                True: begin
                        glColor4f(1,0.5,0,0.3);
                        glEnable(GL_BLEND);
                        glPushMatrix;
                        glTranslatef(0,0,glfRmw -
0.01);
                        gluDisk(Koerper,glfRf, glfRf
+ 2, 64, 1);
                        gluCylinder(Koerper, glfRf +
2, glfRf + 2, 0.05, 64, 1);
                        glPopMatrix; glPushMatrix;
                        glTranslatef(0,0,-glfRmw +
0.01);
                        gluDisk(Koerper,glfRf, glfRf
+ 2, 64, 1);
                        gluCylinder(Koerper, glfRf +
2, glfRf + 2, 0.05, 64, 1);
                        glDisable(GL_BLEND);
                        glTranslatef(0, glfRf + 1.5,
0);
                        gluCylinder(Koerper, 0.03,
0.03, 2* glfRmw - 0.02, 64, 1);
                        gluCylinder(Koerper, 0.03,
0.3, 2.0, 32, 1);
                        glTranslatef(0, 0, 2* glfRmw
- 0.02 - 2.0);
                        gluCylinder(Koerper, 0.3,
0.03, 2.0, 32, 1);
                        glPopMatrix;

                        end;
        end;

                case ShowEinpresstiefe of
                True: begin
                        glColor4f(1,0.5,0,0.3);
                        glEnable(GL_BLEND);
                        glPushMatrix;
                        gluDisk(Koerper,0, glfRf + 2,
64, 1);
                        gluCylinder(Koerper, glfRf +
2, glfRf + 2,0.05, 64,1);
                        glTranslatef(0,0,glfET/20 -
0.01);
                        gluDisk(Koerper,0, glfRf + 2,
64, 1);
                        gluCylinder(Koerper, glfRf +
2, glfRf + 2,0.05, 64,1);
                        glPopMatrix;

                        glPushMatrix;
                        glDisable(GL_BLEND);
                        glTranslatef(0,0,-glfMW-2.4);
                        gluCylinder(Koerper, 0.03,
0.03, 2*(glfMW + 2.4), 64, 1);
                        glPopMatrix;

                        glPushMatrix;
                        glTranslatef(0,0,-2);
                        gluCylinder(Koerper, 0.3,
0.03, 2.0, 32, 1);
                        glTranslatef(0, 0, glfET/20
+0.01+2);
                        gluCylinder(Koerper, 0.03,
0.3, 2, 64, 1);
                        glPopMatrix;

                        end;
        end;

        case ShowDurchmesser of
                True: begin
                        glColor4f(1,0.5,0,0.3);
                        glEnable(GL_BLEND);
                        glPushMatrix;
                        glTranslatef(0,0,-glfRmw -
2.3);
                        gluCylinder(Koerper, glfRf +
0.01, glfRf + 0.01, 2*(glfRmw +2.4) , 64, 1);
                        glPopMatrix;
                        glDisable(GL_BLEND);
                        glPushMatrix();
                        glRotatef(90,-1,0,0);
                        // y und z getauscht
                        glTranslatef(0,- glfRmw -2, -
(glfRf + 0.01 + 3));
                        gluCylinder(Koerper,0.03,
0.03, 2* (glfRf + 0.01+ 3),32,1);

                        glTranslatef(0,0,+1);
                        gluCylinder(Koerper, 0.3,
0.03, 2.0, 32, 1);
                        glTranslatef(0, 0, 2* (glfRf +
0.01)+2);
                        gluCylinder(Koerper, 0.03,
0.3, 2.0, 32, 1);
                        glPopMatrix;
                        end;
        end;




        glEnable(GL_CULL_FACE);
        gluDeleteQuadric(Koerper);
        glEndList;
end;
procedure TOpenGl_Box.DrawScene;

begin
        // Hintergund-Farbe oben
        glClearColor(0.05, 0.23, 0.59, 1);
        glClear(GL_COLOR_BUFFER_BIT or
GL_DEPTH_BUFFER_BIT);

        glMatrixMode(GL_PROJECTION); //
Projekton wählen
        glLoadIdentity;
        //glOrtho(-Zoom, Zoom, -Zoom/glfAspect,
Zoom/glfAspect, -20, 20);
        glOrtho(-Zoom, Zoom, -Zoom/glfAspect,
Zoom/glfAspect, -40, 40);
        glPolygonMode(GL_FRONT_AND_BACK,
GL_FILL);

        glMatrixMode(GL_MODELVIEW); // Rotation
berechnen
        glLoadIdentity; // Einheitsmatix laden
        gldisable(GL_CLIP_PLANE0);

        glEnable(GL_BLEND); // Farbabstufung mit
Blend
        glBlendFunc(GL_SRC_ALPHA,
GL_ONE_MINUS_SRC_ALPHA);
        glBegin(GL_QUADS); // Rechteck zeichnen
```

```
                glColor4f(0.4, 0.53, 0.86, 0.0);
                glNormal3f(0.4, 0.53, 0.86); // Hintergrund-Farbe unten
                glVertex3f(-Zoom, -Zoom/glfAspect, -10);
                glVertex3f( Zoom, -Zoom/glfAspect, -10);
                glColor4f(0.4, 0.53, 0.86, 1.0);
                glVertex3f( Zoom, Zoom/glfAspect, -10);
                glVertex3f(-Zoom, Zoom/glfAspect, -10);
        glEnd;
        gldisable(GL_BLEND);
        glRotatef(Alpha, Axis[0], Axis[1], Axis[2]);
                // drehen der Objekte
        Alpha:=0;
                        // Rücksetzen vom Winkel
        glMultMatrixf(@Matrix );
                // zusätzliche Drehung mit letzem Wert
        glGetFloatv(GL_MODELVIEW_MATRIX, @Matrix);
                // Rotation sichern
        glLoadIdentity;
                                // Einheitsmatix Laden
        glMultMatrixf(@Matrix );        // Multiplikation mit Matix
        glClipPlane(GL_CLIP_PLANE0, @Clip_Data); // Clipplane definieren
        glEnable(GL_CLIP_PLANE0);
        case Shademode of               // Darstellungs-modus der Polygone

        1:GlPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

        2:GlPolygonMode(GL_FRONT_AND_BACK, GL_LINE);

        3:GlPolygonMode(GL_FRONT_AND_BACK, GL_POINT);
        end;
        glCallList(Wuerfel);                    // Aufruf der Displaylist
        if GLGetError <> GL_NO_ERROR then // Fehlerprüfung
                MessageDlg('Error in OpenGL scene gefunden!', mtError, [mbOk], 0);
        SwapBuffers(DC);                        // Sichtbar machen
end;
procedure TOpenGl_box.WMPaint(var Msg: TWMPaint);
var  ps : TPaintStruct;
begin
        BeginPaint(Handle, ps);
        DrawScene;
        EndPaint(Handle, ps);
end;
procedure TOpenGL_box.FormResize(Sender: TObject);
begin
        OpenGL_Box.FrameSteuerung1.Height := OpenGL_Box.ClientHeight;
        // *** Aktualisierung des Viewports, wenn sich die Fenstergröße ändert
        if ClientWidth  < 800 then ClientWidth :=800; // min:800
        if ClientHeight < 600 then ClientHeight:=600; // min:600
        glfAspect := ClientWidth / ClientHeight;
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity;
        gLOrtho(-Zoom, Zoom, -Zoom/glfAspect, Zoom/glfAspect, -20, 20);
        glViewport(100, 0, ClientWidth,ClientHeight);//Viewport Transformation
        InvalidateRect(Handle, nil, False);     // Neu-zeichnen anfordern
end;
procedure TOpenGL_box.FormDestroy(Sender: TObject);
begin
        // *** Löschen des rendering context (RC)
        gldeleteLists(Wuerfel,1);
        wglMakeCurrent(0, 0);
        wgldeleteContext(hrc);
        ReleaseDC(Handle, DC);
        if (Palette <> 0) then DeleteObject(Palette);
end;



end.
```

```
end.
```

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
            Dialogs, StdCtrls, ExtCtrls, Buttons, jpeg;

type
  TFrameSteuerung = class(TFrame)
                        Label1: TLabel;
                        Shape2: TShape;
                        SpeedButton1: TSpeedButton;
                        SpeedButton2: TSpeedButton;
                        SpeedButton3: TSpeedButton;
                        SpeedButton4: TSpeedButton;
                        SpeedButton5: TSpeedButton;
    Shape4: TShape;
    Label2: TLabel;
                        Label3: TLabel;
    lblMaulweiteText: TLabel;
    lblDurchmesser: TLabel;
    btnMaulweitePlus: TButton;
    btnMaulweiteMinus: TButton;
    SpeedButton6: TSpeedButton;
    btnStyle: TButton;
    btnSchnitt: TButton;
    Label4: TLabel;
    Label5: TLabel;
                        Shape1: TShape;
                        btnShowMaulweite: TButton;
    Label6: TLabel;
    Shape3: TShape;
    btnShowDurchmesser: TButton;
    lblEinpresstiefe: TLabel;
    btnEinpresstiefePlus: TButton;
    btnEinpresstiefeMinus: TButton;
    btnShowEinpresstiefe: TButton;
    lblLochkreise: TLabel;
    btnLochkreisePlus: TButton;
    btnLochkreiseMinus: TButton;
    lblFelgenhorn: TLabel;
    lblFelgenschulter: TLabel;
    lblHump: TLabel;
    lblFelgenbett: TLabel;
    lblFelgenschuessel: TLabel;
    lblRadscheibe: TLabel;
    btnStyleFelgenbett: TButton;
                        btnStyleHump: TButton;
    btnColorFelgenhorn: TButton;
    btnColorFelgenschulter: TButton;
    btnColorHump: TButton;
    btnColorFelgenbett: TButton;
    btnColorFelgenschuessel: TButton;
    btnColorRadschuessel: TButton;
    procedure btnShowEinpresstiefeClick(Sender: TObject);
    procedure btnShowDurchmesserClick(Sender: TObject);
    procedure btnShowMaulweiteClick(Sender: TObject);
    procedure btnColorRadschuesselClick(Sender: TObject);
    procedure btnColorFelgenbettClick(Sender: TObject);
    procedure btnColorHumpClick(Sender: TObject);
    procedure btnColorFelgenschulterClick(Sender: TObject);
    procedure btnColorFelgenschuesselClick(Sender: TObject);
                        procedure btnColorFelgenhornClick(Sender: TObject);
                        procedure btnStyleHumpClick(Sender: TObject);
    procedure btnStyleFelgenbettClick(Sender: TObject);
    procedure btnEinpresstiefeMinusClick(Sender: TObject);
    procedure btnEinpresstiefePlusClick(Sender: TObject);
    procedure btnLochkreiseMinusClick(Sender: TObject);
    procedure btnLochkreisePlusClick(Sender: TObject);
                        procedure checkTiefbettClick(Sender: TObject);
    procedure btnSchnittClick(Sender: TObject);
                        procedure btnMaulweiteMinusClick(Sender: TObject);
    procedure btnMaulweitePlusClick(Sender: TObject);
    procedure btnDurchmesserMinusClick(Sender: TObject);
                        procedure btnDurchmesserPlusClick(Sender: TObject);
                        procedure btnStyleClick(Sender: TObject);
                        procedure SpeedButton1Click(Sender: TObject);
                        procedure SpeedButton2Click(Sender: TObject);
                        procedure SpeedButton3Click(Sender: TObject);
                        procedure SpeedButton4Click(Sender: TObject);
                        procedure SpeedButton5Click(Sender: TObject);
                        procedure SpeedButton6Click(Sender: TObject);
                        { Private-Deklarationen }
            public
                        { Public-Deklarationen }
            end;

implementation

uses ct_ogl1;

{$R *.dfm}

procedure TFrameSteuerung.SpeedButton6Click(Sender: TObject);
begin
            // verkleinern
            if Zoom < 50 then Zoom:= 1.1 * Zoom; opengl_box.DrawScene;
end;
procedure TFrameSteuerung.SpeedButton5Click(Sender: TObject);
begin
            // vergrößern
            if Zoom > 5 then Zoom:= 0.9 * Zoom; opengl_box.DrawScene;
end;

procedure TFrameSteuerung.SpeedButton4Click(Sender: TObject);
begin
            // nach rechts drehen
            Alpha:= 15; Axis[0]:= 0; Axis[1]:= 1; Axis[2]:= 0;
opengl_box.DrawScene;
end;
procedure TFrameSteuerung.SpeedButton3Click(Sender: TObject);
begin
            // nach links drehen
            Alpha:= -15; Axis[0]:= 0; Axis[1]:= 1; Axis[2]:= 0;
opengl_box.DrawScene;
end;
procedure TFrameSteuerung.SpeedButton1Click(Sender: TObject);
begin
            // nach oben drehen
            Alpha:= -15; Axis[0]:= 1; Axis[1]:= 0; Axis[2]:= 0;
opengl_box.DrawScene;
end;
procedure TFrameSteuerung.SpeedButton2Click(Sender: TObject);
begin
            // nach unten drehen
```

```
	Alpha:= 15; Axis[0]:= 1; Axis[1]:= 0; Axis[2]:= 0;
opengl_box.DrawScene;
end;


procedure TFrameSteuerung.btnDurchmesserPlusClick(Sender: TObject);
begin
	//glfRf:=glfRf+0.5;
	glfDf:=glfDf + 1;
	If glfDf = 18 then glfDf := 17;
	glfRf:=glfDf * 2.54 / 4;
	lblDurchmesser.Caption := 'Durchmesser: ' +
floattostr(glfDf) + ' "';
	OpenGL_Box.Buildlist_Wuerfel;
	opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnDurchmesserMinusClick(Sender: TObject);
begin
	//glfRf:=glfRf-0.5;
	glfDf:=glfDf - 1;
	If glfDf = 12 then glfDf := 13;
	glfRf:=glfDf * 2.54 / 4;
	lblDurchmesser.Caption := 'Durchmesser: ' +
floattostr(glfDf) + ' "';
	OpenGL_Box.Buildlist_Wuerfel;
	opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnMaulweitePlusClick(Sender: TObject);
begin
	glfMW:=glfMW+0.5;
	if glfMW > 8 then glfMW:=8;
	glfRmw:=glfMW * 2.54 / 4;
		lblMaulweiteText.Caption:='Maulweite:
' + floattostr(glfMW) + ' "';
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnMaulweiteMinusClick(Sender: TObject);
begin
	if (glfMW - 0.5) / 2 - 1.2 >= glfET/20 then
glfMW:=glfMW-0.5;
	if glfMW < 4 then glfMW:=4;
	glfRmw:=glfMW * 2.54 / 4;
	lblMaulweiteText.Caption:='Maulweite: ' + float-
tostr(glfMW) + ' "';
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;


procedure TFrameSteuerung.btnSchnittClick(Sender:
TObject);
begin
	case ShowHalbschnitt of
	True: begin btnSchnitt.Caption := 'Halbschnitt';
	Clip_Data[0]:=0; end;
	False: begin btnSchnitt.Caption := 'Volldarstel-
lung';
	Clip_Data[0]:=-1; end;
	end;
	ShowHalbschnitt:=Not(ShowHalbschnitt);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.checkTiefbettClick(Sender:
TObject);
begin
	Showtiefbett:=Not(ShowTiefbett);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
```

```
procedure TFrameSteuerung.btnStyleClick(Sender: TOb-
ject);
begin
	Inc(Shademode);
	If Shademode = 4 then shademode:= 1;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnLochkreisePlusClick(Sender: TObject);
begin
	Inc(gliRadbolzen);
	if gliRadbolzen = 6 then gliRadbolzen:=5;
	lblLochkreise.Caption:='Radbolzen: ' + int-
tostr(gliRadbolzen);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnLochkreiseMinusClick(Sender: TObject);
begin
	Dec(gliRadbolzen);
	if gliRadbolzen = 2 then gliRadbolzen:=3;
	lblLochkreise.Caption:='Radbolzen: ' + int-
tostr(gliRadbolzen);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnEinpresstiefePlusClick(Sender: TObject);
begin
	if (glfET +1)/20 <= glfRmw - 1.2 then
glfET:=glfET + 1;
	lblEinpresstiefe.Caption := 'Einpresstiefe: '+
floattostr(glfET)+'mm';
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnEinpresstiefeMinusClick(Sender: TObject);
begin
	if (glfET - 1)/20 >= -glfRmw + 3.4 then
glfET:=glfET - 1;
	lblEinpresstiefe.Caption := 'Einpresstiefe: '+
floattostr(glfET)+'mm';
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnStyleFelgenbettClick(Sender: TObject);
begin
	Showtiefbett:=Not(ShowTiefbett);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnStyleHumpClick(Sender:
TObject);
begin
	Inc(gliHumpStyle);
	if gliHumpStyle=3 then gliHumpStyle:=0;
	case gliHumpStyle of
	0: begin ShowHumpLinks:=True; Show-
HumpRechts:=True; end;
	1: begin ShowHumpLinks:=True; Show-
HumpRechts:=False; end;
	2: begin ShowHumpLinks:=False; Show-
HumpRechts:=False; end;
	end;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
```

```
procedure TFrameSteuerung.btnColorFelgenhornClick(Sender: TObject);
begin
	ColorFelgenschuessel:= False;
	ColorFelgenhorn:= Not(ColorFelgenhorn);
	ColorFelgenschulter:= False;
	ColorHump:= False;
	ColorFelgenbett:= False;
	ColorRadscheibe:=  False;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnColorFelgenschuesselClick(Sender: TObject);
begin
	ColorFelgenschuessel:=
Not(ColorFelgenschuessel);
	case ColorFelgenschuessel of
	True: begin
		ColorFelgenhorn:= True;
		ColorFelgenschulter:= True;
		ColorHump:= True;
		ColorFelgenbett:= True;
		ColorRadscheibe:=  False;
		OpenGL_Box.Buildlist_Wuerfel;
		Opengl_Box.DrawScene; end;
	False: begin
		ColorFelgenhorn:= False;
		ColorFelgenschulter:= False;
		ColorHump:= False;
		ColorFelgenbett:= False;
		ColorRadscheibe:=  False;
		OpenGL_Box.Buildlist_Wuerfel;
		Opengl_Box.DrawScene; end;
	end;
end;
procedure TFrameSteuerung.btnColorFelgenschulterClick(Sender: TObject);
begin
	ColorFelgenschuessel:= False;
	ColorFelgenhorn:= False;
	ColorFelgenschulter:= Not(ColorFelgenschulter);
	ColorHump:= False;
	ColorFelgenbett:= False;
	ColorRadscheibe:=  False;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure  TFrameSteuerung.btnColorHumpClick(Sender:
TObject);
begin
	ColorFelgenschuessel:= False;
	ColorFelgenhorn:= False;
	ColorFelgenschulter:= False;
	ColorHump:= Not(ColorHump);
	ColorFelgenbett:= False;
	ColorRadscheibe:=  False;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnColorFelgenbettClick(Sender: TObject);
begin
	ColorFelgenschuessel:= False;
	ColorFelgenhorn:= False;
	ColorFelgenschulter:= False;
	ColorHump:= False;
	ColorFelgenbett:= Not(ColorFelgenbett);
	ColorRadscheibe:=  False;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnColorRadschuesselClick(Sender: TObject);
begin
	ColorFelgenschuessel:= False;
	ColorFelgenhorn:= False;
	ColorFelgenschulter:= False;
	ColorHump:= False;
	ColorFelgenbett:= False;
	ColorRadscheibe:=  Not(ColorRadscheibe);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

procedure TFrameSteuerung.btnShowMaulweiteClick(Sender: TObject);
begin
	ShowMaulweite:=Not(ShowMaulweite);
	ShowDurchmesser:= False;
	ShowEinpresstiefe:= False;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
	end;
procedure TFrameSteuerung.btnShowDurchmesserClick(Sender: TObject);
begin
	ShowMaulweite:= False;
	ShowDurchmesser:=Not(ShowDurchmesser);
	ShowEinpresstiefe:= False;
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;
procedure TFrameSteuerung.btnShowEinpresstiefeClick(Sender: TObject);
begin
	ShowMaulweite:= False;
	ShowDurchmesser:= False;
	ShowEinpresstiefe:=Not(ShowEinpresstiefe);
	OpenGL_Box.Buildlist_Wuerfel;
	Opengl_Box.DrawScene;
end;

end.
```