Anhang A

Grafik-Operationen unter Borland-Pascal - Eine Referenz

initgraph

(VAR grafiktreiber : INTEGER; VAR modus : INTEGER; VAR pfad : STRING);

Initgraph ist hauptsächlich eine Prozedur zur Überprüfung der aktuellen Hardware. Zuerst wird der Typ der Grafikkarte festgestellt. Danach wird ein Modus eingeschaltet.

Die Parameter lauten:

grafiktreiber : legt den vom Anwender gewünschten Treiber fest und übergibt den tatsächlich benutzten als Nummer zurück. Als Besonderheit kann initgraph bei grafiktreiber=0 den benötigten Treiber selbst herausfinden.

modus : wählt je nach Grafikkarte einen möglichen Auflösungsmodus. Der tatsächliche Modus wird zurückübergeben.

pfad : legt den Suchweg für den Grafiktreiber fest (".BGI"). Wird hier ein Leerstring angegeben, so wird im aktuellen Verzeichnis gesucht.

closegraph

(ohne Parameter)

Dadurch wird das Grafikpaket beendet. Der durch Treiber und Grafikdaten belegte Speicher wird wieder freigegeben, der Textmodus wird danach wieder eingeschaltet.

cleardevice

(ohne Parameter)

löscht den Grafikbildschirm und setzt alle anderen Parameter auf die Standardwerte. U.a. wird auch automatisch die erste Grafikseite eingeschaltet.

setviewport

(x1,y1,x2,y2 : INTEGER; clipping : BOOLEAN);

Damit wird ein Grafikfenster definiert, in dem sich alle nachfolgenden Prozeduren abspielen sollen. Bei clipping=true werden nur Grafikoperationen innerhalb des definierten Fensters ausgeführt. Ohne diese Definition ist der gesamte Bildschirm als Grafikfenster definiert.

x1,y1,x2,y2 : Koordinaten des Grafikfensters

clipping : legt fest, ob das "Clipping" aktiviert werden soll.

clearviewport

(ohne Parameter)

löscht das aktuelle Grafikfenster bzw. den gesamten Bildschirm.

setactivepage

(seite : WORD)

legt die aktuelle Grafikseite fest (sofern mehrere verwaltet werden. Die erste aller Seiten hat immer die Nummer 0. Alle nachfolgenden Grafikoperationen werden ausschließlich auf dieser Seite durchgeführt.

seite : Die logische Nummer der Seite

setvisualpage

(seite : WORD)

Damit wird bestimmt, welche Grafikseite jetzt auf dem Bildschirm angezeigt werden soll. Es ist also möglich, verschiedene Grafikoperationen im Hintergrund ablaufen zu lassen, später dann erst anzeigen zu lassen.

putpixel

(x,y : INTEGER; farbe : WORD)

Zeichnet einen Punkt der Farbe *farbe* an die Stelle (*x,y*).

x,y : Koordinaten der Stelle, an die der Punkt (Pixel) gesetzt werden soll.

farbe : Die Zeichenfarbe des zu setzenden Punktes.

line

(x1,y1,x2,y2 : INTEGER);

zeichnet eine Linie von (x1,y1) nach (x2,y2) in der aktuell aktiven Zeichenfarbe.

x1,y1 : Startpunkt der Linie

x2,y2 ; Endpunkt der Linie

rectangle

(x1,y1,x2,y2 : INTEGER)

zeichnet ein Rechteck, welches korrekt vertikal/horizontal ausgerichtet ist. Die notwendigen Punkte werden aus den beiden angegebenen Punkten berechnet.

x1,y1 : Koordinaten der oberen, linken Ecke

x2,y2 : Koordinaten der unteren, rechten Ecke

arc

(x,y : INTEGER; wStart, wEnde, Radius : WORD);

Zeichnet einen Kreisbogen vom Start- zum Endwinkel um den angegebenen Mittelpunkt.

x,y : Koordinaten des Mittelpunktes, um den der Kreisbogen gezogen werden soll.

wStart : Winkel, bei dem der Kreisbogen beginnen soll. Hier ist das Gradmaß (0..360 Grad) anzugeben. Das Maß 0 Grad liegt horizontal rechts neben dem Kreismittelpunkt. Die Zählung erfolgt im mathematisch positiven Sinn, also gegen den Uhrzeigersinn.

wEnde : Winkel, bei dem der Kreisbogen enden soll.

Radius : Abstand des Kreisbogens zum Mittelpunkt.

circle

(x,y : INTEGER; Radius . WORD);

Es wird ein Kreis mit angegebenen Radius um (x,y) gezogen.

x,y : Koordinaten des Kreismittelpunktes

Radius : Radius des zu zeichnenden Kreises

ellipse

(x,y : INTEGER; Startwinkel, Endwinkel : WORD; x_Radius, y_Radius : WORD);

x,y : Mittelpunkt der Ellipse

Startwinkel, Endwinkel : geben den Start- bzw. Endwinkel der zu zeichnenden Ellipse an (siehe auch "arc").

x_Radius, y_Radius : geben die beiden Radien der Ellipse an.

getmaxx

: INTEGER;

Stellt fest, wie groß die maximale x-Koordinate des Bildschirmes ist. Dieser Aufruf ist eine Funktion.

getmaxy

: INTEGER;

Stellt fest, wie groß die maximale y-Koordinate des Bildschirmes ist. Auch dieser Aufruf ist eine Funktion.

setcolor

(farbnummer : WORD);

setzt die aktuelle Zeichenfarbe.

farbe : Nummer der entsprechenden Farbe in der Palette.

getcolor

: WORD

liefert die aktuelle Zeichenfarbe zurück. Dieser Aufruf ist eine Funktion.

getbkcolor

: WORD

liefert die Hintergrundfarbe des Bildschirmes zurück. Dieser Aufruf wird dazu benötigt, festzustellen, mit welcher Farbe durch Überzeichnen gelöscht werden soll.

Anhang B

Grundstrukturen von Borland-Pascal

Allgemeine Form eines Borland-Pascal-Programms

Die allgemeine Form eines Programms in Borland-Pascal ist nicht allgemeingültig. Das bedeutet, daß nicht immer alle hier aufgeführten Struktur-Kennzeichen (z.B. der "USES"-Aufruf, der "CONST"-Teil, ...) auch in jedem Programm auftauchen müssen. Welche Teile welchen Sinn haben, wird nun aufgedeckt.

PROGRAM legt den Namen des Programms fest. Er darf keine Leerzeichen, Großbuchstaben oder Satzzeichen beinhalten. Dieser Aufruf muß immer eingegeben werden. Er muß als allererste Anweisung das Programm eröffnen.

USES legt fest, welche Werkzeug-Units benutzt werden sollen. Neben den gebräuchlichen Borland-Pascal Units ("CRT", "DOS", "PRINTER", ...) können auch Units selbst geschrieben und so eingebunden werden. Der Vorteil: Es bleibt viel mehr Speicher für das eigentliche Programm.

CONST deklariert programminterne Konstanten.

TYPE legt selbstdefinierte Datentypen und deren Struktur fest. Das gebräuchlichste Beispiel ist der Typ "Liste" oder "Record".

VAR deklariert die Variablen, die vom Hauptprogramm benutzt werden. Alle anderen Variablen, die nur innerhalb einer Prozedur oder Funktion benutzt werden, werden dort deklariert. Als Variablennamen können auch die neuen, unter TYPE deklarierten Typen angegeben werden.

FUNCTION/PROCEDURE deklariert eine Funktion/Prozedur. Die Struktur ähnelt der des Gesamtprogramms: Es können (müssen aber nicht) TYPE, CONST, VAR-Deklarationen erfolgen. Auch Unterprozeduren/Funktionen können hier deklariert werden. Alle Funktions-/prozedurinternen Deklarationen haben nur innerhalb dieses Unterprogrammes Gültigkeit.

BEGIN/END ist schließlich der Bereich für das Hauptprogramm. Zur Beachtung: "END" muß als allerletzte Anweisung im Programmtext stehen. Ihm muß immer ein Punkt (".") folgen!

```
PROGRAM programmname;

USES unitname1, unitname2, ...

CONST
  konstante1 = ausdruck1;
  konstante2 = ausdruck2;
  ...

TYPE
  typenname1 = typ1;
  typenname2 = typ2;
  ...

VAR
  variable1,
  variable2,
  ...
  variable n : typ1;
  variable a,

  variable b  : typenname1;


FUNCTION funktionsname(Parameter) : Typ;

 BEGIN
 ...
 END;

PROCEDURE prozedurname(Parameter);
```

```
  BEGIN
    ...
  END;

BEGIN
    ...
END.
```

Allgemeine Form einer Borland-Pascal-Unit

Ähnlich wie beim Programm verhält es sich auch bei der Unit von Borland-Pascal. Es müssen auch hier nicht immer alle Teile existieren. Welche jedoch unerläßlich sind, und welchen Sinn die anderen haben, wird nun erläutert:

UNIT legt den Namen der Unit fest. Sie sollte dem Dateinamen entsprechen und somit nicht länger als acht Zeichen sein.

INTERFACE leitet den Beginn des "öffentlichen" Teils ein. Dieser stellt die Schnittstelle zum später übergeordneten Programm dar. Nur die hier deklarierten Prozeduren, Funktionen, Variablen und Typen können dann auch benutzt werden.

USES,CONST,TYPE,VAR siehe Struktur des Programmes.

PROCEDURE/FUNCTION legt den Namen und die Parameterliste der öffentlichen Prozeduren und Funktionen fest. Die Anweisungen selbst werden hier noch nicht mit angegeben.

IMPLEMENTATION leitet den inoffiziellen, Unit-internen Teil ein. Hier wird explizit ausgeführt, wie die Prozeduren im einzelnen aussehen sollen. Hier können außerdem Konstanten, Typen und Variablen, aber auch Prozeduren und Funktionen deklariert werden, die später nicht mehr benutzt werden, sondern nur als Hilfsprozeduren existieren sollen. Der Kopf von Prozeduren/Funktionen braucht keinerlei Parameter mehr zu beinhalten.

BEGIN/END schließen die Anweisungen ein, die direkt beim Aufruf durch das Programms aufgerufen werden sollen. Das könnten z.B. Initialisierungen sein, die zum ordnungsgemäßen Ablauf der Unit notwendig sind. Achten Sie bitte auch hier auf den Punkt hinter dem letzten END.

Und der Nutzen der ganzen Sache? Richtig, er ist sicherlich auf den ersten Blick nicht zu sehen. Im Computer jedoch zeigt sich die Lösung; eine Unit wird nämlich nur soweit in

den Hauptspeicher gelassen, wie sie tatsächlich benutzt wird. Prozeduren, die nicht benutzt werden, werden nicht geladen. Das ist bei großen Programmen ein erheblicher Vorteil, wenn man nur wenige Prozeduren auf der Tool-Box benutzen will. Es entfällt auch eine lästige und zeitraubende Neukompilierung beim Laufenlassen eines übergeordneten Programmes.

```
Unit unitname;

INTERFACE

 USES ...;

 CONST ...;

 TYPE ...;

 VAR ...;

 PROCEDURE prozedurname(Parameter);

 FUNCTION funktionsname(Parameter) : funktionstyp;

IMPLEMENTATION

 USES ...;

 CONST ...;

 TYPE ...;

 VAR ...;

 PROCEDURE prozedurname;
   BEGIN
     ...
   END;

 FUNCTION funktionsname;
   BEGIN
     ...
   END;
```

```
BEGIN
  ...
END.
```

Anhang C

Der Inhalt der Programmdiskette

23 Datei(en) 265216 Bytes frei
Inhaltsverzeichnis von A:\
Speichermedium in Laufwerk A ist BP_GRAFIK

3D_SIM.PAS	782	20.08.91	10.50
ANIMATI.PAS	12148	23.01.93	18.00
ANIMATI2.PAS	8428	23.01.93	18.00
ANIMAT3D.PAS	9832	22.08.91	17.19
ANIMAT3D.TPU	12160	22.08.91	17.19
ANIMATIO.PAS	8680	19.08.91	13.22
ANIMATIO.TPU	9456	20.08.91	10.50
D1.PAS	574	23.01.93	18.00
GENENST.TXT	62	23.01.93	18.00
GRAFIK2D.TXT	206	19.08.91	13.25
GRAFIK3D.TXT	276	22.08.91	16.45
HIDDENLI. PAS	696	25.08.91	11.27
MÜNZE.PAS	703	20.08.91	13.32
SPIEGEL.TXT	70	23.01.93	18.00
TEST.PAS	505	23.01.93	18.00
TEST_2D.PAS	276	22.08.91	16.42
TEST_3D.PAS	580	22.08.91	17.23
V1.PAS	636	14.07.91	14.36
V10.PAS	2759	19.08.91	13.11
V11.PAS	5577	19.08.91	13.14
V2.PAS	3116	14.07.91	14.52
V3.PAS	3117	14.07.91	15.19
V4.PAS	5436	14.07.91	15.37

V5..PAS	6666	18.08.91	12.02
V6.PAS	5828	18.08.91	12.14
V7.PAS	6583	18.08.91	12.43
V8.PAS	456	19.08.91	13.06
V9.PAS	729	19.08.91	13.08
VRECHNG.PAS	4204	23.01.93	18.00
WÜRFEL.TXT	274	20.08.91	13.02
WÜRFELHI.PAS	490	23.01.93	18.00

Anhang D

Literaturempfehlungen

Bereich Informatik

Bartel:
Grafikprogrammierung mit Turbo-Pascal 6.0 (Vieweg Verlag)

Weidner/Strauss:
Grafik und Animation in C (Vieweg Verlag)

Aupperle:
Objektorientierte Programmierung mit Turbo-Pascal (Vieweg Verlag)

Bereich Physik

Berkeley Phasik Kurs:
Band 3: Schwingungen und Wellen (Vieweg Verlag)

Dorn/Bader:
Physik Oberstufe - Band O (Optik) (Schroedel Verlag)

Bereich Mathematik

Fischer:
Lineare Algebra Band 1 (Vieweg Verlag)

Dallmann/Elster:
Einführung in die Höhere Mathematik 1 (Vieweg Verlag)

Dörfler/Peschek:
Mathematik für Informatiker (Hanser Verlag)

Bronstein/Semendjajew:
Taschenbuch der Mathematik (Verlag Harri Deutsch)

Index

!

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z

Formelsammlung der Grafikanimation

Hier soll nun als letzter Buchanhang eine Gesamtübersicht der benutzten Formeln gegeben werden. Als Formelsammlung gedacht sollten Sie diese aus dem Buch herausnehmen und beim Programmieren Ihrer eigenen Animation benutzen. Sie sollten aber immer folgendes beachten:

- Wenn Sie eine Figur/ ein Objekt um einen bestimmten Winkel drehen wollen, so bezieht sich der Winkel ω immer auf das Bogenmaß. Die Umrechenvorschrift wurde als letzte Formel (nächste Seite) eingefügt.

- Wenn Sie in einem Bildschritt mehrere Manipulationen gleichzeitig durchführen möchten, kann es unter Umständen eine nicht unerhebliche Rechen- und damit Zeit-Ersparnis bedeuten, die Zuweisungsformeln nicht einfach für jede Manipulation aneinanderzuhängen, sondern eine neue, mathematisch vereinfachte Manipulationsvorschrift zu finden.

- Alle Operationen in der Formelsammlung beziehen sich auf folgende Anfangsvektoren:

$$\vec{x} = \begin{Bmatrix} x \\ y \end{Bmatrix} \text{ für 2 Dimensionen, bzw. } \vec{x} = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix} \text{ für 3 Dimensionen}$$

Ich wünsche Ihnen bei Ihren Experimenten mit der Grafik viel Erfolg!

Andreas Bartel

Art der Manipulation	Zuweisungsvorschrift 2-dimensional	Zuordnungsvorschrift 3-dimensional
Verschiebung von $\vec{x}$ um $\vec{V}=\begin{Bmatrix} V_x \\ V_y \\ V_z \end{Bmatrix}$	$x_{neu} = x_{alt} + V_x$ $y_{neu} = y_{alt} + V_y$	$x_{neu} = x_{alt} + V_x$ $y_{neu} = y_{alt} + V_y$ $z_{neu} = z_{alt} + V_z$
Skalierung von $\vec{x}$ um $\vec{S}=\begin{Bmatrix} S_x \\ S_y \\ S_z \end{Bmatrix}$	$x_{neu} = x_{alt} * S_x$ $y_{neu} = y_{alt} * S_y$	$x_{neu} = x_{alt} * S_x$ $y_{neu} = y_{alt} * S_x$ $z_{neu} = z_{alt} * S_z$
Spiegelung x von $\vec{x}$	$x_{neu} = x_{alt} * (-1)$ $y_{neu} = y_{alt}$	$x_{neu} = x_{alt} * (-1)$ $y_{neu} = y_{alt}$ $z_{neu} = z_{alt}$
Spiegelung y von $\vec{x}$	$x_{neu} = x_{alt}$ $y_{neu} = y_{alt} * (-1)$	$x_{neu} = x_{alt}$ $y_{neu} = y_{alt} * (-1)$ $z_{neu} = z_{alt}$
Spiegelung z von $\vec{x}$	- nicht ausführbar! -	$x_{neu} = x_{alt}$ $y_{neu} = y_{alt}$ $z_{neu} = z_{alt} * (-1)$
Drehung um die x-Achse von $\vec{x}$ um den Winkel ω	- nicht ausführbar! -	$x_{neu} = x_{alt}$ $y_{neu} = y_{alt} * \cos\omega - z_{alt} * \sin\omega$ $z_{neu} = y_{alt} * \sin\omega + z_{alt} * \cos\omega$
Drehung um die y-Achse von $\vec{x}$ um den Winkel ω	- nicht ausführbar! -	$x_{neu} = x_{alt} * \cos\omega + z_{alt} * \sin\omega$ $y_{neu} = y_{alt}$ $z_{neu} = z_{alt} * \cos\omega - x_{alt} * \sin\omega$
Drehung um die z-Achse von $\vec{x}$ um den Winkel ω	$x_{neu} = x_{alt} * \cos\omega - y_{alt} * \sin\omega$ $y_{neu} = x_{alt} * \sin\omega + y_{alt} * \cos\omega$	$x_{neu} = x_{alt} * \cos\omega - y_{alt} * \sin\omega$ $y_{neu} = x_{alt} * \sin\omega + y_{alt} * \cos\omega$ $z_{neu} = z_{alt}$
Umrechung von Grad α auf Bogenmaß ω	$\omega = \alpha * \pi / 180$ *mit* $\pi \approx 3.141592654$	