CHIP PROGRAMMIER-PRAXIS 84029/92001
29,– DM

Sources
auf Diskette
im Heft

Ausgabe
19

PASCAL

Turbo-Pascal

- Optimierung: Schnelle Assemblerroutinen
- Erfahrungsbericht: Stony-Brook-Pascal 6.0
- Toolbox: Netzwerk-Programmierung
- Grafik: Fenster und Sprites
- Grundlagen: Kopierschutz für eigene Programme
- Einsteigerseiten: Matrizenberechnung
- TP für Profis: Kampf den Speicherfressern

Markus Mück

Grafiken und ihre Animierung in Turbo Pascal

Ziel dieses Artikels ist es, den Anwender in die Lage zu versetzen, aus beinahe jedem (Zeichen-)Programm auf dem Bildschirm dargestellte Grafiken abzufangen, sie zu bearbeiten und sinnvoll in eigenen Programmen zu verwenden. Gleichzeitig wird es ihm möglich sein, Grafiken an verschiedene Grafikkarten anzupassen und damit schnell und einfach Konvertierungen zu erstellen. Schließlich geht es noch um die grundlegenden Möglichkeiten eines scheinbaren Bewegungsablaufs, sogenannte Animation.

Bevor wir eine Grafik bearbeiten können, muß sie von ihrer Umgebung getrennt werden. Diesen Zweck erfüllt das Programm SAVE.ASM. Es speichert auf dem Bildschirm dargestellte Grafiken in einer für uns weiterverwendbaren Datei. SAVE.ASM ist sehr flexibel und unabhängig von der Grafikkarte. Die Installation erfolgt nach der Assemblierung (für Nicht-Maschinensprache-Kenner ist eine bereits assemblierte SAVE-Version beigelegt) folgendermaßen:

```
SAVE XXX, YYY, FILENAME[, Palette]
```

Die beiden Parameter XXX und YYY stehen für die Auflösung in X-, beziehungsweise Y-Richtung. Die Werte müssen jeweils dreistellig eingegeben werden (zum Beispiel "009" für 9, "090" für 90, "640" für 640 etc.). Setzen Sie beispielsweise anstelle von XXX "008" und von YYY "009" ein, wird ein 8 Punkt breites und 9 Punkt langes Rechteck vom linken oberen Bildschirmrand ausgeschnitten und gespeichert.

Filename steht anstelle eines Dateinamens, nach welchem die spätere Bilderdatendatei benannt sein wird. Hier sollte der Pfadname nicht fehlen, da SAVE die Datei andernfalls in dem beim Speichern aktuellen Verzeichnis ablegt. Der Parameter Palette ist nur für VGA- oder MCGA-Kartenbesitzer interessant. Er erlaubt das Speichern sämtlicher Farbinformationen mit den Grafikdaten. Dies ist besonders bei 256-Farbanwendungen sehr nützlich, da diese fast ausschließlich eine veränderte Palette (Farbreihenfolge) verwenden. Folgende Werte sind anstelle von Palette zulässig:

1 Speichern der 16-Farb-VGA-Palette
2 Speichern der 256-Farb-VGA/MCGA-Palette

Sollten Sie kein Speichern dieser Informationen erwünschen, so lassen Sie den Palette-Parameter mit dem dazugehörigen Komma einfach aus. All diese Informationen stellt Ihnen auch das Programm SAVE.ASM in Kurzform zur Verfügung, wenn Sie es ohne jegliche Angabe von Parametern aufrufen. Nach der Installation genügt ein kurzes Berühren der Prt-Scr-Taste - und schon ist SAVE.ASM bei der Arbeit.

Im weiteren Verlauf dieses Abschnitts geht es um den Aufbau einer von SAVE.ASM erstellten Grafikdatei. Dieses Wissen ist aber nicht unbedingt notwendig, deshalb können Sie den Abschnitt ohne weiteres überspringen.

Aufbau einer SAVE-Grafik-Datei

- Größe in Y-Richtung (Word)
- Größe in X-Richtung (Word)

Alle zu diesem Beitrag gehörenden Files finden Sie im SubDir ANIMAT auf beiliegender Diskette

- Palette, entweder 0, 1 oder 2 (Byte)
- Paletteanteile (16 oder 256*3 Bytes)
- Pixel, geordnet in X-Richtung (Bytes)
  Dieser Eintrag in der Grafikdatei weist lediglich darauf hin, ob keine Palette (0), eine 16-Farbpalette (1) oder eine 256-Farbpalette (2) gespeichert wurde. Je nachdem folgen entweder keine weiteren Einträge, 16 Bytes oder 768 (3*256) Bytes. Die Fülle der Bytes bei der 256-Farbpalette erklärt sich daraus, daß für jeden Farbwert je ein Grün-, Blau- und Rotanteil gespeichert werden muß - also drei Bytes je Wert. Die Farbanteilswerte werden dabei in der gerade beschriebenen Reihenfolge abgelegt.

Konvertieren eingefangener Grafiken

Die Konvertierung von Bildern reduziert die Qualität

Viele Programmierer setzen einen bei Anwendern nicht sehr beliebten Trick ein; sie schreiben eine EGA- oder VGA-Fassung ihres Programms, setzen alle Grafiken mit Hilfe eines Konverters in andere Grafikmodi (CGA, Hercules etc.) um und sparen sich dabei viel Arbeit. Der von den Anwendern gefürchtete Nachteil: Die konvertierten Bilder erreichen ohne Nachbearbeitung nur in den seltensten Fällen das Niveau ihres Originals.

Das Programm CONVERT.PAS:

Das Programm wird unmittelbar nach seinem Start nur den Menüpunkt 0 (Angabe des Grafikdateinamens) zur Auswahl stellen. Wählen Sie diesen an, und geben Sie den Namen einer zuvor mit SAVE.ASM gespeicherten Grafik und den gewünschten späteren Namen der konvertierten Grafik an. Daraufhin bietet CONVERT verschiedene Konvertiermöglichkeiten zur Auswahl. Suchen Sie sich die gewünschte aus, geben Sie die Farbzusammenhänge ein, und erfreuen Sie sich an dem folgenden Aufbau. Dieses Programm achtet bei der Konvertierung auf Proportionen. Es kann daher vorkommen, daß der rechte oder untere Rand nicht ganz angefüllt ist. Dank dieser Vorgehensweise wirken spätere Bilder nicht gestreckt oder zusammengedrückt. Shift-Q beendet CONVERT.PAS. Folgendermaßen können Sie CONVERT.PAS um Konvertiermöglichkeiten erweitern (alle Erweiterungen erfolgen in der DEFINEKASTEN-Prozedur unmittelbar nach dem Variablendeklarationsteil im Programmkopf):

- Verändern Sie in der DEFINEKASTEN-Prozedur am Anfang des Programms die Variable ANZAHLKONV. Sie gibt die Anzahl der zur Verfügung stehenden Konvertiermöglichkeiten an (maximal sind neun erlaubt).
- Belegen Sie in dieser Prozedur die Variable MTEXT[x] (x gibt die Nummer der Konvertiermöglichkeit an) mit dem Text, der zur Beschreibung im Menü erscheinen soll.
- Belegen Sie die Variable KONV[x, 1-8] wie folgt (x steht anstelle der Nummer der Konvertiermöglichkeit):

KONV[x,1]	Auflösung der Ausgangsgrafik in X-Richtung
KONV[x,2]	Auflösung der Ausgangsgrafik in Y-Richtung
KONV[x,3]	Anzahl Farben im Ausgangsmodus - 1 (Beispiel EGA: 16 Farben, KONV[x,3]=15)
KONV[x,4]	Auflösung der Zielgrafik in X-Richtung
KONV[x,5]	Auflösung der Zielgrafik in Y-Richtung
KONV[x,6]	Treibercode des Zielmodus von Turbo-Pascal (zum Beispiel für EGA-Treiber: "EGA")
KONV[x,7]	Grafikmoduscode des Zielmodus von Turbo-Pascal (zum Beispiel für eine EGA-Auflösung von 640 mal 350 Punkten: "EGAHI")
KONV[x,8]	Anzahl Farben im Zielmodus - 1

Anmerkung: CONVERT.PAS muß jederzeit auf die Turbo-Pascal-BGI-Treiber zugreifen können. Kopieren Sie sich dieses Programm daher in Ihr Turbo-Pascal-Unterverzeichnis auf die Festplatte.

Rückholen konvertierter Bilder in Grafikprogramme

CONVERT.PAS ist beim Umstellen von Programmen auf andere Grafikkarten eine große

Hilfe. Dennoch wird es nur selten angebracht sein, eine konvertierte Grafik sofort weiterzuverwenden. Ein kurzes Nachbearbeiten mit einem Grafikprogramm ist in den meisten Fällen mehr als angebracht.

Dies ist mit dem Programm LOAD.ASM möglich. Bei seiner Installation ist nur der Datei- und Pfadname notwendig, alles andere erledigt LOAD.ASM. Nun genügt ein Druck auf die Prt-Scr-Taste und schon erscheint die gewünschte Grafik auf dem Bildschirm.

Das Problem bei der Übernahme liegt bei den Grafikprogrammen selbst. Nur selten nehmen sie ein ihnen auf diese Art und Weise aufgezwungenes Bild an. Damit sie dies dennoch tun, genügt meist je ein Punkt an den äußeren Ecken oder ein Umwandeln in eine Pinselgrafik des Bildes. Experimentieren Sie ruhig ein wenig mit ihrem Zeichenprogramm.

Teile eines eingefangenen Bildes ausschneiden

Sie werden nur sehr selten den gesamten Grafikbildschirminhalt in ein eigenes Programm einbinden wollen. Zu diesem Zweck liegt das Programm CUT.PAS bei. Es erlaubt ein benutzerfreundliches Ausschneiden und Speichern von Teilen des Grafikbildschirms. Die ausgeschnittenen Teile werden dabei automatisch in das IMAGE-Format der Turbo-Pascal-Befehle GETIMAGE und PUTIMAGE umgewandelt. Dadurch ist es möglich, die gespeicherten Daten sofort und ohne Änderung in eigene Programme zu übernehmen. Näheres dazu im nächsten Abschnitt.

Starten Sie erst einmal CUT.PAS, geben Sie den Dateinamen eines mit SAVE.ASM gespeicherten Bildes an, und wählen Sie den Grafikmodus aus, in welchem Sie die Grafik editieren möchten. Der Grafikmodus muß dabei keinesfalls mit dem bei der Speicherung des Bildschirmes übereinstimmen. Überstehende Teile werden einfach abgeschnitten.

Nach dem Ladevorgang steht die Grafik zum Editieren bereit. Bewegen Sie dazu den anfangs im linken oberen Eck befindlichen, zwei mal zwei Punkte großen Cursor mit den Cursortasten (Einerschritte) - oder schneller mit den Nummernblocktasten (Zwanzigerschritte) - zur linken oberen Ecke der auszuschneidenden Grafik. Bedenken Sie dabei, daß der Computer auch die von dem Cursor verdeckten Teile zu ihrem gewünschten Ausschnitt hinzurechnet. Ist der Cursor einmal richtig positioniert, drücken Sie die RETURN- beziehungsweise ENTER-Taste. Der alte Cursor wird nun an der von Ihnen bestimmten Position verharren, während Sie einen zweiten Cursor zur rechten unteren Ecke des gewünschten Grafikausschnitts bewegen können.

Verwaltungstechnische Probleme erlauben leider keine Ausschnitte, welche mehr als 64-KByte-Speicher beanspruchen. Sollte der von

Cursortasten	Bewegen des Cursors in Einerschritten
Nummernblocktasten	Bewegen des Cursors in Zwanzigerschritten
RETURN beziehungsweise ENTER	Ein-/Ausblenden des zweiten Cursors und gleichzeitiges Erlauben, den Grafikausschnitt mit "S" zu speichern
SPACE	Einrahmen des gewählten Ausschnittes (nur bei eingeblendetem zweiten Cursor)
"C"	Ändern der Cursorfarbe auf schwarzem Hintergrund und der Farben der im unteren Bildschirmbereich gezeigten Informationen
"N"	Aktuellen Dateinamen für Speichern ändern
"S"	Speichern eines Ausschnittes (nur bei eingeblendetem zweiten Cursor)
"F"	Wenn Farbpalette in der Bilddatei vorhanden, diese in gesondertem File ablegen
"L"	Koordinaten ein/ausblenden
"I"	Dateiname der Ausschnittsdatei ein/ausblenden
Shift-"Q"	Verlassen des Editiermodus/Verlassen des Programms im Auswahlmodus

Diese Tastaturkommandos sind im Editiermodus zulässig

Ihnen gewünschte Ausschnitt diese Größe erreicht haben, läßt er sich nicht mehr vergrößern. Ein Druck auf die SPACE- beziehungsweise Leertaste umrahmt den eben eingestellten Bereich. Hier können Sie sich, ohne gleich endgültige Entscheidungen treffen zu müssen, genaue Informationen über das spätere Aussehen des Ausschnitts verschaffen. Jede weitere Taste läßt dieses Fenster wieder verschwinden. Haben Sie Ihren gewünschten Ausschnitt mit beiden Cursorn umrissen, so verewigt ein Drücken der S-Taste (für Speichern) diesen unter dem gewünschten Namen. Dieser ist übrigens im linken unteren Eck, die Koordinaten des Cursors sind im rechten unteren Eck angezeigt. Ein weiteres Drücken der RETURN- beziehungsweise ENTER-Taste beendet den Zwei-Cursor-Modus.

Jetzt sind die Grafiken gespeichert

Das Ergebnis unserer bisherigen Bemühungen ist nun also eine Datei, die Daten über eine bestimmte Grafik enthält. Die nächsten beiden Abschnitte werden sich dem Problem widmen, wie diese Daten eigenen Programmen schnell zur Verfügung stehen können.

Anmerkung: CUT.PAS muß jederzeit auf die Turbo-Pascal-BGI-Treiber zugreifen können. Kopieren Sie sich dieses Programm daher in Ihr Turbo-Pascal-Unterverzeichnis auf die Festplatte.

Nachladen von Grafiken

Bisher ist es uns geglückt, bestimmte, für uns wichtige Bestandteile einer Grafik in einer Datei dauerhaft festzuhalten. Nun möchten wir diese Bilder von einem Turbo-Pascal-Programm aus nachladen und auf dem Bildschirm darstellen.

Zu diesem Zweck stellen wir uns die Bilderdatei BILD.IMG mit der Filelänge 2000 Bytes vor. Die Länge des Files ist sehr wichtig und kann durch den DIR-Befehl dem Directory unter DOS entnommen werden. Der wesentliche Teil eines Ladeprogramm könnte folgende Gestalt haben:

```
Program Laden;
  Uses Crt, DOS, Graph;
  var P    : Pointer;       {Zeiger auf Bilddaten    }
      F    : File of Byte;  {Zum Laden notwendig     }
  ...
  Begin
    ...                     {Grafikmodus aktivieren  }
    getmem(P,2000);         {2000 Bytes Speicher für
                             die Grafik reservieren  }
    Assign(F,'BILD.IMG');   {Bilddatei öffnen        }
    Reset(F,1);             {Bilddatei zurücksetzen  }
    BlockRead(F,P^,2000);   {Die 2000 Bytes in re-
                             servierten Speicher
                             lesen                   }
    Close(F);               {Bilddatei schließen     }
    PutImage(0,0,P^,0);     {Bild darstellen         }
    ...
  end.
```

Die Vorgehensweise ist an diesem Beispiel leicht erkennbar: Zuerst mit Hilfe einer Pointer-Variable den von der Grafikdatei benötigten Speicher, in unserem Fall 2000 Bytes, mit GETMEM reservieren, dann die Daten aus der Datei in diesen reservierten Bereich laden und das Bild darstellen.

Einbinden von Grafiken in eigene Programme

Das feste Einbinden eigens erstellter Grafiken ist wohl das gebräuchlichste Verfahren, wenn Bilder in eigenen Programmen benötigt werden. Besonders bei einer Vielzahl von Dateien ist wiederholtes Laden beim Programmstart auf die Dauer lästig.

Bevor wir die Grafikdaten in eigene Programme einbinden können, müssen wir die zugehörigen Dateien in ein fest einbaubares Format, in das OBJ-Format verwandeln. Dies geschieht mit dem Turbo-Pascal-Hilfsprogramm BINOBJ. Seine Syntax lautet wie folgt:

```
BINOBJ Quelldateiname Zieldateiname Prozedurname
```

Anstelle der Parameter "Quelldateiname" und "Zieldateiname" geben Sie einfach die gewünschten Dateinamen der Grafikdatei und der konvertierten Datei ein. Für den Parameter "Prozedurname" lassen Sie sich ein Wort einfallen, das auf den Bildinhalt schließen läßt; unter ihm wird für die Grafikdaten im späteren Turbo-Pascal-Programm eine Prozedur erstellt. Für jede einzelne Grafikdatei ist eine solche Prozedur mit jeweils verschiede-

nem Namen notwendig. Sie hat folgende Gestalt:

```
Procedure PROZEDURNAME; external;
{$L Zieldateiname}
```

Achtung: Die Angaben in den geschweiften Klammern dürfen nicht übergangen werden. Sie stellen für Turbo-Pascal einen Befehl wie jeder andere dar.

Nun benötigen wir nur noch je eine Pointer-Variable für die einzelnen Grafikdaten-Prozeduren, der wir die jeweilige Adresse des Speicherbereichs zuordnen, an der die Grafikdaten untergebracht sind, und schon ist das Bild im Programm verewigt. Eine Speicheradresse kann wie folgt bestimmt werden:

```
P:=ADDR(PROZEDURNAME);
    { P ist eine Pointer-Variable }
```

Ein Befehl wie

```
PutImage(0,0,P^,0);
```

brächte die eingebundene Grafik nun auf den Bildschirm.

Animation

Dieser Abschnitt wird erläutern, wie einfache Animationen in eigenen Programmen realisiert werden können. Leider sind die Möglichkeiten auf Grund der beschränkten Anpassungsfähigkeiten der Turbo-Pascal-BGI-Treiber nicht sehr vielfältig. So können beispielsweise nur auf einfarbigem Hintergrund zufriedenstellende Ergebnisse erzielt werden.

Die Voreinstellung des Grafikmodus setzt die Hintergrundfarbe auf Schwarz. Da dies nicht immer eine ideale Hintergrundfarbe für Animationen ist, müssen wir die Hintergrundfarbe unmittelbar nach dem Einschalten des Grafikmodus ändern. Dies kann beispielsweise auf folgende Art und Weise geschehen:

```
SetBKColor( Farbe )
```

Bevor wir eine Animationssequenz auf dem Bildschirm darstellen können, müssen wir sie erst einmal zeichnen. Dies kann mit jedem handelsüblichen Malprogramm geschehen. Achten Sie hierbei darauf, daß ein möglichst großer Teil aller später animierten Bilder gleich beschaffen ist.

Ein Beispiel: Das Demo-Programm DEMOEGA.PAS für EGA beziehungsweise VGA-Kartenbesitzer und DEMOCGA.PAS für CGA-Kartenbesitzer zeigen die Vorgehensweise. Bei beiden Programmen wird ein Diskettenlaufwerk dargestellt, in welches eine Diskette einfährt. In diesem Fall wird ein Großteil der Laufwerksgrafik während der ganzen Animation nicht verändert. Es genügt, diesen Teil nur ein einziges Mal darzustellen. Lediglich die einzelnen Diskettenbilder müssen nach jedem Schritt gelöscht und das nächste neu dargestellt werden. Dadurch erreichen wir eine enorme Zeitersparnis und eine viel fließendere Animation. Selbstverständlich ist diese Verfahrensweise nur bei festen Animationen und nicht bei sich über den Bildschirm bewegenden Bildern sinnvoll.

Bei einer Animation werden Bewegungsabläufe als Einzelbilder gespeichert

EGA-Kartenbesitzer haben eine weitere Möglichkeit, noch fließendere Animationssequenzen zu erstellen: Durch Wechseln von Bildschirmseiten. Mit Hilfe dieser Technik ist es möglich, auf einem imaginären, unsichtbaren zweiten Bildschirm eine Grafik aufzubauen und sie nach ihrer Fertigstellung blitzschnell sichtbar zu machen. Nun ist die zuvor sichtbar gewesene Bildschirmseite verschwunden und steht Veränderungen zur freien Verfügung - ohne daß der Anwender etwas bemerkt. Ein solches Animationsprogramm könnte folgende Form haben:

```
SetVisualPage(0);      { Bildschirmseite 0 sichtbar                }
SetActivePage(1);      { Veränderungen auf Bildschirmseite 1 }
....                   { Entwerfen einer Grafik auf Seite 1 }
SetVisualPage(1);      { Blitzartiges Darstellen der Grafik }
SetActivePage(0);      { Veränderungen nun auf Seite 0      }
....                   { Entwerfen der nächsten Grafik      }
SetVisualPage(0);      { Blitzartiges Darstellen der Grafik }
....
```

Zwischen den einzelnen Bildschirmumschaltungen müssen nur jeweils die richtigen Bilder in der korrekten Reihenfolge dargestellt oder gelöscht werden. Ein Ergebnis solcher Animationsversuche können Sie mit den schon vorhin genannten Programmen DEMOEGA.PAS und DEMOEGA2.PAS bestaunen, falls Sie glücklicher EGA- oder VGA-Kartenbesitzer sind. CGA-Eigner können sich an DEMOCGA.PAS und DEMOCGA2.PAS erfreuen.