

OPL Workshop 6 - Gezielte Textausgabe

Ulrich Krinzner, 06/1999

Bisher haben wir OPL kennengelernt und Programme erstellt, ohne uns großartig um eine wesentliche Funktion, nämlich die Ausgabe auf den Bildschirm zu kümmern. In dieser Ausgabe unseres Workshops soll es eben darum gehen - um ganz genau zu sein: um die Textausgabe.

Doch vorher noch zwei Nachrichten. Die schlechte zuerst: Wegen gar keiner Resonanz seitens der Workabout- und Siena-Nutzer klammern wir diese Geräte zumindest vorerst aus. Die gute: Wir unterstützen ab sofort die neu hinzugekommene Serie 5mx PRO.

Und nun zur Sache. Gerade Einsteiger stolpern immer wieder über die verwirrend vielfältigen Möglichkeiten, Texte gezielt auf dem Bildschirm auszugeben. Aber keine Bange, wir schildern Ihnen auch diesmal alles haarklein!

Bildschirm & Fenster

Wir haben im wesentlichen mit zwei begrifflich zu trennenden Dingen zu tun.

Da ist einerseits der Bildschirm. Er ist lediglich die physikalische Ausgabeeinheit, die über einzeln angesteuerte Bildpunkte (pixel, picture cells) ein Bild für unser Auge aufbereitet. Der Serie 3a/c/mx verfügt über immerhin 480*160 Pixel, der Serie 5 bereits über 640*240.

Andererseits organisiert das Betriebssystem die optische Ausgabe intern über eine ausgefeilte Fenstertechnik. Fenster - den meisten von Ihnen vom PC oder Mac bekannt - sind in der Größe frei definierbare rechteckige Flächen, die Raum für Darstellungen jeglicher Art bieten. Der Serie 3a/c/mx kann davon 8, der Serie 5 sogar 64 gleichzeitig offen halten!

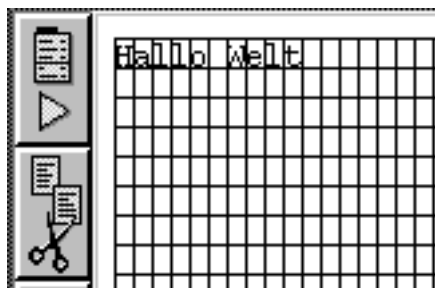
Uns interessiert vorerst jedoch nur einziges davon, nämlich das Basisfenster. Dieses steht uns ungefragt und ohne unser Zutun nach dem Start eines OPL-Programmes immer zur Verfügung und nimmt den gesamten Bildschirm ein. Überzeugen Sie sich, indem Sie ein kurzes Programm eintippen, das einen Rahmen um das Fenster zieht. Wie üblich schließen Sie die eigentlichen Programmzeilen zwischen "PROC name:" und "ENDP" ein, übersetzen es und lassen es schließlich laufen:

```
GET : gBORDER 0 : GET
```

Das Basisfenster beschäftigt uns auch deshalb besonders, weil es das einzige ist, das eine Zwitterstellung einnimmt. Die besteht darin, daß dort Text auf zweierlei Art ausgegeben werden kann, und zwar einmal als normaler Text und andermal als grafischer Text.

"Normalen Text" haben wir bereits mehrfach verwendet. Der zugehörige Befehl heißt "PRINT". Durch ihn wird in einen dafür reservierten Bereich des Basisfensters geschrieben - in das sogenannte Textfenster.

Kariert



Das Textfenster ist optisch nicht ohne weiteres zu erkennen und eigentlich auch nur eine Vorstellungshilfe für uns Menschen. Beim Programmstart hat es fast die gleichen äußeren Abmaße wie das Basisfenster und kann auch nie größer als dieses sein. Um damit vertraut zu werden, stellen Sie sich bitte einmal vor, das Textfenster wäre ein Stück kariertes Papier und auch so unterteilt. Die Besonderheit: es gibt nur vollständige Karos. Auf diese Weise können sie jeden Platz im Textfenster

adressieren, indem Sie eine Spalten- und Zeilennummer angeben. Genau das machen Sie mit dem AT-Befehl. Mit ihm werden Spalte und Zeile benannt - gewissermaßen wird ein unsichtbarer Zeiger gesetzt ("Cursor"). Dort genau beginnt schließlich der folgende PRINT-Befehl mit der Textausgabe. Beispiel:

```
AT 4,2 : PRINT "Hallo!"      REM Text ab Spalte 4, Zeile 2
```

Dabei stehen im Textfenster des Serie 3a/c/mx 60 Spalten und 17 Zeilen zur Verfügung, beim Serie 5 sind es 91 Spalten und 21 Zeilen. Jedenfalls solange man die Schriftart nicht ändert. Verschiedene Schriften ("Fonts") sind aber ein Gestaltungsmittel und daher interessant und wichtig. Ein Merkmal von Fonts ist ihre räumliche Ausdehnung, insbesondere die in "Punkt" gemessene Größe (Höhe). Sie können sich sicherlich sehr gut vorstellen, daß ein größerer Font nicht mehr in die Karostruktur eines kleineren paßt. Würde man also mitten beim Schreiben den Font ändern, käme das Betriebssystem mit der Buchstabenplatzierung ins Schleudern. Also schiebt das System einen Riegel davor, was wiederum nun uns ins Grübeln kommen läßt: Nach dem Befehl zum Ändern des Fonts wird einfach der Textbildschirm gelöscht! Das Betriebssystem berechnet daraufhin die Karostruktur und deren Anordnung im Basisfenster neu. Beispiel für den S5:

```
PRINT "Teil1" : GET : FONT 12,0 : PRINT "Teil2" : GET
```

(am S3a/c/mx benutzen Sie bitte "FONT13,0".)

Der Text "Teil1" wird im Standardfont dargestellt, nach einem Tastendruck wird der Font mit dem entsprechenden Befehl gewechselt. Die erste Zahl sagt dabei, welcher Font benutzt werden soll, die zweite, welche Auszeichnung Verwendung findet (fett, kursiv ..) - die Null steht für "normal", andere gültige Werte entnehmen Sie bitte dem unten stehenden Programm "txtpos1.". Die Neuordnung des virtuellen Karorasters im Basisfenster können Sie gut beobachten, wenn Sie einmal darauf achten, daß der Querbalken des "T" jeweils geringfügig in der Höhe verschoben zur Anzeige kommt.

Fonts

Über diese Fonts verfügen Sie auf dem S3a/c/mx (zugehörige Nummer in Klammern):

Mono 8*8	(4)
Roman 8	(5)
Roman 11	(6)
Roman 13	(7)
Roman 16	(8)
Swiss 8(9)	
Swiss 11	(10)
Swiss 13	(11)
Swiss 16	(12)
Mono 6*6	(13)

Die Zahlen hinter der Fontbezeichnung ist die Schriftgröße in der Maßeinheit „Punkt“; ein kleines Programm zeigt uns die Auswahl noch einmal auf dem Bildschirm:

```
PROC   txtpos1:
      GLOBAL n% : n%=4
      DO
          FONT n%,0          REM entspricht STYLE 0
          PRINT "FONT Nr. ";n%
          PRINT "Text normal"
          STYLE 32 : PRINT "Text kursiv"
          STYLE 4  : PRINT "Text invers"
          STYLE 2  : PRINT "Text unterstrichen"
          n%=n%+1 : GET
      UNTIL n%>13
ENDP
```

Das Beispiel beweist gleichzeitig: Solange man den Font in seiner Größe nicht ändert, kann man sein Aussehen durch den STYLE-Befehl in gewissen Grenzen durchaus beeinflussen. Die im Programm verwendeten Zahlen können auch addiert werden und ergeben dann eine kombinierte Darstellung (32+4 = kursiv und invers). Die Auszeichnung "fett" würde die Ausdehnung des Fonts ändern und ist daher nicht als Option vorgesehen.

Diese Fonts funktionieren zwar auch am S5, fallen dort aber kleiner aus. Probieren sie daher einmal systematisch die Nummern aus dieser Auswahl:

Arial: 268435951 .. 268435961

Times: 268435962 .. 268435972

Courier: 268436062 .. 268436072

REM Beispiel: FONT 268435959,0

Um herauszufinden, wieviele Spalten und Zeilen das Textfenster nach einem Fontwechsel hat, bedienen Sie sich des SCREENINFO-Befehls. Der legt seine Informationen in einem vorher zu deklarierendem Variablenfeld mit 10 Variablen ab und wir fischen uns die wichtigsten heraus:

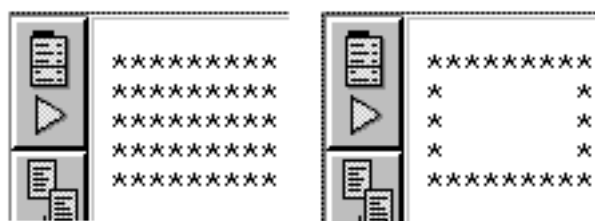
```
LOCAL i%(10)
SCREENINFO i%() : PRINT i%(3),"Spalten" : PRINT i%(4),"Zeilen" : GET
FONT 12,0      REM Fontaenderung
SCREENINFO i%() : PRINT i%(3),"Spalten" : PRINT i%(4),"Zeilen" : GET
```

Änderungen

Ein weiterer Befehl läßt uns aktiv auf Größe und Anordnung des Textfensters Einfluß nehmen: SCREEN. Ein Beispiel:

SCREEN 7,3,2,2

Unter Verwendung der aktuellen Font-Einstellungen wird das Textfeld dadurch auf einen 7 Karos (oder besser Zeichen) breiten und 3 Zeilen hohen Bereich neu eingestellt. Die Entfernung vom Basisfenster-Rand in Zeichen und Zeilen bezieht sich bis zur endgültigen Aktivierung des veränderten Textfensters noch auf das zuletzt benutzte Karo-Raster. In unserem Falle liegt das neue Fenster also ab dem 2. Zeichen vom linken und ab der 2. Zeile vom oberen Rand entfernt. In unserem folgenden Beispiel wird zuerst der linke obere Teil des Standard-Textfensters mit Sternchen gefüllt und dann das Textfenster neu eingestellt.



```
PROC txtpos2:
  GLOBAL n%
  n%=1
  DO : PRINT "*****" : n%=n%+1 : UNTIL n%>5
  SCREEN 7,3,2,2 : GET : CLS : GET
ENDP
```

Sichtbar wird die neue Form und Position allerdings erst nach dem Text-Löschbefehl "CLS". Die Wirksamkeit des neuen Fensters bestätigt ein kurzer Anhang an das gerade geschriebene Programm:

```
PRINT "134567890" : GET
```

Der auszugebende Ausdruck ist länger als die Breite des Textfensters - er wird daher am Rand umgebrochen und in der nächsten Zeile fortgeführt.

Dimensioniert man das Textfenster für ein eigenes Programm richtig, kann man mit den wenigen Befehlen AT x,y / PRINT / CLS einen recht einfach zu handhabenden Anzeigebereich verwirklichen. Zur Erinnerung: PRINT gibt üblicherweise einen oder mehrere Ausdrücke aus und erzeugt dann einen Zeilenumbruch, wenn die Zeile nicht gerade mit einem Komma oder Semikolon endet. Ein Komma in einer Folge von Ausdrücken hinter dem PRINT ergibt auf dem Bildschirm eine Leerstelle. Die Ausdrücke hinter PRINT können sowohl Text als auch Zahlen als auch komplexe Befehle, Funktionen oder Unterprogramme sein, wenn sie nur am Ende eine Zahl oder einen Text ergeben:

```
PRINT "Font Nr.",n%,"ist Arial und",640/8,"Zeichen breit",CHR$(7),ASC("A")
```

Schrift grafisch

Nach dem Ausflug in die "Untermenge" Textfenster kehren wir nun zum Basisfenster zurück. Aus all den vorhergehenden Erklärungen ahnen Sie schon, daß es wir jetzt über die grafischen Eigenschaften reden müssen. Lassen wir in dieser Folge einmal noch die tatsächlichen Grafiken außen vor und bleiben wir bei der Ausgabe von Text.

Die meisten Befehle zur Ausgabe ins Textfenster finden ein Gegenstück für die Ausgabe ins Basisfenster. Weil der Text dort pixelgenau - eben wie auch Grafiken - positioniert werden kann, redet man von "grafischen Text". Die verwendeten Befehle beginnen daher folgerichtig mit einem "g". Die Gestaltungsmöglichkeiten sind weitaus umfangreicher und flexibler, setzen aber mehr Kenntnisse und Sorgfalt beim Programmieren voraus:

- gAT

gibt die (Cursor-)Position vor, ab der der Text ausgegeben werden soll.

- gPRINT

sorgt für die Ausgabe des Textes. Achtung! gPRINT interpretiert die Angaben von "gAT" als einen Punkt auf der Grundlinie des Fonts! Geben Sie eine gPRINT-Anweisung beim ersten Mal ohne eine Positionsangabe ein sehen Sie gar nichts bzw. lediglich ein paar Buchstaben, die unter die Grundlinie schreiben, denn beim Start steht der Cursor an den Grundkoordinaten 0,0:

```
gPRINT "hello - good bye!"
```

Um präzise zu arbeiten, benötigen Sie daher bereits bei der Positionsangabe Kenntnisse über den Font! Ein weiterer Unterschied zum PRINT-Befehl: gPRINT sorgt nicht automatisch für einen Zeilenumbruch.

- gCLS

löscht den GESAMTEN Bereich des Basisfensters - einschließlich den des Textfensters! Dem Geschick des Programmierers bleibt es überlassen, sich vor den unerwünschten Folgen zu schützen, z.B. das Textfenster nach einem gCLS wieder neu zu beschreiben, damit eine evtl. darin enthaltene Information nicht verloren geht.

- gFONT

legt den zu benutzenden Font fest. Im Gegensatz zum Textfenster läßt sich der Font "im Galopp" verändern, ohne daß dabei das Fenster gelöscht wird. Das Text-Attribut wird in der gFONT-Anweisung nicht benannt, sondern bei Bedarf mit gSTYLE manipuliert:

```
gat 10,30 : gFONT 9 : gPRINT "H" : gFONT 10 : gPRINT "A"  
gFONT 11 : gPRINT "LL" : gFONT 12 : gPRINT „O“
```



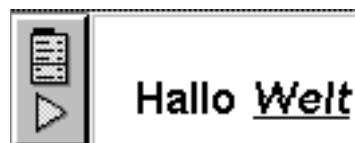
- gSTYLE

verhilft dem Text zu seinen Attributen (fett, kursiv ..), kennt aber im Gegensatz zu seinem Pendant STYLE eine größere Auswahl. Sie darf ebenfalls durch Addition der Werte kombiniert werden kann:

gSTYLE 0	normal
gSTYLE 1	fett
gSTYLE 2	unterstrichen
gSTYLE 4	invers
gSTYLE 8	doppelt hoch
gSTYLE 16	mono
gSTYLE 32	kursiv

Zum Beispiel:

```
gat 10,30 : gFONT 12 : gPRINT "Hallo " : gSTYLE 2+32 : gPRINT "Welt"
```



- gTMODE setzt die Art, wie sich die Schrift auf die Bildschirmdarstellung auswirkt:

gTMODE 0	Pixel setzen
gTMODE 1	Pixel löschen
gTMODE 2	Pixel invertieren
gTMODE 3	Pixel setzen / Hintergrund löschen

Sehen Sie sich dazu dieses Beispiel an:

```
gAT 65,0 : gFILL 40,130,0 : GET : gFONT 12 : gSTYLE 1
gTMODE 0 : gAT 20,30 : gPRINT "Pixel setzen"
gTMODE 1 : gAT 20,60 : gPRINT "Pixel loeschen"
gTMODE 2 : gAT 20,90 : gPRINT "Pixel invertieren"
gTMODE 3 : gAT 20,120 : gPRINT "Pixel setzen / Hg. loeschen"
GET
```



Hier wird auch besonders deutlich, was "grafisch" bedeutet: Die Schrift wird Pixel um Pixel gesetzt - im Normalfall wird dabei der "weiße Block" um den Buchstaben nicht mit ausgegeben, wie das aber im Textfeld üblich ist. Zur Verdeutlichung ergänzen Sie einmal das vorstehende Programm um diese Zeile:

```
AT 11,2 : PRINT "OX" : GET
```

- gSETWIN

ist vergleichbar der SCREEN Anweisung. Die Parameter werden allerdings anders benutzt. Bei zwei angegebenen Parametern wird lediglich die Position des Basisfensters versetzt, bei vier Parametern auch noch Breite und Höhe. Die Angaben sind Pixelwerte. Beispiel:

```
gBORDER 0 : gSETWIN 40,40 : GET : gSETWIN -40,-40 : GET
gSETWIN 40,20,gWIDTH-80,gHEIGHT-40 : gCLS : gBORDER 0 : GET
```

Durch gBORDER erkennt man das Fenster erst so richtig. gSETWIN positioniert nun die linke obere Ecke des Fensters neu auf die Koordinaten 40,40. Daß dabei das Fenster tatsächlich nur verschoben und nicht verändert wird, zeigt die zweite Verschiebung auf die Anfangskoordinaten -40,-40, bei der der Rahmen erhalten bleibt. Jawohl, richtig gelesen! Das grafische Fenster verarbeitet auch negative Koordinaten, ohne daß das Programm abbricht.

Erst die Verkleinerung des Fensters um je 40 Pixel an der linken und rechten Seite sowie die Verminderung der Höhe um je 20 Pixel oben und unten unterbricht den Rahmen. Der muß deshalb in unserem Beispiel auch neu gezeichnet werden. Damit seine Ecken auch wieder schön rund werden, steht noch ein gCLS vor dem Rahmenbefehl.

Die Verwendung von gWIDTH und gHEIGHT, deren aktuellen Werte das Betriebssystem selber kennt und die der aktuellen Weite und Höhe des Fensters in Pixel entsprechen, ersparen entsprechendes Kopfrechnen - lediglich die Seitenabstände werden noch subtrahiert, damit das neu dimensionierte Fenster zentrisch auf dem Bildschirm zu sehen ist.

Wer dem Frieden nicht traut, ob das Programm auch wirklich das getan hat, was ihm gesagt wurde, prüfe, ob der Abstand des neuen Basisfensters tatsächlich 40 und 20 Pixel vom Bildschirmrand liegt mit diesen beiden neuen Befehlen, die die Position der linken oberen Ecke des Fensters auf dem Bildschirm melden:

```
gAT 20,20 : gPRINT gORIGINX,gORIGINY
```

Wer darüber hinaus neugierig ist, wird sich mit gINFO auseinandersetzen müssen. Nur soviel: Ähnlich SCREENINFO wird mit gINFO i%() ein Array mit 32 Variablenwerten gefüllt, die hinterher abgefragt werden können und Auskunft über diverse aktuelle Parameter des Basisfensters geben.

Für den Fortgeschrittenen sind aber noch nicht alle Möglichkeiten der grafischen Textausgabe erschöpft. Zur Verfügung stehen nämlich noch weitere Formen, für deren sinnvollen Einsatz man immer erst bei den entsprechenden Spezialfällen zurückkommen wird. Deshalb nur kurz gestreift:

- gPRINTB

zeigt einen Text in einem leeren rechteckigen Feld mit einer frei bestimmbaren Breite. Zusatzparameter bestimmen dabei die Anordnung des Textes darin (links, Mitte, rechts) oder legen gezielt den Freiraum über und unter der Schrift fest.

- gPRINTCLIP

erlaubt, die Textlänge auf eine selbst anzugebende Breite in Pixeln zu begrenzen

- gXPRINT

schreibt den Text vergleichbar dem PRINT-Befehl mit den "leeren" Anteilen, aber auch unterstrichen oder invers.

Mehr

Die Information, wie breit eigentlich ein Schriftzug in Pixel ist, ermittelt für uns gTWIDTH in der Form

```
b%=gTWIDTH text$.
```

So scheinbar unnütz die letzten Befehle waren, so brauchbar sind die beiden letzten für heute. Zuerst

```
GIPRINT "maximal 64 Zeichen lang!",0
```

Der Wert hinter dem Komma (0,1,2 oder 3) bestimmt, in welcher der 4 Bildschirmecken der Text für etwa 2 Sekunden auftaucht. Er ist dabei keinem der Fenster zugeordnet. Da jeder auch sonst übliche Text ausgegeben werden kann, ist das eine exzellente Methode, sich während der Programmentwicklung Variablenwerte etc. anzeigen zu lassen, da ja die Fensterinhalte in keiner Weise betroffen sind. Diese Art der Ausgabe ist auch ganz hervorragend für Nutzerinformationen geeignet, z.B. der Hinweis, wenn es etwas länger dauert: "Initialisierung" oder "Ich rechne"

Gerade der letzte Fall wird noch besser durch den Befehl BUSY abgedeckt, weil man ihn nämlich wieder abschalten muß. So setzt man am Beginn einer langen Berechnung einen blinkenden Text auf eine der vier Ecken des Bildschirms und schaltet ihn danach wieder ab. Die Parameter sind der Text, und optional die Ecke (0,1,2 oder 3) und eine Einschaltverzögerung (in 0,5 Sekunden Schritten):

```
BUSY "Ich rechne",0,2 : PAUSE 60 : BUSY OFF
```

Wie eng grafische Schrift und Grafik selber zueinander stehen, werden wir noch sehen, wenn wir uns mit dem Thema Grafik auseinandersetzen. Alle Grafikbefehle lassen sich nämlich auch auf Schrift anwenden. Aber mit diesem Hinweis wollen wir es dann auch diesmal belassen.

Zum Abschluß des trockenen Stoffes gibt es noch ein kleines Programm, in dem ein großer Teil des neu Gelernten wieder auftaucht - und zu mehr soll es auch gar nicht gut sein ... die Interpretation bewältigen Sie nun spielend selbst. Ergänzend 2 Tips: Der Wartebefehl "PAUSE", verbunden mit einem negativen Wert, ermöglicht die Pause durch einen beliebigen Tastendruck abubrechen und "gFILL " sorgt für das Zeichnen eines schwarz gefüllten Rechtecks.

Und nun: variieren Sie fleißig!

```

PROC sterne:
    GLOBAL n%, z%, zmax%, x%(10), y%(10), font%(10)
    RANDOMIZE SECOND
    GIPRINT "Sternen-Simulation",0
    FONT 4,0 : SCREEN 8,2 : PRINT "**Sterne*"
    n%=1
    DO
        getvars:(n%) : n%=n%+1
    UNTIL n%>10
    zmax%=2 : z%=0
    DO
        n%=1
        DO
            PAUSE 10
            gTMODE 1 : gAT x%(n%),y%(n%) : gFONT font%(n%) : gPRINT "*" REM
loeschen
            getvars:(n%)
            gTMODE 0 : gAT x%(n%),y%(n%) : gFONT font%(n%) : gPRINT "*" REM
schreiben
            n%=n%+1
            UNTIL n%>10
            z%=z%+1
            UNTIL z%>zmax%
            gAT gWIDTH/2-50,gHEIGHT/2-40 : gFILL 100,80,0
            PAUSE -40
            CLS : PRINT : PRINT "***ENDE**"
            PAUSE -100
        ENDP
    ENDP

PROC getvars:(i%)
    x%(i%)=10+zufall%:(gWIDTH-20)
    y%(i%)=10+zufall%:(gHEIGHT-20)
    font%(i%)=8+zufall%:(4)
ENDP

PROC zufall%:(max%)
    RETURN 1+INT(RND*max%)
ENDP

```