

## OPL Workshop 7 - FensterIn mit dem Psion

Ulrich Krinzner, 09/1999

*In der letzten Folge haben wir uns mit der Positionierung und Formatierung von Text auseinandergesetzt. Dabei haben wir unseren Exkurs auf das Standardfenster beschränkt. Dieses Mal erweitern wir den Horizont und stoßen weitere Fenster auf.*

Für viele Programme ist die Verwendung von Fenstern ein hervorragendes Mittel, bestimmte Aufgaben bequem und elegant zu lösen. Darüber hinaus sind neben der eigentlichen Grafik gerade Fenster ganz besonders geeignet, dem eigenen Programm ein "Gesicht" zu geben.

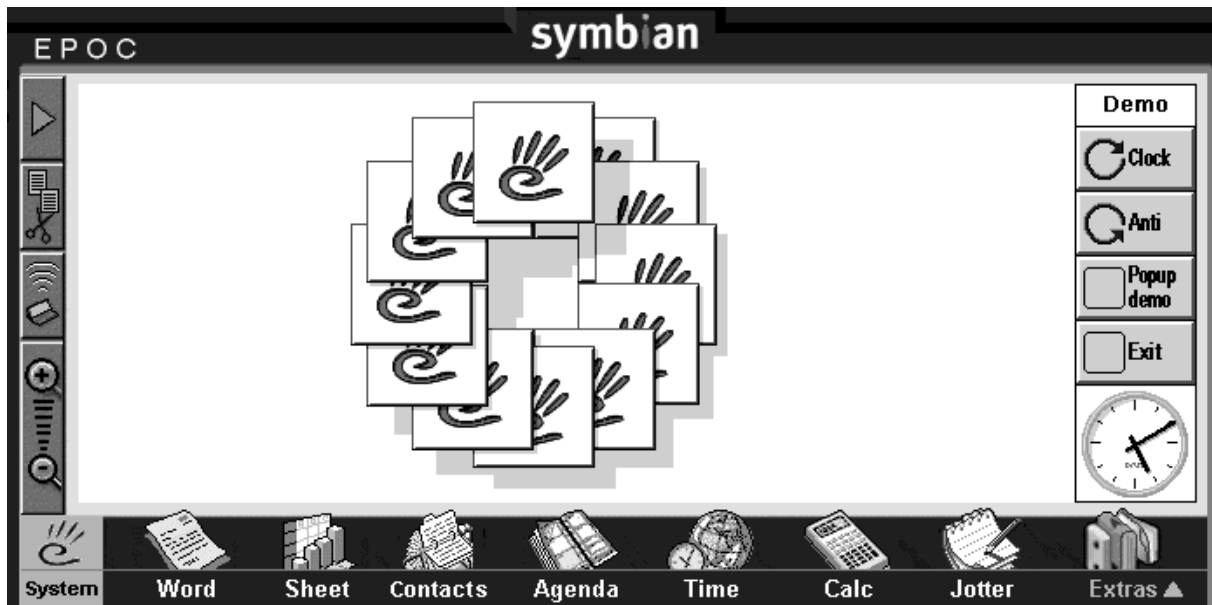


Abb.: Eindrucksvoll: Fenstertechnik aus einer on-board OPL-Demodatei

Zur ihrer Erinnerung: Ein Fenster ist ein selbst definierter, rechteckiger Bereich auf dem Bildschirm mit unabhängigen eigenen Eigenschaften. Der Inhalt des Fensters wird vom Betriebssystem eigenständig verwaltet, so daß man selber weniger Rücksicht beim Programmieren nehmen muß.

### Ein anschauliches Beispiel

Unter Verwendung des Standardfensters wollen Sie erreichen, daß z.B. in der rechten oberen Ecke des Bildschirms ständig das aktuelle Datum angezeigt wird. Der Rest des Bildschirms wird für andere Aufgaben benutzt, muß aber von Zeit zu Zeit mit CLS oder gCLS gelöscht werden. Die Befehle zum Löschen des Bildschirms arbeiten leider nicht selektiv, so daß unerwünschter Weise auch das Datum mit gelöscht wird.

Wenn Sie das Datum nun nicht immer wieder neu schreiben wollen, benutzen sie einfach ein kleines separates Fenster und schreiben es einmalig dort hinein. Der nächste Löschbefehl im Standardfenster ist in einem solchen separaten Fenster nicht wirksam und der gewünschte Effekt erreicht!

Da der Bildschirm lediglich das Ausgabemedium ist, haben sich für die Bewältigung des Problems die kleinen grünen Männchen dahinter ziemlich flott zu bewegen und eine Menge Rechenarbeit zu leisten - aber uns braucht das nicht zu kümmern. Wir genießen einfach das Ergebnis.

### Fenster erzeugen

Auf den Geräten der Serie 3a/c/mx lassen sich nun 8, auf den 5ern bis zu 64 Fenster gleichzeitig anlegen ("öffnen") und verwenden.

Für das Anlegen eines Fensters steht ein komplexer Befehl mit mehreren Parametern bereit:

```
gCREATE (xpos%, ypos%, breite%, hoehe%, sichtbar%)
```

Über die Parameter in der Klammer, allesamt Integerzahlen, teilt man mit, wo das Fenster auf dem Bildschirm erscheinen soll und welche Abmaße es hat. Das verwendete Maß ist "Pixel". Der fünfte Parameter sorgt dafür, daß das Fenster sofort auf dem Bildschirm sichtbar wird (sichtbar% = 1) oder eben nicht (sichtbar% = 0). Ein Grund, ein Fenster unsichtbar anzulegen, besteht darin, es erst mit dem Inhalt zu füllen und danach sichtbar zu machen. Der Betrachter soll den Bildaufbau nicht sehen können, insbesondere, wenn es sich um langsam aufbauende Grafiken handelt. Ein gVISIBLE ON ist dann in der Folge gewissermaßen der Einschalter für das Fenster. Für ein neuen Bildaufbau läßt es sich mit gVISIBLE OFF aber auch erneut verstecken:

```
PROC fenster1:
  GLOBAL win%
  win%=gCREATE(25,25,200,50,0)
  gBORDER 0
  gAT 5,15
  gPRINT "Fenster Nr", win%
  GET
  gVISIBLE ON
  GET
  gVISIBLE OFF
  GET
ENDP
```

Im Beispiel wird ein unsichtbares Fenster mit einer Umrahmung angelegt, in das ein Text geschrieben wird. Nach einem Tastendruck wird es sichtbar und wieder unsichtbar.

Der Befehl "gCREATE()" liefert nach der Ausführung quasi eine Art Fertigmeldung in Form einer Zahl ab, die in diesem Falle die Nummer des Fensters enthält. Dieser Wert wird hier zur Weiterverwendung in der Variablen win% aufbewahrt. Was wir damit anfangen können, zeigt das Beispiel ebenfalls. Der tiefere Sinn besteht allerdings darin, das Fenster gewissermaßen mit einem Namen ansprechen zu können, sonst haben Sie keine Chance, jemals wieder auf andere Fenster (einschließlich dem Standardfenster) zugreifen zu können.

## "Du bist gemeint"

Wenn Sie ein Fenster neu öffnen, ist es automatisch das "aktuelle", mit dem man sofort etwas anfangen kann. Will man in ein beliebiges anderes wechseln, muß man das auch sagen und die Fenster-Nummer, oder wahlweise die Variable, die die Nummer enthält, kennen. Der Befehl dazu heißt "gUSE fensternummer%".

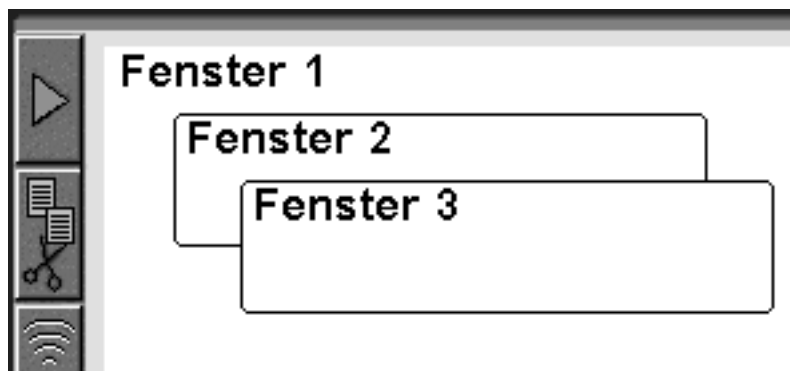


Abb.: Fenster stapeln sich selber

Probieren wir das gleich einmal aus:

```
PROC fenster2:
  GLOBAL win1%, win2%, win3%, x%, y%
  win1%=gIDENTITY
  win2%=gCREATE(25,25,200,50,1)
  gBORDER 0
  win3%=gCREATE(50,50,200,50,1)
  gBORDER 0
  GET
  gAT 5,15
  gPRINT "Fenster", gIDENTITY
  GET
  gUSE win2%
  gAT 5,15
  gPRINT "Fenster", gIDENTITY
  GET
  gUSE win1%
  gAT 5,15
  gPRINT "Fenster", gIDENTITY
  REM 1, Marke f.Ergaenzung
  REM 2, Marke f.Ergaenzung
  REM 3, Marke f.Ergaenzung
  GET
ENDP
```

Beim Programmstart steht lediglich das Standardfenster zur Verfügung. Das fragen wir gleich nach seiner Identifikationsnummer ("ID"), schließlich wollen wir irgendwann auch einmal wieder dahin zurückkehren. "gIDENTITY" ist der entsprechende Befehl, der die gewünschte Angabe liefert. Die ID für das Standardfenster steht eigentlich mit "1" fest, aber der Pingeligkeit halber haben wir auch dort nachgefragt ...

Im weiteren erzeugen wir zwei sofort sichtbare Fenster. Erst auf weitere Tastendrücke (Haltepunkte in vertrauter Weise mit GET ...) werden die Fenster nun mit Text beschrieben. In welches Fenster geschrieben wird, teilen wir dem Betriebssystem per "gUSE id%" mit. Daß wir uns tatsächlich im angegebenen Fenster befinden, bestätigt uns der ausgegebene Wert von "gIDENTITY".

Ist die Fenster-ID einmal genannt, beziehen sich alle weiteren Zeichen- und Schreibbefehle auf eben dieses. Bis zur nächsten Änderung.

## Der Reihenfolge nach

Wenn Sie aufmerksam hinsehen, bemerken Sie, daß neue Fenster immer oben aufgelegt werden, gerade wie beim Ausspielen von Spielkarten bei einem zünftigen Skat. Dadurch können schon mal weiter unten liegende Fenster verdeckt werden - ein Effekt der durchaus auch erwünscht sein kann. Noch dazu, weil man die Schichtungsebene der Fenster beliebig ändern kann. Das Zauberwörtchen lautet "gORDER id%,ebene%".

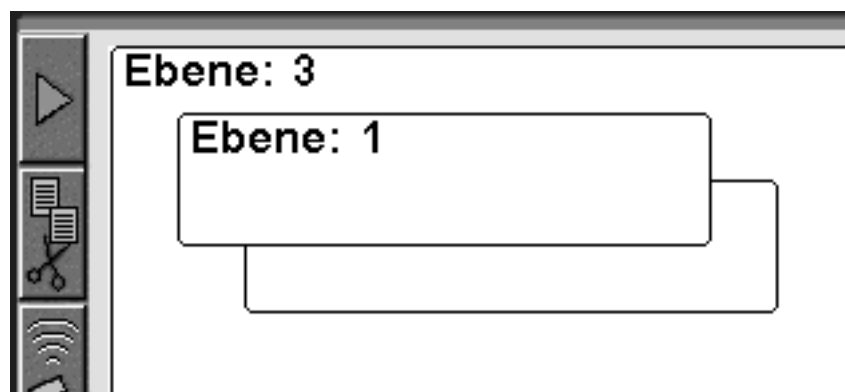


Abb.: Ebenen-Reihenfolge läßt sich frei festlegen

Dabei trägt die oberste Ebene die Nummer 1. Gibt man eine Ebene an, deren Nummer größer ist, als die Anzahl der offenen Fenster, landet das Fenster automatisch in der tiefstmöglichen Ebene - ohne Fehlermeldung. Wenn man also nicht aufpaßt, verschwindet ein kleines Fenster sogar völlig hinter dem großen Standardfenster! Dieses ließe sich dadurch aber ebenfalls als großer Vorhang ausnutzen.

## OPL Workshop 7

Tragen Sie die folgenden Zeilen im obigen Programm "fenster2" anstelle der Zeile "REM 1" ein und verfolgen Sie, was geschieht ...

```
GET : gORDER win2%,100
GET : gORDER win2%,1
```

Ist Ihnen nicht klar, welches Fenster in welcher Ebene liegt, aber Sie benötigen diese Angabe dringend, benutzen Sie "gRANK". Das liefert die Ebene des aktuellen Fensters zurück. Der Wert kann natürlich auch wieder in einer Variablen gespeichert werden. Ersetzen Sie "REM 2" im Programm "fenster2" durch diese Zeilen:

```
GET
gUSE win1%
gCLS : gBORDER 0
gAT 5,15
gPRINT "Ebene:",gRANK
gUSE win2%
gCLS : gBORDER 0
gAT 5,15
gPRINT "Ebene:",gRANK
```

## Wo bin ich?

Neben diesen Informationen erfahren Sie weitere Eigenschaften von Fenstern per "gX" und "gY" (Cursorposition innerhalb des aktiven Fensters in Pixel) sowie "gORIGINX" und "gORIGINY" (Position der linken oberen Ecke des aktiven Fensters im Standardfenster, in Pixel). Probieren Sie das aus, indem Sie im Programm "fenster2" die Zeile "REM 3" durch folgenden Programmtext ersetzen:

```
GET
gCLS : gBORDER 0
gAT 5,15
gPRINT "Fenster ID:",gIDENTITY
x%=gX : y%=gY
gAT 5,30
gPRINT "Cursor: X=",x%,"Y=",y%
gAT 5,4
x%=gORIGINX
y%=gORIGINY
gPRINT "Fenster: X=,,x%,"Y=",y%
```

Weitergehende Informationen stellen "gINFO" (alle Serie 3 Geräte) und "gINFO32" (alle 5er) zur Verfügung, aber für den Anfang kommen Sie mit den genannten Befehlen aus.

## Zwischentöne

Nachdem Sie nun über das Basiswissen verfügen, gehen wir auf ein bisher ausgespartes Thema ein, nämlich die Darstellung von Grautönen.

Das ist insofern diffizil, als Serie3a/c/mx und die 5er Serien das Thema anders behandeln.



Abb.: Grautöne - kein einfaches Thema

### Serie3a/c/mx

Grundsätzlich stehen anfangs selbst im Standardfenster nur schwarz und weiß (oder besser: kein schwarz) zur Verfügung. Den einzigen zur Verfügung stehenden Grauton kann man durch entsprechende Befehle für Schrift und Grafik erst "nach Ansage" nutzen. Für das Standardfenster wendet man dazu den Befehl "DEFAULTWIN 1" an. Bei weiteren Fenstern erweitert man den Befehl zur Erstellung um einen Parameter:

```
gCREATE (xpos%, ypos%, breite%, hoehe%, sichtbar%, grau%)
```

wobei grau% der Wert 1 besitzen muß. Mit welcher "Farbe" nun geschrieben und gezeichnet wird, bestimmt der Parameter des "gGREY modus%" -Befehl. Die Null steht für schwarz, eine 1 für grau und eine 2 zeichnet gleichzeitig schwarz und grau.

Verstärken wir das Verständnis dafür, wie der Psion mit den "Farben" umgeht: Stellen Sie sich vor, daß die "weiße" Ebene unten liegt. Es folgt die graue und oben die schwarze. D.h., die schwarze Ebene deckt die graue zu. In einem auf dem Bildschirm schwarz erscheinenden Gebiet können sie zwar grau zeichnen, das wird aber durch schwarz abgedeckt und ist demzufolge nicht sichtbar. In diesem Falle ist es erforderlich, den schwarzen Bereich wegzulöschen, damit der graue sichtbar wird!

```
PROC fenster3: REM f.Serien 3a/c/mx
  gCREATE (25,25,200,50,1,1)
  gBORDER 0
  GET
  gAT 40,15
  gGREY 0
  gFILL 50,20,0
  GET
  gAT 50,30
  gGREY 1
  gPRINT "GANZ IN GRAU"
  GET
  gAT 50,30
  gGREY 0
  gTMODE 1
  gPRINT "GANZ IN GRAU"
  GET
ENDP
```

"gGREY" benennt also die "Farbebene", in der gezeichnet und geschrieben werden soll. Auf welche Art das geschieht, legt man für Text mit "gTMODE n%" (Einzelheiten siehe vorherige Ausgabe) und für Grafik mit "gGMODE n%" fest. Hat n% der Wert 0, werden auf dem Bildschirm die Pixel in der entsprechenden Farbe gesetzt, bei n%=1 werden sie gelöscht. Ist n%=2, wird invertiert dargestellt. D.h. das Betriebssystem prüft, welche Farbe dem zu beschreibenden Bereich unterliegt und stellt dann Pixel für Pixel den entgegengesetzten Wert dar.

In unserem Beispiel steckt das "gGMODE n%" gleich mit im "FILL"-Befehl als letzter Parameter, so daß diese Angabe hier nicht explizit zu finden ist.

Der einzige Grund, weshalb man sich doch relativ intensiv um dieses Thema kümmern muß, besteht in der Speicherbelegung, die bei Verwendung von Grautönen erheblich anwächst. Unser Tip: Wenn immer möglich vermeiden sie Gestaltung mit Grautönen. Sie sparen Programmzeilen und Speicher! Sollten Sie ein Fenster nicht mehr benötigen, schließen Sie es mit "gCLOSE id%".

### Serie 5/5mxPRO

Das Standardfenster stellt neben schwarz und weiß zwei dazwischenliegende Grautöne dar. Reine schwarz/weiß Darstellung erreicht man mit "DEFAULTWIN 0", der 16-Farb-Modus wird durch "DEFAULTWIN 2" eingestellt.

Besonders im 16-Farb-Modus wird der Farb-/Grauton für Stift und Zeichenbefehle vorzugsweise mit dem neuen Befehl "gCOLOR rot%,gruen%,blau%" eingestellt. Die Einzelwerte der Parameter dürfen von 0 (dunkel) bis 255 (hell) reichen. Sind alle drei Farbwerte gleich groß, entsteht ein Grauton. Wird die Palette auch in der Vierfarb- oder Zweifarbdarstellung verwendet, gleicht das Betriebssystem die Farbtöne an das jeweils darstellbare Spektrum an.

Sie können sich sicherlich vorstellen, daß die Darstellung von mehreren Grautönen in diesem Falle nicht mehr über Ebenen gelöst wird. Die Folge: jedes Zeichnen und Schreiben erfolgt direkt, d.h., wenn nach schwarz mit grau geschrieben wird, ist das Grau sofort und "obenliegend" sichtbar.

Anstelle von "gCOLOR()" wird auch "gGREY x%" noch immer unterstützt und daher auch bequem verwendbar. Für x%=1 wird ein "leichtes Grau" abgebildet. Unser Beispiel verwendet 16 Farb-/Graustufen

```
PROC fenster4: REM Serien 5
    gCREATE (25,25,200,50,1,2)
    gBORDER 0
    GET
    gAT 40,15
    gFILL 50,20,0
    GET
    gAT 50,30
    gGREY 1
    REM oder anst. gGREY 1:
    REM gCOLOR 128,128,128
    gPRINT "GANZ IN GRAU"
    GET
ENDP
```

Das "gGREY 0" vor dem "gFILL 50,20,0" beim 3er Beispiel haben wir hier weggelassen - es wäre auch oben nicht unbedingt erforderlich gewesen, da es der voreingestellte Wert des Betriebssystems ist. Bei 5er entspricht das praktisch einem "gCOLOR 0,0,0".

Wer den Workshop am EPOC32/r5-Emulator nachvollzieht, kann einmal die Farbzusammensetzung in der REM-Zeile verändern und nachsehen, was am Bildschirm passiert, wenn dann das REM entfernt wird...

gMODE und gTMODE behalten für spezielle Anwendungen weiterhin ihre Berechtigung, der voreingestellte Wert "0" bewirkt, daß grundsätzlich alles bereits Vorhandene mit der aktuellen Farbeinstellung überschrieben wird.

Und auch am 5er gilt: Je mehr Farben zum Einsatz kommen, desto höher ist der Speicherverbrauch!

## Fensterinhalte sichern

Sie haben die Möglichkeit, die Inhalte von beliebigen Fenstern zu sichern, so daß Sie in der Lage sind, sie wiederzuverwenden.

Besonders einfach ist der Vorgang, wenn ein komplettes Fenster abgespeichert werden soll:

```
gSAVEBIT "bildname"
```

Das funktioniert aber auch mit Fensterausschnitten.

Voraussetzung ist, daß der Cursor im Fenster an die linke obere Ecke des Ausschnittes positioniert wird. Dem Abspeicherbefehl braucht man lediglich noch Breite und Höhe des zu sichernden Bereiches als Parameter mitzugeben:

```
gAT 20,20 : gSAVEBIT name$,breite%,hoehe%
```

Wie man im Beispiel sieht, läßt sich anstelle des direkten Dateinamens wie üblich auch eine Stringvariable, in der der Dateiname steht, verwenden. Der Serie3a/c/mx legt, wenn man kein Zielverzeichnis angibt, die Daten als \*.PIC-Datei im Ordner /OPD auf dem Laufwerk INTERN ab. Die 5er finden ein Plätzchen im Wurzelverzeichnis, setzen aber keine Datei-Endung an den Dateinamen an.

In beiden Fällen handelt es sich um eine sogenannte Bitmap-Datei. \*.PIC ist ein eigenes Grafik-Format, die 5er Datei ist im MBM-Format abgelegt. Während die PIC-Datei nicht durch irgendwelche an Bord befindlichen Programme des Psion eingesehen werden kann, lassen sich MBMs durch die Verwendung von SKIZZE und dort durch "Hinzuladen EPOC-Grafik" ansehen und auch weiterverarbeiten.

Einmal gespeichert, stehen solche Dateien natürlich auch wieder einem geeigneten OPL-Programm zur Verfügung. Bevor man sie allerdings verwenden kann, müssen die Grafiken erst wieder geladen werden. Einfachste Variante:

```
gLOADBIT (name$)
```

Möchte man nicht das Risiko eines versehentlichen Überschreibens eingehen, setzt man im folgenden die Variable schreib% auf Null. Handelt es sich um eine Bitmapdatei, die mehrere Einzelbilder enthält, gibt man in der Variablen i% auch noch die Bildnummer an (für Bild1 ist i%=0):

```
gLOADBIT (name$,schreib%,i%)
```

Beim 3er hat das die Bewandnis, daß Inhalte von Fenstern mit Graustufe durch "gSAVEBIT" automatisch in mehrere schwarz/weiße Bilder zerlegt werden, die jeweils den Inhalt der schwarzen und der grauen Ebene enthalten. Die Einzelbilder werden aber in eine gemeinsame Datei abgelegt und müssen demzufolge beim Laden wieder auseinander dividiert und der richtigen Ebene zugeordnet werden.

Im 5er MBM-Format dagegen können durch externe Programme mehrere "echte" Bilder zusammengefaßt werden. Jedes wird dann durch seine Nummer aufgerufen. Von OPL aus ist das kombinierte Abspeichern allerdings nicht ohne weiteres zu leisten.

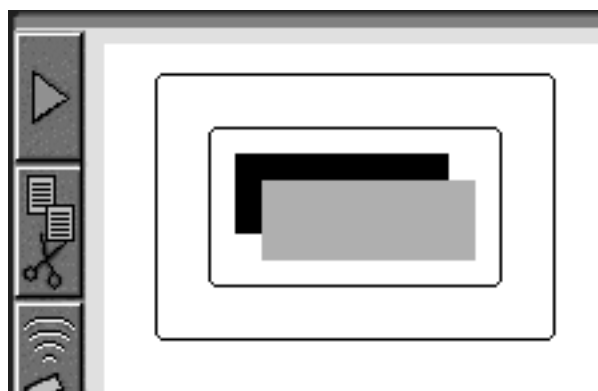


Abb.: Fenster und Farben - speichern und wiederverwenden

Beim Laden einer Bitmap wird ein unsichtbares Fenster geöffnet, das bei der Anzahl der erlaubten offenen Fenster mitzählt. Das kann unter Umständen beim Serie3 Probleme machen, also aufpassen! Schließen Sie grundsätzlich alle nicht benötigten Fenster und Bitmaps! Der Befehl ist, wie bereits genannt, "gCLOSE fenster%". In der Variablen fenster% müssen sie die betreffende Fenster-ID übergeben.

Die nun gerade geladene Bitmap-Grafik muß jetzt noch in ein geöffnetes Fenster oder auch das Standardfenster übertragen werden, sonst kann der Inhalt nicht sichtbar werden. Da durch das Laden

das Bitmap-Fenster zum aktuellen wird, kehrt man zuerst mittels "gUSE" in das Fenster zurück, in dem die Grafik abgelegt werden soll. Schließlich kopiert man die Grafik:

```
gCOPY id%,xpos%,ypos%,breite%,hoehe%,modus%
```

Dem Befehl "gCOPY" gibt man neben der Nummer des Bitmap-Fensters den Bereich mit, aus dem heraus kopiert werden soll, also die Ursprungskoordinaten links oben sowie Breite und Höhe des Bereiches. Soll die gesamte Grafik Verwendung finden, sind das also: 0,0,gWIDTH,gHEIGHT.

Zu guter Letzt wird das ganze noch durch die Angabe des Modus' ergänzt, dessen Werte den Angaben bei "gMODE" entsprechen.

### Beispiel Serie3a/c/mx:

```
PROC fenster5: REM S3x, abspeichern
    GLOBAL win%
    REM Fenster mit grau:
    win%=gCREATE (50,50,110,60,1,1)
    gBORDER 0
    gAT 10,10
    gFILL 80,30,0
    gGREY 1
    gAT 20,20
    gFILL 80,30,0
    gSAVEBIT "testbild"
    GET
ENDP

PROC fenster6: REM S3x, Gr. laden
    GLOBAL win%, bm%, b%, h%
    REM Fenster mit Grau
    win%=gCREATE (50,50,150,100,1,1)
    gBORDER 0
    GET
    kopie:(0,0) REM schwarze Ebene
    GET
    kopie:(1,1) REM graue Ebene
    GET
ENDP

PROC kopie:(grey%,bild%)
    bm%=gLOADBIT("testbild",0,bild%)
    b%=gWIDTH
    h%=gHEIGHT
    gUSE win%
    gAT 20,20
    gGREY grey%
    gCOPY bm%,0,0,b%,h%,0
    gCLOSE bm%
ENDP
```



### Beispiel Serie5/5mx:

```
PROC fenster5a: REM S5x, speichern
    GLOBAL win%
    REM Grafik mit 4 Graustufen
    win%=gCREATE (50,50,110,60,1,1)
    gBORDER 0
    gAT 10,10
    gFILL 80,30,0
    gGREY 1
    gAT 20,20
    gFILL 80,30,0
    gSAVEBIT "testbild_"
    GET
ENDP

PROC fenster6a: REM S5x, Gr.laden
    GLOBAL win%, bm%, b%, h%
    REM Grafik mit 4 Graustufen
    win%=gCREATE (50,50,150,100,1,1)
    gBORDER 0
    bm%=gLOADBIT ("testbild_")
    b%=gWIDTH
    h%=gHEIGHT
    gUSE win%
    gAT 20,20
    GET
    gCOPY bm%,0,0,b%,h%,0
    gCLOSE bm%
    GET
ENDP
```

Nicht verschweigen wollen wir Ihnen, das man Bitmap-Fenster mit OPL auch selber erzeugen kann ("gCREATEBIT breite%,hoehe%"). Jedoch können wir keine Vorteile für die Nutzung entdecken. Im Gegenteil - es sind eine Reihe von Fehlern möglich, wenn man die üblichen Fensterbefehle anwendet. So haben wir darauf verzichtet, ohne daß ein Nachteil daraus erwächst ...

Zusammen mit dem vorhergehenden OPL-Workshop (alle grafischen Textausgaben lassen sich auch und gerade in Fenstern vorteilhaft einsetzen!) verfügen Sie nun über eine gute Grundlage, die bisherigen Demo-Programme oder auch die eigenen mit einer Form zu versehen, die sich sehen lassen kann.

Beim nächsten Mal gehen wir dann auf die direkten grafischen Befehle ein. Bis dahin gilt wie immer: Experimentieren Sie fleißig!