DM 14,90 · SFR 14,90 · ÖS 105 · LFR 326 · BFR 335 · HFL 18

TOS

TOS 1/93

MAGAZIN PLUS SOFTWARE FÜR DEN ATARI ST & TT

FALCON REPORT ● COMDEX USA

Messe COMDEX

- Erste Erfahrungen aus USA

FALCON REPORT

- Die ersten Produkte
- Entwickler packen aus

Test

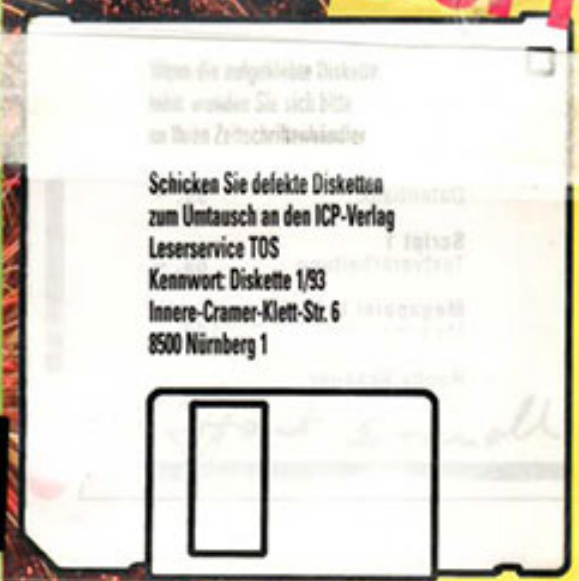
- Mortimer Deluxe
- That's Address
- Interface 2.0
- MO-Laufwerk

BEST OF '92

BESTE HARDWARE

BESTE SOFTWARE

Von der Textverarbeitung zum Laserdrucker – Die megastarken Hits des Jahres '92



7UP

Erstklassiger Editor • Fliegende Dialoge für GFA-Basic und C • Sharewarezeichner MyDraw

AUF DISKETTE

EDITOR 7UP ● MORTIMER DELUXE ● THAT'S ADDRESS ● INTERFACE 2.0 ● MO-LAUFWERK

Grundlagen:
Fliegende Dialoge für C-Programmierer

Fly Deals

Jeder kennt sie und jeder arbeitet mit ihnen in Programmen wie Rufus, Gemini oder SciGraph und Qfax. Gemeint sind die fliegenden Dialoge, wie sie bereits in mehreren Versionen vorliegen. Mit den »Fly Deals« sind runde Radiobuttons, ankreuzbare Buttons und Pop-Up-Menüs kein Problem mehr.

Von Axel Schlüter Viele der Umsetzungen fliegender Dialoge nach Reschke-Art (»Fly Dials«) haben einen Nachteil: Sie verfolgen zwar alle dasselbe Grundkonzept, aber jeder implementiert es grafisch auf eine andere Weise. Damit werden aber alle Standardisierungsversuche bei der Oberflächenentwicklung zunichte gemacht. Denn jedes Programm benutzte bisher seine eigene Oberfläche, so daß der Benutzer sich immer wieder umstellen mußte. Damit dies möglichst nicht passiert, wird hier eine weitere Library veröffentlicht, die sich grafisch gesehen an das Original von J. Reschke und somit an den Gedanken der Einheitlichkeit hält.

Wie funktioniert das?

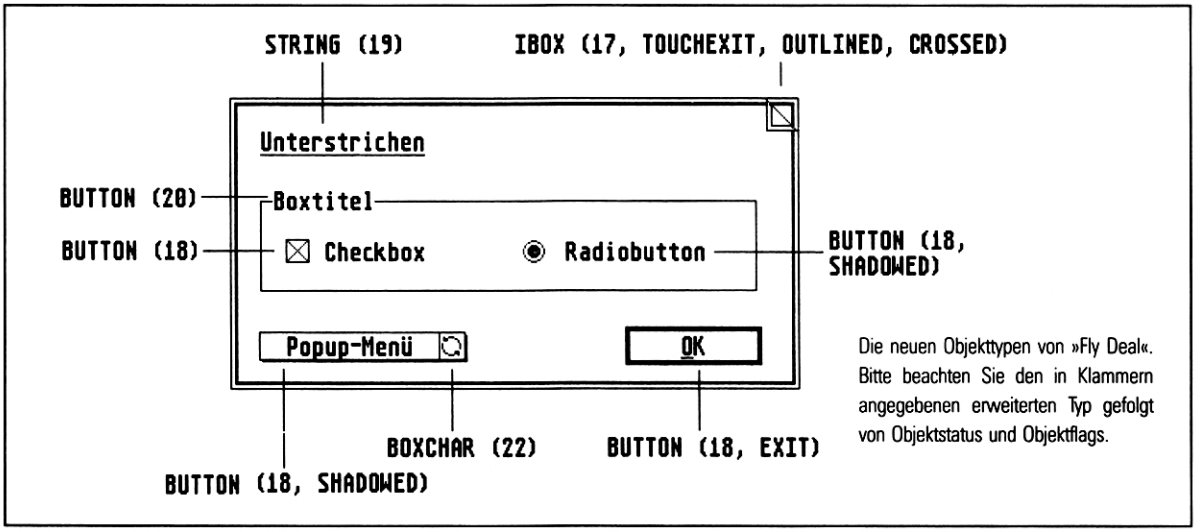
Die meisten Programmierer arbeiten mit C oder dem leider noch nicht allzu weit verbreiten C++, sodaß eine Veröffentlichung in C den meisten Erfolg hat. Wer bereits mit einem Resource-Construction-Set gearbeitet hat, weiß, daß GEM die erweiterten Objekte gar nicht kennt. Digital Research hat aber mit den »user-defined Objects« eine Hintertür freigelassen. Dies ist ein spezieller Objekttyp, bei dem der Programmierer das Aussehen und die Funktion von Objekten selbst bestimmt.

Problem: Erst im laufenden Programm dürfen Sie ein Objekt als erweitert deklarieren. Damit man aber nicht jedes Objekt einzeln aufführen muß, müßten wir schon im Resource-Kit bestimmen, welche Erweiterung unser Objekt tragen soll. Die GEM-Entwickler haben glücklicherweise dem Programmierer ein Byte pro Objekt Platz gelassen, eigene Informationen unterzubringen. Es ist das Highbyte im Objekttyp (GEM benutzt bislang nur das Lowbyte). Diesem Byte weisen wir den Wert des gewünschten Objekttyps im Resource-Kit zu. Das fertige Programm wertet zur Laufzeit diesen Wert aus und installiert den erweiterten Objekttyp.

Entgegen aller Behauptungen geht das auch im Original GEM Resource Construction Kit, wenn auch etwas umständlich:

1. Objekt selektieren 2. Mit <ALT-T> die »Typinfobox« aufrufen 3. Textcursor hinter die Null positionieren und das gewünschte Zusatzbyte eingeben. Die Zahl erscheint jetzt am Anfang der Zeile. 4. An den Schluß der Zeile gehen und das letzte Zeichen löschen. 5. Schritt 4 wiederholen 6. Dialog mit »OK« verlassen und erneut aufrufen 7. Aktuellen Wert bis auf die letzte Stelle löschen 8. Zahl neu eingeben, die letze Stelle löschen und die Box mit OK verlassen. Geschafft!

Die neuen Objekttypen sehen Sie in Bild 1. Zusätzlich



Die neuen Objekttypen von »Fly Deal«. Bitte beachten Sie den in Klammern angegebenen erweiterten Typ gefolgt von Objektstatus und Objektflags.

finden Sie dort die Art des Grundobjekts sowie den erweiterten Objekttyp und den zu setzenden Objektstatus. Bei der Verwendung von Buttons und Pop-Up-Menüs markieren Sie im Text den Buchstaben, den Sie über Tastatur erreichen wollen, mit einer vorangestellten, eckigen Klammer (»[«). Den auf die Klammer folgenden Buchstaben erreichen Sie im Programm über die Tastenkombination <ALTERNATE + Buchstabe>. Die Library bietet außerdem Routinen zur einfachen Initalisierung des AES und VDI sowie einen neuen »form_do()«-Handler und eine neue Dialogverwaltung. Auf Grow-/Shrinkboxen wurde völlig verzichtet, da die original FlyDials sie ebenfalls nicht benutzen, GEM 2.x sie in der bisherigen Form nicht mehr unterstützt und sie eigentlich nur Zeit verbrauchen. Wer sie trotzdem haben möchte, sollte sie vor »FLY_deal_start()« und nach »FLY_deal_stop()« aufrufen.

Die erweiterten Objekttypen

Jeder neue Objektyp basiert auf einem bereits von GEM unterstützten Objekt. Dieses wird ganz normal angelegt und dann mit der Zusatznummer versehen. Für Buttons, Radiobuttons und Exitbuttons gelten weiterhin die bekannten Spielregeln. Das neue Aussehen für runde bzw. ankreuzbare Buttons bestimmt die Zahl 18 als Nummer für den erweiterten Objekttyp.
Der Movebutton (das Eselsohr in der rechten oberen Ecke) ist eine gewöhnliche »IBOX« in der Größe eines »BOXCHARS«. Die zu setzenden Flags sind »TOUCHEXIT«, »OUTLINED« und »CROSSED«, das Zusatzbyte lautet hier 17.
Unterstrichene Texte sind in der Regel für die Überschrift einer Dialogbox reserviert. Das Grundobjekt ist vom Typ »String«, der erweiterte Objekttyp trägt die Nummer 19.
Abtrennboxen sind Buttons ohne Flags, inbesondere nicht »SELECTABLE«! Der Text des Buttons erscheint als Überschrift oben links in der Box. Der erweiterte Objekttyp hört auf die Zahl 19.

Pop-Up-Menüs

Wir kommen zum schwersten Teil: Pop-Up-Menüs. Die Grundlage bildet eine Ibox mit dem Flag »SHADOWED«. Zur Anzeige der aktuellen Einstellung des Menüs dient ein normaler Button mit Touchexit und dem erweiterten Objekttyp 22. Verwenden Sie zudem eine »Circlebox« (Boxchar, Typ 22), plazieren Sie diese rechts vom Button. Außerdem benötigen Sie einen weiteren Dialog, der das eigentiche Pop-Up-Menü enthält. Er sollte »SHADOWED« sein, die einzelnen Einträge sind »Strings«.
Soviel zum RCS. Die restlichen Vorkehrungen treffen Sie in Ihrem Programm. Es gibt eine Struktur namens »POPUP«:

```
typedef struct
{
  int object;
  int tastSel;
  OBJECT *tree;
}POPUP;
```

Ihre Deklaration finden Sie in »FLY_DEAL.H« (auf der nächsten TOS-Diskette). Sie müssen für jeden Dialog mit Pop-Ups ein Array dieser Struktur anlegen, wobei die Anzahl der Elemente des Arrays gleich der Anzahl der Elemente des Pop-Up-Menüs ist. Dann füllen Sie für jeden Dialog ein Element. In »object« kommt die Nummer des Anzeigebuttons, in »tree« der Pointer auf den anzuzeigenden Dialog und in »tastSel« die Nummer des Tastaturstrings. Dieses Array übergeben Sie vor(!) »FLY_deal_start()« mit der Funktion »FLY_radio_set()«. Als Parameter erwartet die Funktion das besprochene Array, sowie die Anzahl der Einträge.
Nach der Abarbeitung des Dialogs mit anschließendem(!) »FLY_deal_stop()« erfragen Sie mit »FLY_test_radio_set()« das gewählte Element. Der erste Parameter ist wieder das Array, der zweite die Position des abzufragenden Anzeigebuttons in dem Array (es könnten sich ja mehrere Pop-Ups in einer Dialogbox befinden).
Um die Trennstriche der Menüs zu ändern, ist weder im Resource-Contruction-Set noch im Programm selbst eine Umarbeitung erforderlich.

Was muß ich im Programm beachten ?

Der erste Aufruf ist immer »FLY_init()«. Hier wird sowohl das GEM als auch die Library initalisiert. Dann kann ganz normal die Resourcedatei geladen sowie die Adressen bestimmt werden. Zum Vorbereiten und Zeichnen des Dialogs verwenden Sie die Funktion »FLY_deal_start()«, die als Ergebnis einen Pointer auf den Hintergrundpuffer liefert. Diese Funktion nimmt die Stelle von »form_center()«, »form_dial()« und »objc_draw()« ein. Die Arbeit innerhalb der Box übernimmt »FLY_deal_do()«. »FLY_deal_do()« ist äquivalent zu »form_do()«. »FLY_deal_stop()« löscht den Dialog und restauriert den Bildschirm.
Die Library ist voll GEM-kompatibel programmiert und sollte daher in jeglicher Auflösung und mit jeder Rechnerkonfiguration arbeiten. Sollten Sie trotzdem noch Fehler finden oder Verbesserungen haben, bitte wenden Sie sich an:
Axel Schlüter, Mertenweg 16, 5014 Kerpen-Brüggen
Auf der TOS-Diskette zur nächsten Ausgabe finden Sie die Library für Pure-C mit vielen Beispielen. **(ah)**