

Was ARExx bisher noch fehlte, war eine grafische Benutzeroberfläche für Programme. Eine Public-Domain-Funktionsbibliothek führt jedoch zum Ziel – wie das hier entwickelte »LhAControl« beweist.

von Karsten Wysocki

Was nützt das schönste Programm, wenn die Bedienung die meisten davon abhält, es zu benutzen. Um auch ARExx-Programmen zu einer wirklich schicken Oberfläche zu verhelfen, gibt es die »apig.library« von Ronnie E. Kelly. Die Besonderheit dieser Library ist, daß sie nicht wie andere Funktionsbibliotheken fest programmierte Funktionen enthält, sondern lediglich als Durchreicher arbeitet.

Tatsächlich werden mit dem Aufruf einer Funktion der »apig.library« die passenden Betriebssystem-Routinen benutzt (s. Kasten

und die Benutzung ist in keiner Weise eingeschränkt.

Wenn Sie die ersten beiden Workshop-Teile durchgearbeitet und bei der ARExxManager-Installation das Programm »CopyLibs« gestartet haben, befindet sich die apig.library schon im Verzeichnis »Libs:«. Andernfalls starten Sie bitte das Programm »CopyLibs«.

### Kleider machen Leute

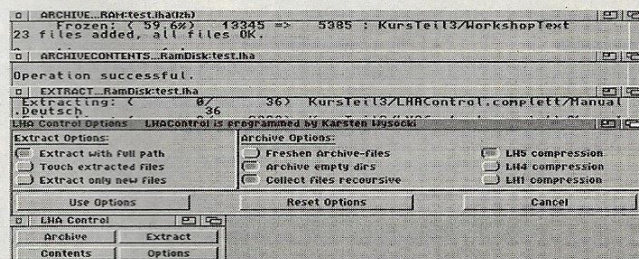
Mit der apig.library lassen sich Bildschirme und Fenster öffnen, alle Gadgets nutzen sowie System- und ASL-Requester auf den Bildschirm bringen – alles per ARExx.

Die apig.library startet für jeden geöffneten Bildschirm und jedes geöffnete Fenster einen eigenen Task mit jeweils einem ARExx-Port. Fenster-Tasks bekommen als ARExx-Port-Namen »apig.task.n«, Bildschirm-Tasks »apig.

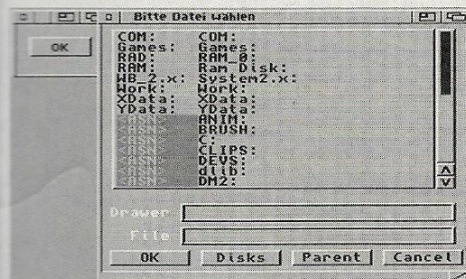
dann eine Gadget-Struktur initialisiert und ein Bildschirm oder Fenster geöffnet. Danach verzweigt das Programm in eine Programmschleife, in der auf Nachrichten von Fenster- und Bildschirm-Tasks gewartet und reagiert wird.

Jedes Gadget erhält bei der Initialisierung eine eindeutige Nummer. Wird vom Benutzer des Pro-

gramms ein Gadget gedrückt, sendet der Fenster-Task, in dessen Fenster sich das Gadget befindet, eine Nachricht an das dazugehörige ARExx-Programm. Die Nachricht enthält einen Verweis auf »GADGETUP« (Maustaste losgelassen) bzw. »GADGETDOWN« (Maustaste gedrückt) und zusätzlich die vorher festge-



Bei der Arbeit: Mit LhAControl geht das Arbeiten mit LhA-Archiven auch ohne Fachkenntnisse leicht von der Hand



APIGWindow: So sieht das Fenster mit Dateirequester aus Listing 1 und 2 aus. Mit APIG schnell gemacht.

»APIG-unterstützte System-Libraries«). Dadurch passen sich mit der apig.library gemachte Programmoberflächen automatisch dem Aussehen des jeweiligen Betriebssystems an.

Ein weiterer Vorteil der apig.library ist, daß sie ein Produkt aus der Public-Domain ist. Der Autor behält sich keinerlei Rechte vor

screen.n.«, wobei in beiden Fällen für das »n« eine Nummer eingesetzt wird.

Ein ARExx-Programm, dessen Oberfläche mit der apig.library erstellt wird, funktioniert im Prinzip genauso, wie das Programm »TestMsg« im ersten Kursteil. Am Programmanfang wird ein ARExx-Port für das Programm angelegt,

### Images (Bilder)

Images sind vordefinierte, grafische Elemente ohne Funktion. In Verbindung mit Bool-Gadgets (das sind einfache Schaltflächen, die lediglich Mausklicks registrieren) kann man sich damit eigene Gadget-Funktionen konstruieren. In der Tabelle sind verschiedene vorhandene Images aufgeführt und kurz beschrieben.

|            |                                                      |
|------------|------------------------------------------------------|
| DEPTHIMAGE | wie Gadget zum Fenster nach hinten oder vorn stellen |
| ZOOMIMAGE  | wie Gadget zum Fenster größer oder kleiner schalten  |
| SIZEIMAGE  | wie Gadget zur Größenänderung des Fensters           |
| CLOSEIMAGE | wie Gadget zum Schließen des Fensters                |
| LEFTIMAGE  | wie Gadget mit Pfeil nach links                      |
| UPIMAGE    | wie Gadget mit Pfeil nach oben                       |
| RIGHTIMAGE | wie Gadget mit Pfeil nach rechts                     |
| DOWNIMAGE  | wie Gadget mit Pfeil nach unten                      |
| CHECKIMAGE | wie Gadget mit/ohne Haken                            |
| MOXIMAGE   | kleiner runder Knopf wie bei Radiobuttons            |

```

/* APIGWindow */
/* Vorbereitungen für apig-Oberfläche ----- */

call addlib("rexsupport.library",0,-30,0) /* zus. Libraries */
call addlib("apig.library",0,-30,0) /* einbinden */
call SET_APIG_GLOBALS( /* apig-Funktionen initialisieren */

portname = "apigWindow" /* Portnamen festlegen */
portlist = Showlist('p') /* Portliste einlesen */
if find(portlist,portname) > 0 /* Wenn portname in portlist */
then exit /* dann Prg beenden */
p = openport(portname) /* ARExx-Port öffnen */

scr = LOCKPUBSCREEN("Workbench") /* Screenpointer */
scrinfo = GETVISUALINFO(scr) /* Alle nötigen */
scrfont = GETVALUE(scr,40,4,'p') /* Pointer für */
glistpointer = ALLOCVEC(4,MEMP_CLEAR) /* Gadget-Struktur */
conxgad = CREATECONTEXT(glistpointer) /* einlesen */
previousgadget = conxgad

/* Gadget-Struktur erstellen ----- */
okbutton = 10 /* ID für Button festlegen */
gadname = "OK" /* Name, */
gadid = okbutton /* Identifier, */
top = 10 /* Position der */
lft = 10 /* linken oberen Ecke, */
wdt = 50 /* Länge und */
hgt = 15 /* Höhe des Gadgets festlegen */

/* Gadget-Struktur initialisieren ----- */
newgad = MAKENEWGADGET(scrinfo,scrfont,lft,top,
,wdt,hgt,gadname,PLACETEXT_IN,gadid,0)
previousgadget = CREATEGADGET(BUTTON_KIND,previousgadget,
,newgad,TAG_DONE,0)

/* Fenster erstellen und öffnen ----- */
winlft = 10; wintop = 10 /* Position links,oben */
winhgt = 52; winwdt = 78 /* Höhe, Breite */
winpen1 = 2; winpen2 = 4 /* Farben festlegen */
wintitle = "My first apig-Window" /* Titel des Fensters. */

/* Attribute, auf die per Message reagiert werden sollen ---- */
winidcmp = CLOSEWINDOW+GADGETUP

```

legte Gadget-Nummer. So kann das Programm auf aktivierte Gadgets passend reagieren.

Am besten tippen Sie das Listing 1 (»APIGWindow«) einfach ab und starten es. Für den Fall, daß beim ersten Testlauf etwas schiefgeht, noch ein Tip: Wenn Ihr Programm eine Fehlermeldung verursacht und von Rexx-Mast abgebrochen wird, kann es passieren, daß ein schon geöffnetes Fenster nicht automatisch geschlossen wird.

Um das zu erreichen, müssen Sie dem dazugehörigen APIG-Task (in der Regel dem letzten mit der höchsten Nummer) von der ARExxShell aus die Message »DIE« senden:

```
address "apig.task.3" "DIE"
```

Auch dem ARExx-Programm kann man durch Senden dieser Message den Hahn zudrehen, denn die aus dem ersten Teil von »MsgTest« bekannte Abfrage nach einer Message mit dem Inhalt »DIE« ist auch hier als Notbremse eingebaut.

Schwerwiegende Fehler können bei der Software-Entwicklung

mit der apig.library auch zu Systemabstürzen führen: Sie greifen mit der apig.library indirekt auf System-Funktionen zu. Falls bei Ihnen solch ein schwerer Fehler auftritt, booten Sie Ihren Rechner neu, versuchen Sie, den Fehler zu korrigieren und probieren Sie es nochmal.

Wenn Sie das Programm »APIGWindow« zum Laufen gebracht haben, erscheint nach dem Start auf der Workbench ein kleines Fenster mit einem Gadget und der Aufschrift »OK«. Wenn Sie darauf klicken, verschwindet das Fenster wieder und das Programm beendet sich.

Die Basis ist damit gelegt, jetzt gilt es, das Programm ein wenig aufzubohren, um alle Funktionen kennenzulernen, die für das abschließende Projekt »LhAControl« gebraucht werden.

Als nächstes soll »APIGWindow« einen ASL-Dateirequester öffnen. Hängen Sie dazu im ersten Schritt das Unterprogramm »asl\_file\_request« ans Ende des bestehenden Programms an. Danach ändern Sie in der Hauptprogramm-Schleife die Zeilen

```
/* Fenstereigenschaften festlegen und Fenster öffnen -----*/
winflags = WINDOWCLOSE+WINDOWDRAG+WINDOWDEPTH+GIMMEZEROZERO
win1 = OPENWINDOW(portname,win1ft,wintop,winwdt,winhgt,winpen1,
,winpen2,windcmp,winflags,wintitle,scr,0,0,0,conxgad)

/* Fensterinhalt-update, da mit erstellte Gadgets erscheinen. */
call GT_REFRESHWINDOW(win1,null())

/* Hauptprogramm-Steuerschleife ----- */
/* funktioniert fast wie TestMsg aus Folge 1 */

exitme = 0
do forever
  x = waitpkt(portname)
  do forever
    msg = getpkt(portname)
    if msg = "0000 0000"x then leave
    class = getarg(msg,0)
    if class = DIE then exitme = 1
    if class = CLOSEWINDOW then exitme = 1
    if class = GADGETUP then do
      gadid = getarg(msg,9)
      select
        when gadid = okbutton then exitme = 1
        otherwise nop
      end
    end
  end
  x = reply(msg,0)
end
if exitme = 1 then leave
end

/* Fenster schließen und Ressourcen freigeben ----- */
call CLOSEWINDOW(win1) /* Fenster schließen und */
call UNLOCKPUBSCREEN(null(),scr) /* sämtliche */
call FREEVISUALINFO(scrinfo) /* Zeiger */
call FREEGADGETS(conxgad) /* und */
call FREETHIS(newgad) /* Ressourcen */
call FREEVEC(glistpointer) /* wieder freigeben */

exit /* Programmende */
```

**Listing 1: Mehr ist nicht nötig, um mit ARExx und der »apig.library« ein kleines Fenster mit OK-Button zu öffnen**

## HotHelp-ARExx-Spezialversion



Am schnellsten lernt man, wenn man bei Fragen nachschlagen läßt, statt selbst zu blättern. Maxon hat speziell für unseren ARExx-Kurs eine Demo-Version von HotHelp 3 – einem interaktiven Hilfesystem ähnlich AmigaGuide – zusammengestellt, die zusätzlich zur normalen Demo eine vollständige ARExx-Beschreibung enthält. Diese ist normalerweise nur im OS-3.0-Paket enthalten.

HotHelp 3 läßt sich per ARExx in jeden Editor einbinden, unterstützt Grafiken im Text, liest und verwaltet auch AmigaGuide-Dokumente und sucht auch Projekt-übergreifend nach Schlüsselwörtern – sogar mit Platzhaltern. Von HotHelp 3 gibt es zwei verschiedene Pakete:

Paket 1: Die Entwicklerversion mit deutschsprachiger Dokumentation zu ANSI-C und Amiga-OS 2.1,

Online-Projekt-Entwicklung und Projekt-Editor. Preis 69 Mark.

Paket 2: Deutschsprachige OS-3.0-Dokumentation mit Beschreibung aller Devices und Libraries, IFF- und ARExx-Dokumentation sowie Infos zu allen Shell-Befehlen. Preis 89 Mark.

Weitere Infos: MAXON Computer GmbH, Industriestr. 26, 65760 Eschborn, Tel. (0 61 96) 48 18 11, Fax (0 61 96) 4 18 85

```
select
  when gadid = okbutton
then exitme = 1
  otherwise nop
end
```

folgendermaßen ab (und fügen Sie die then-Zeile mit der folgenden zu einer zusammen):

```
select
  when gadid = okbutton
  then call asl_file_request
    ("Bitte Datei wählen",,)
  otherwise nop
end
```

Dabei fällt auf, daß beim Unterprogrammaufruf nur ein Parameter und dann nur noch zwei Kommas stehen. Keine Angst, mit ARExx funktioniert das einwandfrei, da der Interpreter die Funktion dann genauso aufruft, als hätte man leere Strings mit »""« eingesetzt.

### ... Kontrolle ist besser

Damit sind die Grundlagen für LhAControl gelegt. Es ist ein Aufsatz für das Archivierungsprogramm »LhA«, um alle Aktionen wie »Archivieren«, »Entpacken« und »Auflisten« bequem mit der Maus zu erledigen. Zu diesem Programm gehören »LhAControl« (aus Platzgründen nur auf PD-Diskette) und »LhAOptions« (Auszüge in Listing 3).

LhAControl öffnet ein Fenster mit vier Schaltern (Gadgets) und steuert LhA, indem es den Archivierer per »address command« mit den passenden Parametern startet. Die Ausgabe von LhA wird auf eine Konsole umgeleitet. Die LhA-Aufrufe befinden sich im Programm LhAControl in den Unterprogrammen am Ende der Datei. Da LhA als Hintergrundprozess läuft, lassen sich mehrere Archive gleichzeitig bearbeiten.

Wenn Sie sich »LhAControl« näher ansehen, werden Sie feststellen, daß sich im Programm nicht viel Neues befindet. Fast alle dort benutzten Funktionen haben Sie schon einmal ausprobiert. Die einzige Ausnahme ist die Kommunikation mit LhAOptions. LhAOptions wird von LhAControl mit »Run« gestartet, läuft also ebenfalls als eigener Task und unabhängig von LhAControl.

In LhAOptions lassen sich die gängigen LhA-Optionen einstellen. Vor dem Aufruf von LhAOptions werden von LhAControl die aktuellen LhA-Parameter mit der Funktion »setclip()« als ARExx-Clip gespeichert (Das ist ein ARExx-interner Zwischenspeicher). Wenn LhAOptions dann gestartet ist, liest es diesen Clip und präsentiert auf der grafischen

## Kursübersicht

**ARExx in Perfektion, für Leute, die sich bisher nie aufraffen konnten, diese einfache aber mächtige Sprache zu lernen, die zu jedem Amiga gehört: Das ist das Ziel dieses Kurses.**

**Folge 1:** Einführung in ARExx, Variablen-Handling, Schleifen, Abfragen, Blöcke, Nachrichten; »TestMsg« verschickt Nachrichten.

**Folge 2:** Funktionsaufrufe, Dateizugriffe und die Konsole, Unterprogramme, Steuern anderer Programme, Anbindung von GoldED mit Skripten.

**Folge 3:** Die »apig.library«: grafische Oberflächen mit Schaltern, Fenstern und Requestern; eine Oberfläche für »LhA«.

**Folge 4:** Tabellenverarbeitung, ein Listview-Gadget, Fenster-Refresh, TAC- (TransAktionsCode-) Programmieretechnik, weitere ARExx-Funktions-Bibliotheken.

Oberfläche die aktuellen Einstellungen. Eventuelle Änderungen dieser Einstellungen werden LhA-Control durch das Senden einer Message mitgeteilt. Zum Schluß beendet sich LhAOptions. Auf diese Weise kann man die LhA-Optionen während eines LhA-Laufs ändern. Die neuen Einstellungen werden dann beim nächsten LhA-Aufruf berücksichtigt.

Das Listing LhAOptions enthält dagegen einige neue Programmteile: Zunächst wird am Programmanfang per »getclip()« der Options-String eingelesen. Dieser String hat ein bestimmtes Format: Er besteht aus Schaltern und Kommandos, die LhA zum Extrahieren und Archivieren benötigt. In dem String sind Schalter- und Kommandoarten durch das Zeichen »|« getrennt.

Nach dem Einlesen wird dieser String aufgeteilt und die Teile in Variablen übertragen, die genauso wie die Schalter- und Kommandoarten heißen. Danach kommt wieder das schon bekannte Einrichten einer Gadget-Struktur. Neu hinzugekommen ist der Gadget-Typ »TEXT«, der nur der Beschriftung dient und daher nicht in der Hauptprogrammschleife auszuwerten ist. Danach öffnet

```
/* Unterprogramm asl_file_request ----- */
/* Aufruf: datei = asl_file_request(titel,dir,suchmuster) */
/* Zurückgegeben wird der Name der Datei inklusive Pfad oder */
/* null(), falls keine Datei oder Cancel gewählt wurde. */

asl_file_request: PROCEDURE /* Deklaration d. Routine */
    parse arg title,dir,filepattern /* Parameterübersame */

    top = 0; lft = 0 /* Fensterausmaße festlegen */
    wdt = 200; hgt = 212
    freq = ALLOCFILEREQUEST() /* Struktur allozieren */
    /* Requester auf den Schirm ! */
    file = REQUESTFILE(freq,lft,top,title,dir,filepattern,
        ,0,0,0,-10,-2,wdt,hgt,0)
    call FREEFILEREQUEST(freq) /* allozierten Speicher freigeben */

    echo file /* Zur Kontrolle ausgeben (Test) */
    return file /* Dateinamen zurückliefern */
```

**Listing 2:** Mit diesem Unterprogramm für Listing 1 erscheint nach Drücken des OK-Buttons ein Datei-Requester

das Programm wieder ein Fenster und setzt – zur besseren Optik und Bedienung – Images (grafische Bildsymbole) und Bool-Gadgets (einfache Schaltflächen) ein.

Das neue Window läßt sich in drei Bereiche gliedern: Im linken kann man gängige Entpack-Optionen einstellen, im rechten die Archivierungs-Optionen festlegen und darunter die Einstellungen übernehmen, verwerfen oder zurücksetzen. Damit dieses Fenster schön klein bleibt, werden keine »Checkmarks« (die mit

dem »Häkchen«), sondern MX-Gadgets (Multiple Exchoice, Radiobuttons) eingesetzt. Da normalerweise aber immer mindestens zwei davon erzeugt werden müssen und in einer MX-Schalterleiste immer nur ein Schalter aktiviert sein kann, lassen sich MX-Gadgets bei LhAOptions nicht einsetzen.

Da trotz des begrenzten Platzes nicht auf Schalter mit diesem Aussehen verzichtet werden soll, wird jeder Schalter auf folgende Weise zusammengebaut:

Diese sind so definiert, daß sie unsichtbar unter den Bildern der MX-Gadgets liegen. Wir erzeugen also ein unsichtbares Gadget und legen darüber ein Bild, damit man etwas erkennt. Klickt der Benutzer auf das Bild, »denkt« das Betriebssystem, man habe das Gadget darunter gemeint.

Zusätzlich werden noch Variablen definiert, die den Programm-Code enthalten, um die beiden Zustände (gedrückt und nicht gedrückt) der Bilder anzuzeigen – dies geht ja nicht automatisch, man muß es selbst erledigen.

Sind die Vorbereitungen soweit abgeschlossen und alle Bool-Gadgets definiert, muß die Liste der Schalter mit einer OS-Funktion ans Fenster angehängt und damit dem Amiga-OS bekannt gemacht werden. Dies übernimmt das APiG-Pendant »addglist()«. Bool-Gadgets kann man erst nach Öffnen eines Fensters definieren, weil man dazu den Zeiger auf das Fenster benötigt – und den gibt es nach dem passenden Aufruf zurück.

Zum Schluß werden vor der Hauptprogrammschleife noch Vergleichs-Variablen definiert sowie die Zustände aus dem AREXX-Clip gelesen und in den Variablen ge-

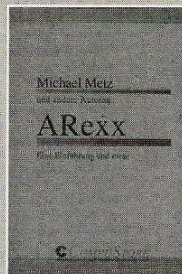
## Fenster-Attribute

Die Funktion »openwindow()« der »apig.library« erlaubt die Angabe zahlreicher »Compares« und »Flags«. Mit den Compares sagt man dem Fenster-Task, welche Ereignisse (wie Mausklicks, Tastendrücke) in Form von AREXX-Botschaften an das dazugehörige AREXX-Programm zu verschicken sind. Im Programm kann man dann z.B. auf eine Veränderung der Fenstergröße reagieren. Mit den Flags lassen sich die Eigenschaften und das Aussehen des Fensters bestimmen. Nachstehend die gebräuchlichsten Werte:

|                                                                     |                                                  |
|---------------------------------------------------------------------|--------------------------------------------------|
| Compares (in den Listings in der Variablen »winidcmp« gespeichert): |                                                  |
| SIZEVERIFY                                                          | Größenveränderungen des Fensters bemerken        |
| REFRESHWINDOW                                                       | Refresh-Ereignisse verschicken                   |
| MOUSEMOVE                                                           | Mausbewegungen mitteilen                         |
| GADGETDOWN                                                          | Gadget-Aktivierung melden                        |
| GADGETUP                                                            | Gadget-Deaktivierung verschicken                 |
| MENUPICK                                                            | Menüauswahlereignisse weiterleiten               |
| CLOSEWINDOW                                                         | Schließen des Fensters übermitteln               |
| RAWKEY                                                              | Tastaturereignisse (Tastennummern) melden        |
| MENUVERIFY                                                          | Menübestätigung anfordern                        |
| DISKINSERTED                                                        | Diskette wurde eingelegt                         |
| DISKREMOVED                                                         | Diskette wurde entfernt                          |
| ACTIVEWINDOW                                                        | Fenster wurde aktiviert                          |
| INACTIVEWINDOW                                                      | Fenster wurde deaktiviert                        |
| VANILLAKEY                                                          | Tastaturereignisse (ASCII-Werte) melden          |
| INTUITICKS                                                          | (Intuitionticks) wird für Schieberegler benötigt |

|                                                                   |                                                               |
|-------------------------------------------------------------------|---------------------------------------------------------------|
| Flags (in den Listings in der Variablen »winiflags« gespeichert): |                                                               |
| WINDOWIZING                                                       | Für Gadget zur Größenänderung des Fensters                    |
| WINDOWDRAG                                                        | Fenster läßt sich verschieben                                 |
| WINDOWDEPTH                                                       | Fenster erhält Tiefen-Gadget rechts oben                      |
| WINDOWCLOSE                                                       | Fenster erhält Schließ-Gadget links oben                      |
| SIZEBRIGHT                                                        | Breiter Rahmen am rechten Rand                                |
| SIZEBBOTTOM                                                       | Breiter Rahmen am unteren Rand                                |
| SMART_REFRESH                                                     | Intelligenter Fensterneuaufbau                                |
| SIMPLE_REFRESH                                                    | Einfacher Fensterneuaufbau                                    |
| BACKDROP                                                          | Fenster liegt immer hinter allen anderen                      |
| GIMMEZEROZERO                                                     | (0,0) liegt innerhalb des Fensterrahmens                      |
| BORDERLESS                                                        | Fenster ohne Rahmen zeichnen                                  |
| ACTIVATE                                                          | Fenster ist beim Erscheinen aktiviert                         |
| RMBTRAP                                                           | Fängt rechten Mausknopf ab                                    |
| SUPER_BITMAP                                                      | Fenster mit eigener Bitmap (kann größer sein als das Fenster) |

## AREXX-Buch zu gewinnen



Die Firma CompuStore spendet freundlicherweise den zehn pfiffigsten AREXX-Programmierern das Buch »AREXX – Eine Einführung und mehr ...« für das trickreichste AREXX-Skript, das Sie in möglichst wenige Zeilen fassen sollen.

Je kürzer das Skript und je pfiffiger die Konstruktion, um so eher haben Sie die Chance, ein Buch zu gewinnen.

Einsendungen sind direkt an CompuStore Handelsgesellschaft mbH, Fritz-Reuter-Str. 6, 60320 Frankfurt, Stichwort »AREXX-Wettbewerb« zu schicken. Einsendeschluß ist der 30. September 1995. Der Rechtsweg ist ausgeschlossen. Absender nicht vergessen!

Zuerst muß der Zeiger auf den »RastPort« des Fensters ermittelt werden. Dieser ist immer anzugeben, wenn man Funktionen zum Zeichnen von Punkten, Linien und Bildern (wie das des Radiobuttons) benutzt. Dies erledigt die APiG-Funktion »getwindowrastport()«.

Danach wird eine Speicherstruktur für das Image (Bild) besorgt und initialisiert. Das Amiga-OS verfügt schon über eine gewisse Anzahl von Bildern, die für den Einsatz in eigenen Programmen lediglich in eine Speicherstruktur kopiert werden müssen.

Als nächstes wird in LhAOptions ein Interpret-Unterprogramm in eine Variable gefüllt, mit dem man Bool-Gadgets erstellen kann.

speichert. Anschließend lassen sich die MX-Schalter mit der richtigen Stellung ins Fenster zeichnen. Während des Programmlaufs wird dann beim Aktivieren eines Bool-Gadgets der jeweils andere Zustand des darüberliegenden Bildes ins Fenster kopiert.

Als Vorbereitung für den letzten Teil des AREXX-Workshops sollten Sie ein wenig im von CompuStore zu Ihrem Amiga mitgelieferten AREXX-Handbuch lesen, um sich fit zu halten. dg

### Literatur:

- [1] Amiga-OS 3.1 AREXX, Commodore Electronics Limited. Erhältlich bei Village Tronic Marketing GmbH, Sarstedt
- [2] Michael Metz: AREXX, Eine Einführung und mehr ..., CompuStore Handelsgesellschaft, ISBN 3-93073300-5