Madhouse

Carsten Jahn

COLLABORATORS			
	TITLE : Madhouse		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Carsten Jahn	February 7, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Madhouse

1.1 Madhouse: Inhaltsseite

```
            Hallo Freunde der modularen Screenblanker!
Vor Euch seht Ihr die Anleitung zu Madhouse - dem modularen Bildschirm-
"Schoner" den Ihr mögen werdet.
Das beste an Madhouse sind sicherlich die Blanker, denn hier haben wir
uns wirklich Mühe gegeben um interessante Effekte zu erstellen.
Doch auch das Hauptprogramm "Madhouse" welches die Module konfiguriert
und aufruft, hat einiges zu bieten. Aber seht selbst.
Um schnell die Blanker zu sehen bietet sich der Workshop an, der Euch bei
den ersten Schritten begleitet.
```


1.2 Ein Vorwort

Als er mit feuchten Fingern den klobigen Schalter betätigte, schüttelte es die Zahnräder der Festplatte locker hin und her. Der Aluminiumscheibenstapel saugte sich an seine Vertikalachse, vier Schwenkärme zwengten sich jäh zwischen sie und entsandten Magnetfelder. Die Rotorblätter neben den seriellen Stiftreihen zirkulierten um den Lüftermotor. Ein tiefes Röhren durchströmte den Raum. Die gelbe Leuchtdiode begann aufgeregt zu blinken, mit einem Zittern würgte der Monitor das erste Bild hervor. Ein pinkfarbener, spitzer Pfeil nach Nord-West begann eifrig und hungrig Betonklötze mit farbigen Bildern zu attackieren. Ein Beben ging über die Tasten, die sich voller Furcht in der Mitte der Tastatur zusammenkauerten. Dreihunderttausendmal wurde eine von ihnen von einem harten Fingerballen getroffen. Plötzlich erlosch das Bild, der Lüfter verstummte, und die Lampe ging aus. Mist - Stromausfall. Und wieder nicht abgespeichert...

--

Nach diesem kurzen Drama wollte ich aber eigentlich noch etwas sinnvolleres sagen. Als wenn die ersten zehn Zeilen auf der Inhalts-Seite nicht gereicht hätten, werde ich hier noch ein paar Anmerkungen loswerden, die sich sonst in keines der Kapitel so richtig einreihen lassen.

Ich, du, er-sie-es
Die deutsche Sprachsyntax ermöglicht die Verwendung einer Höflichkeits-Form neben den normalen Anreden, die normalerweise benutzt wird wenn man es mit jemanden zu tun hat, den man nicht kennt. Also könnte ich Sie über den gesamten Text hinweg siezen. Andererseits werden viele Leser auch nicht gerade die Ältesten sein, weshalb ich das "Sie" nicht verwenden werde. Das soll natürlich nicht die 80-jährigen User diskreminieren. Kurzum, ich habe mich für Euch und Ihr entschlossen, was hier die gesamte Amiga-Fangemeinde anspricht. Wer sich damit nicht anfreunden kann, bitte sehr:

Sehr geehrte Dame, sehr geehrter Herr!
Wir gratulieren Ihnen recht herzlich zum Empfang der Madhouse-Distribution. Dieses Programmpaket wird Ihnen noch lange Freude bereiten, denn wir haben bei der Herstellung Wert auf ausgesuchte Materialien gelegt. Bitte lesen Sie diese Gebrauchsanweisung sorgfältig durch.
Da wir uns ja nun schon seit 80 Sekunden kennen, erlaube ich mir, Ihnen das Du anzubieten. Ich heiße Carsten, und du? (Stellvertretend für Aicke möchte ich hier anmerken, daß Aicke Aicke heißt.)

Englische Eindeutschung und deutsche Einenglischung
Besonders die Einsteiger, die das "Commodore Benutzerhandbuch Workbench" gelesen haben (möglichst bevor es auseinander fiel), wurden dort mit einer Fülle von neudeutschen Begriffen konfrontiert, die sich in der englischen Sprache viel besser anhören.
Von Blättersymbolen, Amiga-DOS Stapeldateien, Piktogrammen, Voreinstellern, Standardnamensmustern und nicht zuletzt Anmeldedateien ist hier die Rede. Ich bin aber noch mit Cyclegadgets, Scripts, Icons, Prefs, Patterns und Mountfiles aufgewachsen und finde z.B., daß sich "Lokalisationsvoreinstellungsdialogfenster" etwas gedrungen anhört... weshalb ich hauptsächlich Begriffe verwenden werde, wie man sie auch in Fachzeitschriften findet.

Hausgemachtes Sprachen-Wirrwar auf Commodore-Art: die locale.library
Eine der tollsten (wenn auch eine der letzten...) Ideen, die Commodore noch vor dem Konkurs vollbrachte, war die locale.library. Zu ihr gehören auch die ca. 50 Verzeichnisse und hunderte von Dateien, denn die locale.-library ist komplett modular aufgebaut. Falls also Mars-Kolonien entstehen sollten, oder ein Amiga an außerirdische Lebensformen verkauft wird, oder plötzlich ein neuer Kontinent auftauchen sollte, dann kann man ganz problemlos die betreffenden Sprachen und Auslandsvorwahlen inclusive des Zeitformats und der Wochentage einstellen.
Seit der Version 1.1 ist Madhouse lokalisiert. Das heißt: Benutzer von OS 2.1 oder höher kommen in den Genuß eines Madhouse mit deutschen Texten, OS 2.0 - User lesen nach wie vor alles auf englisch. Das stellt diese Anleitung auf eine harte Probe: englische und deutsche Texte müssen gleichzeitig erwähnt werden. Außerdem unterscheiden sich teilweise die Texte

vom MUI-Programmteil von den Texten, die ohne MUI zu lesen sind.

Leider ist Madhouse seit vorgestern Commoditie. Wieso leider? Nun, Aicke und ich sind nicht gerade Fans von CX's (dumme Abkürzung..), weil man als Anwender immer das Problem hat, die Einstellungsfenster der Commodities zu öffnen. Denn viele C. bieten nur zwei Möglichkeiten dazu: den Hotkey und das Exchange-Programm. Ersteres vergißt man dauernd, letzteres muß man erst starten. Weil Exchange selbst Commoditie ist, könnte man es auch per Hotkey aufrufen, aber wenn man den auch vergessen hat...
Naja, Commoditie-Hasser haben keine Probleme mit Madhouse, da man es ja ganz bequem mit dem Tools-Menü aufrufen kann. Und die, die voll auf Commodities abfahren, können Madhouse auch mit Exchange fernsteuern - wenn's denn unbedingt sein muß.
Madhouse ist aber nur deshalb Commoditie, weil sich das vom programmtechnischen Aspekt ganz gut macht, so lassen sich Maus und Tastatur leicht und vor allem systemkonform abfragen.

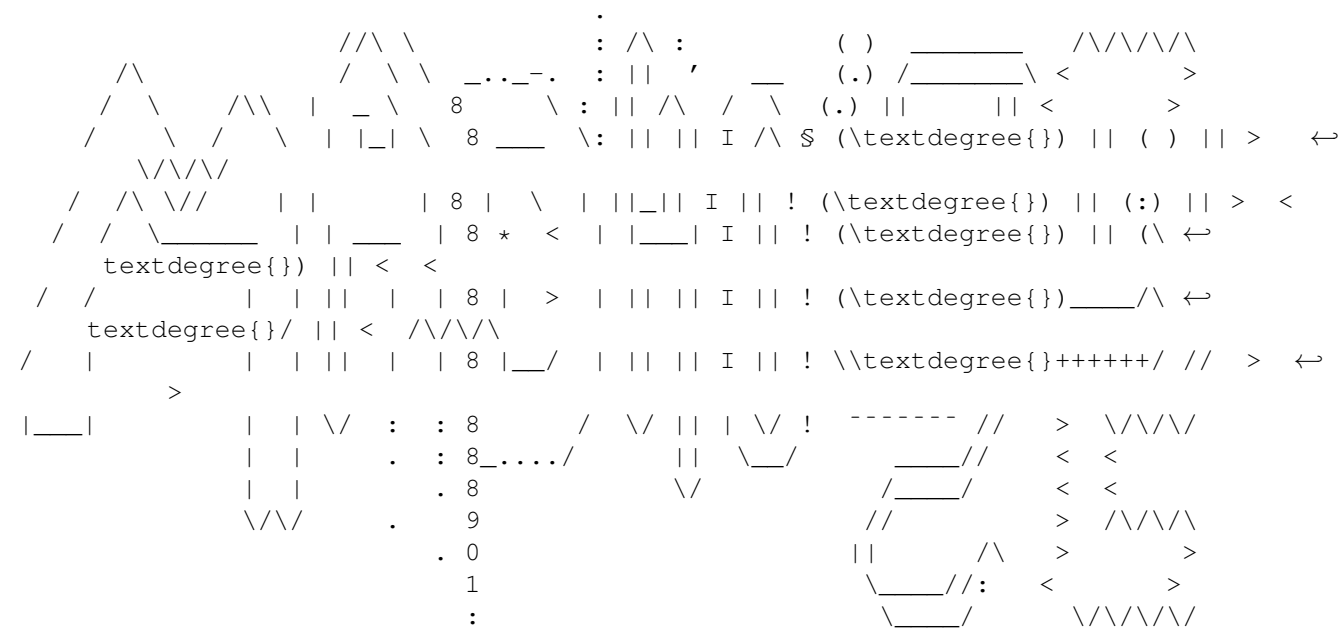
Was Madhouse von anderen herkömmlichen Marken-Blankern unterscheidet:
Die großen Unterschiede liegen in den kleinen Details. So kann mit fast jeder Programmiersprache ein Blanker für Madhouse erstellt werden, und kinderleicht ist es auch. Als Beispiel für ein anderes Detail möchte ich die Madhouse-Zufalls-Funktion hervorheben, die zu gegebener Zeit einen der Blanker auswählt, damit er dann angezeigt werden kann. Weil dieses Kapitel aber eh' schon groß genug ist, und ich Euch den Code auf keinen Fall ersparen möchte, müßt Ihr jetzt Hier klicken!
Die anderen Features werden Euch beim Lesen der Anleitung bestimmt auch auffallen.

Ein paar Worte an die Einsteiger.
Die Bedienungsanleitungen von Programmen, gleich ob kommerziell oder PD, können und sollen die Anleitung zur Bedienung des Computers nicht ersetzen. Wer also seinen Amiga noch nicht so lange besitzt, der sollte einerseits eine gute Computerzeitschrift lesen. Über solche Zeitschriften kann man etwas über Neuerscheinungen auf dem Amiga-Markt erfahren. Zum Einsteigen gehört aber meiner Meinung nach noch ein Einsteigerbuch, das auf Euren Computer zugeschnitten ist. Solche Bücher sind noch erhältlich und sind eigentlich immer um Längen besser als Commodores Orginal-Anleitung.

Zum Schluß bitte ich alle interessierten Programmierer, doch mal ein eigenes Modul für Madhouse zu schreiben, da die Schnittstelle recht gut beschrieben ist (in dieser Datei). Außerdem geht alles wirklich einfach, und kleine Hürden werden mehr als ausführlich erklärt. Viel Spaß dabei.

Zum absoluten Schluß danke ich allen, die sich so lange mein Gequassel durchgelesen haben... Schreibt uns mal! Für Verbesserungsvorschläge, Anregungen, Lob und Kritik sowie geistige Aufbaumaßnahmen ("Wann kommt denn endlich das Update?") sind wir immer offen. Die Adressen findet Ihr einerseits im etwas überdimensionierten (ich geb' es ja zu..) Info-Fenster des Hauptprogramms (über Info-Button erreichbar) und im Autoren- und Copyright-Anhang

Viel Spaß mit



wünschen Euch die Programmierer
Aicke Schulz und Carsten Jahn.

1.3 Die Madhouse-Zufallsfunktions:

Also, beim Durchschnittsblanker würde man so eine Funktion etwa so realisieren:

```
short bl_random()
{
   return Random(b_counter)
}
```

wenn b_counter die Anzahl der Blanker ist.

In Madhouse sieht das so aus:

```
short bl_random()
{
   short rd=0;
   BOOL takeIt = FALSE;
   BOOL Bl_Avail = FALSE;

   short used_min = 30000;
   short used_max = 0;
   short take_me_border = 0;

   // CPU-Tests?
   short cpu = 0;
   if( cpu_sensitive )     // Wenn CPU sensitive Random aktiv ist
```

```
   cpu = cpu_test();    // Zustand der CPU ermitteln (macht sie was?)

// Ist überhaupt ein Blanker verfügbar?
for( int i=0; i<b_counter; i++ ) {
   // Wurde der Blanker ausgewählt und lief bei ihm noch nichts schief?
   if( (b[i].rand_sel == TRUE) && (b[i].error == FALSE) ) {
      // Tritt folgender Fall nicht ein: CPU sensitive Random gewählt
      // UND CPU ist belastet UND der Blanker braucht CPU?
      if( !(cpu_sensitive && cpu && b[i].CPU_need == 1) ) {
         // Nein, dieser ist schon mal tauglich und wir haben einen
         // Dummen gefunden, der die Arbeit macht.
         Bl_Avail = TRUE;

         // Es wird ermittelt, wie oft dieser Blanker schon in dieser
         // Blank-Periode benötigt wurde. Dieser Wert steht in [].used.
         // Öfter oder weniger als alle anderen?
         if( b[i].used < used_min )   used_min = b[i].used;
         if( b[i].used > used_max )   used_max = b[i].used;
      }
   }
}
// Jetzt enthalten: used_min wie oft der Blanker schon dran war, der am
//                          wenigsten dran war und
//                  used_max wie oft der Blanker schon dran war, der am
//                          meisten dran war.

take_me_border = used_min + (used_max-used_min)/2;

// Die Gruppe der Blanker wird zweigeteilt. Einen Auftrittszähler klei-
// ner oder gleich take_me_border bedeutet, er war weniger dran als DIE
// HÄLFTE der Blanker, will sagen daß eine Menge Blanker schon öfter
// dran war als er.
// Besser kann ich das nicht erklären...

if( Bl_Avail ) {
   // Solange noch keiner herausgefischt wurde
   while( !takeIt ) {
      takeIt = TRUE;
      // Einen aus der Menge greifen
      rd=Random(b_counter);

      // Und sehen, ob er den Anforderungen genügt.
      // Wurde er etwa nicht ausgewählt? Dann nix wie weg mit ihm.
      if( b[rd].rand_sel == 0 ) takeIt = FALSE;
      // Oder hatte er in dieser Blank-Periode schon einen Fehler ge-
      // meldet? Dann geht er jetzt sich auch nicht... und tschüß.
      if( b[rd].error == TRUE ) takeIt = FALSE;
      // Oder wurde CPU sensitive Random ausgewählt UND die CPU ist
      // am kochen UND er würde den Prozessor zu stark beanspruchen?
      if( cpu_sensitive && cpu && b[rd].CPU_need ) takeIt = FALSE;
      // Oder haben wir den vorhin schon gesehen?
      if( b[rd].used > take_me_border ) takeIt = FALSE;
   }
} else
   // Kein Wunder - niemand hat sich gefunden...
   rd = -1;
```

```
   return rd;
   // Ergebnis ist -1, wenn kein passender Bl beschafft werden konnte.
}
```

Kleine Anmerkung: Obwohl - wie hier ersichtlich - die Blankerdaten in einem Array gehalten werden, ist die Anzahl der maximal möglichen Blanker nicht limitiert. Dies könnt ihr gerne durch hundertausendfaches Kopieren des CrazyPixel-Blankers nachprüfen. Ich habe das nicht ausprobiert...

1.4 Der Workhop

Soo, da wären wir nun. Eigentlich ist die gesamte Madhouse-Oberfläche selbsterklärend, wie man so schön sagt. Aber wer kann schon von seinenen Programmen behaupten, daß sie jeder versteht? Vielleicht Dr. Peter Kittel von seinen AmigaBASIC-Demos? Also dann, ab gehts:

0. Die Installation.
   ... sollte eigentlich ganz einfach gehen, denn das Installerscript zu Madhouse erledigt alles. Das dazu nötige Programm "Installer" solltet Ihr von Eurer Install-Disk (sofern Ihr eine habt, sonst von einem anderen Programm das den Installer verwendet) in Euer C:-Verzeichnis kopieren, dann müßte alles laufen.
   Wer absolut nicht an den Installer herankommt, was jetzt aber echt verwunderlich ist, der muß halt alles "handy" machen:

   a) Zum schnellen Ausprobieren: die amos.library aus dem Madhouse-Libs/-Verzeichnis in das libs-Verzeichnis der Startdiskette/-Festplatte kopieren. Das kann über die Shell so geschehen:
   "copy Madhouse/libs/amos.library to sys:libs/amos.library".
   Vor "Madhouse" muß ggf. noch ein Pfad mit Diskettenbezeichnung stehen. Das war's dann schon, denn wenn das Programm "Madhouse" aus seiner ursprünglichen Schublade heraus gestartet wird, findet es das Einstellungsprogramm von allein.
   b) Zur dauerhaften Anwendung: Wie das Wort "dauerhaft" schon andeutet, muß hier wieder die Schublade SYS:WBStartup/ herhalten. Aber zuerst die amos.library kopieren, wie unter a) beschrieben.
   Nun das Programm "Madhouse" nach SYS:WBStartup kopieren. Alle Programme in dieser Schublade werden dann beim Hochfahren des Rechners automatisch gestartet. Benutzer von Disketten sollten das gesamte Madhouse-Verzeichnis auf einer neuen Leerdiskette auslagern, HD-User legen ein neues Verzeichnis an und kopieren die Madhouse-Schublade dort hinein.
   Jetzt muß nur noch das ToolType ("Merkmal") im Madhouse-Icon (dem Madhouse-Piktogramm in WBStartup) geändert werden. Die Zeile CONFIGED= sagt Madhouse, wo sich das separate Einstellungsprogramm für Madhouse befindet, welches Madhouse ggf. selbst starten muß. Man muß diesen ToolType so ändern, daß er den neuen Pfad und den Programmnamen des MadhouseConfigEd-Programs enthält - also je nach dem, wohin man das Madhouse-Verzeichnis kopiert hatte.
   Damit - falls dies ein Amiga mit OS 2.1 oder höher ist - Madhouse die mitgebrachte Locale-Datei benutzen kann, um alles auf deutsch anzuzeigen, muß noch die Datei
   Locale/deutsch/madhouse.catalog
   aus dem Madhouse-Verzeichnis nach
   LOCALE:catalogs/deutsch/madhouse.catalog
   kopiert werden - und fertig.

1. Der Programm-Start.
   Das sollte kein großes Problem darstellen. Wie auf dem Amiga üblich, geschieht dies per Doppelklick auf das Madhouse-Icon.
   Falls der Computer nicht mit dem Amiga-OS 2.0 ausgerüstet ist, ist das schlecht, denn in diesem Fall gibt Madhouse nicht mehr als einen simplen Alert aus...
   Außerdem *muß* die amos.library im "LIBS:"-Verzeichnis vorhanden sein, aber das haben wir ja eben erledigt. (Nur, falls jemand mittels "assign add LIBS: xxx" mehrere Suchpfade für LIBS: erstellt hat, noch ein Hinweis: die amos.library muß sich in dem Verzeichnis befinden, was zuerst als LIBS: deklariert wird, also normalerweise sys:libs. Sonst stürzt der nächstbeste AMOS-Blanker ab. Sorry, aber so ist das halt bei AMOS.)
2. Ein Fenster öffnet sich.
   Und das ist schon einen Faszination für sich, denn bei installiertem MagicUserInterface (© Stefan Stunz, folgend MUI genannt, siehe auch MUI-Infos) reagiert der MadhouseConfigEd entsprechend und öffnet ein MUI-Fenster, sonst nur ein normales. Ihr habt richtig gelesen, im Moment und eigentlich immer bedient Ihr den MadhouseConfigEd und nicht Madhouse selbst. (Madhouse hat soeben dieses Programm gestartet).
   Falls nicht, wenn sich also kein Fenster öffnet, ist bereits alles in Ordnung. Beim Erststart wird Madhouse allerdings nicht seine Datei mit den Einstellungen finden können, so daß es das Hauptfenster öffnet.
   Wenn Ihr das Fenster "von Hand" öffnen wollt, dann
   - braucht Ihr keinen Hot-Key zu drücken, den Ihr immer vergeßt, und
   - Ihr braucht nicht das Exchange-Programm zu starten, das Ihr immer löscht...

   Ein Blick ins Tools-/Hilfsmittel-Menü bringt Klarheit, Madhouse trägt sich praktischerweise hier ein. Bei Anwahl Fenster.
   Wenn anstatt eines tollen Fensters ein fader Requester erscheint, der etwas von einem falschen Pfad labert, dann folgt den Anweisungen und beendet Madhouse (Requester bestätigen und Madhouse nochmals starten), ändert den CONFIGED-Tooltype (siehe Abschnitt 0) und startet Madhouse neu.
3. Jetzt den Pfad der Blanker angeben!
   Durch einen Klick auf den kleinen Knopf mit dem Bildchen (neben dem Path-Gadget) öffnet sich der ASL-Directory-Requester, in dem ihr jetzt das Verzeichnis mit den Blankern angeben könnt. Auf der ungeänderten Madhouse-Disk heißt dieses Verzeichnis "Blankers". Wichtig: Der Pfad muß mit dem NAMEN einer Diskette beginnen (z.B. "Work:.../Blankers"), sonst macht Euch Madhouse darauf aufmerksam und akzeptiert nix.
   Nachdem der Requester mit "Ok" bestätigt wurde, sollten in der Listbox auf der linken Seite des Hauptfensters Namen wie "FlyingToasters" stehen. Außerdem werden nun die restlichen Gadgets nicht mehr schraffiert dargestellt.
   Für Einsteiger, die ihren Wortschatz aufstocken wollen: Dieses Gadget "mit dem Bildchen" heißt Popup-Button.

- Wer keine Festplatte hat...
  der kann Madhouse auch benutzen. Jedoch muß dann der Schalter "Puffern/Buffering" im Advanced-Options-Fenster bzw. auf der Optionen/Advanced-Seite im MUI-Fenster aktiviert sein.
  Das geschieht schon standard-mäßig.
  Da Madhouse sehr gut auf den Diskettenbetrieb abgestimmt ist, aber oberflächlich davon nur dieses "Puffern/Buffering"-Gadget zu sehen ist, empfehle ich noch die Lektüre von Alles über Buffering.
- Wer aber eine Festplatte hat...

Der kann die Option "Puffern/Buffering" im Advanced-Options-Fenster (bei MUI: auf der Optionen/Advanced-Seite) ausschalten.
Doch dazu später mehr. Natürlich reicht es nicht, eine Festplatte zu haben, Madhouse muß auch dort installiert sein...

4. Nun sehen wir uns die Blanker-Module an!
   Ohne MUI:
   Zuerst muß das Edit-Gadget durch draufklicken weitergeschaltet werden, bis der Text "Einstellung/Prefs..." erscheint. In diesem Modus können die Einstellungen des Blankers geändert werden, den man in der unteren Liste auswählt. Also bitte einen Blanker anwählen!
   Mit MUI:
   Zuerst müßt Ihr das Registerfeld (oder das Cycle-Gadget) von "System" auf "Blankers" umschalten. Nun erscheint wieder fast dieselbe Liste auf der linken Seite des Fensters. Bitte auf einen Blanker doppelklicken.
   Weiter für mit und mit ohne :^) MUI:
   Schon öffnet sich ein Fenster, in dem sich u.a. ein Knopf mit der Aufschrift Test befindet. Anklicken und staunen!
   Durch einen weiteren Mausklick wird der Blanker dann wieder beendet. "Abbruch/Cancel" und "Okay" öffnen wieder das Hauptfenster, wobei "Okay" die Einstellungen abspeichert. "Dauer/Duration" (bei MUI im Hauptfenster, sonst im eben geöffneten BlankerPrefs-Fenster) wird im Referenzteil erklärt.
   Die anderen Gadgets variieren je nach Blanker. Hier bieten sich viele Möglichkeiten zum Herumspielen, beeinflussen diese Gadgets doch den Blanker. Ihre Bedeutungen werden dann im Blanker-Kapitel erklärt.
   Jetzt könnt Ihr erstmal alle Blanker ausprobieren.
5. Die Liste(n) mit den Blanker-Namen und was da so passiert:
   Ohne MUI: Der Edit-Button bestimmt, was passiert, wenn ein Blanker in der Liste ausgewählt wird.
   Steht er auf "Einstellung/Prefs..." kann, wie eben geschehen, das BlankerPrefsFenster geöffnet werden.
   Mit "Auswahl/Selection" wird bestimmt, welche Blanker benutzt werden, wenn Madhouse nach der angegebenen Zeit die Blanker startet.
   Durch Klick werden sie angeschaltet (» CrazyPixel) und wieder ausgeschaltet (CrazyPixel).
   Mit MUI:
   Die Liste, die auf der Blankers-Seite angezeigt wird, ermöglich das Ändern der Einstellungen der einzelnen Blanker. Die andere Liste, auf der System-Seite vorhanden, dient dem Auswählen der einzelnen Blanker. Mit Ihr wird bestimmt, welche Blanker gestartet werden sollen, wenn der Computer über eine einstellbare Zeitspanne hinweg keine Eingaben empfangen hat.
6. Wieviel Zeit meiner Inaktivität soll vergehen, bis Madhouse loslegt?
   Diese Einstellung wird mit dem "Zeit/Time"-Slider erledigt.
7. Konfiguration speichern.
   Ganz einfach mit "Speichern/Save". Dann wird die Konfiguration als "ENVARC:Madhouse.prefs" abgespeichert.
8. Wer keine Festplatte hat oder ein Paßwort braucht,
   Klickt hier: Die (Advanced) Options

1.5 Das Advanced Options Fenster / Die Advanced/Optionen-Seite

Dieses Fenster stellt einige weitere Funktionen bereit, die man selten ändert.

1. Paßwörter

   Für Zeitgenossen, die anderen alles zutrauen und für Leute, die den Amiga beruflich nutzen (ja, das gibt's!), haben wir eine Paßwort-Funktion zur Verfügung gestellt, die sich kaum umgehen läßt (was mir bereits zum Verhängnis wurde...)
   "Kaum" heißt, es wurde durch keinen Versuch die Sicherheit des Eingabescreens widerlegt. Und wir haben wirklich alles ausprobiert. Jedoch kann Madhouse auf Systemen ohne Festplatte den neuen Blanker nach Abbruch der Paßworteingabe nicht so schnell starten, was zur Folge hat, daß in dieser Zeit dann doch Veränderungen an den Programmen im Hintergrund vorgenommen werden können. Also ausprobieren!
   Wenn Ihr ein Paßwort wollt, müßt Ihr zunächst den Haken vors "Password/Paßwort/Benutzen"-Gadget setzen. Dann sollte noch ein möglichst gerissenes Paßwort in das Text-Feld "Name" (bzw. "Text") eingegeben werden. 4711 würde natürlich jeder Gangster zuerst ausprobieren, besser sind da ausgefallenere Dinge: "Schnellhefter", "ALDI", "Rasenmäher", "Hundshai", ... Ach, ja.. es ist recht hilfreich, wenn man das Paßwort nicht vergißt. Nach der Einstellung des Paßworts fragt Madhouse gleich nach demselben; was sich also nicht eingeben läßt wird gleich und ohne Probleme durch das alte Paßwort ersetzt.
   Nachdem ein Blanker gestoppt wurde, erscheint dann der Paßwort-Screen. Es ist zwar kein Cursor im Textfeld zu sehen, aber Ihr könnt einfach lostippen. Jedes Zeichen wird als "*" dargestellt. Es gibt drei Chancen, das richte Paßwort zu finden, aber ist das dritte falsch, verstreichen ein paar Minuten seit der letzten Eingabe, oder es wird Esc gedrückt, startet der nächste Blanker. Backspace funktioniert wie gewohnt, und Return bestätigt das Paßwort. Zwischen Groß- und Kleinschreibung wird fieserweise unterschieden!

2. Optionen für Disketten-Benutzer

   Wenn Ihr keine Hard-Disk habt, ist Madhouse der einzige modulare Screenblanker, den Ihr benutzen könnt (soweit ich weiß..).
   Was passiert wohl, wenn man einen herkömmlichen Marken-Mod.-Screenbl. startet, der seine Module nur auf einer Diskette sucht? Sofern beim Start des Blankvorgangs mal die benötigte Disk nicht im Drive liegt, startet die Workbench ihre eigene Show: Ein netter "Please insert Disk..."-Requester brennt sich in Eure Phosphorschicht.
   Wenn man "Puffern/Buffering" im Advanced-Options-Fenster aktiviert, ist soetwas mit Madhouse nicht möglich. Madhouse lädt dann immer einen Blanker in sein RAM:Madhouse_Storage-Verzeichnis, das es beim Programm-Start automatisch anlegt. Wurde dieser Blanker dann gezeigt, und die Diskette mit den Madhouse-Blankern liegt WIRKLICH im Drive (wird vorsichtig geprüft), dann lädt Madhouse einen frischen Blanker nach.

   Wenn Ihr zusätzlich "Nach Disk fragen/Ask for Disk" anwählt, zeigt Euch Madhouse mit einem kleinen Fenster auf der Workbench an, daß der gepufferte Blanker bereits benutzt wurde und daß Madhouse darauf lauert, daß die Blankers-Diskette eingelegt wird.
   Ein Klick auf das Schließ-Gadget dieses kleinen Fensters bewirkt nicht nur das Schließen des Fensters, sondern stellt auch alle Schnüffelversuche nach der Diskette ein.

   Madhouse kopiert außerdem die libs:amos.library in sein Storage-Verzeichnis, da die Amos-Blanker diese benötigen.

3. Der "Schwarze Hintergrund/Black Background"
   Wer Disketten benutzt, hat sicher schon die oft sekundenlangen Ladezeiten der Blanker bemerkt. Wer mehrere Blanker ausgewählt hat und "Blanker wechseln/Exchange Blankers" benutzt, bekommt beim Blankerwechsel immer kurz die Workbench zu sehen (sofern die Disk mit den Blankern im Laufwerk liegt, sonst kann Madhouse ja nicht wechseln). Wenn Euch das stört, könnt Ihr "Hintergrund öffnen bzw. Benutzen / open Background" aktivieren. Dann bleibt der Screen in den Ladezeiten schwarz. Wer zusätzlich "Info-Texte/and show info-texts" anwählt, bekommt auf diesem schwarzen Screen noch die Texte "Lade neuen Blanker/ Loading new blanker." und das beliebte "Bitte noch etwas Geduld.../Working. Please wait..." angezeigt!

4. Die anderen Optionen: "Other Options"
   Das Fenster der erweiterten Optionen bietet auch noch zwei "andere Optionen": Beginnen wir mit der interessanteren: "CPU active:"
   Zuerst einmal muß erklärt werden, daß verschiedene Programme und bestimmte Dinge den Prozessor Eures Computers mehr oder weniger belasten. Generell kann man sagen, daß die sog. CPU ausgelastet ist, wenn ein Programm etwas tut und Ihr warten müßt. Z.B. bei irgendwelchen Rechnungen.
   Während PC-Freaks noch froh sind, wenn sie gleichzeitig einen Text Drucken UND eine Diskette formatieren können, könnt Ihr auf Eurem Amiga ohne größere Probleme ein Bild berechnen und einen Text schreiben. Man beachte, daß das mit dem Amiga 1986 möglich war, und daß IBM 1994 mit OS 2 dafür Werbung macht...
   Also zurück. Wenn nun zwei rechenintensive Programme gleichzeitig laufen, also praktisch zwei Bilder berechnet werden, werden beide Bilder halb so schnell berechnet.
   Und jetzt kommts: Wer ein Bild berechnet und dabei einen rechenintensiven Bildschirmschoner laufen läßt, merkt 1. daß das Bild halb so schnell berechnet wird und daß 2. der Blanker halb so schnell läuft. Und das ist gleich zweifach ärgerlich.
   Deshalb gibt es in Madhouse das "CPU active:"-Gadget. Es hat drei mögliche Zustände:
   - "alle Blanker/use all Blankers" kümmert sich nicht um die CPU, so daß der o.a. Fall eintreten könnte.
   - "(nur) einfache Blanker/only simple ones" überprüft die CPU, ist sie aktiv wird ein Blanker gestartet, der die CPU fast gar nicht belastet. Wurden nur Blanker ausgewählt, die viel Rechenzeit benötigen, erscheint ein schwarzer Bildschirm und später die Mitteilung, die auf diesen Notstand aufmerksam macht.
   - "nichts anzeigen/show nothing" zeigt den schwarzen Screen immer, wenn die CPU beschäftigt ist.

   Diese Funktion ist übrigens voll kompatibel zur "Puffern/Buffering"-Option. Nur falls der gepufferte Blanker "viel" CPU braucht, ein Progr. auch viel CPU braucht und "CPU active:" auf "only simple Blankers" steht UND die Disk mit den Blankern nicht im Laufwerk liegt, seht Ihr auch den schwarzen Screen.
   Wenn Ihr wissen wollt, wie sowas im Programmcode aussieht, dann fragt lieber nicht...
   Wenn Ihr wissen wollt, welche Blanker denn nun rechenintensiv sind und welche nicht, hier steht es:die Blanker.
   Mit MUI läßt sich das einfacher herausfinden, einfach auf die "Blankers"-Seite wechseln und den fraglichen Blanker einmal anklicken.

Dann wird im Feld "CPU-Belastung/CPU load" folgendes angezeigt:
- "niedrig/low" wenn der Blanker nicht rechenintensiv ist; und
- "mittel bis hoch/medium to high" wenn er es ist.

Die zweite Option dieser Rubrik heißt "Blanker-Pri" und ist über ein (echt niedliches) Cycle-Gadget erreichbar. Wie der Name verspricht, könnt Ihr damit bestimmen, mit welcher Priorität die Blanker laufen sollen. Eine Taskpriorität kann zwar zwischen -128 und 127 tendieren, sinnvoll sind hier aber die Werte 0 und 1. Deshalb das Cycle-Gadget. Für den Fall, daß zwei Programme gleichzeitig rechnen, hat der Amiga die Task-Prioritäten. Dann wird das Programm mit der höheren Priorität zuerst abgearbeitet, das Programm mit der niedrigeren bekommt nur noch ganz wenig vom Prozessor. Normalerweise laufen alle Anwendungen mit der Priorität 0.
Rechnet nun also ein Programm (kanonisches Beispiel: Raytracer) und ein rechenintensiver Blanker wird mit der Priorität 1 gestartet, bleibt dieses praktisch stehen. Deshalb sollte man bei "CPU active:" = "alle Blanker/use all Blankers" immer 0 wählen.
Da bei "CPU active:" = "(nur) einfache Blanker/only simple Blankers" sowieso nur ein Blanker mit niedriger CPU-Belastung gestartet wird, wenn ein Programm rechnet, kann man in diesem Fall ruhig 1 wählen. Dann läuft der Blanker viel flüssiger und die Anwendung kann trotzdem weiterrechnen. Trotzdem muß das nicht die beste Einstellung sein, denn wenn einem Programm mitten beim Blanken anfängt zu rechnen, geht es ziemlich leer aus. Ein Beispiel wäre ein Backup-Programm, was abwechselnd Dateien packt und schreibt. Das Packen ist rechenintensiv, das Schreiben weniger. Wenn nun Madhouse die Prozessorauslastung gerade beim Schreiben überprüft, kann es gut sein, daß ein Blanker mit hoher CPU-Belastung gestartet wird, und das Backup-Programm beim Packen arg verlangsamt wird.

Und dann gibt’s noch - vorerst nur für MUI-User - den "Sound"-Button. Mit ihm läßt sich bestimmen, wie das Sound-System von Madhouse arbeiten soll. Auf der Blanker-Seite kann man ja für jeden Blanker ein Musik-Modul einstellen, was dann natürlich irgendwie abgespielt werden muß. Das ist vielleicht total schade, aber ich war tatsächlich zu faul einen kompletten Musik-Modul-Abspieler mit 70 unterstützen Formaten und modularen Konzept auf Assemblerbasis mit diversen Zusatzfunktionen zu programmieren. Aus zwei Gründen: 1. ich kann kein Assembler (was soll man denn noch alles können, Computer sind in mancherlei Hinsicht echt anspruchsvoll), und 2. sowas gibt’s schon. (Ok, das Argument zieht nicht ganz, es gibt auch schon etliche Screenblanker).
Deshalb bin ich die Sache anders angegangen: Madhouse spielt nicht, sondern läßt spielen. Dabei kann Madhouse zwei Sound-Quellen ansteuern:
1. die medplayer.library vom MED-Programmierer Teijo Kinnunen, die sinngemäß nur MED spielt und
2. DeliTracker von Delirium Softdesign (Peter Kunath and Frank Riffel). Der DeliTracker ist der härteste Modul-Player, den ich kenne. Er spielt so gut wie alles, und wird per ARexx ferngesteuert.

Ihr könnt nun zwischen einer der Varianten wählen, und zwar mittels des Sound-Gadgets. Obwohl Madhouse den DeliTracker mittels ARexx steuert, muß dazu nicht das Programm RexxMast aus der System-Schublade laufen (darf aber). Die ARexx-Kommunikation zwischen zwei Programmen kann vollständig über die rexxsyslib.library erfolgen, die dann automatsch in Euren Speicher gestopft wird.
Wenn Ihr also mehr als MED-Module abspielen wollt, dann muß die Einstellung "via DeliTracker" lauten. ("via" heißt übrigens "auf dem Wege

über" und ist damit das lateinische Pendant für einen ARexx-Port, bloß kürzer!)
Wer
Es ist mir sehr wichtig, dies auch hier nochmals zu erwähnen: für die Sound-Funktionen von Madhouse ist eine Festplatte notwendig, damit die betreffende Datei erreichbar ist.

1.6 Alle Blanker in der Übersicht

Hier ist der Nachschlageteil für unsere Blanker, alphabetisch geordnet und mit allen nötigen Infos.

CrazyPixel
DigitalClock
Drops
Fireworks
FlyingToasters
Glitter
Memory
Note
Shuffle
Skyline
Soccer
SoftwareFailure
Stars
Thunder
Waves
Worldtime

Bitte beachten: Der Duration-Slider und die Gadgets am unteren Rand des BlankerPrefs-Fensters sind in Referenz/BlankerPrefs-Fenster erklärt. Der Wert des "Dauer/Duration"-Sliders wird nicht mit Okay abgespeichert, sondern NUR mit der Save-Funktion im Hauptfenster. Trotzdem muß dann das BlankerPrefs-Fenster mit Okay verlassen werden, damit der Wert akzeptiert wird.

1.7 Die Blanker in der Übersicht - CrazyPixel

CrazyPixel

Von Aicke Schulz, Version 1.

Dieser Blanker zeigt einen springenden Punkt sowie den Wusel.

Mode/Modus: schaltet zwischen Wusel und Bouncing Point um. Ist Random ausgewählt, wird zufällig ein Modus benutzt.
Wusellänge/Wusel lenght: ist die Länge des Wusels.
Toneffekt/Sound: macht Geräusche zum Bouncing Point.

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig. Speicherverbrauch:160 kB. Programmgröße:25 kB.

1.8 Die Blanker in der Übersicht - DigialClock

DigitalClock

Von Aicke Schulz, Version 1.

Wie viele andere Blanker hat auch Madhouse eine Uhr. Diese benutzt aber einen echten Digital-Font, wie er auch bei den beliebten Uhrenradios zu finden ist. Achtung: Wenn die Uhr falsch geht, dann liegt das entweder daran, daß die interne Uhr des Amigas falsch gestellt ist oder daß ihr kein Akku zur Verfügung steht.

```
  Sprache/Language: Hier können auch ein deutsches Datum und deutsche Texte
                    für diesen Blanker gewählt werden.
         Countdown: Wenn dieser Wert nicht Null ist, ertönt ein Klingeln
                    nach <Countdown> Minuten nach dem Start des Blankers.
       Datum/Date: schaltet das Datum ein.
     Rollen/Scroll: bewegt die Uhr.
```

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig. Speicherverbrauch:180 kB. Programmgröße:34 kB.

1.9 Die Blanker in der Übersicht - Drops

Drops

Von Aicke Schulz, Version 1.

Tropfen laufen über den Screen. Sie haben unterschiedliche Geschwindigkeiten und sind in der Farbe änderbar. Aicke hat mit viel Liebe zum Detail die Drops gezeichnet. Von hunderten von Farbsets sind die schönsten ausgewählt worden. Sie faszinieren mit ihren warmen, hellen Eindrücken oder lassen die Gedanken des Betrachters im tiefen Blau versinken. Ohne zu Übertreiben muß ich sagen, daß diese Farben wirklich einmalig in der gesamten Welt der Amiga-Screensaver sind. Doch stellen wir sie einzeln vor: Ocean wird seinem Namen wirklich gerecht. Ocean zeugt noch von der Zeit, als die Ozeane klar und unverseucht waren. Ein heutiges Ocean halte ich für unmöglich, müßten doch Giftmüllfässer und russische Atom-U-Boote durchschimmern. Wine ist von seiner ganzen Farbpracht ein Genuß. Es springt regelrecht aus dem Monitor und und bezaubert durch die Anmutigkeit der Farbe. Grass erzählt von grünen Wiesen, spielenden Kindern und grasenden Schafen. Grass ist wie ein Tag im Wald, läßt den Zuschauer frei assoziieren und die Augen mit dem tiefen Grün verschmelzen. Caramel: Gold-gelb und sooooo sanig-süß. Das hat mir mein Großvater schon geschenkt. Und was gebe ich heute meinem Enkel? Fresh erinnert an einen Sonntagmorgen, an die Strände von Miami Beach und Ibiza. So hell ist die Farbe, daß sich kaum das Licht darin bricht. Fresh ist frisch. Da soll Aicke mal sagen, ich würde seine Blanker kurz abhandeln... Grüße an alle Kunstlehrer!

```
 Anzahl/Number: Anzahl der Drops, die maximal gleichzeitig auf dem Screen
                sind.
   Farbe/Color: Der absolute MEGA-Mix der 5 Farbpaletten. Random läßt den
                Blanker selbst wählen.

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig.  Speicherverbrauch:180 kB.   Programmgröße:17 kB.
```

1.10 Die Blanker in der Übersicht - Fireworks

```
                                Fireworks

                        Von Carsten Jahn, Version 1.

Dieser Blanker zaubert ein tolles Feuerwerk auf Euren Bildschirm. Es
stehen vier Effekte zur Verfügung. Mit den Slidern bestimmt man, wie
oft ein Effekt verwendet wird. Wenn die Slider nicht zu hoch gesetzt
werden, hat FireWorks noch Möglichkeiten, die Farben zu ändern. So ent-
stehen immer neue Farben, obwohl kein Effekt "mittendrin" die Farbe
wechseln muß. Will sagen: man merkt nur, daß plötzlich ein neuer Effekt
in einer neuen Farbe erscheint, Effekte, die schon auf dem Screen sind,
werden nicht beeinflußt.

          Spiralen/Spirals: bestimmt, wie stark die Spiral-Effekte vertreten
                            sind.
             Fontänen/Jets: bestimmt, wie stark die Fontänen vertreten sind.
               Bälle/Balls: bestimmt, wie stark die Kugeln vertreten sind.
 Farb-Fontänen/Color-Jets: bestimmt, wie stark die Mehrfarb-Fontänen ver-
                            treten sind.
  Pixelgeschw./Pixelspeed: Natürlich schafft es Fireworks schneller, wenige
                            Punkte zu zeichnen als viele. Wenn nur wenige
                            Punkte auf dem Bildschirm sind, muß Fireworks
                            deshalb öfter Pausen einlegen, damit die Anzeige
                            nicht optisch langsamer wird wenn mehr Punkte
                            hinzukommen. Mit Pixelgeschw. kann man nun ein-
                            stellen, wie groß die Pausen bei wenigen Punkten
                            werden. Hier muß man einfach etwas herumpro-
                            bieren, bis der Effekt nicht mehr schneller und
                            langsamer wird.

CPU-Belastung: Hoch.  Speicherverbrauch:110 kB.   Programmgröße:17 kB.
```

1.11 Die Blanker in der Übersicht - FlyingToasters

```
                              FlyingToasters

                        Von Carsten Jahn, Version 1.

Boh ey, guck 'ma da fliegt'n Toaster! Kleine, drollige Wesen mit Flügeln
und glühenden Toast-Slots schwingen sich über den Screen. Aus leblosen
Küchenknechten werden niedliche Gestalten, aus dem morgendlichen Frühstück
```

werden Hindernisse, schon beginnt der Spaß. Die auf- und abrutschenden Auswurfhebel erschrecken die Toasts auch nicht, keines springt beiseite. Folge: jeder 20ste Toaster in Deutschland ist eingeklemmt. Wohin soll das noch führen? Was weiß ich. Spendet jetzt aber nicht für eingeklemmte Toaster, sondern lieber für verkohlte Toasts, die haben ein viel schlimmeres Leben.

```
Toaster/Toasters: max. Anzahl der Toasters auf dem Screen.
          Toasts: max. Anzahl der Toasts auf dem Screen.
   Geschw./Speed: bestimmt die Geschwindigkeit des Blankers.
   Marmelade/Jam: schmiert Marmelade, Honig oder Nutella auf die Toasts.
                  Das perfekte Frühstück!
Double Buffering: schaltet Double-Buffering ein. Dadurch wird das Flackern
                  der bewegten Objekte vermieden. Benötigt mehr Speicher.
                  Ist dieser nicht vorhanden, wird vor dem Totalabbruch
                  noch ein Versuch ohne diese Option gemacht.
Leerer Start/Endscreen /
   In/Out Moving: Ist diese Option abgeschaltet, so erscheinen die Toasters
                  und Toasts beim Blankerstart plötzlich und sind auch noch
                  mitten auf dem Screen, wenn der Blanker abbrechen muß.
                  Wird "In/Out Moving" jedoch aktiviert, fliegen die Toasts
                  und Toasters erst rechts herein, der Blanker beginnt mit
                  einem schwarzen Bildschirm. Eine den Einstellungen und
                  Prozessortyp angemessene Zeitspanne vor Ablauf der Dura-
                  tionzeit werden dann keine neuen Toasts mehr erscheinen,
                  so daß sich der Blanker auch wieder mit einem leeren
                  Screen verabschiedet.
                  Das zeitgemäße Herausfliegen klappt natürlich nur bei
                  aktiviertem "Blanker wechseln/Exchange Blankers", da
                  sonst der Blanker ja noch bis ins Jahr 3000 laufen könn-
                  te.
```

CPU-Belastung: Hoch. Speicherverbrauch:297 kB. Programmgröße:29 kB.

1.12 Die Blanker in der Übersicht - Glitter

Glitter

Von Aicke Schulz, Version 1.

Glitter ist sicher nicht das Genialste, was Ihr jemals gesehen habt, aber trotzdem ganz nett.

```
              Anzahl/Number: Anzahl der Sterne. Dies verlangsamt den
                             Start des Blankers, jedoch nicht die Ani-
                             mation.
  Geschwindigkeit/Cyclespeed: Geschwindigkeit des "Glitterns".
```

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig. Speicherverbrauch:218 kB. Programmgröße:15 kB.

1.13 Die Blanker in der Übersicht - Memory

Memory

Von Aicke Schulz, Version 1.

Das ist eine gute Möglichkeit, mal einen Blick in den Speicher des Amiga zu werfen. Hier werden alle Daten als Grafik abgebildet. So erscheinen die geöffneten Screens und viele andere Dinge, die man natürlich nicht erkennt. Da der Amiga nach einem Reset nicht alle Daten löscht, sondern nur zum Überschreiben freigibt, lassen sich oft noch Grafiken eines zuvor gestarteten Spiels erkennen - natürlich in die Bitplanes aufgespalten und deshalb zweifarbig. Jedenfalls sieht man immer was Neues... Übrigens wird nur das erste MegaByte ChipRAM angezeigt.

Geschwindigkeit/Speed: bestimmt die Geschwindigkeit des Scrollens.

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig. Speicherverbrauch:180 kB. Programmgröße:14 kB.

1.14 Die Blanker in der Übersicht - Note

Note

Von Aicke Schulz, Version 1.

Ganz nach Belieben erfüllt Note gleich zwei Aufgaben in einem Blanker: Einerseits kann den Leuten, die am Computer vorbeigehen oder die etwas vom Benutzer des Computers wollen, eine Nachricht angezeigt werden. Andererseits ist auch der umgekehrte Weg möglich, die Anderen können eine Mitteilung eingeben und der Benutzer kann sie lesen. Um die Sache abwechslungsreich zum machen, hat Aicke noch drei verschiedene Papier-Typen gezeichnet, auf denen wird dann geschrieben und angezeigt.

Papier/Paper: Mit diesem Schalter wird der Papier-Typ ausgewählt. "Granit(e)" ist ein Betonklotz, "Mittelalter/Medival" ist ein Mittelalter(-Papier natürlich...) und "gegen/Against AIDS" ist gegen AIDS. Anders als Granit(e) und Mittelalter /Medival ist Against AIDS keine eigene Erfindung, diese Notizblöcke gibt’s wirklich. Sie sind erhältlich bei der Bundeszentrale für gesundheitliche Aufklärung
Postfach 910152
5000 Köln 91 (Sorry - die neue Postleitzahl stand nicht da).
Bestell-Nr. 70551000.

Modus/Mode: Schaltet zwischen dem Empfänger- und Versender-Modus um. Bei "Nachricht geben/Give Message" werden die nebenstehenden Texteingabefelder zur Anzeige eines Textes benutzt. Dabei muß jede Eingabe mit <Return> abgeschlossen werden. "Nachricht bekommen/Get Message" läßt die Besucher eintippen und den Benutzer lesen. Wenn "Blanker wechseln/Exchange Blankers" nicht aktiv ist, oder der Blanker noch arbeitet wenn der Benutzer zurückkommt, dann kann er den Text noch lesen.Wenn aber die Duration-Zeit von Note um war und ein neuer Blanker

gestartet wurde, kann der Benutzer die Nachricht nach dem Wegklicken des Blanker in der Error-List lesen.

ruhend/Static: Damit sich der Notizzettel nicht in der Phosphorschicht des Monitors einbrennt, läßt sich mit "Static" die Zeit bestimmen, nach der der Zettel wieder seine Position wechselt.

Textzeilen/ Textlines: Dies sind die Schreibfelder. Jedes Schreibfeld steht für eine Zeile auf dem Notizblatt. Bei Against AIDS können die letzten drei Zeilen nicht verwendet werden, weil da unten Platz fehlt. Auf einer MUI-Oberfläche haben diese Gadgets die Shortcuts 1 bis 7 (linke Spalte) und a bis h (rechte Spalte), c ausgenommen.

Fehlermelungen: Die Nachricht, wenn dies so eingestellt wurde.
Außerdem die üblichen AMOS-Fehlermeldungen.

CPU-Belastung: Niedrig. Speicherverbrauch:210 kB. Programmgröße:48 kB.

1.15 Die Blanker in der Übersicht - Shuffle

Shuffle

Von Carsten Jahn, Version 1.

Habt Ihr schon einmal einen modularen Screenblanker ohne Shuffle-Modul gesehen? Ich nicht... Jedoch ist dieser Shuffler erwartungsgemäß nicht ganz gewöhnlich. So wird eine AGA-Workbench unterstützt, und mit Grafikkarten sollte es zumindest mit einem Register- (d.h. nicht Echtfarb-) Modus keine Probleme geben. Bitte ausprobieren. Außerdem - und jetzt kommt der Hammer - shuffled Shuffle den vordersten Screen nicht nur, sondern restauriert ihn auf Wunsch auch wieder. Da seid Ihr platt, was?

Modus/Mode: bestimmt den Shuffle-Modus. "Mischen/Shuffle" ist die Standard-Einstellung. Hier wird der Screen nur geshuffled. Die beiden anderen funktionieren nur mit aktivierter "Blanker wechseln/Exchange Blankers"-Option (Hauptfenster), da der Blanker dafür wissen muß, wie lange er maximal blanken soll. "Mischen & Auflösung / Shuffle & Restore" Shuffled die Hälfte der Zeit, danach wird restauriert, also werden die Tiles wieder so verschoben, daß alles wieder richtig wird. "Auflösung/Restore" bringt den Screen zuerst (unsichtbar) so durcheinander, daß nach der eingestellten Zeit der Screen wießder hergestellt ist. Danach wird restauriert.

Geschw./Speed: bestimmt die Geschwindigkeit. Diese wird für die Zeitberechnungen (s.o.) mitberücksichtigt.

Gitter/Grid: schaltet den 3D-Rahmen ein oder aus. Falls die Farben des Bildschirms überhaupt nicht passen (3D-Rahmen würde blöd aussehen) benutzt der Blanker nie einen Rahmen.

Fehlermeldungen: "The "Shuffle & Restore" mode and the "Restore" mode can be only used if "Exchange Blanker" is activated." Diese Fehlermeldung will uns sagen, daß es doch ein User probiert hat, diese Modi ohne "Blanker wechseln/Exchange Blankers" zu benutzen.

CPU-Belastung: Niedrig. Speicherverbrauch:??? kB. Programmgröße:13 kB.

(Der Speicherverbrauch von Shuffle hängt sehr stark von der Auflösung des zu "shufflenden" Screens und der Mode-Option ab, da hier etwas mehr Speicherplatz benötigt wird.)

1.16 Die Blanker in der Übersicht - Skyline

Skyline

Von Aicke Schulz, Version 1.

Na Ihr Hinterwäldler, wollt Ihr mal was anderes sehen als Kühe und Berge? Als Stadtmensch kann ich mir jetzt gar nicht so vorstellen, wie das auf dem Land abgeht. Zweiunddreißig Einwohner, eine Schule, einen Lehrer, einen Briefkasten, eine Straße, eine Telefonzelle, zwei Amigas?
Oder habt Ihr etwa PC's? Doch bevor ich Euch nun über die 20 Nachteile eines PC's aufkläre, sage ich noch etwas zu Skyline.
Skyline stellt Hochhäuser dar, wie sie in richtig großen Städten vorkommen. Weil es aber dunkel ist, sieht man sie nicht. Man sieht nur die Fenster, in denen noch Licht an ist. Die Helligkeitswerte für die einzelnen Fenster werden über eine quadratische Binominalsublimationsfunktion trianguliert, die Aicke in nächtelanger Kleinarbeit nach ihren Koeffizienten aufgelöst hatte. Dagegen wird der Mond nur auf einer billigen Parabel bewegt.
Skyline gehört zu den Blankern, deren voller Funktionsumfang nicht auf den ersten Blick ersichtlich ist. Deshalb solltet Ihr hier Duration ruhig einmal hochstellen, dann kommen die vielen Fassadentypen zum Vorschein.
Die verschiedenen Dächer kann man gleich bewundern.

Mond-/Moonphase: Skyline stellt am Horizont einen Mond dar, der sich langsam über den Screen bewegt. "Zufall/Random" läßt den Zufall über die Größe der Sichel entscheiden, "Einstellung/Setting" macht die Größe einstellbar (mittels des gleichnamigen Sliders darunter).
Einst./Setting: Falls unter Moonphase "Einstellung/Setting" eingestellt ist, läßt sich hiermit die Größe der Mondsichel in Prozent einstellen. 100% entspricht einem Vollmond, 0% einem Neumond.
Sternfarbe/
Starcolor: Die Sterne am Himmel lassen sich in ihrer Farbe beeinflussen. "Aus/Off" schaltet sie ganz aus, "Zufall/Random" läßt den Computer wählen.
Anzahl/Number: Anzahl der Sterne.
Y-Limit: gibt an, wie weit die Sterne nach unten gehen bzw. in welcher Bildschirmzeile sich der unterste Stern befinden darf.
Himmel/Sky: schaltet zwischen verschieden Copper-Listen im Hintergrund um. "Zufall/Random" ist wieder der Zufallsgenerator.

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig. Speicherverbrauch:200 kB. Programmgröße:32 kB.

1.17 Die Blanker in der Übersicht - Soccer

Soccer

Von Carsten Jahn, viele Spieler-Grafiken von Aicke Schulz; Version 1

Hier mal wieder eine echt innovative Idee: Fußball in den Arbeitspausen hat wohl vielen gerade noch gefehlt. Was Bundesliga-Fans bestimmt begeistert, wird wohl auch anderen (mich eingeschlossen...) viel Freude machen. Die kleinen Fußballer rennen, was das Zeug hält (jedenfalls auf schnelleren Amigas), um die gegnerische Mannschaft fertigzumachen.
Ab und zu fällt auch mal ein Tor, mangels Sound ist aber leider nicht viel von den Fans zu hören. Und hier haben wir auch schon die Aufgabe für Euch da draußen: anfeuern, mitjubeln, ausrasten. Viel Spaß damit.
Wenn möglich (Speicher verfügbar) arbeitet Soccer mit Double Buffering.

Aufstellung Weiß/Rot /
Line-up White/Red: hiermit stellt Ihr die Spieltaktik ein, nach der die kleinen Erdnuckel spielen.
1. Zahl: Spieler in der Abwehr,
2. Zahl: Spieler im Mittelfeld,
3. Zahl: Spieler im Sturm.

Torwartsintelligenz/
Keeper intelligence: als ob irgendetwas im Computer intelligent wäre, läßt sich hier die Intelligenz der beiden Torwärte einstellen, was die Ergebnisse natürlich fatal beeinflußt.

CPU-Belastung: Hoch. Speicherverbrauch:378 kB. Programmgröße:45 kB.
Bei Speichermangel:260 kB.

1.18 Die Blanker in der Übersicht - Stars

Stars

Von Carsten Jahn, Version 2

Auch hier ein Blanker, der dem Beobachter den Eindruck gibt, räumlich in einem unendlichen Weltraum herumzufliegen. Auch hier ein paar Extras: da wäre zuerst der Rückwärtsgang zu nennen, den viele schon in anderen Star-Blankern vermißt haben werden. Der absolute Clou ist aber der Dreh-Effekt, der sich in Häufigkeit, Beschleunigung und Geschwindigkeit beeinflussen läßt. Besonders der "Waschmaschinen-Effekt" (hohe Drehzahl bei niedriger Geschwindigkeit) löst immer wieder Übelkeit unter den Betrachtern aus... üüüüürrah!
Fast schon beängstigend wirken die Shift-Effekte: die Kamerafahrt durchs Sternenfeld ist dann äußerst kurvenreich.
Diesen Blanker habe übrigens optimiert wie keinen anderen. Besonders die Dreheffekte verlangsamen den Blanker kaum, und die Performance insgesamt ist schon ganz gut. Wie jeden anderen Blanker habe ich die Stars zwar in C geschrieben, aber mit Hilfe des Compiler-Assembler-Listings optimiert.

Anzahl der Sterne/
Number of Stars: äääh.. Anzahl der Sterne?

Normale Bewegungen/Normal Movements (Bewegungen auf der Z-Achse):

```
         Maximum: max. Fluggeschwindigkeit des Betrachters.
         Manimum: min. Fluggeschwindigkeit des Betrachters. Negative
                  Zahlen lassen auch den Rückwärtsgang zu.
  Ändern/Changes: bestimmt, wie oft die Geschwindigkeit gewechselt wer-
                  den soll.
           Start: Startgeschwindigkeit.

Drehungen/Turns (Bewegungen UM die Z-Achse, Drehungen des Betrachters):

         Maximum: max. Drehmoment.
Beschleunigung/
    Acceleration: Beschleunigung der Drehung.
  Ändern/Changes: bestimmt, wie oft die Drehrichtung wechselt.
           Start: Startgeschwindigkeit.

Verschiebungen/Shifts (Bewegungen der X- und Y-Achse)

   Geschw./Speed: Dieser Wert legt fest, wie eng die Kurven sind.
  Gebrauch/Usage: bestimmt, wie oft der "Kurvenmodus" verwendet wird;
                  0 = nie, 10 = immer.
             X/Y: Hiermit wird definiert, ob sich die Verschiebungen
                  nur auf eine Achse, beide oder gar keine beziegen.

CPU-Belastung: Hoch.  Speicherverbrauch:70 kB.   Programmgröße:11 kB.
```

1.19 Die Blanker in der Übersicht - SoftwareFailure

```
                          SoftwareFailure

                     Von Aicke Schulz, Version 1.

Tipp, tipp; F10; Compiling; No Errors; Linking; Running: Blink-Blink-Blink.
 -Zap- "Software Failure - Press left mouse button to continue".
Von erfolgreichen Programmieren empfohlen: der leichte Absturz für
Zwischendurch. In Härtefällen ist auch das Sechserpack geeignet. Für An-
wender, die nur ausgereifte Software benutzen, bietet sich aber Software-
Failure für den täglichen Gebrauch an. SoftwareFailure simuliert den
komplexen Ablauf eines Absturzes nur grafisch, was aber den Vorteil hat,
daß man nach dem Blanken weiterarbeiten kann.
Der Überraschungseffekt nach den Arbeitspausen ist immer wieder tödlich!
Damit sich die Alert-Box aber nicht einbrennt, und damit Nervenschwache
nicht jedesmal einen Infarkt bekommen, wird der SoftwareFailure noch in
drei verschiedenen Modi animiert.

 Effekt/Effect: Mit Effekt kann gewählt werden, was nach einer gewissen
                Zeit des Blinkens mit der Alert-Box geschehen soll:
                "Zerfließen/Melt" läßt sie langsam zerfließen, "Springen/
                Bounce" läßt sie auf und ab hüpfen und "Crazy" bewegt die
                einzelnen Buchstaben.
   Warten/Wait: Hier wird die gewisse Zeit (s.o.) bestimmt, in Sekunden.

Fehlermeldungen: siehe AMOS-Errors.
```

CPU-Belastung: Niedrig. Speicherverbrauch:150 kB. Programmgröße:29 kB.

1.20 Die Blanker in der Übersicht - Thunder

Thunder

Von Aicke Schulz, Version 1.

Blitz und Donner im Amiga! Mit realistischem Sound. Wenn es draußen kalt und Winter ist (kann man das so sagen?...), dann kommt dieser Blanker gerade richtig. Die Blitze zucken nur so über den Screen, daß man denkt, es hat den Monitor zerbröselt. Außerdem kann man sehr gut die Übertragungsgeschwindigkeit von Wellen mit unterschiedlicher Wellenlänge in der Luft erkennen: während man den Blitz fast sofort sieht, kommt das Geräusch erst später. Wie lange die Zeitspanne zwischen Blitz und Donner ist, hängt also von der Entfernung des Unwetters ab, die man mit "Distanz/Distance" einstellen kann.

Einschläge/Flash Interval:	Mit dieser Einstellung wird bestimmt, wie oft die Blitze einschlagen. Es sind jeweils Zeitspannen angegeben, die Blitze werden dann in der gewählten Zeitspanne abgeblitzt.
Distanz/Distance:	Gibt die Entfernung des Gewitters in Metern an und bestimmt damit die Zeitspanne zwischen Blitz und Donner.
Toneffekt/Sound:	Hiermit kann man die Geräusche abschalten, falls man das mit den Wellenlängen noch nicht ganz verstanden hat.

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig. Speicherverbrauch:185 kB. Programmgröße:32 kB.

1.21 Die Blanker in der Übersicht - Waves

Waves

Von Carsten Jahn, viele Farbpaletten von Aicke Schulz; Version 1.

Dieser Blanker erzeugt Wellen, die so aussehen, als wären sie einem ganz genialen Algorithmus entsprungen. Tatsächlich wird nur ein einfacher Trick angewandt, wie er in manchen Demos zu finden ist. Zum besseren Verständnis und zum allgemeinen Aha-Effekt will ich ihn erläutern: Zuerst wird (im Hintergrund) mit 1, 2 oder 3 Pixel dicken Querstreifen ein Farbverlauf auf den Screen gezeichnet. Dann kommt eine Sinus-Funktion daher, die aus den Streifen eine Welle macht - hoch, runter, hoch runter, ... Weil das Ergebnis noch eindeutig als Sinus-Funktion erkennbar wäre, wird jede Bildzeile noch um eine weitere Sinus-Funktion nach links oder rechts verschoben. Jetzt bekommt der Screen seine Farben, die kontinuierlich durchlaufen. Zur Steigerung des beliebten Würg-Effekts wird der Screen, der Anfangs mit Übergröße geöffnet wurde, anhand zweier Sinus-Funktionen (ach ja..) nach links/rechts und oben/unten verschoben. Natürlich kann

man keine gewöhnliche Farbpalette nehmen, sie muß echt terrormäßig 'rein-
hauen. Da bleibt kein Teppich trocken -schluck-.
Obwohl auch bei diesem Blanker eine Duration-Einstellung bis 30 min.
möglich ist, empfehlen wir aus gesundheitlichen Gründen nicht mehr als
6 Minuten. Ob die Krankenkassen für Folgeschäden zahlen, ist uns nicht
bekannt. Nebenwirkungen und Gegenanzeigen bitte melden.

Das Prefs-Fenster ist in zwei Bereiche aufgeteilt. Links findet Ihr die
Farbpaletten. Waves wählt eine aus, bei der das Häkchen gesetzt ist. Wer
hinterhältigerweise alle Häkchen löscht, wird mit einer Zufallspalette
nicht unter 32 Farben bestraft. Auf der rechten Seite seht Ihr die Ein-
stellungen zum Zeichnen der Waves:

Wellengröße/WaveSize: beeinflußt die gesamt-Dicke der Waves.
XWellen/XWaves: bestimmt die Breite der Waves. Kleiner Wert = breite
Waves.
YWellen/YWaves: bestimmt die Höhe der Waves. Kleiner Wert = hohe
Waves.
XGröße/XSize: ist der Maximalausschlag der horizontalen Sinus-
Funktion.
YGröße/YSize: ist der Maximalausschlag der vertikalen Sinus-
Funktion.
XGeschw./XSpeed: legt fest, wie schnell sich der Screen in X-Richtung
bewegt.
YGeschw./YSpeed: legt fest, wie schnell sich der Screen in Y-Richtung
bewegt.

Da jetzt sicher nicht jeder Parameter sonnenklar ist (irgendwie kann man
die Waves ja auch nicht beschreiben) rate ich zum gründlichen Auspro-
bieren. Übrigens sind manche Farbpaletten bewußt NICHT schön, sondern
dienen dem Abschrecken von diversen Haustieren. Denn Vogelsch***e auf der
Tastatur ist eine recht unangenehme Sache; Katzen haaren alles voll und
Hunde lutschen glatt die Leertaste weg. Also Vorsicht!
(Im übrigen halte ich gar nichts von Paßwortprogrammen, die den Computer
vor Fehleingaben durch über alles trampelnde Katzen schützen. Ein Paß-
wort hält doch keine Katze davon ab, alles zu verwüsten!)

CPU-Belastung: Niedrig. Speicherverbrauch:212 kB. Programmgröße:14 kB.

1.22 Die Blanker in der Übersicht - Worldtime

Worldtime

Von Aicke Schulz, Version 1.

Wie hängt man eine Weltkarte auf, ohne sich die Finger am Kartenständer
zu klemmen? Ganz einfach: entweder man läßt die Karte weg und benutzt den
Amiga (Buuuh) oder man hängt die Karte mit dem NEUEN, GROSSEN, PRAKTISCHEN,
PASSGENAUEN SCHLONZ-Kartenaufhängehandschuh auf! (Yeah).
Der Schlonz-Arbeitshandschuh wurde EXTRA FÜR SOLCHE AUFGABEN ENTWICKELT
und bietet OPTIMALEN TRAGEKOMFORT. Wir haben uns in einer Berliner Droge-
riefiliale umgesehen. Ah, da ist ja schon der erste Kunde. Interessiert
bleibt er vor dem Schlonz-Warensortiment stehen. Eine FREUNDLICHE Ver-
käuferin hilft ihm sogleich: "Für welche Aufgabe benötigen Sie denn Ihren
zukünftigen Schlonz-Qualitätshandschuh? Ich habe doch gleich gemerkt, daß

Sie sich SPONTAN für eines der Schlonz-Produkte entschieden haben!" - "Ja, ein Schlonz-Produkt wollte ich schon kaufen, da weiß ich schließlich, daß ich QUALTITÄT bekomme. Aber spontan ist meine Entscheidung nicht, mein NACHBAR hat mir SCHLONZ EMPFOHLEN. Ich suche einen Schlonz für's Karten-aufhängen. Wo ist der nur, führen Sie den etwa nicht?" - "SELBSTVER-STÄNDLICH führen wir das gesamten Schlonz-Sortiment. Der zum Kar-tenaufhängen ist im Moment DER RENNER. Welche Größe haben Sie denn?" - "XL." - "Flupp, da ist er schon. Wollen Sie GRÜN oder ROT?" - "Tjaaa, schwere Frage. Ich glaube, zu den Karten paßt der grüne Schlonz am besten." - "Gute Wahl. Dieser Schlonz wird ihnen noch SEHR LANGE Freude be-reiten. Auf Wiedersehen!" Für die User, denen die 2,5m x 1.7m - Karten zu unübersichtlich sind, gibt's aber zum Glück noch andere Alternativen zu unserem Schlonz-Schrott. Einfach Worldtime starten. Als kleiner Gag mit GROSSER Wirkung zeigt der Amiga auch noch die Uhrzeiten in den entlegensten Städten an.

```
  Reihenfolge/Order: "Random" zeigt die Städte nach dem Zufallsprinzip an,
                     "One after another" in alphabetischer Reihenfolge.
        Warten/Wait: bestimmt, wie lange eine Stadt angezeigt wird (in Se-
                     kunden).
Stunden, Minuten,
  Hours und Minutes: verändert die Zeitverschiebung gebenüber der MEZ.
                     Für Deutschland müßt Ihr 0 und 0 einstellen, weil
                     sich Deutschland voll in Mitteleuropa befindet.
   Sprache/Language: Hiermit kann man das Datumsformat umschalten.
                     Für Deutschland müßt Ihr "Deutsch" einstellen.

Fehlermeldungen: siehe AMOS-Errors.

CPU-Belastung: Niedrig.  Speicherverbrauch:290 kB.   Programmgröße:51 kB.
```

1.23 Fehler müssen sein.

Jeder, der auch nur ein noch so kleines Programm in einer nicht-BASIC-Sprache geschrieben hat, weiß, was beim Programmlauf alles falsch gehen kann. Praktisch jede Funktion des Betriebssystems liefert einen Rück-gabewert, der dann eventuell signalisiert, daß kein Ergebnis vorliegt. Und ein korrektes Programm soll schließlich mehr ausgeben als "Fehler: Programmabbruch."

Und bei Madhouse können zusätzlich die Blanker einen Fehler zurücklie-fern, der dann von Madhouse angezeigt werden muß.

Tritt ein Fehler bei einem Blanker auf, und er wurde im BlankerPrefs-Fenster gestartet, zeigt Madhouse den Fehler unten im Blanker-Prefs-Fenster an, welches dafür "ausgeklappt" wird.

Stellt aber der Blanker einen Fehler fest, nachdem er durch Madhouse gestartet wurde (nach Ablauf der Zeitspanne), merkt sich Madhouse erst den Fehler, und ruft ggf. einen anderen Blanker auf. Ist keiner der ausgewählten mehr übrig (oder benötigen die verbliebenen im Moment zuviel CPU-Performance, siehe Advanced Options Fenster, 4.) wird nur ein schwarzer Bildschirm angezeigt. Nach Abbruch eines Blankers / des schwarzen Bildschirms mit Maus oder Tastatur und evtl. Eingabe des Pass-worts wird dem erstaunten Benutzer die Madhouse-ErrorList präsentiert. Hier werden alle aufgetretenen Fehler gelistet. Verlassen wird die Liste

mit dem Schließgadget.
  Zu den Blanker-spezifischen Fehlern, die nichts mit so profanen Dingen wie "Out of memory", "Couldn't open Screen/Window" (beide Fehler sind das Resultat von zuwenig Speicher) zu tun haben, findet sich die Erläuterung in Alle Blanker in der Übersicht.

  In diesem Kapitel möchte ich aber noch die Fehler von Madhouse darlegen, die nicht sofort verständlich sind (also lasse ich die "Cannot open Window"- und "Couldn't write file"-Fehler weg.

  Wenn der MadhouseConfigEd oder Madhouse selbst einen Fehler anzeigt, tut es das mit einem Fenster auf der Workbench, welches ein Abort-Gadget besitzt. Um das gleich klarzustellen: Dieses Fenster ist ein Easy-Request! Wer mit der Bedienung des Gadgets nicht ganz klarkommt, der darf sich ruhig fragen, wie wohl ein Difficult-Request funktioniert...
  In beiden Fällen wird der Fehlertext angezeigt. Uuuund jetzt geht's los:

  Bitte einen KOMPLETTEN Pfad auswählen,...
  Please select a COMPLETE path,...
Für bestimmte Spezialitäten (Buffering) benötigt Madhouse den Namen der Diskette/Festplattenpartition, auf der sich die Blanker befinden. Das liegt ganz einfach daran, daß man nicht abfragen kann, ob z.B. "DF0:" oder "//Blankers" gerade im Laufwerk liegt... Das geht nur mit Pfaden wie "MadhouseDisk:" oder "MadhouseDisk:Blankers". Also bitte etwas entsprechendes im Path-Gadget des Hauptfensters eintragen.

  Die "gadget"-Datei von xy enthält keinen BlankerInfo-Chunk!
  BlankerInfo-Chunk does not exist!
Die Datei "gadget" des betreffenden Blankers (in seinem Verzeichnis) ist fehlerhaft; die BlankerInfo-Informationen fehlen. Madhouse kann nicht lesen, ob der Blanker viel CPU-Performance benutzt oder nicht.

  Unbekanntes Makro: "xx"!
  Unknown macro: xx
In der "gadget"-Datei des aufgerufenen Blankers befindet sich ein Makro namens xx. Dieses ist leider nicht existent.

  Couldn't get a lock on this directory!
Vermutlich wurde im Path-Gadget oder mit dem FileRequester ein falscher Pfad eingegeben. Oder ein Diskettenkopierer hat die gesamte Disk gesperrt, so daß Madhouse nicht darauf zugreifen kann.

  Konnte das Verzeichnis "RAM:Madhouse_Storage" nicht anlegen.
  Couldn't create the subdirectory "Ram:..."
Für verschiedene Zwecke wird ein Unterverzeichnis "Madhouse_Storage" in der RAM: - Disk angelegt. Das geht aber nur, wenn 1. Ram: überhaupt gemounted ist (Normalzustand) und wenn 2. Madhouse nicht bereits gestartet wurde. So wird auch eine Doppelt-Inkarnation von Madhouse verhindert.

  Madhouse wurde doppelt gestart! Soll es beendet werden?
  You started Madhouse the second time. Do you want to remove it?
So ein Schlingel: Madhouse doppelt gestartet und gedacht, jetzt kommt alles durcheinander, oder was? Nur, damit Ihr im Bilde seid:
- Das Madhouse, daß Ihr eben gestartet habt, hat sich sowieso schon beendet.
- Das Madhouse, daß Ihr vorhin gestartet habt, hat hinterhältigerweise davon erfahren. Und nun? Naja, der Doppelstart ist eine Möglichkeit, Madhouse loszuwerden. Einfach den Requester mit "Remove Madhouse" beantwor-

ten, und tschüß. "Do nothing" sieht den Doppelstart eher als einen Unfall an und macht so weiter wie zuvor.

Die amos.library wurde nicht gefunden...
No amos.library

Selbstredend benötigt Madhouse nicht die amos.library, um Himmels Willen! Madhouse wollte die amos.library von LIBS:amos.library in sein Storage-Verzeichnis auf der Ram:-Disk kopieren. Damit die Blanker vollen Zugriff darauf haben. Die Datei LIBS:amos.library war aber nicht vorhanden, was zur Folge hat, daß der Buffering-Modus ausgeschaltet wird. Jetzt solltet Ihr lieber keine AMOS-Blanker (die von Aicke, siehe Blanker-Teil) starten, denn die werden die amos-lib erst recht nicht finden können, und dann ist tschüß. Madhouse wurde offenbar falsch installiert.

Konnte die "gadget"-Datei des Blankers "xy" nicht laden!
Couldn't load the "gadget"-file!

Die gadget-Datei wird benötigt, um das Fenster zu öffnen. Sie ist nicht vorhanden. Diese Datei muß ebenfalls vorhanden sein, damit Madhouse beim Einlesen des Blankers-Verzeichnisses Informationen zu diesem Blanker erhalten kann.

Probleme mit dem Paßwort
Problems with your password

Dieser Fehler erscheint, falls es jemandem nicht möglich war, sein Paßwort bei der Sicherungsabfrage einzutippen. Das kann jetzt verschiedene Gründe haben...
Angenommen, Ihr habt es vergessen (nach 2 Sekunden...), dann sollte ein anderes gewählt werden. War der Eingabebildschirm auf einmal weg? Tjaa, wer hat denn da <Ctrl>, <Alt>, <Caps Lock> oder ähnliches gedrückt? Aus Sicherheitsgründen wird bei diesen Qualifiern abgebrochen (stimmt wirklich; damit man NICHT einen Trick anwenden kann, den ich aber trotzdem nicht verrate - doppelt sicher!). Die Ziffern und Sonderzeichen, die nicht mit o.a. Zeichen erreicht werden müssen, gehen aber.

Madhouse benötigt einen Stack von mindestens 4.096 Bytes! ...
Need a stack minimum of 4.096!

Der Stack ist ein Bereich im Speicher des Amiga, der für jedes Programm bei seinem Start von AmigaDOS reserviert wird. Das gestartete Programm hat keinen Einfluß auf dessen Größe, weil er kurz vor dem Start des Programms eingerichtet wird.
Die benötigte Größe von 4096 Bytes ist bei Amiga-Programmen eigentlich der Standard, deshalb sollte man Madhouse von überall her aufrufen können. Falls dieser Fehler trotzdem auftritt, erkläre ich Euch nun, wie man die verschiedenen Programme doch dazu überreden kann, Madhouse korrekt zu starten.
Die Workench merkt sich die Stack-Größe der Programme in ihrem Icon. Also Madhouse 1x anklicken und im "Icon"- bzw. "Piktogramm"- Menü der Workbench "Information" auswählen. Ein Fenster öffnet sich, in dem sich auch ein Number-Gadget namens "Stack:" befindet. Hier 4096 eintragen und mit "Save" bzw. "Speichern" verlassen.
Die Shell startet ihre Programme immer mit dem aktuellen Stack, der für jedes Shell-Fenster variieren kann. Also Shell öffnen, "stack 4096" eingeben und Madhouse mit z.B. "run Work:Madhouse/Madhouse" starten.
Wer Madhouse mit dem Toolmanager startet, der findet im Ändere Programm-Objekt-Fenster auch hier ein Stack-Gadget.

Bug #1 in program!

Bitte schreibt mir, wenn dieser Fehler auftritt!!

  Auf deiner 1.3-Schüssel läuft Madhouse nicht!
-Schluck!- ist dies etwa ein 1.x-Amiga?!
Ey Leute, es gibt doch wirklich für JEDES Amiga-Modell eine Kickstart-Aufrüstung auf 2.0!! 93% aller PD-Software, die ich benutze, braucht OS 2.0! Und jedes 3. kommerzielle Programm auch! Wie kann man es auf dem Amiga ohne OS 2.0 aushalten, wenn man nicht nur spielt? OS 2.0 ist soooo toll und vergleichsweise billig!
Für pure Einsteiger, die vor ein paar Monaten ihren Amiga gekauft haben: OS 2.0 heißt das Betriebssystem, und Euch hat man einen Amiga angedreht, der nicht mal auf dem Stand von vor 5 Jahren ist. Besorgt Euch also das Upgrade. OS hat nichts mit IBM zu tun :-> sondern ist von Commodore.
Wer meint, OS 2.0 zu haben und bekommt diesen Fehler, der hat ein echtes Problem...
Wer gerade losrennen will, um OS 2.0 zu kaufen, der kann sich auch gleich das aktuellere 3.1 besorgen.

  Die "gadget"-Datei dieses Blankers kann nicht gelesen werden. Dazu wird eine neuere Madhouse-Version benötigt.
  Cannot read the "gadget"-file of this blanker! (You need a newer Version of Madhouse)
Die erste Zeile der "gadget"-Datei dieses Blankers hat nicht den Inhalt "Madhouse, Blankerwindow-Def V1". Entweder Schreibfehler oder Eure Madhouse-Version ist überholt. (Ich denke nicht, daß ich am Dateiaufbau mal was ändern werde. Wenn neue Chunks hinzukommen, ist das noch lange kein Grund, daß die alten Files inkompatibel werden müssen - dazu habe ich mir das ja extra so ausgedacht.)

  File mismatch: Chunk "Dimensions" must be the first chunk!
Dieser Chunk muß der vor allen anderen Chunks stehen!

  BlankerPrefs-window too big for this screen!
Die Workbench ist zu klein für diese Prefs-Fenster. Oder Schreibfehler in der "gadget"-Datei dieses Blankers (Stichwort "CHUNK:WINDOW").

  Enter a higher value in "CHUNK:DIMENSIONS" - Gadgets!
Nicht nachdenken, ... machen!

  Der Chunk MUI-PREFSORDER ist falsch.
  Wrong Statements in CHUNK:MUI-PREFSORDER.
Im Chunk "MUI-PREFSORDER" stehen falsche Sachen drin.

  Konnte den Config-Editor nicht erreichen.
  Couldn’t access the Config-Editor.
Madhouse wollte den MadhouseConfigEd starten, also das Programm, mit dem man die Madhouse-Einstellungen ändern kann. Damit Madhouse den ConfigEd starten kann, muß zunächst Madhouse wieder beendet werden, da die Pfadangabe zum ConfigEd nur beim Programmstart ausgelesen wird.
Um Madhouse zu beenden, muß man zunächst wissen, ob es überhaupt läuft.
- Wenn Madhouse gerade erst seit wenigen Sekunden läuft und man nicht "Madhouse" im Tools/Hilfsmittel-Menü aufgerufen hat, und diese Meldung erscheint, dann muß sich Madhouse sowieso sofort nach Bestätigung des Requesters beenden, da es ohne Einstellungen keine Chance hat.
- Wurde diese Meldung durch die Auswahl von "Madhouse" im Tools/Hilfsmittelmenü "provoziert", dann lagen offenbar schon die korrekten Ein-

stellungen vor. Madhouse muß nun beendet werden. Dies kann über das Exchange-Programm oder durch den erneuten Programmstart und anschließendem "Remove Madhouse" geschehen.

Ob sich Madhouse nach diesem Requester beenden muß, ist auch dem Requester-Text zu entnehmen, der auf diese Situation angepaßt wird. Ob Madhouse gerade läuft oder nicht, läßt sich jederzeit einfach durch einen Blick ins Tools-Menü feststellen.

Nun sollte man im ToolType "CONFIGED" von Madhouse den Pfad und Namen des ConfigEds eintragen.

Will man Madhouse nicht mit der Workbench, sondern von der Shell starten, so wird der ConfigEd im Programmverzeichnis von Madhouse angenommen. In diesem Fall können die ToolTypes meines Wissens nach nicht ausgelesen werden.

Der MadhouseConfigEd kann nur von Madhouse selbst gestartet werden. Klare Sache: der ConfigEd läßt sich nur über das Hilfsmittel- / Tools-Menü der Workbench starten, nicht direkt durch den User. Diese Einschränkung mußte sein, da Madhouse und der ConfigEd beide das Madhouse_Storage-Verzeichnis in der Ram-Disk benutzen. Wenn beide gleichzeitig laufen oder Madhouse nicht gestartet wurde, kann es zu Kollisionen kommen.

1.24 Die AMOS-Pro-Fehlernummern und ihre Bedeutung

Dieses Kapitel beschreibt die Fehlernummern, die von den AMOS-Blankern angezeigt werden, wenn etwas nicht klappte. (Dieses Kapitel kann nur aus den Blanker-Beschreibungen der AMOS-Blanker abgerufen werden, deshalb könnt Ihr Euch jetzt sicher sein, daß der Blanker ein AMOS-Blanker ist.)

Zuerst muß noch gesagt werden, daß ein bestimmter Fehler nicht als Nummer, sondern als Klartext in der Errorlist / im BlankerPrefs-Fenster angezeigt wird. Er lautet "Out of memory." und signalisiert, daß nicht genügend freier Speicher zur Verfügung stand, um den Blanker zu starten.

In den anderen AMOS-Fehlern steht etwas von einer AMOSPro-Errornumber, und diese Numbers bekommen gleich eine Bedeutung. Da jedoch das AMOS-Pro-Handbuch 13 Seiten über dieses Thema enthält, zähle ich hier nur die Fehler auf, die sich von Euch beheben lassen.

Nummer	Beschreibung
30	Falsches IFF-Format. Vielleicht konnte man diesem Blanker den Dateinamen eines Bildes übergeben, das er laden sollte. Diese Bilddatei ist jedoch in Wirklichkeit eine Datei anderen Typs.
187	Kann die med.library nicht laden. Dieser Blanker wollte wohl Musik abspielen, und ohne med.library geht das (fast) nicht. Keiner von Aickes AMOS-Blankern benötigt diese Library, weshalb sie auch nicht in Madhouse enthalten ist.
32	Das IFF-Bild paßt nicht auf den Bildschirm. Eigentlich ein Fehler wie Nummer 30, an Eurem Bild stimmt etwas nicht.
86	Gerät (=Diskette) nicht verfügbar. Auch diese Fehlermeldung kann nur von einem Blanker kommen, der die Eingabe eines Dateinamens erlaubt. Ohne Buffering darf man hier den gesamten Pfad eingeben, weil dann die Diskette (höchstwahrscheinlich die Festplatte) immer "eingelegt" sein muß. Mit Buffering muß man seine Datei in das Un-

terverzeichnis des Blankers kopieren und nur den Dateinamen angeben. Wenn das so geschehen ist, arbeitet der Blanker nicht nach der Madhouse-Konvention aus Das Blanker-Unterverzeichnis.

101 Diskettenfehler. Hier ist wohl die Diskette/Festplatte mehr oder weniger kaputt.

88 Diskette voll. Dieser Fehler kann eigentlich nur entstehen, wenn ein AMOS-Blanker gerade seine Fehlermeldung in RAM:Madhouse_Storage ablegen wollte. Der Speicher ist wohl so knapp, daß selbst eine kleine Datei nicht mehr in den Speicher paßt. Vermutlich ist der Speicher dann auch so knapp, daß Madhouse diesen Fehler gar nicht anzeigen kann. Und dieser Fehler kann ja auch nur über die RAM:-Disk an Madhouse übermittelt werden. Also ist DAS hier der Fehler und nicht irgendein anderer, der diesen Fehler auslöste. Ein Fehler, der nur beim Schreiben eines anderen Fehlers auftritt, muß hier eigentlich nicht mehr erwähnt werden. So einen Fehler darf man ja eigentlich noch nicht einmal abfragen, oder habt Ihr eine Ahnung, was passiert, wenn beim Schreiben des Fehlers ein Schreibfehler auftritt, der dann auch geschrieben wird und sicherlich seinerseits den nächsten Schreibfehler verursacht?!
Ach, vergeßt es.
Tatsächlich gibt es auch hier den Sonderfall 48d (siehe Problemkapitel) mit der statischen Ram-Disk als RAM:. Falls Eurer Stat-Ram also gerade das letzte Byte allegegangen ist, (bei den Typen, bei denen man den max. Speicherverbrauch angeben kann/muß), dann ist ist dieser Fehler auch hier möglich.

81 Datei nicht gefunden. Entweder habt Ihr eine Datei gelöscht, die der Blanker unbedingt benötigt, oder Ihr habt Euch bei einem eigenen Dateinamen vertippt.

44 Font (Schriftfamilie) nicht verfügbar.

31 IFF-Kompressionsalgorithmus unbekannt. Abhilfe: Die eigene IFF-Datei in ein Malprogramm laden und gleich wieder unter demselben Namen abspeichern. Diesen Vorgang mit allen zur Verfügung stehenden Malprogrammen und Bildbearbeitungen wiederholen, bis der Fehler nicht mehr auftritt.

82 Buchstabensalat im Dateinamen. Da hat sich einer mächtig vertippt, denn dieser Dateiname paßt noch nicht einmal von Format her zu AmigaDOS. (Beispiel: "Work:Bla:Hallo")

93 Keine Diskette im Laufwerk. Siehe Fehler 86.

92 Keine AmigaDOS-Diskette. Wirkung wie 101.

186 Kein Tracker-Modul. Hier konnte man wohl den Dateinamen eines Noise-Tracker-kompatiblen Musikmoduls angeben. Diese Datei ist nicht kompatibel.

0 Kein freier Raum im Stack mehr. Abhilfe: in einem Editor (z.B. c:ed) die "gadget"-Datei laden, die im Verzeichnis des Blankers steht, der den Fehler verursacht hat. In der Datei nach einem Text "CHUNK:BLANKERINFO" suchen. Vier Zeilen darunter befindet sich eine große Zahl, z.B. 5000. Daraus jetzt z.B. 9000 oder mehr machen. Die veränderte Datei abspeichern und das Madhouse-Hauptfenster öffnen. Jetzt in das Path-Gadget klicken und <Return> drücken. Nun sollte der Blanker funktionieren, wenn nicht, die Zahl noch größer wählen.
Ein Versuch mit absichtlich zu niedrigem Stack zeigte, daß der Blanker zwar tolle Probleme (recoverable Alert, Mauszeiger bleibt stehen) verursacht, aber nicht diesen Fehler erzeugt. Trotzdem ist dann die Vorgehensweise richtig, um einen eigenen Blanker zum Laufen zu bringen.

1.25 Alles über die Buffering-Option!

Da hängt nun so ein kleiner Haken im Advanced-(Options-Fenster) rum... was kann der schon machen? Und wer ihn einfach ausprobiert, wird auch kaum etwas merken. Doch der unscheinbare Haken ist ein starker Vorteil für jeden, der die Blanker auf eine Diskette installiert hat.

Denn man kann eine Diskette bekanntlich aus dem Laufwerk nehmen. Wenn dann ein Programm darauf zugreift, erscheint ein Requester, der Euch freundlich auffordert, diese Disk einzulegen. Währenddessen ist das aufrufende Programm praktisch "leblos".

Das Dilemma: Den meisten Programmen ist es egal, ob sie ihre Disk nun gleich oder ein paar Stündchen später kriegen. Madhouse nicht, denn wenn geblankt werden muß, dann gleich.

Also wird sicherheitshalber ein Blanker in die Ram:-Disk kopiert, wo er auch erreichbar ist, wenn die Disk nicht da ist. Das macht Madhouse nur bei eingeschaltetem Buffering. Ist die Disk aber doch da, kann ein Blanker von dort nachgeladen werden, falls der Blanker im Speicher schon benutzt wurde. Wenn die Disk aber nicht da ist, geblankt wurde, der User abbricht, evtl. ein Paßwort eingegeben wird, dann ggf. die Fehlermeldungen der Blanker in der Errorlist vom User bestätigt wurden, die Disk immer noch nicht da ist, im Advanced-Options-Fenster "Nach Disk fragen / Ask for Disk" gewählt wurde UND wenn der Speicher dazu reicht, wird noch ein kleines Fenster auf der Workbench oder dem aktuellen PublicScreen geöffnet, welches auf diesen Zustand aufmerksam macht...

Zusätzlich wird die amos.library auf die RAM:-Disk kopiert, damit die AMOS-Blanker eine haben, wenn sie sie brauchen.

Wenn ein Blanker es erlaubt, einen Dateinamen einzustellen (ist momentan noch nicht der Fall) dann müssen Disketten-User diese Datei in das Blankerunterverzeichnis kopieren, weil sie nur dort ggf. in die Ram:-Disk gerettet wird. Im Stringgadget dieses BlankerPrefs-Fensters muß dann nur der Dateiname eingetragen werden, ohne Pfad. Bei einer Eingabe von FileXY sucht der Blanker also nach RAM:Madhouse_Storage/FileXY oder Work:Madhouse/Blankers/NiceBlanker/FileXY, je nachdem, von wo er gestartet wurde.

Aber das ist - wie gesagt - momentan noch unwichtig, da im Moment keine Blanker existieren, die nach Dateinamen fragen.

1.26 Der Referenz-Teil

Dieser Teil der Madhouse-Anleitung dient dem Nachschlagen der Gadgets einzelner Fenster.

Ohne MUI:

Das Hauptfenster
Das BlankerPrefs-Fenster
Das Advanced-Options-Fenster
Die Error-List
Das AskForDisk-Fenster

Mit MUI:

Tooltypes

1.27 Referenz: Tooltypes - CONFIGED

CONFIGED

Der komplette Pfad des MadhouseConfigEd muß direkt hinter dem "=" angegeben werden. Beispiel:

CONFIGED=Work:Madhouse/MadhouseConfigEd

Neben dem "=" dürfen keine Leerzeichen stehen.

1.28 Referenz: Tooltypes - QUIETQUIT

QUIETQUIT

Wenn dieser Tooltype eingeschaltet ist, nervt Euch Madhouse nicht mit einem Sicherungsrequester, wenn Ihr es durch erneuten Start beenden wolltet. Wer den Requester haben will, muß diesen Tooltype entfernen oder in Klammern setzen.

1.29 Referenz: Das Hauptfenster

I. Wie man dieses Fenster öffnet.
Um dieses Fenster zu öffnen, muß man zuerst auf die Workbench klicken und dann deren Tools- (Hilfsmittel-) Menü wählen. Darin befindet sich der Eintrag "Madhouse". Nach dessen Answahl öffnet sich dann das Fenster.

II. Die Gadgets
Witzigerweise wird hier jedes Gadget durch ein Gadget repräsentiert.
Bei Anklick Infos.

Entfernen/Remove
Info

III. Wie man die Einstellungen speichert.
Auf "Sichern/Save" klicken.

IV. Wie man das Fenster schließt.
Dazu dienen die Gadgets "Speichern/Save" und "Benutzen/Use". Achtung: solange diese Fenster, das Advanced-Options-Fenster oder die Error-List offen ist, wird Madhouse nach Ablauf der Zeit keinen Blanker starten.

1.30 Referenz - Hauptfenster: Edit

Dieses Gadget beeinflußt nur die Liste. Man kann hier zwischen der Funktion der Liste umschalten. Es gibt zwei Möglichkeiten: "Selection/Auswahl" schaltet die Liste in den Blanker-Auswahl-Modus. Jetzt kann man wählen, welche Blanker gezeigt werden sollen, wenn die Zeit nach einer Schaffensphase um ist.
Ist "Einstellung/Prefs..." aktiv, bewirkt ein Klick in die Blanker-Liste das Öffnen das BlankerPrefs-Fensters für diesen Blanker.
Ist das Gadget schraffiert und damit unbrauchbar gemacht, dann ist ein falscher Pfad eingestellt.

Weitere Infos in Der Workshop, Punkt 5.

1.31 Referenz - Hauptfenster: Das Listengadget

Die Liste ist mit dem Edit-Gadget verbunden. Wenn im Edit-Gadget "Einstellung/Prefs..." eingestellt ist, zeigt die Liste die Namen aller Blanker. Ein Klick auf einen Namen öffnet das BlankerPrefs-Fenster, in dem die Einstellungen für diesen Blanker gesetzt werden können.
Steht das Edit-Gadget jedoch auf "Auswahl/Selection", erscheinen vor den Namen jeweils "» " oder " ". Ein "» " bedeutet, daß dieser Blanker verwendet wird, ein " " markiert einen ausgeschalteten Blanker. Durch Klick auf die Namen wechseln die Blanker zwischen den verschiedenen Betriebszuständen. Es muß mindestens ein Blanker eingeschaltet bleiben.
Nur aus eingeschalteten Blankern wird dann im Lotterie-Verfahren einer zum Blanken ausgewählt.

Ist die Liste schraffiert dargestellt (geht nur unter OS3.0) oder reagiert sie nicht, oder ist sie leer, dann stimmt der Pfad nicht. Bei Path korrigieren.

Es kann sein, daß etwas am Blankers-Verzeichnis geändert wurde und Madhouse noch den alten Verzeichnisinhalt anzeigt. In diesem Fall braucht man nur den Cursor in das Path-Gadget setzen (anklicken) und <Return> zu drücken. Wenn zusätzlich abgespeichert wird, sind die Einstellungen auch beim nächsten Programmstart wieder vorhanden.

Weitere Infos in Der Workshop, Punkt 5.

1.32 Referenz - Hauptfenster: Pfad/Path

Wenn das "Benutzen/Use"-gadget und einige andere unzugänglich sind, wurde ein falscher Pfad eingestellt. In diesem Gadget muß der Pfad eingetragen werden, in dem Madhouse seine Blanker sucht. Normalerweise endet der Pfad mit /blankers, sofern der Name dieses Verzeichnisses nicht geändert wurde. Madhouse erkennt den Pfad übrigends nur als richtig, wenn sich noch die Datei "the_right_drawer" darin befindet.
Der Pfad muß absolut sein, d.h. den Namen der Diskette enthalten.

1.33 Referenz - Hauptfenster: Weitere Optionen/Advanced Options

Madhouse hat mehr Optionen als in ein Hauptfenster passen... Deshalb gibt es noch die Weiteren Optionen/Advanced Options, was nichts mit Chipsets zu tun hat. Nach einem Klick auf den Button öffnet sich ein Fenster mit satten 9 Gadgets zum Einstellen bestimmter Dinge, die Madhouse zu etwas Besonderem machen. Die erkläre ich aber nicht hier, sondern in Die Advanced Options.

1.34 Referenz - Hauptfenster: Zeit/Time

Nachdem die letzte Eingabe schon eine bestimmte Zeit her ist, wird der erste Blanker gestartet. Diese Zeit kann mit dem "Zeit/Time"-Slider eingestellt werden. Die Zeit wird in Sekunden gemessen.

1.35 Referenz - Hauptfenster: Blanker wechseln/Exchange Blankers

Mit diesem Gadget wird bestimmt, ob Madhouse die Blanker mitten im Blanken auswechseln soll. So läßt sich im BlankerPrefs-Fenster mit dem "Dauer/Duration"-Slider für jeden Blanker bestimmen, wie lange er laufen soll. Der Slider wird freigegeben, wenn "Blanker wechseln/Exchange Blanker" aktiviert ist.
Da das Wechseln des Blankers aber nur Sinn macht, wenn auch zwei oder mehr Blanker in der Liste aktiviert sind, ist dieses Gadget auch nur dann verfügbar.

1.36 Referenz - Hauptfenster: Speichern/Save

Schließt das Fenster und sichert die Optionen. Diese Dinge werden dann nach "ENVARC:/ENV:Madhouse.prefs" gespeichert:
- Die Optionen des Hauptfensters bzw. der System-Seite.
- Die Optionen des Advanced-Options-Fensters bzw. der Advanced/Optionen-Seite.
- Die aktuelle Position des Waiting-For-Disk-Fensters; siehe Advanced-Options-Beschreibung.
- Die Duration-Einstellung für jeden Blanker.
- Die Namen und ein/aus-Einstellungen der Blanker.

1.37 Referenz - Hauptfenster: Benutzen/Use

Dieses Gadget schließt das Hauptfenster und läßt Madhouse 'am Leben'.
Die Einstellungen werden übernommen, aber nicht abgespeichert.

1.38 Referenz - Hauptfenster: Entfernen/Remove

Beendet Madhouse.

1.39 Referenz - Hauptfenster: Info

Ein kleines Fenster öffnet sich und zeigt einige Informationen über Madhouse und die Autoren.

1.40 Referenz: Das AskForDisk-Fenster (bei Nach Disk fragen)

Das AskForDisk-Fenster (also das Fenster mit der kleinen Diskette) zeigt an, daß der gepufferte Blanker bereits benutzt wurde und daß Madhouse darauf lauert, die Blankers-Diskette in die Finger zu bekommen.
Ein Klick auf das Schließ-Gadget dieses kleinen Fensters bewirkt nicht nur das Schließen des Fensters, sondern stellt auch alle Schnüffelversuche nach der Diskette ein.

1.41 Reference: The BlankerPrefs-window

I. Wie dieses Fenster geöffnet wird:
Dieses Fenster läßt sich ganz einfach öffnen: Einfach das Hauptfenster öffnen, dann das Edit-gadget auf "Prefs..." stellen und einen Blanker anklicken! MUI-User müssen auf die Blankers-Seite wechseln und dann einen Blanker anklicken.

II. Die Gadgets
Genialerweise ändern sich die Gadgets, je nachdem welcher Blanker angeklickt wurde. Einige sind jedoch immer gleich:

II.1 Das Okay-Gadget
Speichert die Einstellungen unter "blankers/blankername/prefs".
Achtung: Der Wert des Duration-Sliders wird hiermit nicht gespeichert, sondern nur mit dem Save-Button des Hauptfenster!

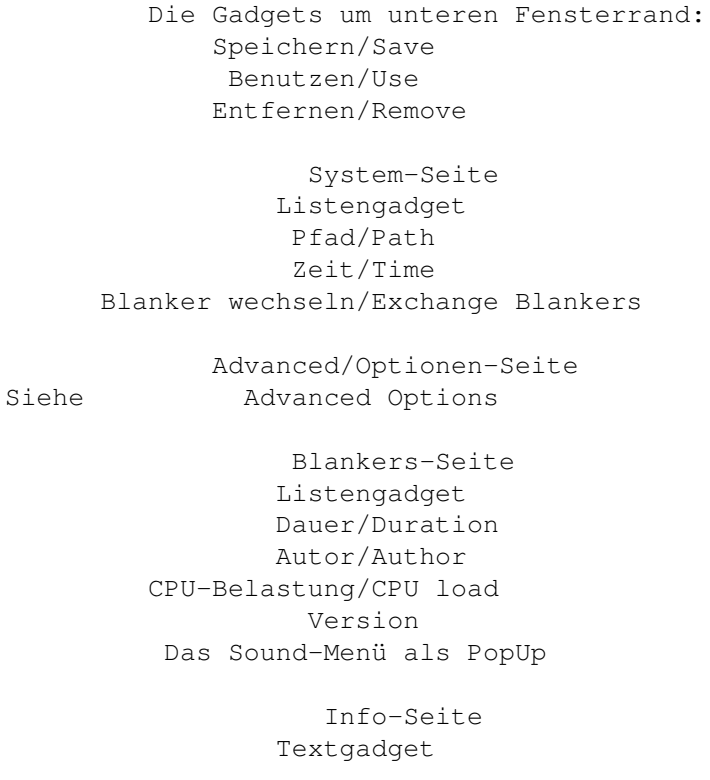
II.2 Das Testen/Test-Gadget
Dieses Gadget starte den Blanker mit den aktuellen Einstellungen.

II.3 Das Abbruch/Cancel-Gadget
Verläßt das BlankerPrefs-Fenster ohne die Einstellungen zu übernehmen.

II.4 Der Dauer/Duration-Slider (MUI: auf der Blankers-Seite vorhanden)

Mit diesem Gadget wird eingestellt, wie lange Madhouse diesen Blanker
zeigt. Da die Blanker aber nur mit aktiviertem "Blanker wechseln/Exchange
Blankers" mittendrin abbrechen und wechseln, macht dieser Slider nur mit
"Blanker wechseln/Change Blanker" Sinn. Andernfalls ist er nicht bedienbar
(schraffiert).

1.42 Referenz: Das MUI-Hauptfenster



1.43 Referenz - Hauptfenster/System: Listengadget

Die Liste links ermöglich die An- und Abwahl der einzelnen Blanker. Nur die
Blanker, die hier ausgewählt werden, werden später von der Zufallsfunktion
herangezogen. Die Auswahl mit der Liste kann umständlich sein, wenn MUI
unpraktisch konfiguriert wurde.
Im MUI-Einstellungsfenster sollte auf der Listen-Seite das Gadget Auswahl
auf "immer" stehen. Zusätzlich wird eine MUI-Multiselect-Liste übersicht-
licher, wenn man auf der Bilder-Seite folgende Zuordnungen vornimmt:
BG Listview -> Stift / System / Hintergrund oder Fremd / irgendein Muster
BG Listview Cursor -> Muster / Scheinstift
BG Listview Selected -> Stift / System / Füllstift
BG Listview Selected+Cursor -> Muster / Schatten/Füll-Raster
Wird die Liste schraffiert dargestellt, dann ist die Pfad/Path-Einstellung
falsch.

1.44 Referenz - Hauptfenster/Blanker: Listengadget

Die Blanker-Liste auf der Blankers-Seite ist für zwei Aufgaben zu gebrauchen: zwischen den Funktionen wird mit der Art des Mausklicks entschieden.

- Einfachklick:
Die Informationen zum angeklickenten Blanker werden in den beiden Textboxen angezeigt, die sich rechts von der Liste befinden.

- Doppelklick:
Das BlankerPrefs-Fenster wird geöffnet. Finden sich in der gadget-Datei des angeklickten Blankers keine Informationen zum Aufbau eines MUI-Fensters, wird eben ein Normales aufgemacht.

1.45 Referenz - Hauptfenster/Blanker: Dauer/Duration

Mit diesem Gadget wird eingestellt, wie lange Madhouse den aktiven Blanker zeigt. Da die Blanker aber nur mit aktiviertem "Blanker wechseln/Exchange Blanker" mittendrin abbrechen und wechseln, macht dieser Slider nur mit "Blanker wechseln/Exchange Blanker" Sinn. Andernfalls ist er nicht bedienbar (schraffiert).

1.46 Referenz - Hauptfenster/Blanker: Autor/Author

In diesem Textfeld wird der Name des Autors angezeigt, der den gerade angewählten Blanker programmiert hat.

1.47 Referenz - Hauptfenster/Blanker: CPU-Belastung/CPU load

Dort ist die Stärke ersichtlich, mit der der Blanker das System belastet. Danach entscheidet Madhouse - falls das Gadget "CPU aktiv/CPU active" auf der Advanced/Optionen-Seite auf "nur einfache Blanker/only simple ones" steht und ein Programm arbeitet - welcher Blanker verwendet werden kann. Siehe auch Advanced Options. Die "Rechenintensivität" des Blankers wird im Feld "CPU-Belastung/CPU load" angezeigt. "niedrig/low" bedeutet eine niedrige Belastung, "mittel bis hoch/medium to high" eine mittlere bis starke Belastung.

1.48 Refernz - Hauptfenster/Blanker: Version

In diesem Textfeld wird die Versionsnummer des vorliegenden Blankers angezeigt.

1.49 Referenz - Hauptfenster/Blanker: Sound

Mit diesem Sound-Menü kann man jedem Blanker einen Sound zuordnen, außerdem ermöglicht es dieses PopUp, sich die Musik-Module anzuhören.

Dazu wird entweder die medplayer.library oder der DeliTracker benutzt, je nachdem, was auf der Advanced/Optionen-Seite eingestellt ist. Falls der DeliTracker benutzt wird, muß der DeliTracker laufen; sonst muß die medplayer.library verfügbar sein und dann darf das Modul auch nur ein MED-Modul sein.

Zunächst muß man in der Liste links einen Blanker auswählen (1x klicken), der "vertont" werden soll. Das Stringgadget neben "Sound" enthält nun das für diesen Blanker eingestellte Modul - also im Moment noch nichts. Das PopUp-Gadget (MUI wird heute mal voll ausgereizt *(:-)) auf der rechten Seite muß auch noch betätigt werden, schon öffnet sich ein liebevoll designtes PopUp-Fenster. Dieses PopUp besteht aus mehreren Elementen:

Hinzufügen... / Add...
Dieser Button muß zunächst betätigt werden, um ein Sound-Modul in die Liste zu bekommen

Löschen / Del
Hiermit wird ein Modul komplett aus Madhouse gestrichen. Jeder Blanker, bei dem dieses Modul eingestellt war, hat nun keine Sound-Einstellung mehr, und das Modul wird aus der Liste entfernt.

Tapedeck-Play-Pfeil
Nun, es soll ja Leute geben, die sich die Bedienungsanleitungen von Radiorekordern, Tapedecks, CD-Playern und Videorekordern durchlesen. Die haben jetzt keine Chance, weil ich nicht erklären werde, was dieser Button macht. Notfalls schaut halt in die Anleitung vom Auto-Radio, Euch ist ja eh' nicht zu helfen.

Tapedeck-Stop-Quadrat
Mit diesem Knopf wird das Musik-Modul wieder gestoppt. %·)

Liste
Ach ja, die Liste: die hat eigentlich gar keine Bedeutung. Die ist nur da, weil die Leute sowas erwarten, wenn sie ein PopUp öffnen. Anstatt des Sound-Menüs hätte es nämlich auch ein simpler File-Requester getan, aber wenn man nun mal gerne programmiert...

Zugegeben, man könnte das Sound-Modul auch direkt in das Stringgadget eintragen. (Der Hypermegasupergeheimtip.)

Übrigens wird die Liste selbst nie abgespeichert. Falls also ein Modul hinzugefügt, aber nicht benutzt wird, befindet es sich beim nächsten Start des MadhouseConfigEd nicht mehr in der Liste. Die Liste wird vielmehr am Programmstart des ConfigEd anhand der eingestellten Module erstellt.

Einen interessanten Tip von Aicke möchte ich auch nicht verschweigen: wenn Ihr nicht wollt, daß der DeliTracker nach jedem Blanken wieder eines seiner eigenen Module nachlädt (wenn er also ohne Madhouse ruhig sein soll), dann schaltet doch einfach die Option "Play at Start" ab.

Halt, nach dieser sehr sinnvollen Beschreibung des Sound-PopUps noch was Wichtiges: Eigentlich kann man die Sound-Funktion nur mit einer Festplatte benutzen (die sowieso jeder Anwender haben sollte), weil Madhouse immer davon ausgeht, daß das Modul ohne Diskettenwechsel (und damit ohne "Please insert Disk..."-Requester) ladbar ist.
Uneigentlich geht es aber auch ohne, dazu muß man das betreffende Sound-Modul jedes Blankers in sein Unterverzeichnis kopieren. In das Sound-Gadget (nicht ins PopUp, das wird nicht gehen) muß man dann
RAM:Madhouse_Storage/mod.Musikmodul
eintragen, vorausgesetzt "Puffern/Buffering" ist eingeschaltet und das Modul heißt "mod.Musikmodul". Diesen Trick sollte man aber nur mit dem DeliTracker anwenden, weil Madhouse die medplayer.library erst kurz vor dem Abspielen öffnet, was auch ein "Please insert Disk..." hervorrufen kann.
Siehe auch Puffern/Buffering-Option.

1.50 Referenz - Hauptfenster: Info

In diesem Gadget werden einige Informationen über uns angezeigt. Außerdem findet Ihr hier das Madhouse-Logo (was Ihr bestimmt gesucht habt!)

1.51 Für Programmierer: eigene Module schreiben!

Da Madhouse ein offenes System ist, kann man leicht eigene Module hinzufügen. Eure erste Frage ist sicherlich
Kann ich meine Programmiersprache benutzen?

Danach könnt Ihr einen Blanker schreiben und ihn Compilieren.
Wie soll ich den Blanker schreiben?

Jetzt fügt Ihr Euren Blanker in den Madhouse-Verzeichnisbaum ein.
Mein eigenes Verzeichnis!

Dies ist alles was Ihr braucht, um einen einfachen Blanker zu schreiben. Aber ein richtiger Blanker hat schließlich auch Einstellungen, und die verändert der User ja in Eurem zukünftigen BlankerPrefs-Fenster. Madhouse erstellt es nach Euren Wünschen - oder besser gesagt, nach Eurer Datei. Sie heißt "gadget" und muß sich im Verzeichnis des neuen Blankers befinden. Und die wird jetzt erstellt.
Die gadget-Datei.

Und fertig! Wer hiermit Probleme hat, der sollte uns unbedingt schreiben. Schließlich hat jeder Programmierer einmal angefangen und ich hatte keine Versuchsperson, die nur anhand dieser Anleitung einen Blanker schreiben sollte. Gerade weil für einen modularen Screenblanker die verfügbaren Module das Wichtigste sind, werden wir hier jede uns mögliche Unterstützung liefern. Natürlich kennen wir nicht jede Programmiersprache, aber das ist oft auch nicht nötig.

Noch ein paar Worte an Leute, die tatsächlich einen Blanker schreiben und veröffentlichen wollen:
1. Wir haben bestimmte Dinge bewußt nicht gemacht (?). So haben wir bewußt

keinen Lines-Blanker geschrieben, weil der nicht gut aussieht und nicht originell genug ist. Natürlich könnt Ihr veröffentlich, was Ihr wollt, aber es wäre schon, wenn das Niveau von Madhouse erhalten bliebe. Die Idee sollte also irgendwie ausgefallen sein.

2. Vielleicht denkt Ihr, es ist praktisch, Madhouse zusammen mit Eurem Blanker zu veröffentlichen. Tut das bitte nicht, denn Madhouse darf nur unverändert weitergegeben werden. Wenn Ihr Euren Blanker zwischen unseren sehen wollt, könnt ihr ihn uns schicken. Dann ist er in der nächsten Version dabei. In diesem Fall muß er sich dann aber unserer Qualtitätskontrolle unterziehen (das hört sich jetzt arrogant an, aber irgendwie muß man sich ja absichern wenn doch einer mit Lines ankommt). Bitte seid nicht böse, wenn der Blanker mit drei Seiten Verbesserungsvorschlägen zurückkommt.

1.52 Kann ich meine Programmiersprache benutzen?

Ja.

Wenn Ihr jetzt denkt, daß Ihr mit BASIC-mäßigen Dialekten keine Chance habt, dann liegt Ihr falsch. Immerhin wurden viele Blanker in AMOS geschrieben, und das will schon was heißen (ich will hier nicht das Gerücht verbreiten, AMOS wäre nur zu sich selbst kompatibel. Nachher folgt dann in der nächsten Version dieser Anleitung eine Gegendarstellung, die ich dann noch nicht einmal unterschlagen darf... oder so).

OK, wer also meint AMOS benutzen zu müssen, der soll das eben tun.

Um es auf dem Punkt zu bringen, mir fallen nur zwei "Sprachen" ein, mit denen Ihr kein Glück haben werdet. ARexx und AmigaDOS. "Waaaas, letzteres ist eine Programmiersprache?" Streng genommen schon. Noch nie was von Batch-Dateien gehört? Gut so.

ARexx geht auch nicht, weil man da keine Screens öffnen kann. "Und das müßt Ihr mir jetzt einfach glauben", würden die Physik-Lehrer sagen.

ARexx würde es theoretisch schaffen, wenn jemand vorher eine ARexx-Extension-Lib mit neuen Befehlen für Screens und Zeichnen und so in Assembler oder C schreiben würde. Und da hört dann der Sinn irgendwie auf, oder? Tatsächlich existiert eine solche Library, die müßt Ihr jetzt aber schon selbst suchen, und außerdem bräuchtet Ihr dann noch den 250,-DM teuren ARexx-Compiler, und dafür bekommt man ja schon eine richtige Programmiersprache.

Tja, längst habt Ihr es gemerkt. Ich bemühe mich immer, einleuchtend, klar verständlich und für jeden begreiflich zu schreiben. Kurz fassen kann ich mich aber nicht.

Um also wieder auf den Punkt zu kommen: hiermit definiere ich eine Madhouse-Blanker-mögliche Sprache als alle Amiga-Sprachen abzüglich der Sprachen für die es keinen Compiler gibt oder die keine Möglichkeit bereitstellen, Screens oder Fenster zu öffnen.

Für AMOS und GFA-BASIC (und Amiga-BASIC...) gibt es den Compiler extra zu kaufen. Für ARexx auch, aber diesen Punkt hätten wir ja nun abgehakt. BlitzBasic, C, Pascal, Assembler, Oberon, Modula, Amiga-E, und alles was mir sonst einfallen könnte sind sog. Compiler-Sprachen und werden deshalb GARANTIERT mit Compilern ausgeliefert, da der Compiler sozusagen

die Sprache ist, wie sich ein Einsteiger ausdrücken würde.

  Alles klar? AmigaDOS (und ARexx?) sind aus dem Rennen, dem Rest wünsche ich viel Spaß.

1.53 Wie ich meinen Blanker schreibe.

Wie Ihr Euren Blanker schreibt müßt natürlich Ihr entscheiden, aber ich kann Euch sagen, wie er an die Daten von Madhouse herankommt und wie er Madhouse Infos übermitteln muß.
Um die Kommunikation zwischen Blanker und Madhouse "programmiererfreundlich" zu gestalten (schließlich wollen wir hier niemanden zum Umsteigen auf C zwingen...), haben wir uns entschlossen, alles mit Dateien abzuwickeln. Das heißt, daß Madhouse eine bestimmte Datei, die immer denselben Namen und Pfad hat, mit Informationen für den Blanker füllt. Dann ruft es den Blanker auf und macht erst weiter, wenn sich der Blanker beendet hat. Der Blanker liest die Datei und zieht daraus seine Schlüsse - die Datei namens "RAM:Madhouse_Storage/prefs" enthält die Einstellungen des Blankers sowie die Duration-Zeit in Minuten (entspricht dem Duration-Slider im BlankPrefs-Fenster) und einen neuen Pfad, der ihm sagt, wo sich sein Hauptverzeichnis befindet. Diesen Pfad braucht man in den seltensten Fällen, nämlich genau dann, wenn man eine Datei aus dem Verzeichnis nachlädt. Alle numerischen Angaben sind im Kartext geschrieben, z.B. "23" statt "æ". Die Frage "Wieviele Einstellungen hat denn mein Blanker?" solltet Ihr selbst beantworten können... Da ein Programmlisting solche Sachverhalte am besten verdeutlichen kann, gibt es drei Beispielprogramme. Das Erste ist in dieser Datei enthalten und folgt gleich. Es ist sozusagen allgemein gehalten. Außerdem tut der Blanker hier nicht etwas bestimmtes und auch auf die Parameter wollte ich mich nicht festlegen. Die beiden anderen Programme befinden sich im developers-Verzeichnis und sind in C und AMOS geschrieben. Sie enthalten richtige, handfeste Beispiele. Im Unterverzeichnis mit dem AMOS-Beispielblanker befindet sich noch eine Configurations-Datei für die AMOSPro-Compilershell. Diese könnt Ihr einfach laden, schon sind die Einstellungen korrekt. Übrigens sollte man auf die abartigen "Amos lock/unlock"-Kommandos achtgeben.

```
Datei "RAM:Madhouse_Storage/prefs" zum Lesen öffnen.
Ersten eigenen Parameter lesen.
Zweiten eigenen Parameter lesen.
:
:
n-ten eigenen Parameter lesen.
Blank-Dauer in Minuten lesen.
Pfad auf die eigenen Dateien lesen.
Pfad ist z.B. "RAM:Madhouse_Storage" oder "Work:Madhouse/Blankers/MeinBl".

// Wenn die Zeit in Ticks ermittelt wird (DateStamp, der Timer)
Blankdauer = Blankdauer * 300
// Wenn die Zeit in Sekunden ermittelt wird (mit Include <time.h>)
Blankdauer = Blankdauer * 60

Screen öffnen.

Wenn Fehler, dann
```

```
  Datei "RAM:Madhouse_Storage/errors" zum ANHÄNGEN (!) öffnen.
  Den Fehler in eine Zeichenkette kopieren,
  an diese Zeichenkette das ASCII-Zeichen nummer 13 (0x0D) anhängen,
  Fehler in Form EINER kurzen Zeichenkette schreiben.
  Datei schließen.
  Programm beenden.
)


Aktuelle Systemzeit (Ticks) in eine Variable legen.


Für immer
  Blanken
  Blanken
  Blanken
  (was der Blanker halt so in einer Hauptschleife macht...)

  Wenn Maustaste gedrückt oder Tasteneingabe auf der Tastatur dann
    Datei "RAM:Madhouse_Storage/Blankstop" zum Schreiben öffnen.
    Datei schließen.
    Screen schließen.
    Programm beenden.
  )

  Wenn Blanker-Dauer ungleich 0 dann
    Zeit messen.
    Wenn (aktuelle Zeit - Startzeit)  >  Blank-Dauer dann
      Screen schließen.
      Programm beenden.
    )
  )
)


Ein paar Erläuterungen sind sicher nötig. So bedeutet die ) ein ENDIF-
mäßiges Sprachkonstrukt.
Die Umrechnung der Blankdauer (*300 oder *60) muß erfolgen, da Madhouse
die Zeit in Minuten übergibt. Da man aber die Zeit in Sekunden oder Ticks
stoppt und die Startzeit in die Variable legt, muß man die Madhouse-
Blankdauer mit 60 (Stoppeinheit Sekunden) oder mit 300 (Stoppeinheit Ticks)
multiplizieren. Dies wird auch in den beiden Beispielprogrammen veran-
schaulicht: das C-Demo benutzt die Funktion time() und die Struktur time_t
aus <time.h> und arbeitet in Sekunden. Das AMOS-Demo liest den Timer aus,
welcher die Zeit nach dem Einschalten in Ticks beinhaltet.
In die Errors-Datei darf deshalb nur eine Zeile ANGEHANGEN werden, weil
Madhouse die Datei erst auswertet, nachdem evtl. mehrere Blanker liefen.
So wird ein Überschreiben der Datei verhindert.
Außerdem ist noch wichtig, daß jeder Blanker nur EINMAL EINE ZEILE in die
"errors"-Datei schreiben darf, die dann von Madhouse umbrochen wird. Und
noch viel wichtiger ist, daß diese eine Zeile maximal 500 Zeichen lang
sein darf.

Die "prefs"-Datei sieht im Übrigen z.B. so aus:
5
10
$Ein toller Text!
```

```
22
2
RAM:Madhouse_Storage
```
Dann hatte der Blanker selbst vier Einstellungen, die man im Prefs-Fenster wählen kann. Sie werden hier in der Reihenfolge übergeben, in der auch die Gadgets in der "gadget"-Datei definiert sind, doch dazu später.
Hinter die Blanker-Einstellungen hängt Madhouse die Zeit in Minuten, die der Blanker maximal blanken kann. Ist sie null, läuft er tatsächlich bis der Benutzer ihn abbricht. Als "Abbrechen" gilt übrigens nur eine Maus- oder Keyboard-Taste. Strings werden Madhouse-Technisch mit einem $ angeführt. Die Blanker-Einstellungen können alles mögliche sein, z.B.
- der Wert eines Cycle-Gadgets
- der Wert eines Sliders
- der Zustand eines Häkchen-Gadgets (0 oder nicht 0 [0 heißt "aus".])
- u.v.m.

Wie der Blanker das auswertet, bleibt ihm selbst überlassen.
Ein sehr einfacher Blanker hat selbst gar keine Einstellungen und kein Prefs-Fenster. Er liest dann z.B. nur

```
1
Work:Madhouse/Blankers/MeinBlanker
```
und reagiert darauf. Die "gadget"-Datei kann sich solch ein Blanker schenken, dafür muß der Programmierer so eines Blankers aber eine prefs-Datei selbst erzeugen, mit 0 Bytes Länge. (So eine "leere" Datei befindet sich auch im developer-Verzeichnis, unter dem Namen "emptyprefs".)

Das Executable, also die Datei, die hinten beim Compiler rauskommt, darf sich übrigens auf keinen Fall vom CLI abkoppeln. Dies muß auch im AMOSPro-Compiler in der Compiler-Shell eingestellt werden (Option: CLI-Program to run in the Background... No - oder so ähnlich). Übrigens ist es für AMOS-Programmierer schon recht wichtig, die mitgelieferten Compiler-Settings zu benutzen. Aicke und ich hatten damals eine Menge Ärger, als wir erstmals einen AMOS-Blanker in das System integrierten. Den solltet Ihr Euch mit der mitgelieferten Config-Datei ersparen.

1.54 Mein eigenes Verzeichnis!

Daß Madhouse für jeden Blanker ein eigenes Verzeichnis erwartet, sollte nun langsam klar sein. Deshalb solltet Ihr jetzt ein Verzeichnis mit dem Namen des Blankers im Madhouse-Blanker-Verzeichnis-Pfad (Ihr wißt, was ich meine...) anlegen.

Ein Verzeichnis hat im Übrigen nicht nur die Aufgabe, alles beisammen zu halten, User ohne Festplatte können - falls Euer Blanker die Angabe eines Dateinamens erlaubt - diese Datei in das Blankerverzeichnis kopieren. Da bei eingeschaltetem Buffering immer das gesamte Verzeichnis im RAM gehalten wird, wird dann diese Datei auch erreichbar sein.
Der User gibt dann nur den puren Dateinamen an. Der Blanker versucht erst, den Pfad, den Madhouse in die letzte Zeile der Prefs-Datei schreibt + "/" + den Dateinamen zu öffnen, wenn das fehlschlug, dann nur den Dateinamen allein.
Bei eingeschaltetem Buffering geht die erste Variante: aus dem Dateinamen "File" wird dann "RAM:Madhouse_Storage/File". Bei ausgeschaltetem Buffering schlägt diese Variante fehl, denn der User mit Festplatte muß ja seine Datei nicht in das Unterverzeichnis des Blankers kopieren. Er gibt z.B. an: "Work:Pictures/File" und es entsteht

"Ram:Madhouse_Storage/Work:Pictures/File", was sich natürlich nicht öffnen läßt. Dafür funktioniert dann aber Variante 2 (nur der Dateiname): "Work:Pictures/File" müßte sich jetzt öffnen lassen.

Das Verzeichnis kann natürlich auch zusätzliche Dateien beinhalten, die der Blanker dann beim Start immer nachlädt. Wichtig ist aber, daß diese Dateien nicht "errors" oder "stopblank" heißen dürfen. Ebenfalls dürfen keine weiteren Unterverzeichnisse angelegt werden.

Also dann, macht ein Verzeichnis und legt das Kompilat Eures Blanker hinein. Der Blanker muß den Namen "blanker" tragen (ach so).
Außerdem solltet Ihr noch eine leere Prefs-Datei erzeugen, wenn ihr den Blanker zuerst ohne gadget-File und ohne BlankerPrefs-Fenster von Madhouse aus starten wollt. Diese Datei muß dann "prefs" heißen (ach ja) und sich in Eurem Blankers-Verzeichnis befinden. Da es nicht so leicht ist, eine Datei wirklich leer zu machen, könnt Ihr die leere prefs-Datei aus dem developers-Verzeichnis benutzen.
Diese leere Datei könnt Ihr Euch sparen, wenn Ihr jetzt die folgenden beiden Absätze nicht mitmacht und gleich an die gadgets-Datei geht. Danach ruft Ihr einfach das BlankerPrefs-Fenster Eures Blankers auf, bewegt alle Slider und klickt auf Save. Schon habt ihr eine fertige Prefs-Datei.

Jetzt muß Madhouse gestartet werden. Klickt einmal ins "Path"-String-gadget und drückt <Return>, damit Madhouse das Verzeichnis neu einliest. Das BlankerPrefs-Fenster kann natürlich noch nicht geöffnet werden, da die Definitionsdatei dafür nach fehlt. Wenn der Blanker schon so programmiert wurde, daß er eigene Einstellungen liest, müßt Ihr halt selbst eine prefs-Datei schreiben, die NUR DIE EINSTELLUNGEN DES BLANKERS enthält. Wie so etwas auszusehen hat, wißt Ihr ja. Aber soetwas würde ich selbst nicht gleich ausprobieren, besser erst den Blanker anpassen, so daß er keine eigenen Einstellungen lädt.

Bei "Selection" könnt Ihr noch Euren Blanker allein auswählen, auf "Use" klicken und - warten.

1.55 Die gadget-Datei

Da ja mittlerweile auch den Lesern der Commodore-Dokumentation klar ist, was Gadget heißt (Ihr wißt schon, die Dinger auf die Ihr dauernd klickt..), dürfte die Bedeutung der "gadget"-Datei, die eigentlich jeder Blanker in seinem Verzeichnis hat, klar sein. Sie enthält u.a. die Beschreibung, wie das BlankerPrefs-Fenster des jeweiligen Blankers aussieht.
Und damit der Anfang wieder einfach wird, schauen wir uns zunächst eine solche Datei an, die ich zwecks Dateifüllung mal hier eingefügt habe.

(Anm.: Diese Datei ist nicht lokalisiert, d.h. ist nur einsprachig. Zur Erzeugung mehrsprachiger BlankerPrefs-Fenster komme ich später.)

```
Madhouse v1.0, BlankerPrefs-Window

CHUNK:DIMENSIONS
Gadgets 3
Texts 5
Arrays 1
```

```
CHUNK:WINDOW
338,62

CHUNK:DURATION_POS
112,53,150,12


CHUNK:GADGETS

Cycle,112,4,150,12,Mode,TextLeft
1,Done
3,Bouncing Point,Wusel,Random

Slider,112,19,150,12,Wusel lenght,TextLeft
5,SlMin,1,SlMax,20,Done

Checkmark,112,34,26,12,Sound,TextLeft
1,Done

(.... Hier steht noch die MUI-Fensterbeschreibung, die ich weiter unten
erkläre. Sie ist nicht zwingend notwendig.)

CHUNK:BEVELBOXES
2
0,0,330,49
0,49,330,20

CHUNK:BLANKERINFO
Aicke Schulz
1
0
5000
```

Ha! Obwohl Ihr bisher nicht die leiseste Ahnung vom Aufbau der Datei habt, kann jetzt jeder sofort sehen, von welchem Blanker die Datei stammt.
Es tauchen nämlich bestimmte Schriftzüge auf, die der erfahrene (ahem..) Madhouse-User schon gesehen hat.
Na klar, von CrazyPixel stammt diese Datei. Und offensichtlich ist sie auf eine bestimmte Art strukturiert. So ist des öfteren das Wort CHUNK darin zu finden.
Auf fünf Dinge ist zu achten:

- Der Text "Madhouse v1.0, BlankerPrefs-Window" MUSS sich in der ersten Zeile befinden.
- Die Chunks können gemischt werden.
- Der erste Chunk muß der "CHUNK:DIMENSIONS" sein. Hier ist auch leider der etwas invariable Programmierstiel erkennbar, den ich für die Daten dieses Fenster angewandt habe... (mit MUI ist das alles schon etwas genialer programmiert worden, aber egal.)
- Es dürfen keine Leerzeilen innerhalb des Chunks liegen. Auf Recht-, Groß- und Kleinschreibung wird geachtet. Ausnahmen nerven die Regel: weil sonst die Übersicht echt futsch gegangen wäre, sind im CHUNK: - GADGETS doch Leerzeilen möglich, aber auch dort nicht überall...!
- Es dürfen Leerzeilen zwischen den Chunks stehen, ein Chunk (CHUNK:GADGETS) erlaubt auch Leerzeilen zwischen den einzelnen Gadget-Definitionen. (Also zwei Regeln und eine Ausnahme.==)

Als kleinen Überblick will ich noch die Funktionen der einzelnen Chunks erklären, bevor ich weiter unten genau auf die Details eingehe.
Der CHUNK:DIMENSIONS sagt Madhouse, wieviele Gadgets das BlankerPrefs-Fenster haben wird, damit es rechtzeitig den Speicher für die Daten besorgen kann. Die beiden anderen Angaben haben ebenfalls mit der Speicherverwaltung zu tun.
Der CHUNK:WINDOW bestimmt die Ausmaße des BlankerPrefs-Fensters (Breite und Höhe).
Der CHUNK:DURATION_POS erlaubt die Positionierung des Duration-Sliders, der von Madhouse selbst erzeugt wird und der sich in allen BlankerPrefs-Fenstern befindet.
Im CHUNK:GADGETS wird jedes Gadget des BlankerPrefs-Fensters gesondert definiert. Die Buttons Okay, Test und Cancel sowie der Duration-Slider werden hier nicht erstellt. Anhand des CHUNK:GADGETS bastelt Madhouse die Daten zusammen, die vom Betriebssystem zum Öffnen des Fensters benötigt werden.
Der CHUNK:LOCALE muß nicht vorhanden sein, erlaubt es jedoch, mehrsprachige BlankerPrefs-Fenster zu erzeugen. Nur sinnvoll und möglich, wenn der CHUNK:GADGETS und evtl. der CHUNK:MUI-WINDOWLAYOUT entsprechend angepaßt werden. Diese Anpassung wird ebenfalls hier erklärt.
Der CHUNK:BEVELBOXES ist (wie auch der CHUNK:TEXTS) optional, d.h. er muß sich nicht in der gadget-Datei befinden. Der CHUNK:BEVELBOXES ermöglicht, wenn vorhanden, die Erzeugung von plastischen Rahmen im BlankerPrefs-Fenster. Damit lassen sich verschiedene Gadgets prima gruppieren; in den Madhouse-Blankern wird dieser Chunk immer eingesetzt, um den Duration-Slider von den Blanker-Gadgets zu trennen. Mit dem CHUNK:TEXTS lassen sich beliebige Texte ins BlankerPrefs-Fenster einfügen.
Der CHUNK:BLANKERINFO muß dagegen wieder in jeder gadget-Datei enthalten sein. Er wird hauptsächlich nur eingelesen, wenn Madhouse nach einer Eingabe im Path-Gadget des Hauptfensters das gesamte Blankers-Verzeichnis durchwühlt und die dort vorhandenen Blanker in der Liste einträgt. Aus dem CHUNK:BLANKERINFO kann Madhouse ersehen, ob der Blanker rechenintensiv ist (siehe Advanced Options, Punkt 4), und mit wieviel Stack er gestartet werden muß.

Noch ein kleiner Tip: Wer nicht weiß, was Radio-Gadgets u.s.w. sind, der kann sich auch den recht informativen Artikel in der Amiga-Plus 11/93, Seite 95f durchlesen. Hier werden auch die Einsatzgebiete der Gadgets erläutert. Wessen Fenster auch noch total (Commodore-) Standard aussehen sollen, der kann sich noch (aber das ist für diesen Zweck echt übertrieben) den "AMIGA User Interface Style Guide", Addison-Wesley, 206 Seiten, ca. 60 DM ’reinziehen. Dieses Buch ist aber eigentlich nur für die Programmierer gedacht, die ein Programm mit Fenstern schreiben. Madhouse ist auch "Style-Guide"-konform, obwohl ich nie ein Blick in den Interface Style Guide warf. Mit der Zeit kapiert man einfach, daß man die Gadgets nicht wild im Fenster herumstreuen sollte...

Und jetzt folgt die Erklärung der einzelnen Chunks. Mit "*" markierte Chunks müssen nicht auftauchen, können aber. (Optional.)

CHUNK:DIMENSIONS
CHUNK:WINDOW
CHUNK:DURATION_POS
CHUNK:GADGETS
CHUNK:BLANKERINFO

```
                    * CHUNK:BEVELBOXES
                    * CHUNK:TEXTS
                    * CHUNK:LOCALE
```

Bis jetzt habe ich Euch ganz diskret verschwiegen, wo denn nun die Blanker-Prefs-Oberfächen des MUI-Programmteils herkommen. Auf meine Kosten kann nämlich auch jeder Blanker, der in AMOS oder sonstwas geschrieben wurde, eine MUI-Oberfäche haben. Die wird - wie konnte es anders sein - auch über einen Chunk erzeugt.
Man sollte den Text, der da oben steht, bereits verstanden haben um hier weiterzumachen. Ich baue jetzt nämlich ein bischen auf dieses Wissen auf. Außerdem sollten sich auch in jeder fremden gadget-Datei die Informationen zum Aufbau eines normalen Fensters (ohne MUI) finden, sonst kann Euer Blanker nur von Usern der MUI-Version genutzt werden. Umgekehrt können aber die MUI-Angaben fehlen.
Doch MUI-Oberflächen werden nicht einfach so pixelweise in den Raum gestellt, sondern bauen sich aus Beziehungen einfacher Gestaltungsmöglichkeiten auf. Deshalb muß ich jetzt auch etwas weiter ausholen, denn die Beschreibung von MUI bleibt mir wohl nicht erspart. Vorher jedoch noch

```
     Der MUI-Madhouse-Schnelleinstieg für erfahrene MUI-Programmierer.
```

Alle anderen Leute mit MUI-Ambitionen lesen besser

```
              MUI von Null auf Hundert für MUI-Greenhorns.
                        Die MUI-Gruppentypen
                          Die MUI-Gadgets
                      Der CHUNK:MUI-PREFSORDER
```

1.56 CHUNK:DIMENSIONS

Dieser Chunk muß der erste Chunk sein. Er beinhaltet drei Zahlen:

```
CHUNK:DIMENSIONS
Gadgets x
Texts y
Arrays z

x = Die Anzahl der Gadgets in Eurem Fenster. Hier werden nur selbst-
    definierte gezählt, die Duration- und Test-Ok-Cancel-Buttons nicht.
y = Die Anzahl der Texte. Sie ist Anzahl der Gadgets (für den Namen) +
    die Anzahl der gesamten Auswahlmöglichkeiten von Cycle- und Radio-
    gadgets.
    Die "richtigen" Texte des Text-Chunks werden nicht mitgezählt.
    Also einfach die Menge der eigenen Gadgets und die Anzahl der Cycle-
    Einträge addieren, und man hat y.
y = Die Menge der Cycle- und Radio-Gadgets.
```

1.57 CHUNK:WINDOW

```
CHUNK:WINDOW
x,y

x = Die Höhe des BlankerPrefs-Fensters in Pixeln.
y = Die Breite des BlankerPrefs-Fensters in Pixeln.
```

1.58 CHUNK:DURATION_POS

```
CHUNK:DURATION_POS
x,y,w,h

Wie schon mehrmals angesprochen, gehört der Duration-Schieber nicht zu
den selbstdefinierbaren Gadgets. Madhouse erledigt hier die Arbeit.
Da aber ein frei in der Gegend positionierter Slider nicht gut aussieht,
gibt es trotzdem die Möglichkeit, den Platz und die Größe des Duration-
Sliders zu bestimmen:

x = Linke Ecke des Duration-Sliders.
y = Obere Ecke des Duration-Sliders.
w = Breite des Sliders.
h = Höhe des Sliders.
```

1.59 CHUNK:GADGETS

```
Dieser Chunk enthält 1 bis n-mal eine solche Angabe:

Typ,x,y,w,h,Name,Flags
Anzahl der Tags,Tag1,Data1,Tag2,Data2...
[Anzahl der Array-Elemente,Element1,Element2,...]

Vor jeder dieser Zeilenblöcke dürfen beliebig viele Leerzeilen (zur Über-
sichtlichkeit) stehen, jedoch nicht IN einem solchen Block (vgl. auch an-
dere Gadget-Dateien).

Das sieht jetzt sehr schwierig aus, aber man kann ja schließlich bei
den anderen Blankern abgucken... Außerdem werde ich noch Beispiele bringen.

Die erste Zeile ist immer ähnlich:

    Typ = Typ des Gadgets. Hier wird zwischen Stringgadgets (für Zeichen-
          ketten), Sliders (die Schieberegler), Cycles (die 'Blättersym-
          bole'), Integergadgets (für Ganzzahlen), Checkmarks (die Häkchen-
          gadgets) und MX's (auch Radiobuttons genannt, hat nichts mit
          Rundfunk zu tun) unterschieden.
          Je nachdem, was für ein Gadget man erstellen möchte, schreibt
          man hier
          String       für Stringgadgets,
          Checkmark    für Checkmark-gadgets,
          Slider       für Sliders
          Cycle        für Cycles
          Number       für Integer-gadgets oder
```

```
          Radio        für Radio- bzw. MX-gadgets.
          Die anderen Typen sind in Madhouse nicht möglich oder nicht re-
          levant.
x,y,w,h = Die Positionen und Dimensionen des Gadgets. x,y = linke, obere
          Ecke, w = Breite, h = Höhe.
   Name = Text, der z.B. neben dem Gadget stehen soll.
  Flags = Hiermit wird definiert, wo der Text "Name" stehen soll:
          TextLeft     der Text steht links neben dem Gadget,
          TextRight    der Text steht rechts neben dem Gadget,
          TextTop      der Text steht über dem Gadget oder
          TextBottom   der Text steht unter dem Gadget.
```

Die zweite Zeile ist die Tag-Zeile. Denn manche Gadgets benötigen noch weitere Informationen, und die werden mit Tags übergeben. Ein Slider muß z.B. wissen was sein Wertebereich ist.
Die Tag-Zeile beginnt immer mit der Anzahl der Tags, z.B. "1".
Viele Gadgets benötigen so etwas aber nicht, deshalb schreibt man da

```
1,Done
```

Jede Tag-Zeile muß mit einem "Done" aufhören.

Bei einem Slider heißt es

```
5,SlMin,minimum-Wert,SlMax,maximum-Wert,Done
```

Für "minimum-Wert" und "maximum-Wert" muß natürlich etwas nach eigenen Vorstellungen eingesetzt werden.

Auch ein Stringgadget kann Tags besitzen:

```
3,StMaxChars,maximale Anzahl Zeichen,Done
```

Hier wird für "maximale Anzahl Zeichen" ein Wert eingesetzt, der angibt, wieviel Zeichen das Stringgadget maximal aufnehmen darf. Wird mehr als 500 angegeben, oder lautet die Zeile nur

```
1,Done
```

dann wird der Default-Wert von 500 Zeichen benutzt.

Das waren schon alle Gadgets, die Tags benötigen.

Die Schlüsselwörter ("String", "Checkmark", "TextRight", "SlMin", "Done", u.s.w.) können in Egalschreibung geschrieben werden (also Pascal-groß oder C-klein oder ARexx-gemischt oder irgendwie).

Die dritte Zeile ist die Array-Zeile. Cycle-Buttons und Radio-Gadgets muß noch mitgeteilt werden, was als mögliche Auswahl zur Verfügung steht. Achtung: sind die Werte im CHUNK:DIMENSIONS richtig gesetzt? (Man muß die Radio- und Cycle-Gadgets durchzählen und das Ergebnis unter "Arrays" im CHUNK:DIMENSIONS eintragen. Dies steht aber dort erklärt.)
Als Beispiel dient jetzt ein Cycle, welches zwischen vier Farbpaletten

```
umschalten läßt.

Cycle,140,20,200,12,Farbauswahl,TextLeft
1,Done
4,Farbset 1,Farbset 2,Farbset 3,Farbset 4

Die erste Ziffer ist wieder die Anzahl der Auswahlmöglichkeiten.
```

1.60 CHUNK:DURATION_POS

1.61 CHUNK:LOCALE

```
CHUNK:LOCALE
sprache1,sprache2,...

Bei der Lokalisierung von Madhouse stoß ich auf das Problem der Blanker-
Prefs-Fenster, die ja nicht auch noch alle ihren eigenen catalog haben
sollten.
Deshalb müssen - falls dieser CHUNK in der Datei vorkommt - die Chunks
GADGETS und MUI-WINDOWLAYOUT in den Angaben für Gadgettexte jeweils
alle verschiedenen Sprachen beinhalten. Um das nicht zu kompliziert zu
erklären, zeige ich Euch zwei Beispiele:

CHUNK:GADGETS
CYCLE,112,4,170,12,"Mode,Modus",TEXTLEFT
1,DONE

3,"Bouncing Point|Wusel|Random,Springender Punkt|Wusel|Zufall"
SLIDER,112,19,170,12,"Wusel lenght,Wusellänge",TEXTLEFT
5,SLMIN,1,SLMAX,20,DONE

CHECKMARK,112,34,26,12,"Sound,Toneffekt",TEXTLEFT
1,DONE


CHUNK:MUI-WINDOWLAYOUT
ColumnGroup(2),
   Label( "_Mode,_Modus" ),
   Cycle( "m", "Bouncing Point|Wusel|Random,Springender Punkt|Wusel|Zufall" ),
   Label( "Wusel _lenght,Wusel_länge" ),
   Slider( "l", 1, 20 ),
   Label("_Sound,To_neffekt"),
   HGroup,
      CheckMark( "s,n" ),
      HVSpace,
   End,
End,

Diese beiden Chunks aus CrazyPixel/gadget zeigen wohl alle Problemfälle,
die mir spontan einfallen. Da es in der Gadgetdatei keine Texte gibt,
die sich nicht irgendwie auf die Bildschirmausgabe beziehen, sind alle
Textangaben (erkennbar an den "-Zeichen) vom Locale-Chunk betroffen.
Leser, die bis jetzt noch keinen CHUNK:MUI-WINDOWLAYOUT vor Augen hatten,
brauchen sich den ja nicht anzusehen, erst nachdem sie die MUI-Kapitel
```

gelesen haben.
Wie man problemlos erkennen kann, hat jede Textangabe ein , in der Mitte. Dieses ist auch echt wichtig, da die Kommas die Sprachen voneinander trennen. Hier sind die linken Teile (also die 1. Sprache) englisch, und die rechten Teile (2. Sprache) deutsch. Also sieht der passende CHUNK:LOCALE so aus:

```
CHUNK:LOCALE
english,deutsch
```

Weitere Sprachen könnten folgen, dann müssen aber auch alle Texte dementsprechend mehr Kommas haben. Madhouse geht nun folgendermaßen vor:
- Wenn am Programmstart des MadhouseConfigEd bereits festgestellt wurde, daß dies kein OS 2.1-oder-höher-Amiga ist, wird ab sofort "english" als die voreingestellte Sprache angesehen; sonst die Sprache, die man im Locale-Editor der Workbench eingestellt hat (also z.B. "deutsch").
- Bevor nun ein BlankerPrefs-Fenster geöffnet wird, wird die zugehörige gadget-Datei nach dem CHUNK:LOCALE durchsucht (Ihr glaubt gar nicht, was das arme Ding alles mitmachen muß...). Ist der nicht da, werden die Texte innerhalb der Anführungszeichen einfach ins Fenster übernommen, und diese Aufstellung hier hat ein Ende.
  Ist er aber da, wird nachgesehen, ob die Zeile mit den Sprachennamen die eingestellte Sprache (also für OS 2.0 "english") beinhaltet, und wenn ja, nach dem wievielten Komma. Wenn nicht, wird die Sprache verwendet, die zuerst aufgeführt wird (SOLLTE "english" sein).
- Trifft nun Madhouse beim Aufbau der Gadgets auf Texte, also auf etwas das mit " anfängt, so werden einfach dementsprechend viele Kommas übersprungen, je nachdem, an welcher Stelle die ausgewählte Sprache stand. Das Ergebnis wird weitergeleitet.

Wie oben im GADGET-Chunk ersichtlich ist, hat bei Cyclegadgets das Komma eine höhere Priorität als das Trennzeichen |. Es wird also nicht jeder einzelne Cycleeintrag umgesetzt, sondern der gesamte Text. Außerdem sind - für MUI-Programmierer - auch die Tastaturbelegungen als Texte anzusehen. Diese sind also auch lokalisiert, für den Fall, daß ein deutsches Wort keinen Buchstaben des englischen Wortes enthält. Dementsprechend können die Buchstaben in den Gadgettexten anders unterstrichen werden, und die Gadgettasten können unterschiedlich sein, wie bei "_Sound,To_neffekt". Wenn sich jedoch die Gadgettexte oder Tasten nicht unterscheiden, muß nicht extra "_Level,_Level" oder "l,l" geschrieben werden, "_Level" oder "l" reicht aus. Wenn Madhouse nämlich kein Komma findet, wird die Locale einfach nicht beachtet.

Der LOCALE-Chunk selbst muß die Sprachennamen in Kleinschrift enthalten, und zwar in der Sprache auf die sie sich beziehen. Alles klar? Nö.
Was ich meine, ist, daß man français anstatt von französisch schreiben muß.

Hoffentlich ist der Locale-Chunk nun nicht zu einem der schwersten geworden. Es gibt da halt soviele Sonderfälle, OS 2.0 oder drüber, verwendet oder nicht, vorn oder hinten, ...
Aber das Beispiel hat doch alles klar gemacht, oder? Natürlich, Carsten!
Na prima, dann kann ich ja wieder beruhigt einschlafen.

1.62 CHUNK:BLANKERINFO

Dieser Chunk wird nicht für das BlankerPrefs-Fenster benötigt, sondern
er wird von Madhouse während des Verzeichnisbaum-Lesens gelesen.
Er enthält allgemeine Infos zum Blanker. Einige Daten werden nur vom MUI-
Programmteil ausgewertet, aber bitte trotzdem immer alles ausfüllen!

EXTREM WICHTIG: Madhouse schaut sich die Daten in diesem Chunk NUR
beim Lesen des Verzeichnisses (Path-Gadget benutzen) an. Das heißt, daß
eine Änderung an diesen Werten nur dann eine Auswirkung auf Madhouse
und den Blanker hat, wenn das Blankers-Verzeichnis neu gelesen wird.
Bis dies nicht geschieht, arbeitet Madhouse mit den alten CPU-Usage und
Stack-Werten.

```
CHUNK:BLANKERINFO
Name
Version
CPU-usage
Stack
```

```
     Name = Der Name des Autors.
  Version = Die Versionsnummer des Blankers. Fängt ab 1 an und wird dann
            ganzzahlig heraufgezählt. 1.4 ist also falsch; 3, 6, 8008743
            gehen aber.
CPU-Usage = kann 1 oder 0 sein. Wenn "CPU active:" auf "only simple Blan-
            gers" gestellt ist, muß Madhouse wissen, ob der Blanker die CPU
            stark (1) oder fast gar nicht (0) beansprucht. 1/3 Belastung
            würde ich noch zu 1 zählen! Den ganzen CPU-Krams habe ich schon
            in Advanced Options erklärt. Entscheidet
            selbst, ob Ihr neben Eurem Blanker einen Raytracer laufen lassen
            würdet... Die CPU-Belastung kann mit einem entsprechenden Tool
            gemessen werden.
    Stack = hängt von Eurer Programmierweise und dem Compiler ab. Zuerst
            sollte man es mit 5000 probieren und den Blanker gründlich
            austesten. Bei Problemen, die sich in Form von Abstürzen
            äußern sollten, einfach mehr probieren. Eventuell legt der
            Compiler (in C kann man das mit static bestimmen) alle Daten
            des Programms auf den Stack, dann kann man schnell die benö-
            tigte Menge Speicherplatz überschlagen.
            Wenn sich Rekursionen im Programm befinden, sollte der Stack
            ebenfalls großzügig bemessen sein. (Hi Einsteiger: wer nicht
            weiß, was Rekursionen sind, der hat auch keine im Programm!)
```

1.63 CHUNK:BEVELBOXES

Was ist plastisch, dreidimensional, sieht aus wie ein Button und ist
(fast) völlig nutzlos?! - Bevelboxen!
Bevelboxes sind die Umrandungen, die mehrere Gadgets zu einer Gruppe zu-
sammenfassen. Im nicht-MUI-Fenster des Stars-Blankers kann man sie gut er-
kennen. (Kleiner Tip: wer sich die BlankerPrefs-Fenster ansehen will, die
von Madhouse erzeugt werden, wenn MUI nicht vorhanden ist, der kann
- entweder die betreffende gadget-Datei um einen MUI-WINDOWLAYOUT-Chunk
  ärmer machen oder

```
- den ConfigEd mittels Use oder Save beenden, alle anderen MUI-Anwen-
  dungen beenden, mit "avail flush" in der Shell die muimaster.library
  aus dem Speicher räumen und diese Library umbenennen. Dann kann der
  ConfigEd sie nicht mehr finden, wenn er aufgerufen wird, und öffnet von
  nun an auch die normalen BlankerPrefs-Fenster.)

CHUNK:BEVELBOXES
Menge
x,y,w,h
[x,y,w,h]
[x,y,w,h]
[...]

 Menge = Die Anzahl der Bevelboxes.
     x = X-Position der Box.
     y = Y-Position der Box.
     w = Breite der Box.
     h = Höhe der Box.

Hinter dem Chunk-Header folgt die Anzahl der Boxen. Dann dieselbe Anzahl
Zeilen, jede Zeile definiert eine Box.
```

1.64 CHUNK:TEXTS

```
Der Text-Chunk erlaubt das Schreiben in die BlankerPrefs-Fenster. So
können Texte erzeugt werden, die nicht direkt zu einem Gadget gehören
sondern frei in der Gegend positioniert werden können.

CHUNK:TEXTS
Menge
Farbe,x,y,Text
[Farbe,x,y,Text]
[Farbe,x,y,Text]
[...]

 Menge = Die Anzahl der Texte.
 Farbe = Die Farbe des Textes (am besten nur 1 - 3).
     x = X-Position des Textes.
     y = Y-Position des Textes.
  Text = Der Text selbst. Bitte ohne Anführungszeichen.

Ein solcher Text-Chunk ist ebenfalls im Stars-Blanker-"gadget"-file zu
finden. Der Aufbau des Chunks ähnelt dem der Bevelboxes, erst die Anzahl
der Texte und dann genausoviel Zeilen darunter, jede Zeile spezifiziert
einen Text.
```

1.65 Madhouse-MUI für Leute, die sich schon auskennen.

```
In diesem Kapitel werden nur die Unterschiede zwischen dem C-Makro-
Headerfile "libraries/mui.h" und dem Chunk MUI-WINDWLAYOUT besprochen.

Eigentlich sieht so eine Deklaration in der gadget-Datei genauso aus wie
```

im C-Quelltext. (Bitte mal in einer gadget-Datei nachsehen!)

Der Chunk MUI-WINDOWLAYOUT enthält jedoch nur den Auszug einer MUI-Applikationsdeklaration, wo der Fensterinhalt erstellt wird. Madhouse kennt die Gruppentypen HGroup, VGroup sowie ColumnGroup(Anzahl der Spalten).

Die Gadgets werden deklariert, indem man den Gadget-Typ zwischen ein Group,...End, schreibt und in Klammern die nötigen Parameter übergibt (siehe Gadget-Beschreibungen).

Die Gruppen erhalten einen Titel, indem man diesen mit in die Gruppendeklaration schreibt:
VGroup("Laber"), bzw. HGroup("Bla"), oder ColumnGroup("Sülz", 2).

Jede Gruppen- oder Gadgetdeklaration wird in eine Zeile geschrieben, immer von einem Komma gefolgt. Auch die letzte Zeile muß mit einem Komma abgeschlossen werden.

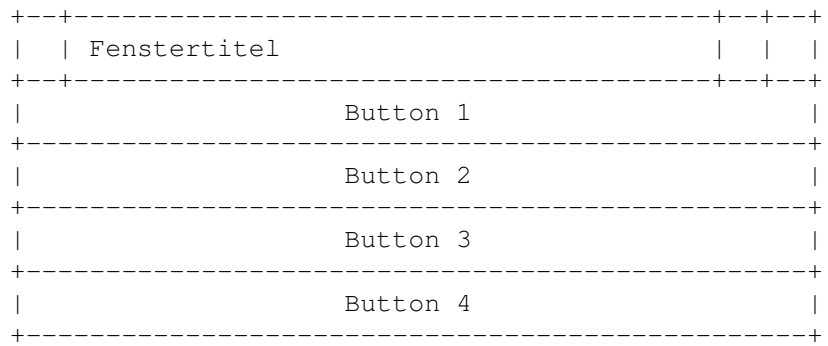
Soll man einen Buchstaben angeben, der zur Tastatursteuerung von Gadgets benutzt werden soll, so muß dieser in Anführungszeichen eingeschlossen sein und nicht in Hochkommas, wie in C sonst üblich.

Diese Beschreibung reicht natürlich noch nicht allein aus, um ein MUI-BlankerPrefs-Fenster zu erzeugen. Deshalb solltet Ihr noch die beiden unteren Kapitel (Gadgets, Gruppen) lesen.

1.66 MUI von Null auf Hundert für MUI-Greenhorns.

Damit Ihr die Erstellung einer MUI-Oberfläche schafft, muß ich Euer Hirn erstmal etwas durchschütteln. Denn aus dem normalen Programmiereralltag seid Ihr absolute und pixelgenaue Positionierung von Gadgets und anderen grafischen Dingen gewohnt. Und jetzt kommt da einer daher, der Euch erklären will, wie man eine komplette Oberfläche ganz ohne Koordinaten aufbaut. Nachdem ich Euch darauf hingewiesen habe, was Euch gleich erwartet, kommt jetzt Psycho-Trick Nummer 1:

Hier seht Ihr eine stilisierte Oberfläche:



Beschreibt einfach verbal, was Ihr seht. Wie sind die vier Knöpfe in diesem Fenster angeordet? Na klar: untereinander. Oder anders ausgedrückt: vertikal.

Mit diesem kleinen Beispiel wird schon einmal deutlich, wie man Gadgets ohne Angabe von Positionen anordnen lassen kann. Man sagt MUI: Die Gadgets kommen untereinander. Außerdem sagt man MUI, was die Gadgets sind, also die üblichen Beschreibungen (Cyclegadget, was sind die Cycleeinträge, was ist der Wertebereich eines Sliders u.s.w.).

Daraus kann man sich ja schon ableiten, was wohl die zweite Anordnungsform sein wird: horizontal, also nebeneinander. Diese Anordnungsformen werde ich demnächst als "Gruppen" bezeichnen. Wir hätten da also horizontale und vertikale Gruppen.

Aber das reicht ja noch lange nicht. Schließlich sind Oberfälche meist viel komplizierter. Deshalb gibt es ein weiteres Feature:
neben normalen Gadgets können die Gruppen noch weitere Gruppen enthalten.

<Lange Denkpause.>

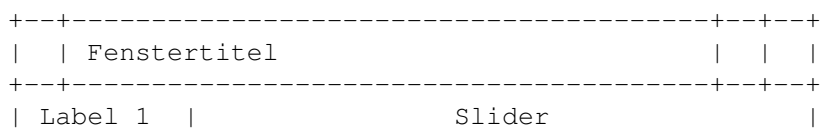
Wie sieht also das oben dargestellte Fenster aus, wenn wir Anstelle von Button 3 eine horizontale Gruppe mit zwei Buttons setzen? Das solltet Ihr Euch jetzt mal auf Papier aufzeichen. Die Lösung ist folgende: .

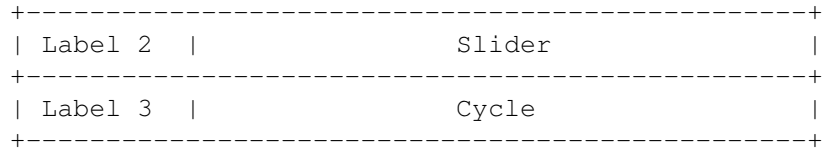
Natürlich können die Untergruppen noch Unteruntergruppen enthalten, so geht das dann weiter bis in alle Ewigkeit. Damit kann man dann schon eine Menge anstellen. In der Regel arbeitet man aber nicht nur mit Buttons, sondern viel mehr mit allen anderen Gadgettypen.
Und die anderen Gadgettypen bestehen nicht nur aus dem Gadget selbst, sondern auch aus dem Text davor, der das Gadget beschreibt. Dieser Text, der z.B. dem "Blanker wechseln/Exchange Blanker"-Checkmark erst seinen Namen gibt, nennt man Label.

Für viele Gadgets kennt Madhouse eine Kurzform, die nicht nur das Gadget selbst, sondern auch das entsprechende Label davor erzeugt. Diese Kurzformen verwendet man aber in der Regel sehr selten.

Der Grund dafür ist, daß alle Gadgets verschiedene Breiten und Ausdehnungs-Möglichkeiten haben, und daß MUI die Gadgets von z.B. einer horizontalen Gruppe immer zur Mitte der Gruppe zentriert. Will man drei Slider übereinander erzeugen, die ja wahrscheinlich mit drei unterschiedlich langen Labels beschriftet werden, so gleicht keine linke Sliderkante der anderen. Man hat den Eindruck, daß die Gadgets wild umherfliegen.

Um die Oberfächen also schöner zu machen, bietet MUI neben den horizontalen und den vertikalen Gruppen noch die Spalten- bzw. die Column-Gruppen an. Wie hat man sich das vorzustellen? Eine Column-Gruppe steht nicht einfach so da und wird von oben nach unten gefüllt, sondern sie hat eine Spaltenanzahl. Eine Column-Gruppe hat eine Struktur wie Karo-Papier, die Kästchen werden von links nach rechts ausgefüllt. Wenn die Spaltenanzahl erreicht ist, erzeugt die Column-Gruppe die nächste Zeile. Hier kommt ein Fenster mit einer Column-Gruppe mit zwei Spalten. Dieser Column-Gruppe wurden folgende Objekte zugeordnet:
ein Label, ein Slider, ein Label, ein Slider, ein Label, ein Cycle-Gadget.





Aber wie würde sowas denn nun eigentlich in der Gadget-Datei aussehen?
Zuerst mal der ungefähre MUI-CHUNK für das erste Beispiel (eine einfache horizontale Gruppe mit vier Buttons):

```
CHUNK:MUI-WINDOWLAYOUT
VGroup,
   Button1,
   Button2,
   Button3,
   Button4,
End,
```

Hier werden einer vertikalen Gruppe (VGroup) vier Buttons zugeordnet. Das End "schließt" diese Gruppe. Hinter jeder Zeile steht ein Komma, trotzdem darf man diese Ausdrücke nicht in einer Zeile zusammenfassen. Die zugeordneten Buttons rückt man ein paar Zeichen ein, damit klar wird, daß die Buttons zu dieser Gruppe gehören. Man sagt auch: "Die Buttons sind Kinder der VGroup."
Mit der VGroup (oder HGroup, falls die Gadgets nebeneinander gehören) und dem End verhält es sich wie mit den Klammern in einem Text, so gehört einer Gruppe immer das End, was in derselben Spalte steht. Madhouse liest das aber nicht an der Spaltennummer ab, sondern macht es anders (was ich hier aber nicht auch noch erkären will). Deshalb muß man nichts einrücken, aber es ist doch bedeutend übersichtlicher.

Und was ist nun mit der Column-Gruppe von eben? Da kommt sie:

```
CHUNK:MUI-WINDOWLAYOUT
ColumnGroup( 2 ),
   Label( "_Label 1" ),
   Slider( "l", 1, 10 ),
   Label( "L_abel 2" ),
   Slider( "a", -5, 20 ),
   Label( "La_bel 3" ),
   Cycle( "b", "Erster Eintrag|Zweiter Eintrag|Dritter Eintrag" ),
End,
```

Das ist schon viel Stoff, oder? Nach dem Chunk-Header "CHUNK:MUI-WINDOWLAYOUT", der Madhouse sagt, daß hier die MUI-Beschreibung losgeht, folgt die Column-Gruppe mit den zwei Spalten, wie in der Skizze oben. Die folgenden Kinder der Gruppe werden von links nach rechts und von oben nach unten in die Spaltengruppe eingefügt. Man sollte darauf achten, daß die Spaltenzahl Teiler der Anzahl der Kinder einer Column-Gruppe ist. Anders ausgedrückt: wir haben sechs Kinder für eine Gruppe mit zwei Spalten. 6 durch 2 geht glatt auf, 7 durch 2 z.B. nicht. Eine Column-Gruppe muß nämlich immer bis in die letzte Spalte aufgefüllt sein. Wenn einem das nicht paßt, behilft man sich anders, aber dazu später.

Oben können wir noch sehen, wie man ein Label erstellt. (Dieses Beispiel könnte man schon so wie es dasteht in eine Gadget-Datei schreiben.)
Also, in die Anführungszeichen kommt der Text, der dann in das betreffende

"Kästchen" der MUI-Gruppe eingefügt wird. In diesem Text kann ein Underscore ("_") vorkommen, der angibt, daß das nachfolgende Zeichen unterstrichen wird.

Das ist auch nötig, damit der Benutzer weiß, mit welcher Taste er ein Gadget steuern kann. MUI ist nämlich extrem tastaturfreundlich.
Der nachfolgenden "Deklaration", ein Slider, wird schon als erstes Argument der Buchstabe übergeben, mit dem der Slider gesteuert wird. Diese Buchstaben sind immer klein zu schreiben. Dann kommen Minimal- und Maximalwert des Sliders, also reicht der erste Slider von 1 bis 10.
Nun wird der nächste Slider beschriftet. Er hat den Buchstaben "a", und sein Wertebreich geht von -5 bis 20.
Jetzt kommt noch das dritte Label, Hotkey "b". Dann kommt etwas Neues: die Deklaration eines Cycle-Gadgets. Wie auch beim Slider wird zuerst der Hotkey angegeben, dann folgen die Einträge des Cycle-Gadgets, getrennt durch "|".

Wo wir gerade bei den Hotkeys, also den Tastatursteuerungsbuchstaben für Gadgets, sind: Diese Hotkeys dürfen nur einmal vorkommen. "Verbotene" Buchstaben sind o, t und c: Damit werden schon die Gadgets am unteren Rand des BlankerPrefs-Fensters gesteuert.

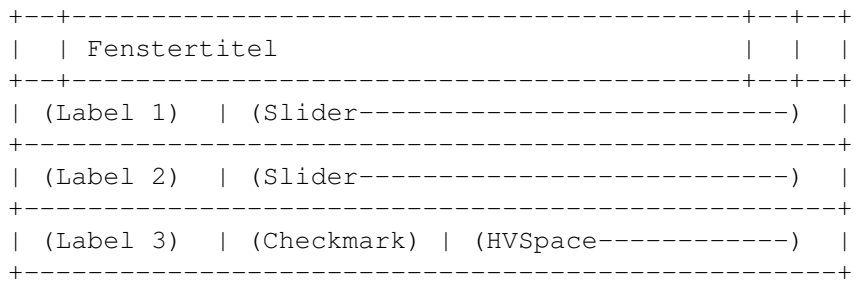
War nicht vorhin davon die Rede, daß man die Gruppen schachteln kann? Wann benötigt man denn sowas?

Sicherlich habt Ihr gemerkt, daß MUI die Gadgets nicht nur automatisch, sondern auch dynamisch anordnet. So ziemlich jedes MUI-Fenster hat ein Größen-Gadget, und die Gadgets passen sich automatisch der neuen Fenstergröße an, nachdem man es benutzt hat.
Viele MUI-Fenster lassen sich nur in die Breite ziehen, die vertikale Größe läßt sich weniger oft verändern (an einigen BlankerPrefs-Fenstern ausprobieren).
Warum eigentlich? Nun, jedes MUI-Objekt (also vor allem die Gadgets, die Gruppen auch, aber von denen reden wir jetzt lieber nicht auch noch) hat eine minimale und maximale Höhe und Breite. Ein Slider, ein Stringgadget, ein Cyclegadget und noch ein paar andere Sachen haben eine feste Höhe. Klar, ein horizontaler Slider kann ja nur breiter werden, würde er höher, würde das komisch aussehen. Bei vertikalen Slidern, die sich von oben nach unten slidern lassen, ist es umgekehrt.
Listen haben aber sowie horizontale als auch vertikale Vergrößerungsfreiheit. Will man mehr von ihnen sehen, zieht man das Fenster einfach auf. Checkmarks stehen als Objekt da und lassen sich nicht größer und kleiner machen. Also lassen sich viele MUI-Fenster nur horizontal vergrößern, weil die meisten Gadgets vertikal beschränkt sind.

Wir hatten aber eigentlich gerade das Problem für die Lösung gesucht, bei der man die Gruppen schachtelt. Nun stellt Euch mal vor, im letzten Beispiel würde man das Cycle-Gadget durch ein Checkmark-Gadget ersetzen. Die beiden Slider würden ihre Minimalbreite bekommen (beim Versuch, sich Längenmäßig an das kleine Checkmark-Gadget anzupassen), und das Fenster wäre jetzt gar nicht mehr vergrößerbar.
Also machen wir das Checkmark seitlich ausziehbar. Das geht aber nicht. Also ersetzen wir das Checkmark durch eine horizontale Gruppe, die ein Checkmark und einen Slider enthält. Nun wäre das Fenster wieder horizontal vergrößerbar, weil die ColumnGruppe wieder vergrößerbar wurde, weil ihre Kinder (die Slider und die HGroup) vergrößerbar sind. Die HGroup wurde übrigens vergrößerbar, weil EINS ihrer Kinder (nämlich der

Slider) vergrößerbar ist.
Die Labels sind nicht vergrößerbar. Genau müßte es also heißen:
Die Column-Gruppe ist horizontal vergrößerbar, weil mindestens in einer Spalte alle Kinder der Gruppe vergrößerbar sind.
Nun würden aber alle MUI-Applikationen höchst seltsam aussehen, wenn man immer Sliders als Füllstoff verwenden würde. Deshalb gibt es ein Objekt, das nach nichts aussieht und das auch nichts kann, das aber in alle Richtungen ausziehbar ist. Wenn nötig, bis nach Tokio.
Dieses Objekt heißt "HVSpace", also wörtlich übersetzt, horizontaler und vertikaler Raum.

Also greifen wir wieder das letzte Beispiel aus. So würde die Skizze aussehen, wenn man das Cycle-Gadget durch eine horizontale Gruppe mit Checkmark und HVSpace ersetzen würde:



Die Objekte haben jetzt Klammern und Spiegelstriche bekommen, um anzudeuten, wie ihre Ausdehungsmöglichkeiten sind.

Und wie sieht der MUI-WINDOWLAYOUT-Chunk aus? Voilà:
(Anm. des Autors: 4½ Jahre Französischunterricht hatten wenig zur Folge. Aber den acent von voilà kann ich schon richtigherum setzen, ohne im dictionnaire nachschlagen zu müssen.)

```
CHUNK:MUI-WINDOWLAYOUT
ColumnGroup(2),
   Label( "_Label 1" ),
   Slider( "s", 1, 10 ),
   Label( "L_abel 2" ),
   Slider( "a", -5, 20 ),
   Label( "La_bel 3" ),
   HGroup,
      CheckMark( "b" ),
      HVSpace,
   End,
End,
```

Mit diesem Beispiel sollte nun der MUI-Groschen gefallen sein. Mit HV-Space können auch ColumnGruppen aufgefüllt werden, bei denen man nicht jede Spalte einer Zeile benötigt.

Eine ausgiebige Lektüre der mitgelieferten gadget-Dateien ist sicher auch hilfreich, falls man mit MUI einen bestimmten Effekt erzeugen will, dessen technische Lösung einem nicht einfällt.
Natürlich war das jetzt noch nicht die gesamte MUI-Anleitung, aber erstmal das Grundwissen zum Layout-Prozess. Die einzelnen Gadgettypen und noch eine Spezialität bei den Boxen werden in den nächsten Kapiteln beschrieben.

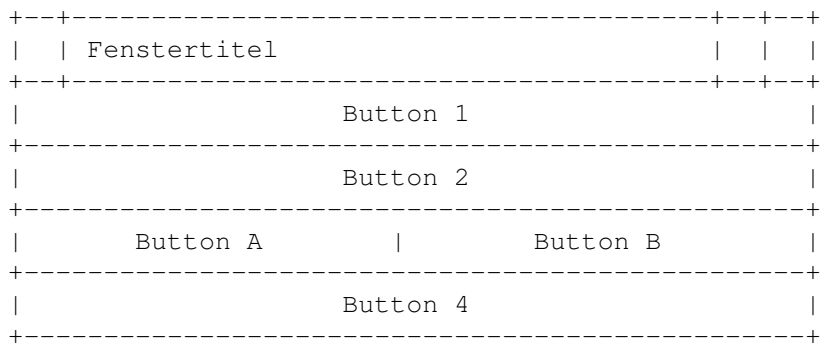
Wie auch das Programmieren, kann das Erzeugen von MUI-Windowlayout-Chunks nicht "auf dem Trockenen" erlernt werden. Etwas Übung gehört auch dazu.

Wer das Grundprinzip verstanden hat, der hat es nun auch schon leichter, in die MUI-Programmierung von richtigen Programmen einzusteigen. Allerdings schreibt man dann diese Group-Geschichten in den Programmcode und läßt sie von einem waschechten Compiler interpretieren, nicht von einem kleinen Programm namens MadhouseConfigEd. Wo da der Unterschied liegt? Nun, ein Compiler entsteht in mehreren Mannjahren Arbeit, der Madhouse-Programmteil zum Lesen des MUI-WINDOWLAYOUT-Chunks entstand in einer halben Woche. Deshalb sei es mir verziehen, wenn ich

a) nicht jedes Feature in den Interpreter eingebaut habe. Es lassen sich keine Page-Gruppen erzeugen, die Ausdehnungs-Gewichte der Objekte stehen fest auf 100, und es gibt auch keine CustomClasses (was wohl meilenweit übertrieben wäre) und manche Spezialgadgets fehlen auch. Die wichtigsten Dinge können aber problemlos erzeugt werden, deshalb sehen die Blanker-Prefs-Fenster auch genauso gut aus wie "richtige" MUI-Appliationen wie Madhouse und alle anderen.
b) die Fehler in einer gadget-Datei nicht peinlich genau abfrage. Madhouse ist mir zwar infolge von solchen Fehlern noch nie abgestürzt, und das wird es sicher auch nicht. Aber es gibt nicht zu jedem Fehler eine Fehlermeldung, denn da hätte ich ja fast alle der 260 Fehler, die mein C-Compiler melden kann, auch auf die gadget-Dateien übertragen müssen. Aufmerksam wird man auf die Fehler sowieso, und finden tut man sie auch schnell, denn der MUI-Chunk ist ja immer nur ein paar Zeilen lang.

Übrigens arbeitet auch Vionas EGS auf dem Boxenprinzip, die Gadgets werden ähnlich wie bei MUI angeordnet. EGS wurde auf dem Amiga durch die vielen Grafikkarten, mit denen es ausgeliefert wird, bekannt. Deshalb gibt es wohl auch kaum EGS-Anwendungen, die nichts mit Grafik zu tun haben.

1.67 Die Lösung



1.68 Die Gruppen

Natürlich erkläre ich hier nicht das gesamte Boxenprinzip erneut. Das steht ja schon alles im Greenhorn-Kapitel.

HGroup und HGroup("Gruppentitel")

Die HGroup, die ihre Kinder ja übereinander anordnet, kann auch umrahmt
werden, und wird damit sichtbar. In diesem Rahmen wird dann ein Gruppen-
titel dargestellt, der die Gruppe sozusagen überschreibt.
Will man so ein Ding erzeugen, benutzt man die HGroup so wie es in der
Überschrift steht.

Dasselbe geht auch mit VGroups.

ColumnGroup(x) und ColumnGroup("Gruppentitel", x)
x steht nach wie vor für die Anzahl der Spalten.
Wie auch die anderen Gruppen kann die ColumnGroup einen Titel bekommen.

1.69 Die Gadgets

Einige Standard-Bezeichner in diesem Kapitel

- Taste enthält einen Kleinbuchstaben, der angibt, mit welcher Taste
auf dem Keyboard des Computers ein Objekt gesteuert werden kann.
Beispiel: Slider("a", 5, 10) erzeugt einen Slider, der mit der Taste A
gesteuert wird. Die Tasten "o", "c" und "t" sind bereits für die Standard-
Gadgets am unteren Rand des BlankerPrefs-Fensters reserviert und können
daher nicht benutzt werden.
- Bezeichnung steht für einen Text, auf der linken Seite des betreffenden
Gadgets angezeigt werden soll. Dieser Text soll das Gadget näher erläutern.
Beispiel: LabelCycle("Color _selection", "s", "Red|Green|Blue") würde
ein Cycle-Gadget herstellen, auf dessen linker Seite sich der Text "Color
selection" befindet, bei dem der Buchstabe "s" unterstrichen wird. Das ist
nötig, damit der Benutzer weiß, daß dieses Cycle-Gadget mit dem Buchstaben
"s" gesteuert wird.

Slider(Taste, Minimalausschlag, Maximalausschlag)

erzeugt einen Slider. Minimal- und Maximalausschlag bezeichnen den Aktions-
Radius des Sliders, inklusive dieser beiden Werte (d.h., daß Minimalaus-
schlag selbst ist auch noch anwählbar ist).

LabelSlider(Bezeichnung, Taste, Minimalausschlag, Maximalausschlag)

wie oben, jedoch zusätzlich mit einem Label auf der linken Seite.

CheckMark(Taste)

stellt ein Häkchengadget her.

Cycle(Taste, Auswahlmöglichkeiten)

stellt ein Cycle-Gadget mit mehreren Einträgen her. "Auswahlmöglichkeiten"

ist eine Zeichenkette, die alle Einträge enthält. Ein Beispiel dafür wäre "RGB|HSV|CMYK". Der Vertikalstrich "|" trennt die Einträge. MUI beginnt das Auszählen der Einträge (wie sich das für einen Computer gehört) mit Null, wenn also RGB der aktive Eintrag ist und der User auf "Test" klickt, steht eine "0" in der prefs-Datei.

LabelCycle(Bezeichnung, Taste, Auswahlmöglichkeiten)

wie oben, jedoch zusätzlich mit einem Label auf der linken Seite.

String(Taste, Länge)

erzeugt ein String-Gadget. Es lassen sich maximal so viele Zeichen eingeben, wie bei "Länge" angegeben. Wieder ist die Obergrenze für eine Stringlänge 500, was darüber geht wird automatisch auf 500 gesetzt.

LabelString(Bezeichnung, Taste, Länge)

wie oben, jedoch zusätzlich mit einem Label auf der linken Seite.

Label(Bezeichnung)

erlaubt eine bessere Plazierung der Objekte. Wenn Label und Objekt getrennt werden (macht man normalerweise so), dann schließen die Gadget-Rahmen bündig ab. Man benötigt dann einen Gadgettyp ohne Label... am Anfang und dieses Label-Objekt.
Bezeichnung ist diesmal alles, aus was das Objekt besteht.

LLabel(Bezeichnung)

wie oben, jedoch wird dieses Label nicht rechts ausgerichtet, sondern links. Das ist praktisch, wenn man hinter einen Slider noch eine Angabe klemmen will. Sonst sollten alle Objekte links beschriftet werden, nur bei den CheckMarks bin ich mir da nicht so sicher (da macht auch eine Beschriftung auf der rechten Seite Sinn). Ein Beispiel für linksbündige Labels findet sich in der gadget-Datei von Waves.

HVSpace

der berühmte Platzhalter, wie viele andere Dinge schon bekannt aus dem Vorkapitel.

HBar

Eine sehr elegante Sache, die die Gruppen-Rahmen manchmal ersetzen kann: dieses Objekt zeichnet einen waagerechten Strich an seiner Position, und sollte nur in horizontalen Gruppen zur Trennung von Funktionen benutzt werden.

VBar

wie oben, jedoch in der vertikalen Ausführung. Eine VBar läßt sich gut im BlankerPrefs-Fenster von Stars beobachten.

1.70 Der CHUNK:MUI-PREFSORDER

"Wie kann MUI nur mit einem Chunk auskommen?", werden viele schon gefragt haben. So ganz geht das doch nicht. Aber jetzt wieder zuerst ein Problem.

Ganz oben, in der Erklärung wie man einen Blanker zu schreiben hat, schrieb ich, daß Madhouse die Einstellungen der Gadgets in der Reihenfolge in die prefs-Datei abspeichert, in der die Gadgets im CHUNK:GADGETS angegeben wurden. Und im CHUNK:MUI-WINDOWLAYOUT? Da ist es genauso. Allerdings haben wir da ein gaaaanz kleines Problem: Man kann mit diesem MUI-Chunk nicht ein Gadget in die obere Fensterecke legen und dann seine Einstellung zuletzt auslesen wollen. Natürlich könnte man jetzt den Programmtext ändern, und mir fällt gerade ein, wie sinnlos diese Funktion ist, aber man kann eine bestimmte Funktion von Madhouse benutzen, nämlich den CHUNK:MUI-PREFSORDER.

In diesem Chunk wird die Reihenfolge angegeben, mit der die Einstellungen in die prefs-Datei geschrieben werden. Bei Waves und Stars habe ich diesen Chunk benötigt. Auch Note enthält einen.

Dazu muß man nun zuerst jedem Gadget einen internen Namen geben, zum Beispiel so:

```
CHUNK:MUI-WINDOWLAYOUT
VGroup,
   LLabel( "Ein Ausschnitt aus der Stars-gadget-Datei." ),
   ColumnGroup( 2 ),
      Label( "_Maximum" ),
      Maximum = Slider( "m", 1, 8 ),
      Label( "M_inimum" ),
      Minimum = Slider( "i", -7, 1 ),
      Label( "C_anges" ),
      Changes = Slider( "a", 0, 15 ),
      Label( "_Start" ),
      Start = Slider( "s", -7, 8 ),
   End,
End,
```

Normalerweise würde die prefs-Datei so aussehen:

```
Wert von Maximum
Wert von Minimum
Wert von Changes
Wert von Start
Duration-Zeit in Minuten
Pfad auf das Verzeichnis des Blankers
```

Und jetzt kommts: schreibt man noch

```
CHUNK:MUI-PREFSORDER
Changes, Start, Minimum, Maximum
```

dann sieht die prefs-Datei schon ganz anders aus:

Wert von Changes
Wert von Start
Wert von Minimum
Wert von Maximum
Duration-Zeit in Minuten
Pfad auf das Verzeichnis des Blankers

Naja, sowas braucht man wohl sehr selten bis gar nicht... Wichtig ist aber noch, daß als "interne Bezeichner" nur ganz gewöhnliche Buchstaben in Frage kommen, nicht aber Zahlen und andere Spezialitäten.

1.71 Anhang

A Arrrggh: Bekannte Probleme
B Die Autoren und das: Copyright
C Madhouse ist toll: Registrieren leicht gemacht.
D Die Revolution: MUI
E Nutzlos^3: alle hier verwendeten Smileys

1.72 Bekannte Probleme

Probleme - Kapitel I: Probleme, die von einer Programmiersprache namens AMOS (über 43.655 versch. Befehle...) erzeugt werden:

Probleme mit VGA-Monitoren

Genialerweise öffnen AMOS-Programme keinen normalen Intuition-Screen. Wer einen VGA-Monitor ohne ScanDoubler o.ä. sein eigen nennt, der kann zwar die Blanker, die von mir in C++ geschrieben wurden, mittels eines Screen-Promoters umpatchen; aber die AMOS-Blanker von Aicke müssen hier klein beigeben. In diesem Fall hat Madhouse nur sechs Blankmodes, sorry. Falls tatsächlich mal die AMOS-AGA-Version erscheinen sollte, für die auch das Öffnen von normalen Screens versprochen wurde, wird Aicke alle AMOS-Blanker darauf anpassen. Falls Aicke die Programmiersprache wechseln sollte (ist noch unwahrscheinlicher), hättet Ihr und ich weniger Probleme. Madhouse mußte nämlich extra abgestimmt werden, damit AMOS die erzeugten Textdateien lesen kann.

Probleme mit MousoMeter

MousoMeter macht leider ebenfalls Probleme mit AMOS-Programmen, was sich so äußert, daß MousoMeter wie wild Kilometer zählt. Das bricht jeden Highscore. Der Grund ist wohl, daß AMOS AMOS ist. (Anders kann ich mir das nicht erkären.) Spitze, AMOS!
Abhilfe: AMOS-Blanker inaktivieren.

Probleme, die durch inkompatible Programme entstehen (auch AMOS..)

Probleme mit AMOS und Protracker (und vielleicht mit anderen Tracker-Sound-Editoren, nicht aber mit MED)

Manche Programmierer wollen wohl die Grafikausgabe beschleunigen, indem sie einen Screen öffnen, der kein Intuition-Screen ist. Diesen könnt Ihr nicht wie gewöhnlich umschalten, außer wenn diese Programmierer die Tasten <linke Amiga + n> (und +m) selbst abfragen. Na ja, jedenfalls kann man dieses Etwas nicht nach hinten bringen, indem man selbst einen Intui-Screen öffnet. Deshalb werden die Blanker FlyingToasters, Stars, FireWorks, Waves, Socher und Shuffle (die ich in C schrieb) HINTER AMOS und Protracker angezeigt. Die anderen Blanker, die Aicke in AMOS schrieb (paradox, oder?) können aber vor AMOS und Protracker angezeigt werden. Also dürfen nur diese Blanker bei Benutzung von AMOS und Protracker angewählt sein.
Weiterhin funktionieren mit obigen Programmen der Black Background und der Paßwortscreen nicht, aus demselben Grund.

Probleme, die in Sondersituation 48d entstehen:

Probleme mit einer resetfesten Ram-Disk, die als RAM: gemountet ist

Dieser Tip stammt von Aicke, und vielleicht bleibt er auch der einzige User, der gar keine nicht-resetfeste Ram-Disk hat, dafür eine resetfeste SD0: oder VD0:, die über RAM: angesprochen wird.
Wer also Madhouse bei jedem Systemstart automatisch aufrufen läßt (z.B. mit der WBStartup-Schublade) und wer ein Gerät namens RAM: hat welches nach einem Reset NICHT wieder leer ist, der hat ein Problem: Madhouse wird nach einem Reset beim Start schon die Schublade RAM:Madhouse_Storage vorfinden und denken, es wäre 2x gestartet worden.
Dieses Problem kann man nun umgehen, indem man VOR dem Start von Madhouse dieses Shell-Kommando ausführen läßt, indem man es z.B. in die S:User-Startup einträgt:

```
delete >NIL: RAM:Madhouse_Storage ALL
```

Beachtet bitte, daß es nicht möglich ist, gar kein RAM:-Device zu haben.

1.73 Die Autoren und das Copyright.

Eine Art History-File: kleiner Geschichtsunterricht

Wir brauchten neunzehn Monate, um Madhouse und die Blankmodes zu entwikkeln. Aicke hat früher angefangen, weil mein Computer damals gerade in Reparatur war. So entstanden einige seiner Blankmodes früher. Madhouse wurde von uns schon früher erdacht, denn immer, wenn wir einen modularen Screenblanker in PD-Serien gesehen hatten, gefiel er uns nicht richtig. Wir hatten viele Ideen für ein besseres Hauptprogramm, die größtenteils in Madhouse verwirklicht wurden. Doch am ärgerlichsten waren oft die Blankermodule, so kann man sie bei einigen Blankern gerade noch als Beispielprogramm durchgehen lassen (dabei waren die Dinger echt dazu gedacht, die Leute zu begeistern)...
Jedenfalls können wir die Ursprünge von Madhouse nicht mehr richtig nachvollziehen. Eine Kritzelei, die schon etwas aussieht wie das

Hauptfenster befindet sich noch in meinem alten Deutschordner und hat das Datum 6.8.93. Damals hieß meine Programmiersprache noch GFA-Basic. Im "Entwicklertagebuch" des Hauptprogramms kann ich aber den Beginn noch ablesen: am 25.1.94 entwarfen Aicke und ich das Design des Hauptfensters. Der Name von Madhouse war zu dieser Zeit nicht Madhouse, sondern "BlackHole". Diesen Namen gab es aber schon. So hatte Aicke in den folgenden Wochen diverse Einfälle für neue Namen. Am 26.2. konnten wir uns dann für "Madhouse" entscheiden. Alternativen waren z.B "Joke-Box" oder "Monitor-Holidays".
Wie jedes Programm hatte auch Madhouse viele Bugs. Fast zur Verzweiflung trieb mich ein ganz hartnäckiger, der Madhouse nach Betätigung des Remove-Buttons zum Abstürzen brachte. Dabei stürzte der Computer genau (das konnte ich kontrollieren) beim Aufruf der exit()-Funktion ab, der ein Programm beendet. Nach mehreren Wochen fiel mir dann auf, daß es an der Stackgröße von Madhouse lag...

Arbeitsteilung: Entwicklung beim Einen, Abstürze beim Anderen...
(war umgekehrt nicht der Fall... AMOS als absturzfreie Sprache??!)

Von Aicke stammen 62,5% der Blanker. Wahrscheinlich kann man an seinen Blankern und auch an den anderen Teilen Madhouses erkennen, daß er ein echter Perfektionist ist. Das hat es einerseits nicht einfach gemacht, mit ihm zu arbeiten, aber es hat Madhouse oft entscheidend verbessert. So hat er in allen Dingen, die zu Madhouse gehören, seine Spuren hinterlassen: ob er nun fünf Pixel an der kleinen Diskette geändert hat, die im Ask-For-Disk-Fenster erscheint oder ob er eine gute Idee für FireWorks hatte. Ideen hat er auch viele, so daß er schon neue Blankmodes in Vorbereitung hat. Die AMOS-Demo ist auch von ihm.

Madhouse ist mein zweites C-Programm (eigentlich C++), das Erste waren die FlyingToasters. Von mir kommt auch die englische und deutsche Anleitung. In der Englischen werden viele Fehler sein, in der Deutschen viele Abweichungen vom Thema... (Leider seht Ihr nun gar nichts von der engl. Anleitung, weil die für eine deutsche Zielgruppe nun doch nicht mehr nötig ist.) Da das Einsteigen in eine Sprache wie C recht schwierig ist, danke ich allen die durch ihre Quelltexte im PD-Bereich etwas Licht in das große Compilerdunkel brachten. An die seitenlangen Compilerfehler kann ich mich noch gut erinnern. Die C-Demo habe ich auch geschrieben, das war meine erste echte Konfrontation mit dem ANSI-C-DosLib-File-I/O.

Zusammen haben wir einen Vorteil anderen Programmieren gegebüber, die allein arbeiten: Wir haben verschiedene Computer. Damit kann man Programme besser auf Fehlerfreiheit testen, den die Betriebssysteme 2.0 / 3.0 reagieren bei Programmfehlern so gut wie immer unterschiedlich.

Blick in die Glaskugel

In Zukunft werden hoffentlich neue Blanker erscheinen. Die müssen dann nicht unbedingt von uns sein, denn mit der mitgelieferten Dokumentation (in dieser Anleitung) kann so ziemlich jeder selbst einen Blanker herstellen. Eine Idee wäre noch ein Sound-Teil, Madhouse könnte Musik-Module abspielen während die Blanker laufen. Meine wichtigste Planung für die Zukunft war das MUI-Interface für Madhouse. Wie Ihr sicherlich schon gemerkt habt, haben die MUI-Funktionen schon den Sprung in diese Version geschafft.

Wir haben schon eine Menge Ideen für neue Blanker. Teilweise ist schon Grafik fertig. Auch manche vorhandenen Blanker sind noch Verbesserungsfähig. Falls es einen Geschwindigkeitszuwachs gibt (und das hoffe ich doch) werden die FlyingToasters mit Bobs arbeiten. Ein Aquarium wäre auch langsam fällig, oder fangen wir doch lieber mit einer anderen Idee an?

Madhouse ist Shareware, der Betrag ist 20 DM. Im Registrationskapitel (Anhang C) könnt Ihr alles über die Registration erfahren.

Wenn Ihr einen Bug findet, bitte schreibt uns! Wenn jemand einen guten Blanker geschrieben hat, soll er/sie (?) ihn nicht nur für sich behalten, sondern auch veröffentlichen! Es wäre wirklich super, wenn auch andere Programmierer unser System nutzen würden.

Unser Input-Device: schreibt uns mal!

Ihr könnt uns Ideen, Grafiken und Code senden, dann versuchen wir, was draus zu machen. Wenn Ihr eine Antwort wollt, legt bitte genug Briefmarken bei (am besten deutsche oder internationale). Wer weiß, wieviel da ankommt... (Wahrscheinlich gar nichts, sowas kennt man ja.)

Wenn Ihr uns wegen Bugs schreibt, dann erklärt bitte peinlich genau WAS passiert ist, WIE Ihr das geschafft habt und WELCHE anderen Sachen im Hintergrund liefen und welchen Computer Ihr benutzt. Desweiteren benötigen wir eine genaue Beschreiben der Pflanzen im Computerraum.
Am besten Ihr versucht es auch ohne Tools und mit Eurer Original-Workbench, bevor Ihr uns mit Back-Beschreibungen zudröhnt... Natürlich sind wir auch froh, wenn wir Bugmeldungen erhalten, aber andere Dinge würden uns halt mehr freuen - verständlich, oder?

Leider verfügen wir mangels Modem über keine E-Mail-Adresse.

Aicke Schulz
Neudeckerweg 118
D-12355 Berlin
Telefon: 030 (Berlin) / 664 48 22

Carsten Jahn
Kuckucksruf 34
D-16761 Stolpe-Süd

Vielen Dank für den Brief!

Natürlich dürfen auch die Credits, sprich Danksagungen nicht fehlen. Die Blanker CrazyPixel, DigitalClock, Drops, Glitter, Memory, Note, Skyline, SoftwareFailure, Thunder und Worldtime wurden mit der Programmiersprache AMOS entwickelt. Die Blanker FireWorks, FlyingToasters, Shuffle, Soccer, Stars und Waves sowie das Hauptprogramm wurden mit dem MaxonC++-Light 3 Compilerpaket erstellt und compiliert.

MagicUserInterface wurde von Stefan Stuntz entwickelt, die Rechte dafür liegen bei ihm.

```
+-------------------------------------------------------------+
| Wer Madhouse verfälscht oder nachmacht, oder verfälschte oder |
|  nachgemachte Versionen in Umlauf bringt, wird mit Windows®   |
|              nicht unter zehn Stunden bestraft.               |
+-------------------------------------------------------------+
```

Natürlich wird keinerlei Haftung für Schäden übernommen, die durch den Gebrauch von Madhouse entstehen / entstanden sind.
Für den Gebrauch von Windows® haften wir erst recht nicht.

Und wärend ich hier die Zeile 3591 der Anleitung schreibe, verabschiede ich mich und wünsche allen Amiga-Usern ein langes, schönes Leben, tolle Programme, neue Blanker, einen Multiscanmonitor und uns allen das Überleben des Amiga auf dem engen Markt der Hardwareplattformen.

Tschüß, bis zur nächsten Anleitung,

Euer Carsten.

1.74 Registration

Seit eineinhalb Jahren haben wir Madhouse auf unseren Computern. Endlich ist es uns gelungen, es einigermaßen fertigzustellen. Ein modulares System kann zwar gar nicht fertig sein, schließlich könnten noch hunderte von Blankern hinzukommen, aber die Hauptprogramme Madhouse und MadhouseConfigEd sind langsam ausgereift. Nun sollte es jedem Amiga-User zugänglich sein, und wir hoffen wenigstens auf ein kleines Feedback.

Ohne das Keyfile, das jeder registrierte User von uns per Post erhält, läßt sich Madhouse nicht dauerhaft benutzen. Ist es nicht vorhanden, werden die Einstellungen in ENV: und ENVARC: nicht beachtet, also muß nach jedem Start mindestens der Pfad zu den Blankern neu eingestellt werden.
Nach der Registration erhaltet Ihr von uns eine Diskette, die eine Datei namens Madhouse.key enthält. Diese solltet Ihr in Euer S: - Verzeichnis kopieren. Sonst kopiert Ihr die niemandem! Sie enthält nämlich Eure komplette Anschrift, die Ihr 1. nicht im Keyfile ändern solltet, sonst wird es nicht mehr erkannt, und die Euch 2. sofort entlarvt, wenn sie einmal der Falsche in die Hände kriegt.

Um Euch registrieren zu lassen, benötigt Ihr:
- Das ausgefüllte Registrationsformalur. Dieses könnt Ihr einfach ausdrucken und leserlich ausfüllen.
- Den Betrag von 20 DM, soviel kostet das. Dieser Preis ist doch Säugling-, Kleinkinder-, Jungliche-, Erwachsenen- und Seniorenfreundlich, oder?
- Einen normalen Briefumschlag. Da steckt Ihr o.g. hinein.
- Eine 1DM-Briefmarke. Die klebt Ihr auf o.g. drauf.
- Einen Briefkasten in Eurer Nähe. Da steckt Ihr o.g. hinein.

Mut gehört übrigens nicht dazu - Ich habe schon dreimal Geld für solche Zwecke mit der Post verschickt, nach Deutschland, nach Holland und nach England, und immer hat es geklappt.

Wer das aber will, der kann das Geld auch überweisen:
Aicke Schulz
Deutsche Bank, BLZ 100 700 00, Konto-Nr. 3565611
Bitte trotzdem das Formular ausfüllen und an uns schicken!

Als mordsmäßigen Bonus garantieren wir den ersten zwanzig Bestellern, daß sie eine Diskette mit unserer Handschrift und Autogrammen erhalten! (Schließlich lohnt es sich nicht, für 0 oder 4 Disketten Aufkleber zu drucken...)

Alles klar?

Na, dann wollen wir mal den Ausdruck starten.

1.75 Auch Madhouse benutzt das sagenhafte MUI.

This application uses

MUI - MagicUserInterface

(c) Copyright 1993/94 by Stefan Stuntz

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 30.- or US$ 20.-

to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

Stefan will uns damit sagen, daß die Oberfäche von Madhouse mit installiertem MUI völlig frei von Euch gestaltet werden kann. Dem kann ich nur zustimmen. MUI gehört zu den flexibelsten Programmen, die ich je gesehen habe. MUI ist kein normales Programm, was man für einen bestimmten Zweck benutzt. Jede entsprechend programmierte Anwendung kann es verwenden, und dann profitiert der Anwender davon. In PD-Serien und Mailboxen findet man

die aktuelle Version von MUI, so richtig toll ist aber nur die unter der oben genannten Adresse zu beziehende Vollversion, mit der man die Einstellungen dauerhaft speichern kann. Sonst sehen die MUI-Anwendungen nach jedem Reset wieder ganz normal aus.

1.76 Alle in dieser Anleitung verwendeten Smileys

Neben dem Standard-Smiley :-) wurde der Einsatz weiterer Arten in dieser Anleitung nötig. Die müssen natürlich noch entsprechend gedeutet werden:

```
==)       Smiley mit Darth-Vader-Helm.
*-{:-)    Smiley mit Bommelmütze (Inspiration aus der Winterzeit).
%·)       Smiley mit Designerbrille (und Knollennase).
:^)       Smiley aus der isometrischen Perspektive.
```

Allen Smiley-Fans empfehle ich stark das Archiv misc/misc/SmileyV3.lha aus dem AmiNet.