Multiformatinputfilesorrydoesn’tknowthisargumentisnotusers

COLLABORATORS			
	TITLE : Multiformatinputfilesorrydoesn’tknowthisargumentisnotusersfault.		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Multiformatinputfilesorrydoesn’tknowthisargument

1.1 Madhouse 2.01 Inhalt

```
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" was 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.

        Ein kleines Vorwort: Wer will das schon lesen?
       Was Madhouse braucht: Der nötige Amiga.
               Der Workshop: Schnell einsteigen.
       Der MadhouseConfigEd: Alle Seiten.
                  Madhouse: Die Merkmale/Tooltypes
                Gib' Error: Madhouse-Fehlermeldungen
 Madhouse für Programmierer: eigene Module
                Die Blanker: und alle Optionen
                Die Autoren: Registration
```

1.2 Vorwort

```
Mein kleines Vorwort

Hallo Leute, Ihr seid sozusagen mittendrin - in Carstens Anleitung. Aicke
und ich haben Madhouse erdacht. Vom Programmiertechnischen her hat Aicke
vor allem die Blanker übernommen. Wenn Ihr meint, die wären wenig (drei
Stück in dieser Version), dann schaut Euch mal das util/blank/MadAMOS.lha
im Aminet an. Früher hat er nämlich in AMOS programmiert, und die so
entstandenen Blanker haben durch MUI 3.x leider den Rest gekriegt.
Das heißt konkret, daß man im MadhouseConfigEd bei diesen Blankern nicht
auf 'Test' (im Konfigurationsfenster des Blankers) klicken darf, denn der
Task wird sich nie mehr beenden. Vielleicht kennt ja jemand eine Lösung;
ich kenne keine.
```

Aicke programmiert nun in AmigaE, ist voll begeistert, und hat bereits einige Blanker umgesetzt. Das sind die, die jetzt dabei sind. Wer noch nicht MUI 3.x benutzt, sollte mal einen Blick in das Archiv werfen. Außerdem debugged Aicke meine Anleitungen.

Heute ist der 5. Februar ’96, Madhouse ist kürzlich zwei Jahre alt geworden. Von dem großen ersten Teil der Entwicklung habt Ihr nichts mitbekommen, deshalb hieß die erste Version im Aminet auch 1.1. Die ganze Entwicklungsgeschichte schlummert bei mir noch in diversen history-Dateien, in denen jedes Detail aufgeführt ist. Aicke hat die mal mit Vergnügen durchgelesen, und wenn mich ein paar von Euch darauf ansprechen, mach’ ich vielleicht mal ein kleines Buch als .dvi- und .txt-Datei daraus.

Meine Anrede an Euch habt Ihr sicher eben bemerkt. Ich mache das ganz gerne so, weil mir ein ständiges ‘Sie’ zu konservativ und verfroren für ein spaßiges Programm erscheint. ‘Du’ dagegen ist schon wieder zu familiär, man fühlt sich manchmal richtig vom Text angegriffen (dieses Gefühl habe ich zumindest). Mit ‘Ihr’ meine ich Euch alle, die da draußen vor dem Monitor hocken und nichts besseres zu tun haben, als sich mein Blabla durchzulesen. Find’ ich übrigens echt in Ordnung von Euch, wer liest denn heute noch Vorworte; na bitte, Ihr seid der Beweis!

Für die nächsten 16.000 Wörter bzw. 2800 Sätze begleite ich Euch auf Eurer Reise durch die Programmfunktionen von Madhouse, dem Schirmschoner, der Arbeit machte. Vor allem die Anleitung. Deshalb mute ich keinem von Euch zu, sich das alles durchzulesen. Steht vermutlich sowieso ’ne Menge Schwachsinn drin; denn ich komme kaum nach, alle Änderungen im Programm zu dokumentieren. Wer es tatsächlich schafft, alles zu lesen, kann mir mal eine Postkarte schicken. Im Autoren-Kapitel stehen unsere Adressen. Meine ist erst ab März benutzbar, weil wir im Moment noch nicht dort wohnen. Euer 20,- DM-Schein könnte da also höchstens zuregnen. Eine Postkarte verkraftet der arme Briefkasten aber noch.

Einmal wurde uns der Briefkasten schon mal abmontiert. Also ist das eigentlich schon der zweite Briefkasten, den wir für diese Adresse gekauft haben. Bei der alten Adresse, die schon heute nicht mehr stimmt, war der Briefkasten fest eingebaut. Neben der Klingel. Die war im Winter meist dermaßen zugefroren, daß die Leute keine Chance hatten. Beim Briefkasten ist immer die Klappe abgegangen. Da wo wir jetzt wohnen, nämlich bei meiner tollen Oma, gibt es gar keinen Briefkasten. Wir haben einfach einen Schlitz in der Wohnungstür.

Manchmal muß ich etwas Schrott schreiben, um mich von all dem abzulenken. Wer in der nächsten Anleitung mehr über meine Briefkästen und mich erfahren möchte, soll dies auf dem Registrationsformular vermerken. Ist mein Ernst.

Wobei ich schon beim Shareware-Thema wäre. Aber eigentlich ist das ja gar kein Thema, denn jeder dürfte bemerkt haben, daß nur die wirklich kostbaren Sachen umsonst sind. Paradox, oder?

Hai Einsteiger. Ich bin auch mal eingestiegen. (Ich steige oft ein, in Busse, in Banken,...) Naja, es war ein A500 mit OS1.2, 512 kByte RAM und einem Diskettenlaufwerk. Wie steigt Ihr ein? Bitte notiert dies im Registrationsformular. Und vor allem, ob Madhouse für Euch einfach zu benutzen ist. Bei mir war damals ja alles englisch, und ich konnte noch

kaum welches. Zumindest nicht dieses Computer-Englisch. Ich habe z.Bsp. immer 'Use' und 'Cancel' verwechselt. Und zwar im originalen 1.2-Preferences-Editor. Aber Ihr wißt bestimmt noch nicht mal, wie der überhaupt aussah.

Da war vor allem alles blau. Ja, blau! Heute ist die Workbench grau, früher war sie blau. Ändern konnte man das schon immer. Die Slider hatten keinen geraden Schieber in der Mitte, sondern einen runden. Es wurden nur Dateien mit Icons auf der Workbench angezeigt, für den Rest bemühte man die Shell. Nein, das CLI. Aber das ist eine andere Geschichte...

Immer noch dabei? Wundert mich. Ich würde mich freuen, mal von Leuten zu hören, die mit Ihrem Amiga 1000 angefangen haben. Einige von denen haben ihn nicht verkauft oder gar vermüllt, sondern heben ihn immer noch auf und benutzen ihn. Der hatte nämlich standard-mäßig 256 kByte RAM. Um eine Diskette mit einem Laufwerk zu kopieren, mußte man sie deshalb zwei dutzend mal wechseln.

Ok, ich glaube, das reicht jetzt. In der nächsten Release schreibe ich dann von dem Fahrstuhl in dem Haus, wo Aicke wohnt. Versprochen. Heute hab' ich mich schon wieder erschrocken.

Also dann, habt viel Erfolg mit den folgenden Zeilen.

Viel Spaß mit Madhouse wünschen Euch
die Programmierer Aicke Schulz und Carsten Jahn.

1.3 Minimale Systemkonfiguration für Madhous

So weit ist's nun schon gekommen: sogar Screenblanker stellen Ansprüche. Aber im Grunde fordern wir nix Besonderes, auf die heutige Zeit bezogen:

- Einen Amiga mit AmigaOS 2.0 oder höher, ab OS 2.1 deutsche Programmtexte sichtbar.
- Eine Festplatte, auf der Madhouse und die Blanker installiert sein sollten. Die neuen Madhouse-Versionen sind nicht mehr, wie die 1.1-Version, nur mit einem Diskettensystem lauffähig, weil es da zuviele Probleme und Umstände gab. Mal ehrlich: wer kommt denn heute noch ohne HD aus?!
- MUI (© Stefan Stuntz). Und zwar nicht unbedingt das brandneue 3.1, ältere Versionen tun's auch noch. Siehe auch MUI-Infos.
- Madhouse wurde aber trotz allem so konstruiert, so wenig Speicher wie möglich zu brauchen. Das Hintergrundprogramm 'Madhouse' hat deshalb keine Bedienoberfläche, die hat der Einsteller 'Madhouse-ConfigEd'. Der braucht dann auch erst MUI. Madhouse selbst ist nur ca. 30 kByte groß und öffnet keine Libraries, die sonst nicht auch im Speicher wären.

1.4 Der Workshop

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:

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 kann Madhouse schon aus der ursprünglichen Schublade heraus gestartet werden, solange man die Diskette / CD, in der Madhouse noch ist, nicht aus dem Laufwerk nimmt. Deshalb sollte man Madhouse ja später auf Festplatte installieren.
- b) Zur dauerhaften Anwendung Wie das Wort "dauerhaft" schon andeutet, muß hier wieder die Schublade SYS:WBStartup/ herhalten. Nun das Programm "Madhouse" nach SYS:WBStartup kopieren. Alle Programme in dieser Schublade werden dann beim Hochfahren des Rechners automatisch gestartet. Dann legt ein neues Verzeichnis auf der Festplatte an und kopiert das gesamte Madhouse-Verzeichnis dorthin. 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.

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...

Ein Fenster öffnet sich.

Und das ist schon einen Faszination für sich, denn das Fenster wird vom MadhouseConfigEd über MagicUserInterface (© Stefan Stunz, folgend MUI genannt, siehe auch MUI-Infos) erstellt. 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 Hilfsmittel/Tools-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 erster Abschnitt) und startet Madhouse neu.

Jetzt den Pfad der Blanker angeben!

Durch einen Klick auf den kleinen Knopf mit dem Bildchen (neben dem Pfad/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 Festplatte beginnen (z.B. "Work:.../Blankers"), sonst macht Euch Madhouse darauf aufmerksam und akzeptiert nix.
Nachdem der Requester mit "Ok" bestätigt wurde, sollten in der Liste 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.

Nun sehen wir uns die Blanker-Module an!

Dazu solltet Ihr einfach auf den Namen eines Blankers in der Liste doppelklicken.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’ schließen das Blankerfenster, wobei ‘Okay’ die Einstellungen abspeichert.
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.

Die Liste mit den Blanker-Namen und was da so passiert

Mit den Checkmarks (‘Häkchengadgets’...) wird bestimmt, welche Blanker gestartet werden sollen, wenn der Computer über eine einstellbare Zeitspanne hinweg keine Eingaben empfangen hat. Ein einfacher Klick auf einen Blankernamen zeigt Informationen zu ihm auf der Blanker-Seite an. Außerdem könnt Ihr dort noch einige interne Einstellungen ändern, die ihn betreffen. Den Doppelklick habt Ihr ja soeben kennengelernt.

Wieviel Zeit meiner Inaktivität soll vergehen, bis Madhouse loslegt?

Diese Einstellung wird mit dem Zeit/Time-Slider erledigt.

Konfiguration speichern

Ganz einfach mit Speichern/Save. Dann wird die Konfiguration als "ENV-ARC:Madhouse.prefs" dauerhaft abgespeichert, und zur sofortigen Benutzung nochmals wieder unter "ENV:Madhouse.prefs".

1.5 Der MadhouseConfigEd

Tjaaa, der ConfigEd. Wollt Ihr jetzt echt wissen, was man da alles machen kann? Na gut, hier kommt das Zeuch:

Die Gadgets am unteren Fensterrand
Die Liste
System - Seite
Optionen I - Seite / Options I (Andere Optionen / Other Options)
Optionen I - Seite / Options I (Dimmer)
Optionen II - Seite / Options II
Blanker - Seite
Info - Seite

1.6 Die Import-Funktion

Die Import-Funktion hebt die Barrieren zwischen den verschiedenen Blanker-Formaten auf. Damit sind die Blanker verschiedener Systeme alle in Madhouse verfügbar, so als wären es Madhouse-Blanker.
Vielleicht hat mit dieser Funktion das stupide Aufstöbern und Portieren anderer Blanker auf das eigene System ein Ende, das diverse Autoren von modularen Schirmschonern so exzessiv betreiben.

Im Moment sind Import-Funktionen für Garshne und Swaz vorhanden, weitere werden folgen. Bei allen Systemen gilt, daß der Benutzer sich die Archive mit den Blankern selbst besorgen muß, denn die verfügbaren Blanker sind nicht im Madhouse-Archiv dabei. Wir respektieren den Copyright der anderen Autoren, obwohl die in einigen Fällen auch nur einen fremden Blankerquelltext nehmen, ihn anpassen und dann mit ihrem System zusammen verbreiten.
Also müßt Ihr Euch zunächst ein originales Swaz/Garshne-Archiv holen und entpacken. Für Garshneblanker gibt es bereits Zusatzarchive mit Blankern. Woher Ihr die nehmt, kommt darauf an. Weil Madhouse über das Aminet verbreitet wird, liegt es nahe, erstmal ein paar Aminet-Archive aufzuzählen, die zu downloaden es sich lohnt:

- util/blank/AgBlankers1_1.lha
- util/blank/ARoseGarshn1_4.lha
- util/blank/GBlnk36.lha
- util/blank/GBlankM1.lha
- util/blank/GBlankM2.lha
- util/blank/NoGBlank_1_1.lha - (ist nicht ganz so toll)
- util/blank/SBTimeModule.lha
- util/blank/ShowPics.lha - (um Bilder mit dem Garshneblanker 'Executor' anzuzeigen)
- util/blank/SwazBlanker27.lha

Ich muß dazu sagen, daß ich selbst keinen Aminet-Zugang habe (mangels Modem), und nur die Aminet-CDs kenne. Deshalb können die Versionsnummern schon veraltet sein. Bei dieser Gelegenheit möchte ich mich auch bei Martin Schulze bedanken, der als Aminet-Administrator unser Madhouse-Archiv ins Netz kopiert.

Bitte denkt daran, daß die Garshne- und Swaz-Blanker eine eigene Library haben, die bei den Zusatzpaketen nicht dabei ist! Beginnt also beim Importieren mit dem Inhalt von SwazBlanker27.lha bzw. GBlank36.lha.

Ist das Archiv erstmal bei Euch, müßt Ihr es noch entpacken. Dann auf einen der Import-Buttons klicken, je nachdem, ob Ihr jetzt Swaz- oder Garshneblankers importieren wollt. Schon geht ein Fenster auf.

Hinweise / Hints

Naja, die solltet Ihr ruhig auch noch lesen. Sonst habe ich den ganzen Kram wieder umsonst geschrieben...

"ImportHelp"-Verzeichnis

Sicher habt Ihr gemerkt, daß zur Madhouse-Distribution ein Verzeichnis mit diesem Namen gehört. Darin sind lauter Gadget-Dateien, aber mit dem Namen der Blanker, zu dem sie gehören, z.B. "Garshne.Clock" und viele andere.Dieses Verzeichnis muß in diesem Feld eingetragen werden, rechts ist wieder der Popup-Button, der einen Filerequester aufmacht.
Wozu sind diese Dateien gut? Madhouse kennt diverse Informationen seiner eigenen Blanker. Beim Import wird versucht, diese Infos auch aus den fremden Blankern zu gewinnen, was aber je nach Format nie ganz oder oft teilweise gelingt. Die so gefundenen Infos werden dann in der Gadget-Datei abgelegt, unter "CHUNK:BLANKERINFO". Fehlende Angaben werden durch ‘?’ oder andere Defaults ersetzt.
Z.B. weiß Madhouse bei bisher keinem anderen Format, ob die Blanker die Workbench kopieren und darauf herumzeichnen. Deshalb wird dann erstmal ‘Unbekannt’ eingesetzt, was dann im Fall der Fälle als ‘Nein’ ausgewertet wird. (Der Benutzer kann das ‘Unbekannt’ auf ‘Ja’ oder ‘Nein’ auf der Blanker-Seite ändern, siehe dort). Es gibt noch weitere Beispiele.
Deshalb habe ich im ImportHelp-Verzeichnis die Daten normal importiert und dann so ergänzt, daß sie vollständig bzw. richtig sind. Und zwar für alle Blanker, die ich so erreichen konnte.
Wenn Ihr dann das Verzeichnis richtig einstellt, werden die fehlenden Infos aus den ImportHelp-Dateien übernommen. Ist für einen Blanker keine Datei erreichbar (weil er neu ist oder weil Ihr das Verzeichnis doch nicht eingestellt habt), wird dies bei den Hinweisen/Hints angezeigt.
Habt Ihr neue Swaz- oder Garshne-Blanker, würde ich mich sehr freuen, wenn Ihr die bei der Registration mitschickt. Ich kann nämlich nur zweimal im Jahr ans Aminet, wenn das neue Aminet Set herauskommt...

Blanker-Quelle / Blanker Source

ist das wichtigste Gadget. Mit dem Popup-Button rechts den Filerequester aufrufen, und in diesem das Verzeichnis auswählen, in dem sich die Blankermodule befinden. Wollt Ihr nicht aus dem originalen Distributionsarchiv, sondern aus einem Zusatzpaket importieren, solltet Ihr die Blanker vielleicht vorher in ein gemeinsames Verzeichnis kopieren (meistens haben die noch Dateinamenendungen, die Euch die Wahl des richtigen Blankers für Euren Prozessor ermöglichen).
Das mit den Namensendungen und den verschiedenen Versionen je nach Prozessor halte ich im Übrigen für totalen Blödsinn. Schaut Euch doch mal die Längen der ausführbaren Blankerdateien an, die sind doch alle fast gleich lang! D.h., sie beinhalten alle fast die gleichen Prozessoranweisungen. Wer nicht gerade in Assembler programmiert hat, der stellt einfach ein Cyclegadget in den Compileroptionen um und hat einen prozessorspezifischen Ausgabecode. Installiert doch mal probehalber die 68000-Version, wenn Ihr einen schnelleren Prozessor habt, der Geschwindigk-

eitsnachteil ist gleich Null.

Blanker-Ziel / blanker dest.

ist das normale Madhouse-Blankers/-Verzeichnis. Da gehören die Dinger ja auch hin, oder?!

Library-Datei Quelle / Library-File source

wer nicht im Begriff ist, ein Zusatzpaket zu installieren, kann hier die Library des Blankers auswählen. Sie muß unbedingt im Libs:-Verzeichnis stehen. Wie in den Hinweisen für Swaz erklärt, hat der Swazblanker noch eine zweite Library, die matrix.library. Die muß man auch noch kopieren, entweder mit einem Directory-Tool oder wieder mit der Import-Funktion.

Anleitungs-Quelle / Documentation source - nur Swaz!

Die Swazblanker haben nicht eine Datei mit Anweisungen zum Aufbau des Konfigurationsfensters, (wie die gadget-Datei in Madhouse oder die .ifc-Datei bei Garshne), sondern bauen ihr Fenster auf Befehl selbst auf. Diese Konfigurationsfenster haben dann auch noch ein Menü, über das sich vor allem ein AmigaGuide-Dokument mit der Anleitung für den Blanker aufrufen läßt. Damit dieses Dokument gefunden wird, kann Madhouse es in das Blankerunterverzeichnis einreihen. Mit diesem Gadget kann man deshalb bestimmen, wo diese Dokumente liegen.
Wieder dient der Filerequester nur zur Auswahl eines Verzeichnisses. Und zwar solltet Ihr das Verzeichnis "docs/blankers" aus dem original-Swazblanker-Archiv auswählen. Wer nur ein Zusatzpaket importiert, läßt das Feld einfach leer.

Anleitungs-Ziel / Documentation dest.

Alles klar. Siehe oben.

Fortschritt / Progress

Soll ich jetzt wieder behaupten, ich hätte die Import-Funktion nur geschrieben, um ein Gauge-Object von MUI auf den Screen zu bringen? Nö. Das hat sich einfach so angeboten. Was dieses Gadget anzeigt? Na, den Fortschritt! Eine Umfrage, "Ist die Menschheit fortschrittlich?", hat mal ergeben, daß viele Leute uns zwar für technisch fortgeschritten halten, aber unser soziales Zusammenleben nicht.

Okay und Abbruch / Cancel

Mit Okay geht’s dann halt voll los. Mit Abbruch geht das Fenster voll zu.

Wie man die importierten Blanker updaten kann

Natürlich ist es durchaus möglich, daß eine neue Version eines Blankers erscheint. Der MadhouseConfigEd ist darauf vorbereitet: Importiert einfach das Blankerverzeichnis, das die Updates enthält, neu. Zwangsläufig wird Madhouse dabei auf Blanker stoßen, die den gleichen Namen haben. Sind sie zusätzlich vom gleichen ‘Typ’ (also werden über das gleiche Protokoll angesprochen), fragt Madhouse mit einem Requester nach, ob der betreffende Blanker ersetzt werden soll, oder ob der Import bei diesem

Blanker nicht stattfinden soll ('Überspringen/Skip'). Für die Zwecke eines Updates wählt Ihr natürlich 'Ersetzen/Replace' aus.

Bitte beachtet,

daß Madhouse Shareware ist. Deshalb kann man zwar im ConfigEd andere Blankerformate importieren, starten und konfigurieren, aber Madhouse wird diese Blanker nicht anzeigen. Dies ist die einzige Funktion, die nur mit Keyfile funktioniert. Bitte laßt Euch registrieren, der Preis ist nicht überteuert. Nur 20,- DM.

1.7 Die Gadgets am unteren Fensterrand

Speichern / Save

Schließt das Fenster und sichert die Optionen. Dann werden Die Optionen aller MUI-Seiten, jedoch nicht die Optionen in den blankereigenen Fenstern (die haben einen extra Okay-Knopf) nach ENVARC: bzw. ENV:Madhouse.prefs gespeichert.

Benutzen / Use

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

Entfernen / Remove

Beendet Madhouse, die Einstellungen bleiben so, wie sie am Start des ConfigEds waren.

Auch ein 'Abbruch' ist möglich:

Wie 'Entfernen', aber Madhouse läuft weiterhin im Hintergrund. Ihr könnt den Button nicht finden? Er ist das Close-Gadget, links oben im Fenster des ConfigEds!

1.8 Die Liste

Ab Madhouse 2.01 gibt's im ConfigEd nur noch eine Liste. Sie übernimmt aber genau die Funktionen, die zuvor mit zwei Listen erreicht wurden:

- Der Zustand des Häkchensymbols bestimmt, ob Madhouse die Blanker zum Blanken heranzieht. Aus den hier ausgewählten Blankern wird dann per erweiterter Zufallsfunktion einer ausgesucht, mit dem Madhouse voll abblankt.
- Mit einem einfachen Klick auf einen Blankernamen werden Informationen über ihn auf der Blanker-Seite angezeigt, mann kann auch ein paar seiner Internas auf dieser Seite ändern.
- Ein Doppelklick auf einen Blankernamen öffnet das BlankerPrefs-Fenster des Blankers (wenn nicht schon eines offen ist).

Ich hoffe, die Einteilung in Kategorien gefällt Euch. Die kleinen Icons

dafür hat hauptsächlich Aicke zusammengebastelt. Die alte Liste wurde nämlich mit über sechzig Einträgen etwas unübersichtlich...

1.9 System - Seite

Wenn hier alles schraffiert ist und fast nix geht, sollte man doch erst den Workshop lesen. Stichwort Blankerpfad einstellen.

Import - Gadgets

Die Importfunktionen für Swaz- und Garshne-Blankermodule erweitert Madhouse ganz schön... über sechzig Blanker in der Liste sind so möglich! Alle Details zum Import erkläre ich in einem separaten Kapitel: Die Import-Funktion.

Pfad / Path

muß den Pfadnamen des Verzeichnisses enthalten, daß die Unterverzeichnisse mit den Madhouse-Blankern enthält. Mit dem Popup-Button rechts erhält man einen Filerequester, den man nun solange traktieren sollte, bis er Namen wie ‘FlyingToasters’ u.ä. enthält. Mit dem ‘Ok’-Gadget des Filerequesters ist die Sache dann gegessen, der MadhouseConfigEd lädt fleißig die Blanker und stellt sie in der Liste dar, die sich von nun an benutzen läßt.
Bitte beachten: Madhouse benötigt einen kompletten Pfad mit dem Namen der Festplatte darin. Also sollte schon ein Doppelpunkt im Pfad sein, vor dem etwas steht. Um den Pfad zu überprüfen, benutzt Madhouse die Datei ‘the_right_drawer’, die sich im Blankers-Verzeichnis befindet. Also löscht die nicht!

Blanker-Start / Blank Time

Von Madhouse merkt man beim Arbeiten herzlich wenig. Hört man jedoch auf, den Computer zu benutzen (ja Dödies, Ihr müßt dazu die Maus loslassen), geht’s ab. Dann wird wie wild losgeblankt. Die dazu nötige Zeit (also die Zeitspanne zwischen Mausloslassen und abblanken) kann man mit diesem Slider einstellen. Ja! Das geht! Wieso gibt es überhaupt Anwender, die so’n Schrott in einer Anleitung nachschlagen? Schreibt uns! Baut mich auf! Sonst laß ich bald diese lächerliche Anleitung weg!

Dimmer-Start / Dimm Time

Vorsicht, hier werden gleich zwei Gadgets erklärt. Aber vielleicht sollte ich erstmal sagen was ein Dimmer ist? Nö.
Also, mit dem Häkchen-gadget (Checkmark) vor dem Wort ‘Dimmer-Start’ könnt Ihr erstmal einstellen, ob Ihr überhaupt den Dimmer benutzen wollt. (Hey Schlaffies: wer nicht weiß, was ein Dimmer ist, wählt dieses Gadget natürlich erst recht an. Nicht immer so zaghaft!)
Mit dem Slider auf der rechten Seite bestimmt man, ähnlich wie bei Blanker-Start, wann der Dimmer losgeht. Weil man aber naturbedingt erst dimmen und danach blanken muß, kann man den Slider nicht über die Grenze von Blanker-Start hinausschieben. Wer’s trotzdem tut, der verändert den Slider darüber. Außerdem sollte zwischen den Werten von Dimmer-Start und Blanker-Start eine Differenz von 10 bis 60 Sekunden (ja, wir messen hier

übrigens alles in Sekunden) bestehen, sonst muß der Dimmer aufhören, bevor er richtig fertiggedimmt hat.

Textanzeiger / Text viewer

Nein, das ist kein Scherz. Hier sollte man seinen Lieblings-Textanzeiger einstellen, und zwar folgendermaßen: Wenn der Textviewer C:Muchmore heißt, muß man C:Muchmore RAM:Madhouse_Storage/errors einstellen. Der Textviewer wird dann mit den Blanker-Fehlerdaten aufgerufen, falls die Blanker Fehlerdaten produzieren. Damit das Hintergrundprogramm 'Madhouse' recht klein bleibt, wurde auf eine grafische Oberfläche zur Anzeige dieser Fehler verzichtet, stattdessen erstellt Madhouse eine Datei und läßt sie so anzeigen. Vermutlich wird sowieso kaum ein User in die Situation geraten, diese Funktion überhaupt in Aktion zu sehen...

Blanker wechseln / Change Blankers

Nachdem Madhouse einen Blanker gestartet hat (einen, der in der Liste links ausgewählt wurde), könnte dieser theoretisch solange laufen, bis der Benutzer eine Taste drückt (auf der Tastatur oder der Maus). Das ist praktisch der Fall, wenn man 'Blanker wechseln' ausschaltet. Schaltet man das Ding aber ein, kann man zusätzlich jedem Blanker eine Zeit in Minuten zuordnen, nach der er sich selbst beendet. Dann startet Madhouse einen neuen Blanker. (Diese Zeit kann auf der Blanker-Seite eingestellt werden.)

1.10 Optionen I - Seite / Options I (Andere O

Bei aktiver CPU / 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 machte...
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 dieses Gadget. Es hat drei mögliche Zustände:

- o alle Blanker benutzen/use all Blankers kümmert sich nicht um die CPU, so daß der o.a. Fall eintreten könnte.
- o 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.
- o nichts anzeigen/show nothing zeigt den schwarzen Screen immer, wenn

die CPU beschäftigt ist.
Wenn Ihr wissen wollt, welche Blanker denn nun rechenintensiv sind und welche nicht, dann schaut einfach auf die Blanker-Seite und klickt den fraglichen Blanker einmal an. Die Antwort steht dann im Feld CPU-Belastung/CPU load, doch dazu später mehr.

Sound

Mit diesem Gadget 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:

- die medplayer.library vom MED-Programmierer Teijo Kinnunen, die sinngemäß nur MED spielt und
- 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 automatisssch 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. Wieder was gelernt. Nein, ich kann kein Latein.)

1.11 Optionen I - Seite / Options I (Dimmer)

Na gut, jetzt erkläre ich den Dimmer. Der Dimmer läuft erst ab OS 3.0, weil das Dimmen mit nur 4096 Farben (OS 2.1 und darunter) schrecklich aussehen würde. Deshalb sind in diesem Fall die Dimmer-Einstellungen nicht verfügbar (schraffiert dargestellt).
Ein Dimmer macht den Bildschirm dunkler, und zwar durch langsames (oder schnelles) Ausblenden. In Verbindung mit dem Dimmer-Start Slider auf der System-Seite kann man von Madhouse durch sanftes Abdimmen erstmal vorgewarnt werden, daß bald ein Blanker folgt. Und so wird der Dimmer eingestellt:

Verzögerung / Delay

Hiermit, und mit dem Schalter

Schritt / Step

läßt sich bestimmen, wie schnell der Bildschirm dunkeler wird. Schritt definiert dabei, um wieviele Farbstufen der Bildschirm auf einmal dunkler wird. Verzögerung gibt die Anzahl der 1/50stel Sekunden an, die zwischen den Schritten gewartet wird. Am besten, man läßt immer einen der beiden Slider auf 1.

WB-Copperliste killen / Kill WB-Copperlists

Was soll hier gekillt werden? Es geht um eine sog. Copperliste, also ein Programm für den Copper, einer der Customchips des Amigas. Der Copper kann direkt die Metallstiftchen, wo Ihr Euren Monitor angeschlossen habt, beeinflussen, und zeigt sich bei einem entsprechend installiertem Copperprogramm mit bunten Farbverläufen. Programme wie WBVerlauf (© Christian A. Weber) erzeugen ein Copperprogramm für die Workbench, das normalerweise durch den Dimmer unbeeinflußt bleibt und dann recht häßlich aussieht.
Wählt man jedoch dieses Gadget an, wird die Copperliste beim Dimmen entfernt und nach dem Dimmer wieder hergestellt. Dies ist, genau wie WBVerlauf, ein Hack und läßt sich deshalb abschalten.

Tiefe rot, grün, blau / Depth red, green, blue

Normale Dimmer haben anstatt dieser drei Slider nur einen. Mit dem kann man dann einstellen, wie stark gedimmt wird, von 0 (gar nicht) bis 255 (Bildschirm ist am Ende total schwarz).
Madhouse hat aber drei Slider, mit denen die ‘Dimm-Tiefe’ nicht nur für alle Farbanteile insgesamt, sondern für jede farbkomponente einzeln einstellen kann. Merke: je mehr der Slider rechts steht, desto stärker wird die Farbkomponente gedimmt. Zieht man z.B. Grün auf 255, wird der Bildschirm, wenn der Dimmer fertig ist, kein (überhaupt kein) Grün mehr enthalten.Umgegehrt kann man z.B. Rot auf 40 setzen und Grün und Blau auf 240. Dann bleibt fast nur noch Rot stehen, was zur allseits beliebten Horror-Workbench führt. Um eine nette Farbkombination zu erstellen, bedarf es etwas Übung, also probiert’ das Teil jetzt aus.

Zurückdimmer-Geschwindigkeit / Undimm speed

Normale Dimmer zeigen sofort wieder die ursprüngliche Workbench, nachdem man die Maus bewegt hat. Madhouse bietet noch die Möglichkeit, ebenso sanft zurückzudimmen. Wer das nicht will, stellt den Sider auf as Maximum, ansonsten sind Werte wie 5 ganz brauchbar. Je kleiner der Wert, desto langsamer wird zurückgedimmt.

1.12 Optionen II - Seite / Options II

Sofortiger Start / Immediate Start

Sofortiger Start ermöglicht es, den Mauspfeil in eine Ecke des Bildschirms zu schieben und damit eine Aktion auszulösen. Diese Aktion kann sein:

- Überhaupt nicht blanken. Ist der Mauspfeil in der betreffenden Ecke, wird nicht geblankt, egal wie lange man keine Eingabe macht. Auch der Dimmer wird nicht eingeschaltet.
- Dimmen. Dann wird sofort beim Eintreffen des Mauspfeils in der Ecke

```
      mit dem Dimmen begonnen. Geblankt wird nicht.
    o Blanken. Wie Dimmen, aber der Blanker startet sofort.
Mit den drei Cycle-Gadgets kann man nun jeder Aktion eine Ecke zuordnen.
Will man z.B. die Aktionen ‘Blanken/Blank’ und ‘Kein Blanken/No blanking’
nicht benutzen, stellt man im Cycle ‘Aus/Off’ ein.

Paßwort / Password

Für Zeitgenossen, die anderen alles zutrauen und für Leute, die den Amiga
beruflich nutzen (ja, das gibt’s!), haben wir eine PaßwortFunktion zur
Verfügung gestellt, die sich kaum umgehen läßt (was mir bereits zum
Verhängnis wurde...)
"Kaum" heißt, es wurde durch keinen Versuch die Sicherheit des Einga-
bescreens widerlegt. Und wir haben wirklich alles ausprobiert. Jedoch
kann Madhouse manchmal 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 Benutzen/
Use-Gadget setzen. Dann sollte noch ein möglichst gerissenes Paßwort in
das Text-Feld ‘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 Klein-
schreibung wird fieserweise unterschieden!
```

1.13 Die Merkmale/Tooltypes von Madhouse

```
Madhouse kennt zwei Tooltypes, die im Icon des Madhouse-Programms in
SYS:WBStartup eingetragen werden können.

CONFIGED

Der komplette Pfad des MadhouseConfigEd muß direkt hinter dem "=" ange-
geben werden. Beispiel: "CONFIGED=Work:Madhouse/MadhouseConfigEd". Neben
dem "=" dürfen keine Leerzeichen stehen.

QUIETQUIT

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

1.14 Blanker - Seite

Zur Beachtung

Alle Einstellungen, die auf dieser Seite gemacht werden (also mit Ausnahme der Einstellungen in den BlankerPrefs-Fenstern), sind verloren, wenn Ihr nachher nicht Speichern/Save oder Benutzen/Use benutzt, oder wenn Ihr das Blankers-Verzeichnis neu einlesen laßt (Pfad/Path-Gadget). Mit dem aktiven oder ausgewählten Blanker ist fortan derjenige gemeint, dessen Name in der Liste farbig hinterlegt ist (weil Ihr ein- oder zweimal auf den Namen geklickt habt).

Anzeigedauer / Duration

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

Autor / Author

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

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 ‘nur einfache Blanker/only simple ones’ steht und ein Programm arbeitet - welcher Blanker verwendet werden kann.
‘niedrig/low’ bedeutet eine niedrige Belastung, ‘mittel bis hoch/medium to high’ eine mittlere bis starke Belastung.

Version

In diesem Textfeld wird die Versionsnummer des vorliegenden Blankers angezeigt.

Taskpriorität / Task Priority

Die Exec-Priorität, mit der das Blankerprogramm läuft. Ist normalerweise Null und sollte man eigentlich auch nicht ändern. Bei z.B. BouncingPixel macht es aber Sinn, die Priorität auf ‘1’ zu setzen, denn dieser belastet die CPU fast nicht. Ist in Optionen I, Bei aktiver CPU/CPU active: nur einfache Blanker/only simple ones eingestellt, und wird der Blanker gestartet während der Compi beschäftigt ist, ruckelt der BouncingPixel nicht so und stört trotzdem nicht.
Aber wie gesagt, diese Einstellung würde ich nur selten benutzen.

Protokoll

bezeichnet das Format des Blankerprogramms. Da aber "Format" schon das falsche Wort ist, habe ich die Art, wie Madhouse mit den Blankern umgeht, "Protokoll" genannt. Swaz- Garshne- und Madhouse-Blanker werden nämlich alle unterschiedlich gestartet.

Anzeigedauer für alle Blanker übernehmen / Take Duration for all Blankers

macht genau das, was der Name schon sagt: alle Blanker in der Liste erhalten die Anzeigedauer des aktuellen Blankers. Praktisch, wenn man mit den voreingestellten fünf Minuten nicht zufrieden ist.

Zeigt WB / Shows WB

Madhouse muß wissen, ob der Blanker den vordersten Screen kopiert, und dann darauf Effekte macht, oder ob er sich nicht um den vordersten Screen kümmert. Wie sich der Blanker verhält, könnt Ihr hier ablesen und vor allem auch ändern, wenn ‘Unbekannt/Unknown’ eingetragen ist.
Wozu braucht Madhouse diese Information? Weil Madhouse vor dem Blanken einen kleinen schwarzen Screen aufmacht und diesen nach vorne holt. Werden dann die Blanker gewechselt, sieht der Benutzer nicht immer für ein paar Sekunden die Workbench, sondern den schwarzen Screen, und der stört nicht so.
Schnappt sich aber ein Programm den vordersten Screen, ist das normalerweise dieser ‘Black Screen’, den Madhouse aufmacht. Der wird dann geshufflet oder gefadet oder durchgewormt und so weiter. Damit dies nicht passiert, kann Madhouse mit Hilfe des ‘Zeigt WB’-Gadgets den Screen vor dem Start des Blankers einfach zumachen.
Unbekannt/Unknown sollte übrigens auch nach der Anwendung der Import-Funktion am besten gar nicht bei Euch vorkommen, denn ich habe die Protokolleinstellung für alle Blanker, die ich kenne, vordefiniert.

Blanker löschen / delete Blanker

Der angewählte Blanker wird richtig gelöscht! Nicht nur aus der Liste, sondern auch aus dem Verzeichnis! Diese Funktion wurde nötig, weil sich einige Blanker von Swaz und Garshne überschneiden. Wer braucht denn schon zwei Labyrinth-Blanker? Na eben.

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 ‘Optionen I/Options I’-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 auf den Namen 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 AutoRadio, 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’ im Deli-Tracker ab. Es ist auch praktisch, ‘SongEnd’ auszumachen, dann lädt der DeliTracker kein neues Soundmodul, wenn der Blanker länger läuft als das Lied lang ist.

1.15 Info - Seite

Naja, die Info-Seite zeigt nicht nur das Madhouse-Logo, sondern ist nebenbei auch echt info (rmativ).

1.16 Madhouse - Fehlermeldungen

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 Rückgabewert, 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 zurückliefern, der dann von Madhouse angezeigt werden muß.

Tritt ein Fehler bei einem Blanker auf, und er wurde vom ConfigEd auf Wunsch des Benutzers hin gestartet, zeigt Madhouse den Fehler unten im sog. BlankerPrefs-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 Optionen I), wird nur ein schwarzer Bildschirm angezeigt.
Nach Abbruch eines Blankers / des schwarzen Bildschirms mit Maus oder Tastatur und evtl. Eingabe des Paßworts wird dann der Textviewer, der auf der System - Seite eingestellt sein sollte, aufgerufen. Er zeigt dann an, welcher Blanker welchen Fehler meldete.
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 eventuell die Erläuterung in Alle Blanker.

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...)
Madhouse hätte gern den Namen der Festplattenpartition, auf der sich die Blanker befinden, da relative Pfade immer eine unsichere Sache sind, wenn sich von mehreren Programmen benutzt werden. Also bitte etwas Entsprechendes im Path-Gadget des Hauptfensters eintragen ("Work:Tools/-Madhouse/blanker", etc).

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.

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 ‘Madhouse beende/Remove Madhouse’ beantworten, und tschüß. ‘Nix tun /Do nothing’ sieht den Doppelstart eher als einen Unfall an und macht so weiter wie zuvor.

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 Work-

bench "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 6 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: ... 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 v1.0, BlankerPrefs-Window". 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.)

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 Pfad-angabe 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.

o Wurde diese Meldung durch die Auswahl von "Madhouse" im Tools/-
Hilfsmittelmenü "provoziert", dann lagen offenbar schon die korrekten Einstellungen vor. Madhouse muß nun beendet werden. Dies kann über das Exchange-Programm oder durch den erneuten Programmstart und anschließendem 'Madhouse beenden/Remove Madhouse' geschehen.

Ob sich Madhouse nach diesem Requester beenden muß, ist auch dem RequesterText 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.

(The MadhouseConfigEd can only be started by Madhouse itself.)
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.

1.17 Eigene Blanker erstellen

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:

- 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 schön, wenn das Niveau von Madhouse erhalten bliebe. Die Idee sollte also irgendwie ausgefallen sein.
- 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 einer kleinen Qualtitätskontrolle unterziehen (das hört sich jetzt arrogant an, aber irgendwie muß man sich ja absichern, wenn doch einer mit Lines ankommt).

1.18 Kann ich meine Programmiersprache benutz

Ja.

Das kann man so klar sagen, weil die nötige Programmiersprache nur Screens öffnen, darin zeichnen (klar, wir wollen schließlich einen Blanker) und Dateien lesen und schreiben können muß. Und das kann jede.

Leider enthält jedoch diese Madhouse-Version weniger Blanker, weil Aicke seine alten Blanker alle in AMOS geschrieben hat, die neuen (in E geschrieben) laufen jedoch wieder. AMOS lief nicht in Verbindung mit MUI 3.1, mit älteren MUI-Versionen gab es keine Probleme. Deshalb sind diese Blanker momentan nicht mehr dabei. Also bitte nicht AMOS benutzen.

Außerdem sollte ich noch erwähnen, daß Euer Programm nur im compiliertem Zustand Madhouse-fähig ist, ich rufe nicht extra noch irgendwelche Interpreter auf. Aber Sprachen, zu denen es keine Compiler gibt, fallen mir nicht ein (sogar zu ARexx und AmigaBASIC gibt’s Compiler), und zu jeder aktuellen Sprache gibt es schon gar keinen Interpreter mehr, sondern ausschließlich den Compiler. Solche Compilersprachen heißen z.B. C, E, Modula, Oberon, Pascal, Assembler (ok, hier spricht man von Assemblern und nicht von Compilern, aber das Ergebnis ist das gleiche) und viele mehr.

1.19 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 ‘Anzei-
   gedauer/Duration’-Slider auf der Blanker-Seite) 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
   Blankereigenen Unterverzeichnis 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..., Ihr könnt soviele Optionen einbauen, wie Ihr wollt. Da ein
   Programmlisting solche Sachverhalte am besten verdeutlichen kann, gibt es
   zwei 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. Das andere befindet sich im developers-Verzeichnis und
   ist in C geschrieben. Das ist ein richtiges, handfestes Beispiel.

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 Form einer Zeichenkette anhängen.
  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 im Beispielprogramm veranschaulicht: das C-Demo benutzt die Funktion time() und die Struktur time_t aus <time.h> und arbeitet in Sekunden.
An die Errors-Datei dürfen nur ein paar Zeilen ANGEHANGEN werden, weil Madhouse die Datei erst auswertet, nachdem evtl. mehrere Blanker liefen. So wird ein Überschreiben der Datei verhindert.

Die "prefs"-Datei sieht im Übrigen z.B. so aus:

```
5
10
$Ein toller Text!
22
2
Work:Madhouse/Blankers/MeinBlanker
deutsch
```

Dann hatte der Blanker selbst vier Einstellungen, die man im Prefs-Fenster des Blankers 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. Im Anschluß daran steht der Pfad des blankereigenen Verzeichnisses und die Sprache, die der Benutzer benutzt (bei Amigas mit OS 2.0 "english").
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 (an oder aus, 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
deutsch
```

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.

1.20 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. 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 ‘Pfad/Path’-Stringgadget 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, nicht Anzeigedauer/Duration/’ und Pfad! 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.
Noch besser wäre es, wie gesagt sich gleich an die gadget-Datei heranzuwagen. Ist echt nicht soo schwer.

1.21 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.

Da Madhouse ab der Version 1.2 nicht mehr ohne MUI lauffähig ist, enthält diese Anleitung auch nicht mehr die Erläuterung des Teils der gadget-Datei, die für ein normales Intuition-Fenster zuständig ist. Früher mußte man unbedint ein Intuition-Fenster in der gadget-Datei definieren, und ein MUI-Fester konnte man als Option dazutun, wenn man wollte. Jetzt, wo es nur noch MUI gibt, erkläre ich sofort MUI. Dazu kommt jetzt eine Beispieldatei, deren groben Aufbau ich zunächst erläutere. Weiter unten gibt’s dann erstmal eine Erklärung zu MUI und zu einzelnen Teilen der Datei. Also, jetzt geht’s los:

```
Madhouse v1.0, BlankerPrefs-Window

CHUNK:LOCALE
english,deutsch

CHUNK:MUI-WINDOWLAYOUT
ColumnGroup(2),
  Label( "Co_llision,Ko_llision" ),
  Cycle( "l", "Windowborders|Screen image,Fensterkanten|Bildschirminhalt" ),
End,

CHUNK:BLANKERINFO
Carsten Jahn
1
1
8000
2
Madhouse
```

So, das wars schon. Sicher habt Ihr, wenn Ihr genau hinseht, erkannt, daß es sich um die gadget-Datei von Snow handelt. Die ist nämlich schön einfach, für den Anfang.
Auf vier Dinge ist erstmal zu achten:

- Der Text "Madhouse v1.0, BlankerPrefs-Window" MUSS sich in der ersten Zeile befinden.
- Die Datei ist in ‘Chunks’ aufgebaut, die beliebig gemischt werden können, und die jeweils eine Funktion haben.
- Es dürfen keine Leerzeilen innerhalb der Chunks liegen. Auf Recht-, Groß- und Kleinschreibung wird geachtet.
- Es dürfen Leerzeilen zwischen den Chunks stehen.

Als kleinen Überblick will ich noch die Funktionen der einzelnen Chunks erklären, bevor ich weiter unten genau auf die Details eingehe.

- Der CHUNK:MUI-WINDOWLAYOUT ist das Herzstück der Datei und bestimmt, wie das Fenster Eures Blankers aussieht.
- Der CHUNK:BLANKERINFO wird nicht für das Fenster selbst verwendet, sondern wird beim Einlesen des Madhouse-Gesamt-Verzeichnisses (wenn man einen neuen Pfad einstellt) angesehen. Madhouse erhält hier Informationen, die es dauernd braucht, und merkt sie sich solange, bis der Pfad erneut eingelesen wird. Zu diesen Informationen

gehören z.B. wieviel Stack der Blanker braucht, ob er CPU-Belastend ist usw., doch dazu später mehr.
- Der CHUNK:LOCALE ermöglicht mehrsprachige BlankerPrefs-Fenster ab OS 2.1. Er kann entfallen. Im CHUNK:MUI-WINDOWLAYOUT bezieht man sich auf ihn, indem man Gadgettexte durch Kommas trennt. Auch dazu werde ich gleich ausführlicher.

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

```
  CHUNK:BLANKERINFO
* CHUNK:LOCALE
```

Der CHUNK:MUI-WINDOWLAYOUT muß natürlich auch erklärt werden. 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

1.22 Madhouse-MUI für Leute, die sich schon a

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 Gadgets.

Die Gruppen können einen Titel erhalten, 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.

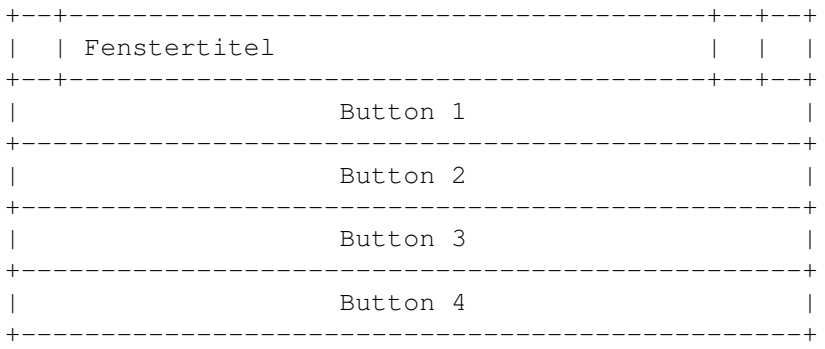
Soll man einen Buchstaben angeben, der zur Tastatursteuerung von Gadgets benutzt werden soll, so muß dieser in Anführungszeichen eingeschlossen sein und nicht in Hochkommas, wie in C sonst üblich.

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. Aber mit den Beispielen kann man schon viel lernen.

1.23 MUI von Null auf Hundert für MUI-Greenho

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 Oberflächen meist viel komplizierter. Deshalb gibt es ein weiteres Feature: neben normalen Gadgets können die Gruppen selbst noch weitere Gruppen enthalten.

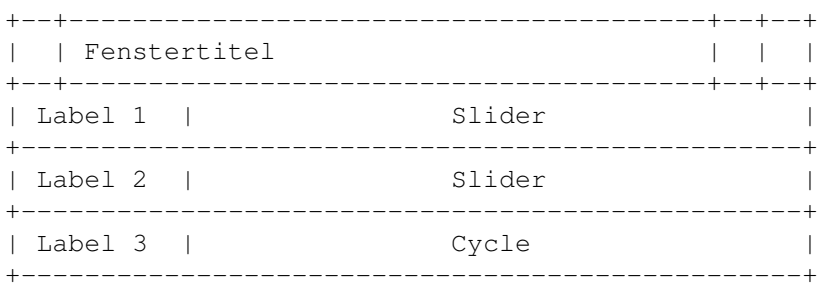
<Lange Denkpause.>

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 Ausdehnungsmöglichkeiten 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 ColumnGruppen 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 KaroPapier, 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, aber natürlich fehlt noch die mehrsprachige Localeanpassung für eine richtig tolle gadget-Datei.) 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 Vergrößerungs-freiheit. 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 horizon-tal 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:

```
          | (Label 1)  | (Slider--------------------------)  |
          +-------------------------------------------------+
          | (Label 2)  | (Slider--------------------------)  |
          +-------------------------------------------------+
          | (Label 3)  | (Checkmark) | (HVSpace------------)  |
          +-------------------------------------------------+
```

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 accent 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 HVSpace 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-WindowlayoutChunks nicht "auf dem Trockenen" erlernt werden. Etwas Übung gehört auch dazu.

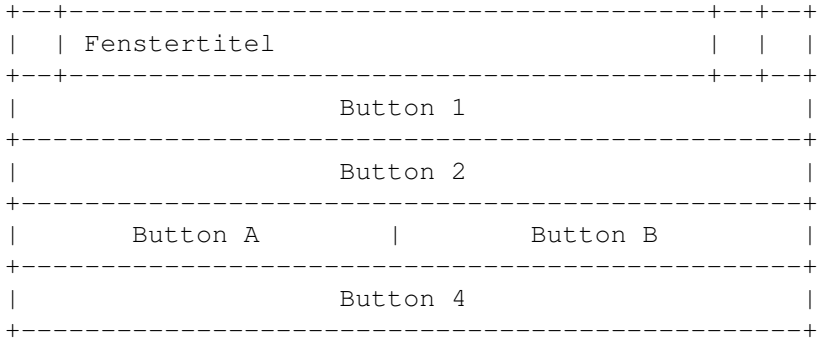
Wer das Grundprinzip verstanden hat, der hat es nun auch schon leichter, in die MUI-Programmierung von richtigen Programmen einzusteigen. Allerdings schreibt man dann diese Group-Geschichten in den Programmcode und läßt sie von einem waschechten Compiler interpretieren, nicht von einem kleinen Programm 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

- 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 BlankerPrefs-Fenster auch genauso gut aus wie "richtige" MUI-Appliationen wie Madhouse und alle anderen.

- 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.

1.24 Die Lösung



1.25 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 Gruppentitel 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.

1.26 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, der 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 Minimal- und Maximalausschlag selbst auch noch anwählbar sind).

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.

1.27 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, hoffentlich daß Madhouse die Einstellungen der Gadgets in der Reihenfolge der Gadget-Deklaration abspeichert. Allerdings haben wir da ein gaaaanz kleines Problem: Man kann mit MUI nicht ein Gadget in die obere Fensterecke legen und dann seine Einstellung zuletzt aus der prefs-Datei lesen wollen. Weil man so ein Gadget für diese Position ja auch nicht zuletzt deklarieren kann.
Natürlich könnte man jetzt den Programmtext, und das Gadget halt zuerst auslesen, 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 (in util/blank/MadAMOS.lha) 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
Sprache des Benutzers

   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
Sprache des Benutzers

   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.
```

1.28 Der CHUNK:BLANKERINFO

```
   Dieser Chunk wird nicht für das BlankerPrefs-Fenster benötigt, sondern er
   wird von Madhouse während des Verzeichnis-Lesens gelesen. Also wenn der
   Benutzer einen neuen Pfad einstellt oder einfach <Return> drückt, wenn
```

der Cursor im Pfad-Gadget steht.

Der Chunk enthält allgemeine Infos zum Blanker.

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-Belastungs- (-Usage) und Stack-Werten.

```
CHUNK:BLANKERINFO
Name
Version
CPU-Usage
Stack
WBDisplay
Protokoll
Kategorie
```

- Name = Der Name des Autors.
- Version = Die Versionsnummer des Blankers.
- CPU-Usage = kann 1 oder 0 sein. Das ist für die 'Bei aktiver CPU/ CPU active:' Einstellung gedacht, Madhouse kann so am schnellsten Ermitteln, ob es sich hier um einen Blanker mit hoher/mittlerer oder kaum vorhandener CPU-Belastung handelt. '0' bedeutet so gut wie keine CPU-Belastung, '1' ist alles andere. Überlegt Euch, ob Ihr auch bei einem laufenden Raytracer Euren Blanker sehen wollt, und vor allem wieviel CPU dem Raytracer dann übrig bleibt. Also im Zweifelsfall lieber 0 als 1 wählen.
  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. Normalerweise legt der Compiler alle Daten, die lokal deklariert werden, auf den Stack. (In C kann man das mit static abschalten, static hat aber auch noch 'Nebenwirkungen', also mal das Compilerhandbuch lesen.)
  Der Wert muß glatt durch vier teilbar sein.
  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!)
- WBDisplay = 1 oder 2. '2' bedeutet, daß der Blanker den vordersten Screen auf einen eigenen kopiert und darauf blankt, wie z.B. Snow oder Shuffle. '1' dagegen heißt, daß der Blanker immer einen eigenen Screen aufmacht. Kann beides der Fall sein (z.B. vom Benutzer einstellbar, sollte man '2' wählen.
  '0' dagegen heißt, daß Ihr Euch nicht sicher seid. Im Cycle-Gadget auf der Blanker-Seite steht dann Unbekannt/Unknown. Dies ist jedoch nur für den Import vorgesehen.
- Protokoll = Madhouse. Jeder, der einen Blanker nach den hier beschriebenen Regeln schreibt, muß hier einfach 'Madhouse' eintragen, natürlich ohne Anführungsstriche.
- Kategorie: Die wird ab Madhouse 2.01 benutzt. Eines der folgenden Schlüsselwörter kann man in diese Zeile schreiben:
    - Anim, für Blanker die mit kleinen bewegten Bildern arbeiten

```
      (wie FlyingToasters);
    o Clock, für alles was mit Uhrzeit und Datum zu tun hat;
    o Pyro, für diverse Feuerwerke und Fontänen;
    o Text, für Blanker die einen (evtl. vom Benutzer eingegebenen
      oder aus einer Datei gelesenen) Text darstellen;
    o Fract, für Effekte die richtigen fraktalen Ursprung haben;
    o View, für Bild- und Animationsanzeiger (im Ggs. zu Anim!);
    o Algo, für algorhythmische Blanker. Weil aber jedes Programm
      irgendwie nur aus Algorhythmen bestehen kann, bezieht sich
      diese Kategorie eher auf Zellularautomaten wie Life;
    o Sim, hauptsächlich für Simulationen im echten Sinne, wenn der
      Blanker Regen oder einen Absturz simuliert;
    o 3D, für alles was vektoriell dreidimensional orientiert ist,
      also für Blanker die (bewegte) Objekte im dreidimensionalem
      Raum zeichnen;
    o Pixel, für Effekte die aus einzelnen, bewegten Punkten
      bestehen, wobei die Punkte keine dreidimensional nachvoll-
      ziehbaren Bewegungen machen und auch kein Feuerwerk sind ;-)
    o Cycle, für Effekte die ohne Colorcycling sehr schnell lang-
      weilig aussehen würden;
    o WB, für alle Effekte die den vordersten bzw. den Workbench
      Screen kopieren und dann damit arbeiten;
    o Geo, wenn zweidimensionale, geometrisch wertvolle Objekte zu
      sehen sind (also Rechtecke, Kreise, Linien in besonderer
      Form);
    o Lines+Splines, wenn einfache Linien oder Splinetypen ge-
      zeichnet werden;
    o ?, wenn Euch mal wieder nix einfällt, oder wenn keine dieser
      Kategorien paßt.
  Das waren also die offiziellen Definitionen, aber wie immer könnt
  Ihr Euch wohl am besten für eine Kategorie entscheiden, wenn Ihr
  Euch die bereits vorhandenen Blanker anseht.
  Die Einteilung in Kategorien betrifft im Moment nur die Liste,
  Madhouse kann Euren Blanker dann richtig einsortieren.
```

1.29 Der CHUNK:LOCALE

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

   Bei der Lokalisierung von Madhouse stoß ich auf das Problem der Blan-
   kerPrefs-Fenster, die ja nicht auch noch alle ihren eigenen catalog haben
   sollten.
   Deshalb muß - falls dieser CHUNK in der Datei vorkommt - der Chunk
   MUI-WINDOWLAYOUT in den Angaben für Gadgettexte jeweils alle verschiede-
   nen Sprachen beinhalten. Um das nicht zu kompliziert zu erklären, zeige
   ich Euch ein Beispiel:

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,
```

Dieser Chunk aus der ehemaligen CrazyPixel-gadget-Datei 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.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 Beispiel 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 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.

1.30 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.

1.31 Alle Blanker in der Übersicht

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

Leider gab es Probleme mit den AMOS-Blankern und MUI 3.1. Alle Blanker von Aicke wurden von ihm zuerst in AMOS geschrieben. Deshalb sind sie momentan nicht dabei, bis sich entweder MUI wieder ändert oder ich es schaffe, ein Problem zu beseitigen. Bis auf weiteres sind sie jedoch unter ‘MadAMOS.lha’ im Aminet erhältlich, falls Ihr trotzdem probieren wollt, ob Euch die Einschränkung stört (sie sind trotzdem, auch mit MUI 3.1, noch recht funktionsfähig).

Aicke ist aber bereits dabei, seine Blanker in AmigaE nachzuprogrammieren, damit sie wieder da sind. Ich hoffe, auch die Zeit zu finden, mal was an meinen Blankern zu machen (Fireworks ist doch schon recht angestaubt...)

```
BouncingPixel
  Fireworks
     Flow
FlyingToasters
   Glitter
    Nautic
   Shuffle
     Snow
    Soccer
    Stars
    Waves
    Wusel
```

1.32 Die Blanker in der Übersicht - BouncingP

BouncingPixel

Von Aicke Schulz, Version 1.1

Nachdem Aicke jetzt mit AMOS schlußgemacht hat, besinnt er sich auf E. Probeweise hat er auch mal was blankermäßiges versucht, und portierte den ehemaligen CrazyPixel auf zwei E-Programme: BouncingPixel und Wusel.

Der Punkt hüpft jetzt noch fröhlicher.
Die Farben sind noch kräftiger.
Der Code ist noch schneller.
Aicke ist noch langsamer.

Der Blanker läuft. (war bei AMOS keine Selbstverständlichkeit. -- Ok, genug auf AMOS ’rumgehackt. =)

--- Keine Optionen ---

CPU-Belastung: Niedrig.

1.33 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 entstehen 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 vertreten 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 einstellen, wie groß die Pausen bei wenigen Punkten werden. Hier muß man einfach etwas herumprobieren, bis der Effekt nicht mehr schneller und langsamer wird.

CPU-Belastung: Hoch.

1.34 Die Blanker in der Übersicht - Flow

Flow

Von Aicke Schulz, Version 1.

Flow hat so gar nichts mit Flöhen zu tun und wird trotzdem so ausgesprochen. (Alles klar; ich hab’ die Dödies schon wieder ‘Floff’ sagen hören.) Was wird bei Floff nun eigentlich geflofft? Das kann ich auch nicht so direkt sagen, aber wahrscheinlich sind das wieder irgendwelche Interferenzen, die durch Colorcycling floffen. Glücklicherweise kann man Floffgeschwindigkeit und Floffrichtung auch noch einstellen, und ab und zu wird das Bild neu durchgeflofft.

- Geschwindigkeit/Cyclespeed Also, das wäre dann die Floffgeschwindigkeit.
- Richtung/Cycledirection Damit könnt Ihr vorwärts, rückwärts und zufällig (also entweder vorwärts oder rückwärts) floffen.
- Warten/Wait Der zeitliche Abstand in Sekunden, den Floff wartet, bis neu durchgeflofft wird.
- Farben/Colors Floff hat mehrere Floffpaletten. Hier könnt Ihr eine aussuchen, und wenn sich Aicke bemüht hat, gibt’s hier noch einen

‘Zufall/Random’-Eintrag. Der ist nur für Leute gedacht, die sich nicht entscheiden können. Dann bestimmt der Zufall, was für eine Palette erscheint. (Natürlich gibt es keinen Zufall. Ihr könnt keine zufällige Entscheidung treffen, die Lottozahlen sind auch nicht zufällig. Sondern nur nicht vorherberechenbar. Wie soll dann bitteschön ein Computer zufällig rechnen können, denn er tut ja nichts anderes als rechnen, und die Mathematik war noch nie zufällig.

CPU-Belastung: Hoch.

1.35 Die Blanker in der Übersicht - FlyingToa

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 eine Nuß-Nugat-Creme (deren Name wir natürlich nicht nennen werden) 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 Durationzeit 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 könnte.

CPU-Belastung: Hoch.

1.36 Die Blanker in der Übersicht - Glitter

Glitter

Von Aicke Schulz, Version 1.1

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 Animation. Warum auch?
- Geschwindigkeit/Speed Geschwindigkeit des "Glitterns".

CPU-Belastung: Niedrig.

1.37 Die Blanker in der Übersicht - Nautic

Nautic

Von Carsten Jahn, Version 1.

Ich habe mal etwas ähnliches in einer Demo gesehen, und beim Nachprogrammieren stark verändert und erweitert. Durch einen Zeichentrick benötigt der Blanker acht Bitplanes, weshalb man einen AGA-Amiga oder eine entsprechende Grafikkarte haben muß, um ihn benutzen zu können.

- Schnell/Fast Schaltet in den etwas schnelleren Modus.
- Warten/Wait Wieviele 1/5tel Sekunden gewartet wird nachdem ein fertiges Bild gezeichnet wurde. Nach der Wartezeit wird es gelöscht.
- Zahl der Kanten/Number of edges bestimmt die Anzahl der Kanten, die die Zeichenschablone hat. Weniger ist mehr: ich benutze im Moment ‘4’.
- Zufall/Random Die Anzahl der Kanten ist immer unterschiedlich, egal was Ihr einstellt.
- Farben/Colors Welche Farbkombination verwendet werden soll.
- Monochrom/e Einige Farbpaletten (die ersten beiden) lassen sich mit kleinen Farbverfälschungen darstellen, die recht nett aussehen. Mit diesem Schalter lassen sich die Verfälschungen abstellen (kein Haken) oder einstellen.

CPU-Belastung: Niedrig.

1.38 Die Blanker in der Übersicht - Shuffle

Shuffle

Von Carsten Jahn, Version 1.1

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 Grafik-

karten 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 wieder hergestellt werden kann. 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.
- X/Y Anzahl/Number: Damit könnt Ihr die Größe der Tiles einstellen.

Bitte beachtet, daß eine lange ‘Anzeigedauer/Duration’-Zeit, eine hohe Geschwindigkeit und der Modus ‘Auflösen/Restore’ zu einer recht großen Rechenzeit vor dem Start des Blankers führen kann.

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.

1.39 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, Bei Speichermangel:260 kB.

1.40 Die Blanker in der Übersicht - Snow

Snow

Von Carsten Jahn, Version 1

Naja, den Schnee wollte ich eigentlich noch vor Weihnachten bringen. Iss halt nix geworden. Aber Dezember haben wir ja immer noch (fast). Und in Alaska schneit’s ja immer. (Oder nie?) Jedenfalls solltet Ihr damit Eure Workbench mal so richtig zuschneien lassen.

- Kollision / Collision bezieht sich auf die Kollisionseigenschaft des Schnees. Der sollte zwar immer an den Fenstern liegen bleiben, kann aber auch auf die einzelnen Pixel des vordersten Bildschirms acht geben (Bildschirminhalt/Screen image).

CPU-Belastung: Hoch.

1.41 Die Blanker in der Übersicht - Stars

Stars

Von Carsten Jahn, Version 3

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 ich 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.
Wenn Stars bei Euch nur Schrott auf den Bildschirm bringt, habt Ihr die falsche Version (für Amigas ohne Grafikkarte) installiert.

- 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.
        - o Ändern / Changes bestimmt, wie oft die Geschwindigkeit gewechselt werden soll.
        - o Start Startgeschwindigkeit.
    - o Drehungen / Turns (Bewegungen UM die Z-Achse, Drehungen des Betrachters)
        - o Maximum max. Drehmoment.
        - o Beschleunigung / Acceleration Beschleunigung der Drehung.
        - o Ändern / Changes bestimmt, wie oft die Drehrichtung wechselt.

        - o Start Startgeschwindigkeit.
    - o Verschiebungen / Shifts (Bewegungen der X- und Y-Achse)
        - o Geschw. / Speed Dieser Wert legt fest, wie eng die Kurven sind.
        - o Gebrauch / Usage bestimmt, wie oft der "Kurvenmodus" verwendet wird; 0 = nie, 10 = immer.
        - o X/Y Hiermit wird definiert, ob sich die Verschiebungen nur auf eine Achse, beide oder gar keine beziehen.

CPU-Belastung: Hoch.

1.42 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 'reinhauen. 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 Einstellungen zum Zeichnen der Waves:

- o Wellengröße / WaveSize beeinflußt die gesamt-Dicke der Waves.
- o 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 Ausprobieren. 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 Paßwort hält doch keine Katze davon ab, alles zu verwüsten!)

CPU-Belastung: Niedrig.

1.43 Die Blanker in der Übersicht - Wusel

Wusel

Von Aicke Schulz, Version 1.1

Ebenfalls aus alten CrazyPixel-Beständen reinkarniert ist Wusel. Der Wusel ist schneller geworden. Die Länge kann man immernoch einstellen. Wusel läuft. Toll.

- Wusellänge / Wusellength Länge des Wusels in Quadratpixeln.
- Geschwindigkeit / Speed Der Wusel wuselt auch unterschiedlich schnell. Wie wohl die Intelligentesten unter Euch schon erwartet haben werden, ist 'ziemlich schnell/quite fast' die schnellste Einstellung.

CPU-Belastung: Niedrig.

1.44 Die Autoren

Die Autoren

Hallo! Wir sind die Autoren, und wir sagen Euch: Madhouse ist Shareware. Das heißt, daß die Benutzung des Programms nicht kostenlos ist.
Also druckt bitte das beiliegende Registrationsformular aus, nehmt einen 20-Mark-Schein und tut beides in einen Briefumschlag. Brav frankieren, und abschicken. In den nächsten Tagen werdet Ihr dann eine Diskette zugeschickt bekommen. Auf der ist in erster Linie das Keyfile, mit dem

Namen "Madhouse.key". Dieses müßt Ihr in Euer S:-Verzeichnis (also die "s"-Schublade Eurer Bootpartition) kopieren.
Wird Madhouse jetzt gestartet, wird es das Keyfile erkennen und Euch

- nicht mehr mit "Nervscreens" nerven
- ermöglichen, die fremden Blankerformate zu nutzen (die Blanker kann man sich sonst nur im MadhouseConfigEd ansehen).

Leider waren diese Einschränkungen nötig, weil ja sonst doch kein Mensch was bezahlt. So ist das nun mal. Liegt vielleicht an unserer total-kapitalistischen Auffassungsgabe: Preis und Leistung. Wenn es für einen Preis keine zusätzliche Leistung gibt, warum dann den Preis bezahlen?

Also, um registrierte Benutzer zu werden, könnt Ihr uns entweder 20 DM schicken oder überweisen. Konto-Nr. steht auf dem Formular. Das Formular solltet Ihr auf jeden Fall schicken.

Vielen Dank!

Bitte schickt Eure Registrationen (oder alles, was Euch so bedrückt und nervt [keine Schwestern und Schwiegermütter schicken!!]) an

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

Oder ab März ’96 an
Carsten Jahn
Kuckucksruf 34
D-16761 Stolpe-Süd
Germany

Wir übernehmen keinerlei Haftung für Schäden, die durch die Benutzung des Programms entstehen. Das Madhouse-Archiv und alle damit in Verbindung stehenden Archive sind frei verteilbar. Wir begrüßen auch die Pressung der Daten auf CD. Auch in entpackter Form darf das Verzeichnis weiter-kopiert werden, solange nichts daran verändert wird. Keine der Daten dieses Archives sind mit Kopier- oder Nutzungsrechten Dritter behaftet. Bei den Verfahren und Vorgehensweisen bin ich mir da nicht so sicher; ich bin doch kein Rechtsanwalt! (Und ich nehme mir auch keinen, da müßten sich schon allein fünfzig Leute registrieren lassen um die Kosten zu decken.)

Tja, in meiner Eingabedatei (ich schreibe diese Anleitung nicht direkt, sondern benutze ein selbstgemachtes Formatierungsprogramm, die Amiga-Guide-Syntax macht mich so alle) bin ich jetzt in Zeile 2608. Unsere gemeinsame Zeit in meiner Anleitung ist bald zu Ende. Aber in der nächsten Anleitung sehen wir uns wieder. Mein besonderer Dank gilt den Leuten, die sich schon in der Version 1.1 registrieren ließen. Ihr seid echt spitze.

Tschau, bis zur nächsten Anleitung

Euer Carsten.