Trashmaster

COLLABORATORS

	TITLE : Trashmaster		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 1, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Trashmaster

1.1   Trashmaster.guide

```
                                    TRASHMASTER V1.7

               The ultimate in byte disposal for the Amiga.

           Copyright (C) 1994 By Aric R Caley and Greywire designs

                                   3 July 1994



                      Who what where when why

                      How

                      Installation

                      Usage

                      Options

                      Wish list

                      Formats and Templates
```

1.2   Trashmaster.guide/Who what where when why

```
Who, what, where, when, why
***************************

WHO
===

   Written by Aric R Caley, AKA Dances V2.0, Dances With Coyotes, Major, Mr
Coyote, and other handles/nicknames.  :) See the readme file if you wish to
```

contact me.

WHAT
====

   See Readme!

WHERE
=====

   Best place is in your WBStartup drawer!

WHEN
====

   The current version, 1.7, was released on 3 July 1994. The first public
release of Trashmaster was Jan 23 1992.

WHY
===

   I had always wanted to write something like this but until now, I couldnt
do it. Not that it was particularly easy to do, even under Workbench 2.0.

1.3 Trashmaster.guide/How

                  How I did it
****

   The main problem is with getting finicky old Workbench to stay "in sync"
with the filing system.. IE, removing its icons when a file is deleted.
Thanks to a new function called DeleteDiskObject(), it's possible to get
workbench to remove an icon. Unfortunately Workbench still is kinda
brain-dead... when you have it display ALL files, it spontaneously creates
icons in memory for files that dont have them... which results in
DeleteDiskObject() not working (no file on disk) and Workbench not removeing
the icon. The only solution I could come up with is to PutDiskObject() and
then delete this icon, which works more or less pretty well. There are
still, however, problems. Workbench, it seems, will "lock" a directory if
its window is open on the Workbench screen. This means I can't delete it
until it's closed... not too big a deal, since when you delete the icon the
window closes automaticaly. But what if something else has that directory
locked? Then I couldn't delete it after all... that means I need to put
the icon back! heheh. But I think I've come up with a good solution.

   Another problem is that there's no way to find out where the AppIcon is,
within the WorkBench window -- so there is no way to implement a snapshot
option within your application (of course, the root problem is that
Workbench doesn't provide any way to "hook" into the Workbench operations
like snapshot, info, etc. See below for my solution to this).

   Finaly, one last problem exists that I couldn't fix. If you drop a
drawer onto Trashmaster that does not have a directory (but obviously has an
.info), then Trashmaster can't delete it. The reason why is that Workbench

tries to get a lock on any directory icons, which it passes to Trashmaster,
but since there is no directory it can't lock it. So Trashmaster gets a
message with no arguments and it can't figure things out. This is only a
minor problem.

   This seems like a good place to voice some opinions on Workbench 2.0, and
what I'd like to see in the future.

                    Wish list

1.4   Trashmaster.guide/Wish list

Wish list
****

   * DiskObjects should be a BOOPSI class.

        * An "icon" class would have methods for all the normal Workbench
          operations; like info, double click, delete, snapshot, copy/move,
          etc. That way, an AppIcon could give an "about" requester with
          the info menu function, and behave in every way just like a
          regular icon. Say you had an icon for a network machine... info
          could put up a requester saying what the node's name was, CPU
          class, whether it was Ethernet or serial, etc. neat huh? A
          Drawer could open it's own type of list window.. perhaps one
          with a "tree" display. You could replace the standard operation
          with new ones (via hooks into the DiskObject class). This makes
          Workbench extremely flexible and extendable and would allow third
          party people to add tons of functionality to Workbench (I should
          be able to make Workbench as powerfull as DiskMaster or DirOpus!)

        * This BOOPSI icon would be able to have more than just a bitmap
          image; like actual structured drawing objects. Of course, this
          introduces the problem of how to store objects on disk.

        *   Any application would now be able to use icons easily (they
          dont have to have anything to do with Workbench). You could even
          drag icons from an application to a Workbench window to save a
          file...

   * Clipboard support, with a "view clipboard" window. Cut and paste icon
     names/file names, icon images, etc.

   * Definable default tools for projects, with file recognition so it knows
     what tool to run (in case there's no icon or the normal tool can't be
     found). And how about a small database that contains application
     names and then a path to where it is? Actualy, I guess you could have
     a "Applications" directory, with icons that are linked to wherever the
     app is -- and Workbench would check here first for tools.

   * "Left out" icons could be done in the same way, so that the app's icon
     is still visible in it's Drawer, and so that Snapshoting the left out
     icon doesn't ruin it's original coordinates.

* Something needs to be done about the list windows.. they are slow and
  clumsy, and are missing what I think is a great feature: a "Tree"
  mode.

* Better tracking of when other apps change/add/delete files.

* ARexx interface!

* Built in Trashmaster :) Built in ToolManager-like features.

   OK, I'll step down off my soapbox now! Anyone want to help me write a
Workbench replacement? No matter, I'll just do it myself.. :) I think I
will code name it - "Encino"! In the tradition of Microsoft ("Chicago",
"Cairo", etc). Maybe I'll put in a Windows NT emulation mode (a simple
AllocMem(15000000,MEMF_ANY) and a busy loop to slow things down... :).

1.5 Trashmaster.guide/Installation

Installing Trashmaster
****

   Installing Trashmaster is simple. Just copy Trashmaster into your
"WBStartup" drawer on your boot disk.

   If you want to run Trashmaster in a language other than english, check in
the "catalogs/" directory for your language (you want the directory, not
the *.ct file). If it's there, then copy it to "LOCALE:catalogs/", or to
whatever directory you put Trashmaster into. If your language isn't there,
then perhaps you could send me a translation and I'll put it in the next
release.

1.6 Trashmaster.guide/Usage

                 Using Trashmaster
****

Usage
=====

   Trashmaster can be run from Workbench or the command line. All options
can be specified either from Tooltypes or from the command line (see

                 Options
).

   Obviously, Workbench must be running (and will be, unless you were
naughty and modified your startup-sequence). Trashmaster will open an icon
on the Workbench screen; this icon will look the same as the icon you ran
Trashmaster from. BTW, the icon is supposed to look like a black hole...

If you want to be able to format disks with Trashmaster, you must have
WBStart-Handler installed in your L: directory. The WBStart-Handler is
included with ToolManager, a must-have utility, or in it's own separate
distribution, both of which should be available on a Fish disk or any good
FTP site, or failing that, from me. ToolManager and WBStart are Copyright
(C) 1991-94 Stefan Becker. The default formating program is the same as
Workbench's formatter.

To use Trashmaster, simply drag files and drop them on the icon. A
requester will come up, similar to Workbench's Delete confirmation
requester. If you really want to delete the file(s) or dir(s), click OK;
if not, hit CANCEL. Remember, once you hit OK, the files are deleted..
they're gone! For reals. Not like the Trashcan. If you have the
VERIFYOFF tooltype set, you won't get this requester!

One difference from the Workbench Delete, is the extra option for
Interactive deletion. With interactive delete, you will get a confirmation
requester for each file you dropped into Trashmaster. For each file, you
can choose to either delete it, delete all the rest of the files
non-interactively, skip this file, or abort completely.

If Trashmaster comes across a file that is protected from deletion, it
will bring up a requester with two options, FORCE and CANCEL. If you
select FORCE, Trashmaster will un-protect the file and delete it!
Selecting CANCEL will of course cancel the operation. If you have the
FORCE option set, you won't get this requester, the file will simply be
deleted without warning.

To quit Trashmaster (which you dont really want to do, do you? :),
double click the AppIcon and select the remove option.

If you want Trashmaster started automaticly (you DO, dont you?), place it
into the "WBStartup" drawer on your boot disk.

1.7 Trashmaster.guide/Options

                 Options
****

   For a description of the Format and Template lines used below, see

                 Formats and Templates
                 .

Format: ICON <name>
Template: ICON/K
     This is the name of a custom icon to be used instead of the default
     icon (the default being Trashmaster's icon)

Format: NAME <name>
Template: NAME/K
     Set this to change the name under Trashmaster's AppIcon.

```
Format: FORMATTER <name>
Template: FORMATTER/K
     Set to the name of the disk-formatting program of your choice.  The
     program will be started as a Workbench application and passed the disk
     icon to format.

Format: VERIFYOFF
Template: VERIFYOFF/S
     If set, the initial verification requester will not appear.  The
     action performed will be the one defined by the TYPE tooltype.  Use at
     your own risk.  Note: This tooltype has been changed from previous
     versions of Trashmaster.

Format: TYPE = <INTERACTIVE>
Template: TYPE/K
     Applicable only when VERIFYOFF is set.  If set to INTERACTIVE, deletes
     will default to interactive deletes (requireing confirmation for every
     file and directory).

Format: FORCE
Template: FORCE/S
     If set, files that have their protection flags set to delete-protected
     will be deleted without warning.  Otherwise, you'll get a requester
     whenever you try to delete a protected file.  Note: This tooltype has
     been changed from previous versions of Trashmaster.

Format: X <n>
Template: X/N
     This is the X coordinate of the AppIcon.  A -1 will tell workbench to
     find a suitable place for the icon.

Format: Y <n>
Template: Y/N
     This is the Y coordinate of the AppIcon.  A -1 will tell workbench to
     find a suitable place for the icon.

   VERIFYOFF, TYPE and FORCE can work in conjuction.  If you don't want any
anoying verification requesters, you can set VERIFYOFF, TYPE to nothing
(leave it out), and FORCE.   If you always want interactive deletes, set
VERIFYOFF and TYPE to INTERACTIVE.

   Be carefull with these!  If you turn off verification and use the FORCE
option, anything that gets dropped on TrashMaster will be simply deleted
without warning, even if it's protected from deletion.
```

1.8 Trashmaster.guide/Formats and Templates

```
Formats and Templates
*********************

Format
======
```

   Commands are described with a Format and a Template. In a Format
specification, the arguments are surounded by brackets to indicate the type
of argument. The brackets are not typed as part of the command.

< >

     Angle brackets are used to indicate that this argument is required; it
     must be provided or the command will fail.

[ ]

     Square brackets indicate that the argument is optional. The command
     will run with or without these arguments.

{ }

     Braces indicate that this argument may be given more than once.

|

     The vertical bar separates multiple options, only one of which may be
     specified (mutualy exclusive).

Template
========

   Templates are a more compact, concise version of Formats (both have their
uses). Templates are directly supported by the Amiga OS. Their main
advantage is that they specify the type of data each argument will (should)
be. Each argument is separated with a comma, and has a specifier code at
the end (always a '/' with a letter). These are the codes currently
supported:

/A

     Always required. This argument MUST be given or the command will fail.
     This is equivalent to the Format specifier "< >".

/F

     Final argument. The entire rest of the line, regardless of any
     keywords or spaces that may appear in it, is taken as the argument
     string.

/K

     Keyword. This option will only be filled if this keyword appears in
     the command line.

/M

     Multiple arguments. This argument will accept any number of strings.
     Anything not matching another option will be added to this option.
     Only one /M will be specified.

/N

     Number. The argument is a decimal number.

/S

     Switch. The argument is a boolean switch. If it is specified, the
     option is true, if it is missing, the option is false (default).

=

     This is used to provide an abbreviation. OPT=OPTION means that this

option can be specified with either OPT or OPTION.

   If for some reason you need to specify an argument string (for instance,
a file name) that is the same as one of the options, enclose it in quotes.
For example:

   Hypothetical command Template: SHOW NAME/A

   You type at the shell:

     >show name "name"

   If you had a file called "name" (I won't ask why...), that is how you'd
do it.