icontools

	COLLABORATORS		
	TITLE : icontools		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		June 15, 2022	

	REVISION HISTORY		
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

icontools

1.1 icontools.guide

This document describes several icon tools for the Amiga which are Copyright (C) 1994 by Tobias Ferber.

Disclaimer
Warranty? No Warranty!

Introduction
What (and why) are IconTools ?

The tools

OptIcon
Optimizing icons for speed and size

Icon2c
Creating C code from icons

IconMaker
Creating Icons from IFF/ILBM brushes

Index

Master Index
Where can I find information about ... ?

1.2 icontools.guide/Disclaimer

```
GNU GENERAL PUBLIC LICENSE
**************************
```

Version 2, June 1991

Preamble
========

The licenses for most software are designed to take away your freedom
to share and change it. By contrast, the GNU General Public License is
intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it. (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.) You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not price.
Our General Public Licenses are designed to make sure that you have
the freedom to distribute copies of free software (and charge for this
service if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone
to deny you these rights or to ask you to surrender the rights. These
restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis
or for a fee, you must give the recipients all the rights that you
have. You must make sure that they, too, receive or can get the source
code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software. If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

Finally, any free program is threatened constantly by software patents.
We wish to avoid the danger that redistributors of a free program will
individually obtain patent licenses, in effect making the program
proprietary. To prevent this, we have made it clear that any patent
must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and

modification follow.

    TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a
   notice placed by the copyright holder saying it may be distributed
   under the terms of this General Public License. The "Program",
   below, refers to any such program or work, and a "work based on
   the Program" means either the Program or any derivative work under
   copyright law: that is to say, a work containing the Program or a
   portion of it, either verbatim or with modifications and/or
   translated into another language. (Hereinafter, translation is
   included without limitation in the term "modification".) Each
   licensee is addressed as "you".

   Activities other than copying, distribution and modification are
   not covered by this License; they are outside its scope. The act
   of running the Program is not restricted, and the output from the
   Program is covered only if its contents constitute a work based on
   the Program (independent of having been made by running the
   Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's
   source code as you receive it, in any medium, provided that you
   conspicuously and appropriately publish on each copy an appropriate
   copyright notice and disclaimer of warranty; keep intact all the
   notices that refer to this License and to the absence of any
   warranty; and give any other recipients of the Program a copy of
   this License along with the Program.

   You may charge a fee for the physical act of transferring a copy,
   and you may at your option offer warranty protection in exchange
   for a fee.

3. You may modify your copy or copies of the Program or any portion
   of it, thus forming a work based on the Program, and copy and
   distribute such modifications or work under the terms of Section 1
   above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices
      stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that
      in whole or in part contains or is derived from the Program
      or any part thereof, to be licensed as a whole at no charge
      to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively
      when run, you must cause it, when started running for such
      interactive use in the most ordinary way, to print or display
      an announcement including an appropriate copyright notice and
      a notice that there is no warranty (or else, saying that you
      provide a warranty) and that users may redistribute the
      program under these conditions, and telling the user how to
      view a copy of this License. (Exception: if the Program
      itself is interactive but does not normally print such an
      announcement, your work based on the Program is not required

```
      to print an announcement.)
```

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the
Program, and can be reasonably considered independent and separate
works in themselves, then this License, and its terms, do not
apply to those sections when you distribute them as separate
works. But when you distribute the same sections as part of a
whole which is a work based on the Program, the distribution of
the whole must be on the terms of this License, whose permissions
for other licensees extend to the entire whole, and thus to each
and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or
contest your rights to work written entirely by you; rather, the
intent is to exercise the right to control the distribution of
derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the
Program with the Program (or with a work based on the Program) on
a volume of a storage or distribution medium does not bring the
other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it,
   under Section 2) in object code or executable form under the terms
   of Sections 1 and 2 above provided that you also do one of the
   following:

     a. Accompany it with the complete corresponding machine-readable
        source code, which must be distributed under the terms of
        Sections 1 and 2 above on a medium customarily used for
        software interchange; or,

     b. Accompany it with a written offer, valid for at least three
        years, to give any third party, for a charge no more than your
        cost of physically performing source distribution, a complete
        machine-readable copy of the corresponding source code, to be
        distributed under the terms of Sections 1 and 2 above on a
        medium customarily used for software interchange; or,

     c. Accompany it with the information you received as to the offer
        to distribute corresponding source code. (This alternative is
        allowed only for noncommercial distribution and only if you
        received the program in object code or executable form with
        such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it. For an executable work, complete
source code means all the source code for all modules it contains,
plus any associated interface definition files, plus the scripts
used to control compilation and installation of the executable.
However, as a special exception, the source code distributed need
not include anything that is normally distributed (in either
source or binary form) with the major components (compiler,
kernel, and so on) of the operating system on which the executable
runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program
   except as expressly provided under this License. Any attempt
   otherwise to copy, modify, sublicense or distribute the Program is
   void, and will automatically terminate your rights under this
   License. However, parties who have received copies, or rights,
   from you under this License will not have their licenses
   terminated so long as such parties remain in full compliance.

6. You are not required to accept this License, since you have not
   signed it. However, nothing else grants you permission to modify
   or distribute the Program or its derivative works. These actions
   are prohibited by law if you do not accept this License.
   Therefore, by modifying or distributing the Program (or any work
   based on the Program), you indicate your acceptance of this
   License to do so, and all its terms and conditions for copying,
   distributing or modifying the Program or works based on it.

7. Each time you redistribute the Program (or any work based on the
   Program), the recipient automatically receives a license from the
   original licensor to copy, distribute or modify the Program
   subject to these terms and conditions. You may not impose any
   further restrictions on the recipients' exercise of the rights
   granted herein. You are not responsible for enforcing compliance
   by third parties to this License.

8. If, as a consequence of a court judgment or allegation of patent
   infringement or for any other reason (not limited to patent
   issues), conditions are imposed on you (whether by court order,
   agreement or otherwise) that contradict the conditions of this
   License, they do not excuse you from the conditions of this
   License. If you cannot distribute so as to satisfy simultaneously
   your obligations under this License and any other pertinent
   obligations, then as a consequence you may not distribute the
   Program at all. For example, if a patent license would not permit
   royalty-free redistribution of the Program by all those who
   receive copies directly or indirectly through you, then the only
   way you could satisfy both it and this License would be to refrain
   entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable
   under any particular circumstance, the balance of the section is
   intended to apply and the section as a whole is intended to apply
   in other circumstances.

   It is not the purpose of this section to induce you to infringe any
   patents or other property right claims or to contest validity of
   any such claims; this section has the sole purpose of protecting
   the integrity of the free software distribution system, which is
   implemented by public license practices. Many people have made
   generous contributions to the wide range of software distributed
   through that system in reliance on consistent application of that

system; it is up to the author/donor to decide if he or she is
willing to distribute software through any other system and a
licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed
to be a consequence of the rest of this License.

9. If the distribution and/or use of the Program is restricted in
   certain countries either by patents or by copyrighted interfaces,
   the original copyright holder who places the Program under this
   License may add an explicit geographical distribution limitation
   excluding those countries, so that distribution is permitted only
   in or among countries not thus excluded. In such case, this
   License incorporates the limitation as if written in the body of
   this License.

10. The Free Software Foundation may publish revised and/or new
    versions of the General Public License from time to time. Such
    new versions will be similar in spirit to the present version, but
    may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the
    Program specifies a version number of this License which applies
    to it and "any later version", you have the option of following
    the terms and conditions either of that version or of any later
    version published by the Free Software Foundation. If the Program
    does not specify a version number of this License, you may choose
    any version ever published by the Free Software Foundation.

11. If you wish to incorporate parts of the Program into other free
    programs whose distribution conditions are different, write to the
    author to ask for permission. For software which is copyrighted
    by the Free Software Foundation, write to the Free Software
    Foundation; we sometimes make exceptions for this. Our decision
    will be guided by the two goals of preserving the free status of
    all derivatives of our free software and of promoting the sharing
    and reuse of software generally.

                              NO WARRANTY

12. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO
    WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE
    LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
    HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT
    WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT
    NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND
    FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE
    QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE
    PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY
    SERVICING, REPAIR OR CORRECTION.

13. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
    WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY
    MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE
    LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL,
    INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR
    INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF

DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU
OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY
OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN
ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs
===

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these
terms.

To do so, attach the following notices to the program. It is safest to
attach them to the start of each source file to most effectively convey
the exclusion of warranty; and each file should have at least the
"copyright" line and a pointer to where the full notice is found.

    one line to give the program's name and an idea of what it does.
    Copyright (C) 19yy name of author

    This program is free software; you can redistribute it and/or
    modify it under the terms of the GNU General Public License
    as published by the Free Software Foundation; either version 2
    of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) 19yy name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
    type 'show w'. This is free software, and you are welcome
    to redistribute it under certain conditions; type 'show c'
    for details.

The hypothetical commands show w and show c should show the appropriate
parts of the General Public License. Of course, the commands you use
may be called something other than show w and show c; they could even
be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

1.3 icontools.guide/Introduction

```
Introduction
************
```

The IconTools are a collection of commands for creating and
manipulating Workbench icon images. Before I started calling this
project IconTools there had been several OptIcon archives with other
tools like Icon2c or IconMaker included into it. The reason for the
the tools included here being so widely spread is last but not least
due to the success of Martin Huttenloher's great MagicWB icon
collection.

1.4 icontools.guide/OptIcon

```
OptIcon
*******
```

```
Abstract
========
```

Even if some users claim OptIcon being the tool of their choice for
changing the number of bitplanes in their MagicWB icons, the main idea
behind OptIcon is and was to optimize icons for size and speed.
OptIcon reads .info files and scans the icon image in order to
optimize the PlanePick and PlaneOnOff fields in the icon's Image
structure. This is a space-saving mechanism for image data.

```
        struct Image {

            /* ... */

            UBYTE PlanePick, PlaneOnOff;

            /* ... */
        };
```

Rather than defining the image data for every plane of the RastPort,
you need define data only for the planes that are not entirely zero or
one. As you define your imagery, you will often find that most of the
planes are just as color selectors. For instance, if you're designing
a two-color icon to use colors one and three, and the icon will reside
in a five-plane display, bit plane zero of your imagery would be all
ones, bit plane one would have data that describes the imagery, and bit
planes two through four would be all zeroes. Using these flags avoids
wasting all that memory in this way: first, you specify which planes
you want your data to appear in using the PlanePick variable. For each
bit set in the variable, the next 'plane' of your image data is blitted
to the display. For each bit clear in this variable, the corresponding
bit in PlaneOnOff is examined. If that bit is clear, a 'plane' of
zeroes will be used. If the bit is set, ones will go out instead.
Note that if you want an Image that is only a filled rectangle, you can
get this by setting PlanePick to zero (pick no planes of data) and set
PlaneOnOff to describe the pen color of the rectangle.

Installing OptIcon
==================

The OptIcon executable comes in two versions: OptIcon.000 for all
Amigas and OptIcon.030 for Amigas with a MC-68030 processor. You
simply have to copy one of those into your path (e.g. to C:) and rename
it to OptIcon:
     Copy CLONE FROM OptIcon.030 TO C:OptIcon

If you want to make use of the recursive-descent ablility of the
OptIcon.rexx script then you should copy this into your rexx: drawer
and make sure the script-flag s ist set:
     Copy CLONE FROM OptIcon.rexx TO rexx:
     Protect FILE rexx:OptIcon.rexx ADD s

Invoking OptIcon
================

OptIcon uses ReadArgs() to parse the command line arguments with the
following template:
     FROM=NAME/A/M,DEPTH=PLANES/N,NOEXPAND/S,CRITICAL/S,REMAPV37/S,VERBOSE/S,SMART ←
       /S,ALL/S

FROM=NAME/A/M (required, multiple)
     The name of the icon image file. A trailing .info is optional but
     not required. Several icon image files can be specified. If the
     ALL switch is given, then OptIcon recursively enters all
     directories passed via FROM, collecting all icons.

DEPTH=PLANES/N (numeric)
     With this option you can specify the number of bitplanes to save.

NOEXPAND/S (switch)
     If the NOEXPAND keyword is not present in the command line, then
     OptIcon will always write as many bitplanes as specified with the
     DEPTH=PLANES/N option, even if bitplanes have to be added. With
     the NOEXPAND switch given, OptIcon will not add any new planes.

CRITICAL/S (switch)
     Commodore's PutDiskObject() currently [icon.library 40.1 (15.2.93)]
     re-expands icon images using the PlanePick/PlaneOnOff mechanism
     and in fact PutDiskObject() has quite a lot of problems doing so!
     For this reason OptIcon will perform the PlanePick/PlaneOnOff
     optimization only if the keyword CRITICAL is given in the command
     line!

REMAPV37/S (switch)
     If the REMAPV37 switch is given in the command line, OptIcon will
     map the colors 4-7 to the last 4 in the palette using the following
     algorithm:

        1. A bitplane mask is generated from all planes > 2 via OR.
            (This mask has ones at those positions where any of the
           bitplanes > 2   has a 1 and has zeros only at those
           positions where all bitplanes > 2   have zeros.)

        2. The result is inverted and

        3. stamped with plane 2 via AND.

        4. The resulting mask is set via OR in all planes > 2

     There is obviously no need to expand the image data if the
     following expression is true for the PlaneOnOff value p10:
          p10 &~ %111 != 0

     When expanding an icon without the REMAPV37 keyword given in the
     command line, the last 4 colors of the input image i are mapped to
     the last 4 colors of the output image o as follows:

        1. A bitplane mask is generated by an OR of all bitplanes but
           the last.   (This mask has zeros only at those positions
           where all bitplanes   but the last have zeros and has
           ones otherwise.)

        2. The resulting mask is stamped via AND with the last bitplane

        3. The result is set in all new bitplanes

     If any plane of i but the last is entirely 1 then we can simply
     copy the last plane of i to all new planes in o

VERBOSE/S (switch)
     This switch tells OptIcon to print out some information about each
     icon and what OptIcon is about to do with it.

SMART/S (switch)
     With this switch, OptIcon will examine WBDRAWER and WBGARBAGE
     icons more closely and if there is not really a drawer (or a file)
     behind the icon then the icon type is changed into WBTOOL. This
     is a great help if you want to use some drag'n drop application to
     update icon images which would have problems otherwise (bug?
     feature? hmmm...).

     Caution: It is dangerous to call opticon ALL SMART on ENV: or

```
     ENVARC: because this would change the type of the default icons
     sys/def_drawer.info and sys/def_trashcan.info (and perhaps some
     more) making them unusable for their initial purpose!

ALL/S (switch)
     If this switch is given, OptIcon recursively enters all
     subdirectories given via FROM, collecting icons.
```

Example: In order to remove all but the first 3 planes of the icon
image for the disk in drive DF0: without adding any bitplanes you can
invoke OptIcon as follows:

```
     OptIcon DF0:Disk PLANES=3 NOEXPAND
```

Notes
=====

Since the IconEdit from Commodore will always save 8 bitplane icons the
above example might be of great use to you. (Note that 3 plane images
are not only smaller but also faster!) Coming with OptIcon is the
script PatchIcons which will recursively descend all subdirectories of
a given path deleting all but the first 3 planes of all icon images in
that path.

OptIcon now also allows you to expand your 8 or more color icons for
the use on a 16 or more color Workbench. This is important due to the
new color system under OS3.x which always shifts the second four colors
to the end of the system palette. Therefore you might want to adapt an
icon's color depth to the actual screenmode it is used on.

Note also: OptIcon will always overwrite your icon and does not support
a recursive descent. This is why we wrote the ARexx script
OptIcon.rexx which offers all this to you. Some people might never
even want to use OptIcon directly but will always use OptIcon.rexx.

Example: I only have 8 colors on my Workbench and I often find icon
collections which come up with full 8 bitplane icons and image drawers
with no real drawer behind them. Now on the one hand I don't want to
waste space and time for 5 bitplanes which I don't really need and on
the other hand I want to be able to use MH's drag'n drop tool
IconUpdate to change my drawer images without having to care about
whether these are really drawers or not.

Okay, I've downloaded pix/mwb/TobiIcons-2.0.lha from the Aminet and
extracted it into my ram: disk. A drawer ram:Tobi-Icons/ had been
created there. What I do now is I invoke my OptIcon.rexx script and
change all the icons there into 3 plane icons and I change all the
faked DRAWER icons into TOOL icons:

```
     rx rexx:OptIcon.rexx FROM ram:Tobi-Icons ALL SMART PLANES 3
```

The magic happening there can also be achived without the .rexx script
like that:

```
     List >ram:doit ALL FILES DIR ram:Tobi-Icons PAT #?.info +
       LFORMAT "OptIcon *"%p%n*" PLANES=3 NOEXPAND SMART"
     Execute ram:doit
```

OptIcon and MagicWB
===================

In these days, OptIcon has become more and more used by people who use
Martin Huttenloher's MagicWB icon collection i.e. on Workbench screens
with more than only two bitplanes. Due to the problems resulting from
the new coloring scheme (first-4/last-4 colors) as introduced with
OS3.x these icons look wrong if the number of bitplanes in the icon is
less than the number of bitplanes on the Workbench screen. Of course
one could avoid this problem by giving each icon all eight bitplanes
(this is what Commodore's IconEdit does) but this is a waste of space
and time since larger icons do not only eat up more disk space but are
also much slower!

An easy way for solving this problem is OptIcon on a ToolManager Dock or
AppIcon. (Thanks to Mark Rose, who contributed his nice 'Plane' icon
for the OptIcon distribution.)

Example: Suppose you have a 16 color Workbench and you find an 8 color
icon which looks wrong then you can simply call OptIcon PLANES=4 from
within ToolManager by simply dropping the icon on the dock.

1.5   icontools.guide/Icon2c

```
Icon2c
******
```

```
Abstract
========
```

Tools which manipulate existing icon images are widely spread but
actually when it comes to the point there most often is exactly one
switch missing: the one you need ;-). Icon2c reads a given .info file
and writes out well documented and directly compilable C code. This
code if compiled with a symbol TEST defined (usual compiler option:
-DTEST) will generate an executable which writes back the icon image
file to disk. This allows you to modify any icon to your own needs -
with a text editor and a C compiler of your choice.

```
Installing Icon2c
=================
```

The Icon2c executable comes in two versions: Icon2c.000 for all Amigas
and Icon2c.030 for Amigas with a MC-68030 processor. You simply have
to copy one of those into your path (e.g. to C:) and rename it to
Icon2c:
      Copy CLONE FROM Icon2c.030 TO C:Icon2c

```
Invoking Icon2c
===============
```

Icon2c uses ReadArgs() to parse the command line arguments with the
following template:
      NAME/A,QUIET/S,TO/K

NAME/A (required)

Name of the icon image file. A trailing .info is optional but not
required.

QUIET/S (switch)
    If this switch is given then Icon2c will not print any warnings.
    (More information about Icon2c's warnings can be found in the
    DrawerData discussion further down.) Note: Warnings are allways
    directed to stderr so that they always appear on the console (and
    not in the source code) even if the output is redirected from
    stdout via > or |.

TO/K (keyword required)
    This option allows you to specify the name of the generated C code
    file. If not specified the standard output stream will be used
    which allows piping in a shell. E.g.:
        icon2c ram:disk.info | more

Example: Suppose you want to generate C code from the disk icon of the
disk in DF0::
    Icon2c DF0:Disk QUIET TO RAM:diskicon.c

The Code Generated by Icon2c
============================

The code generated by Icon2c always begins with some #includes, which
define the structures and constants needed.

```
#include <intuition/intuition.h>
#include <workbench/workbench.h>
#include <workbench/icon.h>
```

We will now discuss the generated output using a WBDISK icon with two
images: one for the 'normal' state and one for the 'selected' state of
the icon. The icon type WBDISK is quite suitable for this tutorial
purpose because it needs all the structures an icon can have. I've
created the code in the following examples by invoking Icon2c like that:
    icon2c ram:disk.info >ram:diskinfo.c

The Image Data

For the 'normal' icon image image (the one which we see if the icon is
not selected), an array gr_data is generated. Each line in the code
represents one line of pixels in the image. As usual in the image
data, bitplanes are stored in ascending order:

```
UWORD gr_data[] = {

  /* plane 0 */

  0x0000, 0x0000, 0x0000, ...
  0x0000, 0x0000, 0x0000, ...
  ...

  /* plane 1 */

  ...
```

```
    };
```

This array gr_data is pointed to by the Image structure gr which – as
we will see later – is pointed to by the do_Gadget.GadgetRender field of
the DiskObject structure. (This is why we use gr here.)

```
    struct Image gr = {
    0,0,          /* LeftEdge, TopEdge    */
    47,36,3,      /* Width, Height, Depth */
    gr_data,      /* ImageData            */
    7,0,          /* PlanePick, PlaneOnOff */
    NULL,         /* NextImage            */
    };
```

If the given icon has an alternate image, an array sr_data and it's
Image structure sr are generated analog to gr_data and gr. The sr
structure is pointed to by the do_Gadget.SelectRender field of the
DiskObject structure.

The ToolTypes Array

The icon ToolTypes are stored into tt, a (char *)NULL-terminated array
of strings. Since our WBDISK icon does not really need ToolTypes, we
only have a dummy here:
```
    char *tt[] = {
       "»»»» Icon by Martin Huttenloher ««««",
       NULL
    };
```

If there are no ToolTypes at all in the given icon, both is possible:
The tt array is created and containes only one NULL entry or the
do_ToolTypes field in the DiskObject structure contains a NULL pointer.

The DrawerData structure

Icons of type WBDISK, WBDRAWER and WBGARBAGE have a DrawerData
structure which holds the information about the window which opens when
double-clicking the icon. Icons of a different type do not have such a
structure but a NULL pointer in the DiskObject's do_DrawerData field.

The dd_NewWindow.FirstGadget field holds a non-NULL pointer every now
and then. Icon2c will print a warning message in these cases and
initialize the dd.dd_NewWindow.FirstGadget field to NULL in the
generated code. The Gadget structure will be included into the
generated code like this:
```
    #ifdef UNDEFINED
      struct Gadget <unknown> = {

        /* ... some strange stuff in here ... */

      };
    #endif /* UNDEFINED */
```

There also happens to be a non-NULL pointer in the dd_NewWindow.Title
field every now and then. In this case as well a warning message will

be printed and the field in the generated code will be initialized to
NULL. However, the original string value will be available in a
comment if it is printable.

Here is the DrawerData structure of our WBDISK icon:

```
    struct DrawerData dd = {
        151,                    /* dd_NewWindow.LeftEdge    */
        54,                     /* dd_NewWindow.TopEdge     */
        347,                    /* dd_NewWindow.Width       */
        150,                    /* dd_NewWindow.Height      */
        255,                    /* dd_NewWindow.DetailPen   */
        255,                    /* dd_NewWindow.BlockPen    */
        NULL,                   /* dd_NewWindow.IDCMPFlags  */
        WFLG_SIZEGADGET
        | WFLG_DRAGBAR
        | WFLG_DEPTHGADGET
        | WFLG_CLOSEGADGET
        | WFLG_SIZEBRIGHT
        | WFLG_SIZEBBOTTOM
        | WFLG_SIMPLE_REFRESH
        | WFLG_REPORTMOUSE
        | WFLG_ACTIVATE
        | WFLG_WBENCHWINDOW,    /* dd_NewWindow.Flags       */
        NULL,                   /* dd_NewWindow.FirstGadget */
        NULL,                   /* dd_NewWindow.CheckMark   */
        NULL,                   /* dd_NewWindow.Title       */
        NULL,                   /* dd_NewWindow.Screen      */
        NULL,                   /* dd_NewWindow.BitMap      */
        92,                     /* dd_NewWindow.MinWidth    */
        68,                     /* dd_NewWindow.MinHeight   */
        65535,                  /* dd_NewWindow.MaxWidth    */
        92,                     /* dd_NewWindow.MaxHeight   */
        WBENCHSCREEN,           /* dd_NewWindow.Type        */
        0,                      /* dd_CurrentX              */
        0,                      /* dd_CurrentY              */
        3,                      /* dd_Flags                 */
        0,                      /* dd_ViewModes             */
    };
```

The values of the dd_Flags and dd_ViewModes fields are not documented
in the includes. Playing around with these values however has revealed
some information. The dd_Flags field is usually set to one of the
following:

1
    If only files with an icon should be visible in this window.

2 or 3
    If all files should be visible, using the def_#?.info default
    icons from env:sys/.

The dd_ViewModes field represents the sorting criteria of the files and
drawers listed in the window:

1
    Graphical, view by icon

2

     Textual, lexicographically sorted by name

3

     Textual, sorted by date

4

     Textual, sorted by size

The DiskObject structure

The DiskObject structure icon is the holder of all the other data.

```
    struct DiskObject icon = {
      WB_DISKMAGIC,                        /* do_Magic             */
      WB_DISKVERSION,                      /* do_Version           */
      NULL,                                /* do_Gadget.NextGadget  */
      5,                                   /* do_Gadget.LeftEdge   */
      7,                                   /* do_Gadget.TopEdge    */
      47,                                  /* do_Gadget.Width      */
      37,                                  /* do_Gadget.Height     */
      GFLG_GADGHIMAGE|GFLG_GADGIMAGE,      /* do_Gadget.Flags      */
      GACT_RELVERIFY|GACT_IMMEDIATE,       /* do_Gadget.Activation */
      GTYP_BOOLGADGET,                     /* do_Gadget.GadgetType */
      (APTR)&gr,                           /* do_Gadget.GadgetRender */
      (APTR)&sr,                           /* do_Gadget.SelectRender */
      NULL,                                /* do_Gadget.GadgetText */
      0,                                   /* do_Gadget.MutualExclude */
      NULL,                                /* do_Gadget.SpecialInfo */
      0,                                   /* do_Gadget.GadgetID   */
      (APTR)WB_DISKREVISION,               /* do_Gadget.UserData   */
      WBDISK,                              /* do_Type              */
      "SYS:System/DiskCopy",               /* do_DefaultTool       */
      &tt[0],                              /* do_ToolTypes         */
      NO_ICON_POSITION,                    /* do_CurrentX          */
      NO_ICON_POSITION,                    /* do_CurrentY          */
      &dd,                                 /* do_DrawerData        */
      NULL,                                /* do_ToolWindow        */
      8192,                                /* do_StackSize         */
    };
```

The TEST code

When compiling the generated code with a symbol TEST defined, then an
executable will be generated which writes the icon to disk via
PutDiskObject(). Together with a C compiler and a text editor of your
choice we now have the most powerful tool for manipulating Workbench
icons you can think of. (-:

Example: Let's assume you called Icon2c and saved your Ram Disk icon to
ram:foo.c:
    Icon2c ram:disk.info >ram:foo.c
Now you compile the file foo.c with Dice C:
    dcc -2.0 -DTEST ram:foo.c

and the resulting executable ram:foo can be used to write the icon back
to ram:disk.info
     ram:foo ram:disk

Here is the main() procedure of foo.c:

```
#ifdef TEST
#include <intuition/intuitionbase.h>
#include <stdlib.h>
#include <stdio.h>

extern struct Library *OpenLibrary(STRPTR, ULONG);
extern void CloseLibrary(struct Library *);
extern LONG IoErr(void);
extern BOOL PrintFault(LONG, STRPTR);
extern BOOL PutDiskObject(char *, struct DiskObject *);

struct IconBase *IconBase;

int main(int argc, char **argv)
{
  if(argc == 2)
  {
    if( (IconBase= (struct IconBase *)OpenLibrary(ICONNAME,36)) )
    {
      if( !PutDiskObject(argv[1],&icon) )
        PrintFault(IoErr(),argv[1]);
      CloseLibrary(IconBase);
    }
    else printf("%s: no %s.\n",*argv,ICONNAME);
  }
  else printf("usage: %s <filename>\n",*argv);

  return IoErr();
}
#endif /* TEST */
```

1.6 icontools.guide/IconMaker

IconMaker
****

Abstract
========

Suppose you painted some brushes and now you want to make icons from
them. Or let's assume you've downloaded some brushes and you don't
know how they look like. This is where it comes to IM - the IconMaker.
IM creates icons from brushes and offers you the best possibility for a
preview: the Workbench with all its functions! (-:

Installing IconMaker
====================

Simply copy IM.000 (or IM.030 if you have a MC-68030 Amiga) somewhere
into your path (e.g. to C:) and rename it to IM. For example:
     Copy im.030 TO c:im

Invoking IconMaker
==================

IconMaker uses ReadArgs() to parse the command line arguments with the
following template:
     FROM=NORMAL/K/A,SELECTED/K,
     IW=ICONWIDTH/K/N,IH=ICONHEIGHT/K/N,MINSIZE/S,
     TYPE/K,HIGHLIGHT/K,
     IX=ICONX/K/N,IY=ICONY/K/N,
     TOOLTYPES/K/M,STACKSIZE/K/N,DEFAULTTOOL/K,
     WX=WINDOWX/K/N,WY=WINDOWY/K/N,WW=WINDOWWIDTH/K/N,WH=WINDOWHEIGHT/K/N,
     TO/K/A

FROM=NORMAL/K/A (required, keyword required)
     The specified IFF/ILBM brush will be used for the normal image of
     the icon. This argument must be present in the command line.

SELECTED/K (keyword required)
     The specified IFF/ILBM brush will be used for the selected image
     of the icon. If present in the command line, an implicit
     HIGHLIGHT=IMAGE is used.

TO/K/A (required, keyword required)
     The name of the icon without the trailing .info which is appended
     automatically by PutDiskObject().

IW=ICONWIDTH/K/N (numeric, keyword required)
IH=ICONHEIGHT/K/N (numeric, keyword required)
     The dimensions of the icon image. Smaller values than those of the
     brush(es) will use the top/left corner of the image, larger values
     will fill up the icon image's bottom/right border with 0's

MINSIZE/S (switch)
     If the two IFF/ILBM brushes differ in size then the resulting icon
     image normally has the dimensions of the larger brush. However,
     if the MINSIZE switch is present in the command line, the
     dimensions of the smaller brush are used.

TYPE/K (keyword required)
     By default, IM will use TYPE=PROJECT and create a project icon
     with the default tool MultiView. The TYPE parameter is parsed
     with the following template:
          DISK/S,DRAWER/S,TOOL/S,PROJECT/S,GARBAGE/S

HIGHLIGHT/K (keyword required)
     This argument specifies the highlighting method of the icon, i.e.
     what happens when the icon is selected. If a SELECTED image is
     specified, then IM implicitly assumes HIGHLIGHT=IMAGE. The
     HIGHLIGHT parameter is parsed with the following template:
          COMPLEMENT/S,BACKFILL/S,IMAGE/S

IX=ICONX/K/N (numeric, keyword required)
IY=ICONY/K/N (numeric, keyword required)

     By default the created icon has no fix position. These two
     options allow an exact positioning of the created icon.

TOOLTYPES/K/M (keyword required, multiple)
     Any number of ToolTypes can be specified. By default, IM uses
          TOOLTYPES "FILETYPE=ILBM"

STACKSIZE/K/N (numeric, keyword required)
     The default stack size for TYPE=TOOL icons.

DEFAULTTOOL/K (keyword required)
     The default tool for TYPE=PROJECT and TYPE=DISK icons.

WX=WINDOWX/K/N (numeric, keyword required)
WY=WINDOWY/K/N (numeric, keyword required)
WW=WINDOWWIDTH/K/N (numeric, keyword required)
WH=WINDOWHEIGHT/K/N (numeric, keyword required)
     The window parameters for TYPE=DRAWER, TYPE=DISK or TYPE=GARBAGE
     icons.

Example: Let's assume you have DOpus5 and you want to see all brushes
in the Images/ and Images2/ drawer. You simply have to do the
following:

     List >RAM:doit ALL FILES DIR DOpus5:Images/ DOpus5:Images2/ +
       PAT ~(#?.info) LFORMAT "IM FROM *"%p%n*" TO *"%p%n""

     Execute RAM:doit

By default, IM creates a project icon with the default tool MultiView
and ToolType FILETYPE=ILBM. This is what I need in most cases when
using IM for the above purpose.

Bugs
====

IM does not remap the colors in your brushes to Workbench colors. This
may or may not be implemented in the future. Please mail me if you
really need such a feature.

1.7 icontools.guide/Master Index

                    Master Index
****

                Bugs

                                              IconMaker

                Bugs

                                              OptIcon

                Coloring Scheme

                                    OptIcon

```
sr_data
                                        Icon2c

struct DiskObject
                    Icon2c

struct DiskObject
                    Icon2c

struct Gadget
                        Icon2c

struct Image
                        Icon2c

struct Image
                    Icon2c

ToolManager
                    OptIcon

tt
                        Icon2c
```