FinalWrapper

NDY's

COLLABORATORS			
	TITLE : FinalWrapper		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	NDY's	July 19, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FinalWrapper

1.1 Hitchhikers guide to FinalWrapper

```
*********************************************
*                                           *
*    FinalWrapper 1.3 (27.04.94) by NDY's   *
*                                           *
*********************************************
```


1.2 What the hell is it?

A Short Overview

FinalWrapper is a FinalWriter(TM) macro that wraps some piece of text or a textblock around an oval! It can also create amazing spiral text effects with just a few mouseclicks and keystrokes!

Features:

- Supports all kind of text attributes and fonts
- Supports any type of text flow
- Unlimited text length when using main text
- Supports rotated ovals
- Additional effects with rotation of characters

- Link oval to wrapped text in different ways
- Built in help requesters for all options
- Uses your preferred language (currently english and german available)
- Can be easily translated into other languages (also available in OS2.0!)
- Different types of spirals
- Automatic/custom reduction of character size towards the inside of a spiral

1.3 Installation

How to Install

Simply copy FinalWrapper to your FWMacros drawer. (I suggest you install it to the user menu or the button strip!).

To run it, you need the "rexxmathlib.library" installed in your LIBS: drawer. Furthermore, of course Rexxmast has to be run before (and you REALLY should own FinalWriter B-)!

If you don't know how to do this, or if you're a lazy guy (like myself 8-]), you can use the Installer-script provided with this release, which will do the whole job for you. To use it, you need Commodore's Installer. It can be found on the OS3.0+ Install disk and is also available from PD (Fish Disk 870).^{1}

Click here to start the script

1.4 Note on installation

Note:

The script internally makes use of the Version command which needs the version.library to work. Unless you've deleted them, they're situated on your Workbench.

1.5 How to use it

Usage

(There's no need to read the footnotes to use this program, they just provide you with some extra information.)

* Before calling the macro

To run the macro, you first have to select an oval plus some text. It doesn't matter whether this is a textblock or a passage of normal text. The whole text will be neatly wrapped around the oval.

Normal text has the advantage of enabling the usage of an unlimitted number of characters in different styles, colours, font etc. Due to the system used for this, it's fairly slower than a textblock, which can be only 33 characters long and can only have one style1.

If the oval had been rotated, the wraptext will be rotated as well.

So: select the text, click on the "mouse-pointer gadget" and select the oval, or select the textblock, press Shift and then click on the oval2. Now you may start FinalWrapper.

* Using the requester

You'll be prompted a requester where you can enter all needed options (the Ok and Cancel gadgets work as usual). Note that it's perfectly OK to enter nothing to use the default settings. All options are to be separated by a space from their precessor and successor3. Unknown ones are ignored.

* Sector size: An = n / A-n = -n (default: 360)

The first option determines the size of the sector to be used in degrees. This allows you to wrap the text around only a part of the oval. You have to enter a number4 between -360 and 360. A negative number makes the bottom of the characters point to the outside of the circle, a positive one to the inside (try both to see the difference). Unless you add an "a" in front of the size, this option must be given in the first place. If the option is omitted or misspelt, it's assumed to be the default value of 360. Too high / low numbers are changed to 360 / -360.

E.g: 360 Use the full circle (a360/a+360/A+360/A360 would be the same)
180 Use one half of the circle

```
          -40     Only a tiny arc is to be used


* Deletion: D+ / D- / D= (default: delete oval)

Unless the D option is specified, the oval will be deleted by  this  macro.  ↩
   The
following  switches  can  be  used: "d+" deletes both the textblock and the  ↩
   oval
whereas "d-" prevents the oval from being deleted. "d-" slows down  the  ↩
   drawing
process  quite  a  bit.  Therefore you can use the "d=" option, which copies  ↩
   the
oval before deleting it for you can simply paste it afterwards, as soon  as  ↩
   you
need it.


* Rotation: R+n / R-n / R+ / R- / R= (default: R+0)

"r\ensuremath{\pm}n" rotates all characters  by another n degrees to the right ↩
    (+) or left (-).
"r\ensuremath{\pm}" is a shortcut for "r\ensuremath{\pm}90". n must be between ↩
    -180 and +180.

"rn" sets the rotation of all letters to n (i.e.  they  all  look  in  the  ↩
   same
direction). n is relative to the oval's rotation, i.e. with "r0", the  ↩
   characters
are drawn parallel to the axes of the oval. "r=" works like "rn", exept that  ↩
   all
chars  now  have the same rotation as the textblock (or 0 for normal text). n  ↩
   is
limitted to 0 ... 360.

   E.g: r+90      The letters build a "file"
        r+180     Looks like "a-360", but the order of the chars is reversed
        r180      All characters are drawn upside down


* Grouping: G- / G+ (default: don't group oval)

The "g" option allows you to group the oval to the created  object.  "g+  ↩
   groups
the oval as it is, whereas "g-" makes it invisible before. This is mainly  ↩
   useful
if you use a sector of less than 360 and want the text to flow around  the  ↩
   full
oval.


* Starting point: Sn = S+n / S-n / S+ / S- (default: S0)

With this option you can determine the place where the text  starts.  "s0"  ↩
   lets
the text be centered relative to the top of the oval. Positive values (the +  ↩
   can
```

```
be omitted) shift the starting point  anticlockwise,  negative  ones   ↩
   clockwise.
"s\ensuremath{\pm}"  is equal to "s\ensuremath{\pm}90". The option allows you  ↩
   to specify any position you want
for the wrapped text, e.g. to use a special part of an oval. By the way: n   ↩
   must
be between -180 and 180.

    E.g:  s90       The text starts on the right side of the oval
          a90 s90   90 degree arc, centered to the left side


* Spirals: @n = @+n / @-n (default: @0)

Spirals can be achieved with the "@n" option: n must be between -100 and 100  ↩
   and
indicates  the  radius  of  the  inside  of the spiral in ratio to the oval's  ↩
   in
percent. Negative values make the text  start  at  the  inside  of  the   ↩
   spiral,
positive  at  the  outside. "@n" starts the spiral with the oval's radii  ↩
   getting
successively smaller upto n% of their original size to the end of the text.   ↩
   The
height  is  reduced  in  the  same  way (unless the "h" option is specified,  ↩
   see
below). The sector size is unlimited when using spirals. "@0" is equal to "  ↩
   @100"
and "@-100" (no spiral).

    E.g. @30 a1080    Start at the outside, going three times round until a  ↩
       size
                      of 30% of the original is reached
         @-1 a720     Start in the middle of the oval, 2 full turns


* Height: Hn = H+n / H-n (default: Value of @ if given, otherwise H0)

The "hn" option,  mainly  intended  to  be  used  in  conjunction  with   ↩
   spirals
(although it can be used without, of course), defines the ratio of the height  ↩
   of
the first character to the height of the last one. It works exactly the same  ↩
   way
as  the  "@" option. "h0" is the same as "h100" and "h-100".

It's main purpose is to keep the height unchanged ("h0") when using spirals,  ↩
   but
you can also set another size reduction rate as for the radius.

    E.g: "h-20"    Lets the first character have one fifth of the original  ↩
       size
                   and each following a bit more, until the last, which is  ↩
                      drawn
                   in normal size.
```

```
* Save options: O+ / O- (default: don't save)

If you intend to use the same options more than once (to experiment with them
for instance), you can use the "o" option to store the current contents of the
requester. Any time you call FinalWrapper from now on, the same options will
reappear. The difference between the two forms is that the "o-" will be removed
from the option list before saving it and then will abort. Use this, if you want
to save the options only once. "o+" saves all options and continues.

    E.g.:   First call          On any further call, the following will appear:
            r+ s-90 d= o- a     R+ S-90 D= A (just add the missing number)
            120 d= s- o+ @120   120 D= S- O+ @120


* Help: ? (default: no help)

If you can't remember the exact syntax or don't know all options, just type "?"
anywhere in the options and a help requester will pop up. You can work your way
through the help by using the "Next" and "Previous" buttons. "Back", as you
might have thought, brings you back to the input requester.

    E.g.:   a123 g+?
```

1.6 Note on usage

```
Note:
  Currently case, super-/subscript and underline etc. are not supported by
  FinalWriter for textblocks.
```

1.7 Note on usage

```
Note:
  Don't select more than one oval or textblock. FinalWrapper being unable to
  figure out which object you selected first, it will simply take the oval and the
  textblock with the highest internal number. If you select a textblock,
  highlighted text will be ignored. Due to Final Writer, you can't select only one
  character (exept with textblocks), but this is no real limitation, I think!
```

1.8 Note on usage

```
Note:
  If an option is used twice or more times, only the first occurence ↩
     is
  recognized. If the sector size is given with and without a preceding "a", ↩
     the
  one with the "a" is used.
```

1.9 Note on usage

```
Note:
  All numbers are integers (pointless numbers ;-S), i.e. they may not contain ↩
     any
  decimal point or comma.
```

1.10 Programmer's corner

```
Some information for Gurus and the like

  If you aren't familiar with Arexx, you'd better skip this section.

  * I've split up the whole code in some subroutines to simplify your (and in ↩
     the
  first place my own) life when altering FinalWrapper. The code growing ↩
     larger
  and larger comments have been reduced to a minimum.

  * The "PROC" in front of all subroutines enables folding them with GoldEd ( ↩
     set
  fold start/end to PROC/RETURN in misc config). This is completely ignored ↩
     by
  Arexx, since it's located before the label (how cunning! ß->).

  * All subroutines have now all parameters listed in their first line, so ↩
     you
  even don't need to unfold them for usage. The number in [brackets] indicates ↩
     the
  version of FinalWrapper in which this routine has changed for the last time. ↩
     So
  you know what is new if you receive a new version. I suggest you add an ↩
     asterisk
  ot something like that, if you change a routine yourself.


How to translate the program into another language

  Copy the english strings in the "locale" sub and translate them. Make sure ↩
     you
  don't remove any doublequote ("), since they're essential. If you use ↩
     single
  quotes ('), add another quote (i.e. use '' instead of ').
```

```
You can make use of the following variables inside your strings:
    "errtext": %n  error number (=RC)
               %l  error line (=SIGL)
               %t  errortext (note that Arexx returns it's  errortexts always  ↩
                  in
                   english,  at least upto OS3.0) (=ErrorText(RC))
    "nolib":   %y  library name (=library)
    "input":   %i  program info (=info)
               %d  default values (=default)
    E.g.:
      info="FinalWrapper 1.3 by NDY's"
      help.7='"This is %i" "It has been enhanced by:" "Al «Kojack» Longhair"'
      expanded to: '"This is FinalWrapper 1.3 by NDY's" "It has been...

"helppages" can be increased for any language, if necessary.

Don't remove my name from the infostring and do include it somewhere, please ↩
   ...

To test your new language, set the variable "test" to 1 and  start  macro,   ↩
   then
set it to 2 and so on. The following things are tested:

    test=1             Error requester
    test=2             Library not found requester
    test=3             Wrong things selected
    test=4             Requester and help (just hit "Ok" and try ALL  ↩
       requesters)
    test=10,20,100,200 FinalWriter returncodes

If everything works fine, reset "test" to 0.

FW  reads  it's  language  from  the  variable  ENV:Locale  which  is  used    ↩
   by
locale.library  V40+.  But don't bother if you don't have this one (I don't  ↩
   have
it either!), the ENV_Locale program initializes this variable and also  the   ↩
   one
in ENVARC:. This is automatically called by the installation script and sets  ↩
   the
variable according to locale's preferred  language.  Unless  you  translate   ↩
   the
whole  doc  and the script (:-), you should duplicate the install button in  ↩
   this
doc, adding "LANGUAGE  name"  just  after  the  "NOPRINTER"  (e.g.  ... ↩
   NOPRINTER
LANGUAGE  français").  Like  that,  OS2.0 users can also use your translation  ↩
   as
well.


If you send me your translations or improved versions, you'll receive my   ↩
   newest
version  including  your additional features, if possible. By the way: Who  ↩
   wants
```

```
to do the translation work of  the english doc to german for me??? It takes me ↩
    a
lot of time which could be used for coding instead...
```

1.11 Known bugs and limitations

```
Bugs and other insects

  * For flat ovals, the results aren't very amazing if the text  flows  round  ↩
    the
tight  arcs,  but  I  didn't  want to spend much time on finding an algorithm  ↩
    to
improve this since I don't consider it useful. Use a sector of 120 degrees or  ↩
    so
and all will be well. The same effect occurs when using spirals.

  * Spiral effects don't look that good  yet.

  * You'd better not change the text while  FinalWrapper  is  working  since  ↩
    this
could lead to malfunction (especially when using normal text!).

  * For normal text, the width of a space is calculated from the  last  used  ↩
    font
only  (for  reasons  of speed). This could slightly change the space between  ↩
    the
words. Just add some more spaces where required. Tabs and ends of paragraphs  ↩
    are
replaced by spaces which can change spacing as well.

  * If the oval is rotated and the sector size is not about \ensuremath{\pm}360  ↩
    then the  created
object  may be placed not exactly over the oval (how terrible! ;). Use g- and  ↩
    it
should be OK.

FinalWrapper was created and tested using  Final  Writer  Release  1  ↩
    (21.10.93,
german version) on an A1200 (OS3.0, 2MB Chip, 4MB Fast, HD). It should run  ↩
    under
OS2.0/2.1 and other hardware configurations as well.
```

1.12 About the history

```
Program history

  (The alpha, beta etc. versions were not released)

  1.0  (24.02.94)  Initial Release
  1.1a (06.03.94)  Changed name from Wrap to FinalWrapper
                   Corrected version string
                   Added usage of normal text for long texts and different  ↩
                       styles
```

```
                  Textflow is now taken from the oval not the standard  ↩
                     textblock
                  Added D option
                  Seperate doc file for saving memory
1.1b (10.03.94)   Added runtime/syntax error checking
                  Thus corrected syntax of RemLib
                  And added test whether no object is selected
                  Added replacing of bad characters (TABs, CRs etc.) by spaces
                  Fixed "sectorsize 6 impossible" bug
                  Sectorsize limited to -360...360, default is now 360
1.1c (11.03.94)   Added RemLib in case of runtime error
                  Corrected SELECTED to "SELECTED" etc.
                  Added support for rotated ovals
                  Changed indention which reduced source size by ~2.5k!
                  Removed requester for wrong input
1.1  (17.04.94)   Corrected bug if using normal text
                  Better german docfile (translation of this one)
1.2a (18.03.94)   Added R option
                  Added L option
                  Added help requesters
1.2b (26.03.94)   Now supports locale's preferred language$^1$
                  Linked strings ("a b c") replaced by dotted variables (x.y)
                  Multiple textblocks abandoned (the selection order is unknown ↩
                     )
                  Now uses Datatype() instead of complicated Verify()s
                  Added S option
1.2  (27.03.94)   Documentation now in the AmigaGuide format
                  Character objects are deleted if an error occurs
                  Added Installer-script
1.3a (02.04.94)   Added @/h options (inspired by CBM's Bullet-lib spiral demo)
                  Corrected silly bug that caused an error when "Abort" was  ↩
                     used
                  Changed procedure headers to enable folding them with GoldEd
                  Installer-script now can be started from within the doc
1.3b (03.04.94)   Completely restructured the whole code (once again saved ~2k ↩
   !)
                  Decimal delimiter now is correctly reset after an error
                  Now all printable characters are handled (;=" didn't work)
                  Helppages are shown in correct order
                  The Installer-script now asks before overwriting the macro
1.3c  (13.04.94)  Added d= option
                  Removed bug in help: Back wouldn't always work
1.3d  (14.04.94)  Added O option
                  L option now is called G (more clear)
                  Somewhat improved parts of the manual
                  Help doesn't trash the option line anymore
1.3   (27.04.94)  Now uses ENV:Locale to determine the language
                  Added ENV_Locale
                  Installer-script now checks whether newer version is  ↩
                     installed

                  Has been tested for quite some time, should be stable now
                  (I can't test all possible option combinations, however!)
```

1.13 Note

```
Note:
  Therefore, there won't be a special german version anymore from now on i.e.  ↩
     all
  comments  in  the  source are english, but I think programmers should  ↩
     understand
  this...
```

1.14 About the future

```
The future of FinalWrapper

  The next release will probably include:
  - A requester with gadgets for all options
  - Faster processing of normal text
  - Better results with spirals and flat ovals (I've got an idea of how to do  ↩
     it,
    let's see how it works out!)
  - Some other little goodies

  Do you have some cool ideas? Don't hesitate to write me!

  If you tell me about bugs in FinalWrapper  or  write  me  a  good  idea  for  ↩
     an
  improvement, you'll receive a new version with the bugs fixed or the new  ↩
     feature
  included (if possible).

  Anyway, do send me a disk (maybe with some cool stuff on  it?)  and  some  ↩
     money
  (sfr., DM, US$) to cover postage (or money for both), if you want me to send  ↩
     you
  a new version.


Other macros

  I've already got some other Final Writer macros in mind:
  - Function plotter with lots of options
  - Sinetext (yeah!)
  - Macro  to make a single textobject out of selected text (with different  ↩
     fonts
    and styles, useful for image descriptions etc.)
  - Effects with oblique (/text\  \text/ etc.)
  - Effects with text size
  - Everything you suggest
```

1.15 Legal stuff

```
Law and order
```

```
* Disclaimer:

You use the program at your own risk. Neither I nor any other author of any
included file can be made responsable for any damage caused by any part of this
distribution.

Especially don't blame me if...
- your machine has crashed because of FinalWrapper and you've lost 10 hours of
work on your new amazing shot'em up.
- your cat has eaten the disk containing FinalWrapper and got ill.
- your local power plant broke down while using FinalWrapper (a known bug, which
only occurs very rarely)!
- you cannot close your mouth anymore because of sheer amazement about the
effects achieved with FinalWrapper (look at a PC, this should cure you!).
                                                                        3-)


* Distribution:

This program is Public Domain, i.e. you may use it or change it as you please.
The whole distribution should be kept together, but at least you should include
the rexxmathlib.library and one of the ShortInfo docs.

The following files are included:

FinalWrapper.rexx     + .info       The macro
FinalWrapper.Guide    + .info       English AmigaGuide doc
FinalWrapperD.Guide   + .info       German AmigaGuide doc
ShortInfo.doc                       A very short description
ShortInfo.dok                       The same in german
FinalWrapper.install                Installer script
DemoDoc               + .info       An example Final Writer document
ENV_Locale                          Used by the install-script
E/ENV_Locale.e                      E source of the program \ Only useful for
E/locale.m                          Needed to compile it    / E programmers!
libs/rexxmathlib.library            For trigonometry


* Copyrights:

Final Writer is (c) by Soft Wood Inc.

The Installer and AmigaGuide are (c) by Commodore, but they both can be found on
Fish Disk 870.

Workbench and Amiga are trademarks of Commodore Amiga Inc.

Arexx is (c) by William S. Hawes and Commodore
```

GoldEd is (c) by Dietmar Eilert

The Guide icons are (c) by Martin Huttenloher

If I should have unintentionally violated a Copyright, please let me know!

1.16 How to contact the author

For suggestions, bug reports, gifts :-} etc. write to:

Andreas Weiss
Dorfstrasse 24
CH-8212 Nohl
(Switzerland)

P.S: This is my very first prog in Arexx! D'ya like it?

Have fun!
ND

1.17 Special thanks

Thanx...

...must go to Dietmar Eilert for his great GoldEd, to SoftWood for their
FinalWriter (I'm waiting for Release 2!) and to the programmer of the
rexxmathlib.library (whoose name I don't know since I haven't got the docs)!

Also thanks to William S. Hawes for his ARexx portation!!!

And not to forget: Matrin Huttenloher for his wonderful MagicWB-icons and Wouter
van Oortmerssen for his E-Compiler!