FinalWrapper

NDY's

COLLABORATORS

	TITLE : FinalWrapper	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	NDY's	December 1, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

FinalWrapper

1.1 Hitchhikers guide to FinalWrapper

```
*********************************************
*                                           *
*    FinalWrapper 2.1 (11.06.94) 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
- 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
- Easy to use due to custom requester (2.x) / online help (1.4)

Click here to look at some demo pics

1.3 Installation

How to Install

(See also the section about FinalWrapper 1.4.^{2})

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", the "rexxsupport.library" and the "apig.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, on your Final Writer Disk 1 and is also available from PD (Fish Disk 870).^{1}

Install FinalWrapper 2.x

Install FinalWrapper 1.4

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.

The rexxsupport.library normally can be found on the Workbench disk as well.

1.5 Note on installation

```
Note:

  The actual version is always being referred to  as  "2.x"  to  prevent  me  ↩
     from
  changing it in the whole doc with every new release :-}
```

1.6 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.)


  * Using the requester

  Having started the macro, you'll be prompted a requester where  you  can  ↩
     select
  all needed options:

  Each option is represented by three gadgets:

   _1   ____________2   ____3
  | |  |@|          |  |    |
   -    ------------    ----

  The checkbox gadget (1) determines, whether the option is to be used  ( ↩
     otherwise
  the  default value is used). The string gadget (3) allows you to enter a  ↩
     number.
  Finally, the cycle gadget (2) specifies the way the  option  works.  Behind  ↩
     the
  name,  you  can see the unit in which (3) is measured (% percent/\textdegree{} ↩
      degrees). The
  entered numbers are internally limited to a certain range (see below).

  The following options can be found:


  * Arc: (0 to 360, default: 360)

  The first option determines the size of the sector to be used. This  allows  ↩
     you
  to wrap the text around only a part of the oval. When writing anticlockwise,  ↩
     the
  bottom of the characters point to the outside of the circle,  otherwise  to  ↩
     the
  inside.


  * Spiral: (0 to 100, default 100)
```

The number indicates the radius of the inside of the spiral in ratio to the oval's in percent. "Outside to inside" 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 "Size" option is specified, see below). "Inside to outside" works the other way round. The arc is unlimited when using spirals.

* Size: (0 to 100, default: Value of "Spiral" if given, otherwise 100)

The "Size" 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 "Spiral" option.

It's main purpose is to keep the height unchanged ("100") when using spirals, but you can also set another size reduction rate as for the radius.

* Start: (0 to 180, 0 to 360 for absolute, default: shift 0)

With this option you can determine the place where the text starts. "Shift" 0 lets the text be centered relative to the top of the oval. Other values shift the start in one direction. "Absolute" sets the absolute starting point (0 is at the bottom of the oval, counting anticlockwise). The option allows you to specify any position you want for the wrapped text, e.g. to use a special part of an oval.

* Rotate: (0 to 180, 0 to 360 for absolute, default: 0 clockwise)

"Absolute" sets the rotation of all letters to n (i.e. they all look in the same direction). The number is relative to the oval's rotation, i.e. with 0, the characters are drawn parallel to the axes of the oval.

"Like textblock" works equal, exept that all chars now have the same rotation as the textblock. The number has no effect. With normal text, it works like "Absolute".

The other two rotate all characters by another n degrees in either direction.

* Delete:

Specifies, which objects are to be deleted. Not deleting the oval slows down the drawing process quite a bit. Therefore, you can use the "Copy oval" option which copies the oval before deleting it for you can simply paste it afterwards, as soon as you need it.

* Grouping:

The option allows you to group the oval to the created object. "Invisible oval" makes the oval invisible before grouping it. This is mainly useful if you use a sector of less than 360 and want the text to flow around the full oval.

* Adjust:

The option provides you with two different ways to improve the generated effects:

"Character width" stretches/squashes each character, so that there are no gaps between them. In addition "Character size" changes the height in the same ratio, so the characters don’t get distorted (gives a fancy effect with ellipses!),

Contrary to the above, "Arc" changes the sector size to achieve the same goal. Therefore it’s only useful, if you don’t need a special sector size (great in conjunction with the "Spiral" option!). If this option is used, the "Arc" option is overridden, but it still determines the direction, the starting point (if "Start: Absolute" isn’t used) and it defines the shape of the spirals.

* Autoshow screen:

FinalWrapper currently can’t open it’s requester on Final Writer’s screen, since it isn’t a public screen. Therefore, it uses the frontmost public screen and brings this one to front. When closing the requester, the screen is put to back again. In case a future version of Final Writer should have a public screen, you

can deselect this gadget so the screen won't be brought to back afterwards.

* Autoactivate gadgets:

Enables/disables the automatic selection of the string gadget, if the cycle gadget is used. In addition, the checkbox gadget is only selected, if you actually changed the value in the stringgadget (if the option is disabled). This is useful when you use the keyboard (see below) to set the options (you can then easily move from one gadget to the next with the TAB key).

* Preserve settings:

Determines, whether all settings are automatically stored when leaving the macro (exept with "Cancel"). They are not saved to disk, however, they only remain active until the next reset. The saved settings can be restored by unselecting "Preserve settings" and "Set default" and clicking on the window's close gadget.

* Rescan:

If you have used the "Draw" gadget before (see below), select this to have FinalWrapper read in new text from the current paragraph. There's no need to use this gadget, if you want to use a new oval or textblock or some new passage of text (just select them).

* Set default:

Saves all current settings. Note that if you use the requester's closegadget and "Set default" is selected, the settings are saved but no action takes place ("Cancel" doesn't save).

* Ok:

Closes the requester and starts the drawing process.

* Draw:

Starts the drawing process without closing the requester. Therefore you don't

have to wait for the requester to pop up for the next tries. You can select any new oval, textblock or passage of text as usual for futher goes. But if you don’t select anything new, the old text/oval will be reused which leads to a shorter calculation time. If you want to use new text from the current paragraph, you have to use the "Rescan" option (see above).

While drawing, you can click on the "Cancel" or close gadget to abort the operation. You can go on working with the requester as usual (Autoactivate Gadgets and keyboard shortcuts might be a little lazy and "Ok"/"Draw" won’t work of course).

* Cancel:

Closes the requester without saving any settings.

* Close gadget:

Closes the requester and saves settings, if the apropriate gadgets are ticked.

* Object selection

Before clicking on "Ok" or "Draw" you first have to select an oval plus some text. This can be done at any time: Before starting the macro or while the requester is open. It doesn’t matter whether this is a textblock or a passage of normal text. If you don’t select any text, the current paragraph is used. 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 style^1.

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 oval^2.
Or: Place the cursor in the wanted paragraph and select the oval.

```
Keyboard shortcuts

  All options can be changed using the keyboard:

  * Fn (i.e. F1 to F8):  Works as if cycle gadget n (counting from  the  top)  ↩
     was
  selected with the mouse.

  * <Left Shift> + Fn: Changes the state of checkbox gadget n (on the  left  ↩
     side,
  therefore left shift key).

  * <Right Shift> + Fn: Activates string gadget n. Please note that  the  ↩
     keyboard
  shortcuts are disabled while a string gadget is active.

  * F9, F10, [, ], / (on numeric keyboard): Toggle the checkbox gadgets.

  * Esc: Cancel button

  * Return: Draw button

  * Enter: Ok button

  * Del: Window closegadget
```

1.7 Note on usage

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

1.8 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.
```

1.9 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.

1.10 Using version 1.4

Why using version 1.4?

Version 1.4 of FinalWrapper is released together with V2.x. I thought some
people might still want to use it, because...

* it doesn't need apig.library which saves quite some memory.

* there's a shorter delay until the requester pops up (it seems impossible to me
to make V2.x remarkably faster since Arexx is not that fast, but you don't have
to reopen the requester on every go).

* it has got an online help (V2.x doesn't need this, I think).

* the only difference between the two versions is the user interface.

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. If you don't
select any text, the current paragraph is used. 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 style^1.

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 oval^2.

```
Or: Place the cursor in the wanted paragraph and select the oval.

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 successor$^3$. Unknown ones are ignored.


* Sector size: An = n / A\ensuremath{\pm}n = \ensuremath{\pm}n / \ensuremath{\ ↩
   pm} = A\ensuremath{\pm} = A\ensuremath{\pm}360 (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
number 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  ↩
   misspleIt,
it's assumed to be the default value of 360. Too high / low numbers are  ↩
   changed
to 360 / -360. \ensuremath{\pm} stands for \ensuremath{\pm}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. "s+0" lets
the text be centered relative to the top of the oval. Positive values with
preceding "+" shift the starting point anticlockwise, negative ones clockwise.
"s\ensuremath{\pm}" is equal to "s\ensuremath{\pm}90". "sn" sets the absolute starting point (0 is at the
bottom of the oval, counting anticlockwise). 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 for the "s\ensuremath{\pm}n" and between 0
and 360 for the "sn" option.

E.g: s270 The text starts on the left side of the oval
a90 s+ 90 degree arc, centered to the left side

* Spirals: @n = @+n / @-n (default: @100)

```
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. "@100" is equal ↩
   to
"@-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 H100)

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. "h100" is the same as "-h100".

It's main purpose is to keep the height unchanged ("h100") 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.


* Adjusting: J+ / J- / J= (default: no adjusting)

The option provides you with two different ways to improve the ↩
   generated
effects:

The j\ensuremath{\pm} option stretches/squashes each character, so that ↩
   there are no gaps
between them. In addition, j+ changes the height in the same ratio, so ↩
   the
characters don't get distorted (gives a fancy effect with ellipses!).

Contrary to the above, the j= option changes the sector size to achieve the ↩
   same
```

```
goal. Therefore it's only useful, if you don't need a special sector size ( ↩
   great
for spirals!). If this option is used, the "a"  option  is  overridden,  but  ↩
   it
still  determines  the  direction (a+/a-), the starting point (if sn isn't  ↩
   used)
and it defines the shape of a spiral (@/h options, see example).

    E.g: a1080 @20 j=  Spiral, 20% of the original radius is reached after  ↩
       three
                       turns (1080\textdegree{}). But maybe  (if there's not  ↩
                          enough text) it
                       won't ever arrive there
         a180 j+       Fixed sector size, both width and height adjusted


* 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+?
```

1.11 Other macros

```
Some other goodies

  There are a few small (and  less  amazing  ;-{)  macros  that  come  along   ↩
     with
```

FinalWrapper. Nevertheless, they're quite useful some times, so I didn't want to hide them from public:

* CentreObjects
Just select some objects, call the macro and they'll be centred to the middle of the current edit area. The vertical positions keep unchanged.

* StretchObjects
Select one or more objects and start this prog. It will expand the objects to the full width of the edit area (only horizontally).

* ExpandObjects
Works similar to the above one, exept that the height is enlarged in the same ratio as the width.

* SizeNOblique
Set size and oblique of the first and the last character of a paragraph (or any two characters and select the passage between including the two chars). Now start the macro. The size and oblique of the characters between are now gradually changed from the first to the last one's.
E.g. You have the word "NICE!" with an 80 point N, and a 40 point !, so the I is going to be 70 points high, the C 60 and the E 50.

* Text2Block
Creates a textblock out of the selected text or - if no text is highlighted - the current paragraph, using all the different fonts, colours, sizes etc. It can be used like a standard textblock.

FinalWaver

Finally, one a bit more sophisticaded one: It creates sine-waves out of some text. It's used similar to FinalWrapper (1.4). If you start it, you'll be prompted a requester to enter your options:

* Waves: Wn (default: 1)
Number of complete sine waved to be created (so 1.5 means one and a half).

* Amplitude: An / A-n (default: 1)
Defines the height of the waves (double the value and they'll be twice as high).
The minus (-) mirrors the wave horizontally.

* Help: ? (default: no help)
Help requester as in FinalWrapper.

```
  As in FinalWrapper, you can select some text or a textblock first. By  the   ↩
     way:
  If  you  don't  use  any textblock, the waves are generated at the middle of  ↩
     the
  page.

  The program is manly based upon FinalWrapper 1.4a. The results are  not  ↩
     perfect,
  but I think you can live with them.
```

1.12 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.

  * The code isn't especially optimized (yet). Even some ugly pieces of  code   ↩
     can
  be  found  because  I  sometimes  wanted  to  keep the code size from growing  ↩
     to
  infinity rather than to make the code faster.


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[.n])
    "wintitle": %i  program info (=info)      (2.x only)
    "input":    %i  (dito)                    \ (1.4
                %d  default values (=default) / only)
    E.g.:
      info='FinalWrapper 2.x by NDY''s'
      wintitle="%i/Al «Kojack» Longhair"
      expanded to: "FinalWrapper 1.4 by NDY's/Al «Kojack» Longhair"

"helppages" can be increased for any language, if necessary (1.4 only)

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             Library not found requester
    test=2             Wrong things selected
    test=3             No OS 2.0 (2.x)
                       Requester/help: hit "Ok", try ALL requesters (1.4)
    test=4             Final Writer not found (text output to the Shell)
    test=6             Error requester
    test=7             FinalWrapper already started (2.x)
    test=8             Couldn't open requester (2.x)
    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 (and other things) instead...
```

1.13 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. Use the "Adjust" (J) option to improve this.

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

  * 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 " ↩
     Group:
  Invisible oval" (G-) and it should be OK.

  * Sometimes, the line above the  Ok/Cancel gadgets gets drawn in  wrong   ↩
     colours
  (or not at all). Why? Don't ask me...

  * "Group" won't work if "Draw" is used and no new oval has been selected  ↩
     before.

  * Bug reported by Amiga Oberland: A  disk  called  "Work:"  was  requested   ↩
     when
  saving  a  Final Writer doc created with FinalWrapper. I couldn't reproduce  ↩
     this
  one, sorry! (Renaming my "Work:" made no difference.) In fact, FWrapper does  ↩
     not
  access  any  files exept ENV:Locale, the prefsfiles and the libs. I really can ↩
     't
  imagine a reason for this behaviour!


  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. In the few tests with  ↩
     Final
  Writer Release 2 (27.04.94, german version) it seemed to work properly.
```

1.14 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  (17.03.94)  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
                    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
  1.4a (10.05.94)   Spaces are no longer treated seperately
                    Floating point numbers are now allowed
                    Procedure and variable "link" renamed
                    Now uses current paragraph if no text was selected
                    Sn option added
                    Included some other macros
  1.4b (14.05.94)   Object-deletion after an error much faster
                    Speedup normal text not much faster but results worse -> RIP
  1.4  (18.05.94)   Added J option
                    Added A\ensuremath{\pm} option
                    Now can be started from outside Final Writer
                    @0/H0 no longer equal to @100/H100
  2.0  (23.05.94)   Added GUI
                    Removed help requesters (not needed anymore)
                    Demodoc replaced by screenshots
  2.1a (04.06.94)   Now requester can be left open (thus added "Rescan")
                    "Preserve settings" switch added
  2.1b (07.06.94)   Faster processing when reusing the same text
                    Window zip-gadget added
                    Requester now works while processing the text
  2.1  (11.06.94)   Program would crash if started twice, now shows a messy
                    Now rescans new ovals/textblocks/selected text automatically
                    Now you can abort the drawing process
                    Ok didn't work anymore from 2.1b, fixed
                    Now shows a messy if it can't open it's requester

  V1.3 has been tested for quite some time, so it should be stable. As  far  as  ↩
      I
  know  until  now,  there  are no bugs in V1.4 either. (I can't test all  ↩
     possible
  option combinations, however!). V2.1 has only been used a few  times  until   ↩
     the
  new functions seemed to work properly.
```

1.15 About the future

```
The future of FinalWrapper

  By the time I'm writing this, I'm run out of improvment ideas (:<).

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

  Idea by Amiga Oberland: Wrap text around two circles at once.  Not  bad,  but  ↩
      I
  would  have  to reprogram the whole code for drawing. Therefore not likely to  ↩
     be
  implemented soon...

  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 plan to write another  macro  to  make  text  flow  alongside  a  ↩
     mathematical
  function  (e.g.  Sin(x)*3*x etc.) while the text size can be modified by  ↩
     another
  function simultaneously, if desired.

  Maybe I'll write a function plotter in near future (as soon as I  need  one ↩
     ...).
  Other suggestions are welcome, of course.
```

1.16 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 needed libraries and one of the ShortInfo docs.
```

```
  The following files are included:

  FW1_4/FinalWrapper.rexx             FinalWrapper 1.4
  FW1_4/FinalWrapper.install          Installer script for 1.4
  FW2_x/FinalWrapper.rexx             FinalWrapper 2.x
  FW2_x/FinalWrapper.install          Installer script for 2.x
  FinalWrapper.Guide    + .info       English AmigaGuide doc
  FinalWrapperD.Guide   + .info       German AmigaGuide doc
  ShortInfo.doc                       A very short description
  ShortInfo.dok                       The same in german
  DemoPics/#?.IFF                     Some demo screenshots
  DemoPics/Showpics.BAT               Script to view them
  OtherGoodies/#?.rexx                Some other macros
  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
  libs/apig.library                   For the requester (2.x only)


  * 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 and rexxsupport.library are (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!
```

1.17 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
```

1.18 Special thanks

Thanx...

...must go to Dietmar Eilert for his great GoldEd, to SoftWood for their FinalWriter 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 to Ronnie E. Kelly for his apig.library!!!

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