ASpellGUI

Fergus Duniho

COLLABORATORS			
	TITLE : ASpellGUI		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Fergus Duniho	July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ASpellGUI

1.1 ASpellGUI v1.6

The AlphaSpell GUI for ARexx capable text editors

1.2 Introduction to the AlphaSpell GUI

This is a graphical user interface for using the AlphaSpell spelling checker with a variety of different text editors. It makes use of Andy Cook's Varexx program to control a GadToolsBox GUI with ARexx. It also makes use of rexxtricks.library and rexxreqtools.library. The GUI comes with a collection of ARexx scripts, each for use with a different text editor. Most of the code in each script is the same, since most of it is just controlling the GUI. The scripts differ only at a few places where editor specific operations are required. These operations include writing your document to a temporary file, finding a word in the text, and replacing a word in the text. You should be able to easily adapt any of the scripts for use with another text editor. If you do, please send it along to me for inclusion in an update.

1.3 Legal Matters

may be distributed through the same channels as AlphaSpell can, and through no others.

I cannot be held accountable for any mishap, loss of data, or major catastrophe that happens as a result of using ASpellGUI or AlphaSpell. When you use the AlphaSpell GUI, you use it ENTIRELY AT YOUR OWN RISK.

If you adapt a script for use with another text editor, you may pass it along to me for distribution, but you must not distribute it on your own. The bulk of the code in a modified script is still mine, and I retain the copyright on it. In exchange for the copyright on the modified part of any script, I will give a free keyfile to anyone whose modifications I use in a future update. In the event that more than one person passes along the same modifications to me, I will provide a free keyfile only to the first person who sent me the modifications.

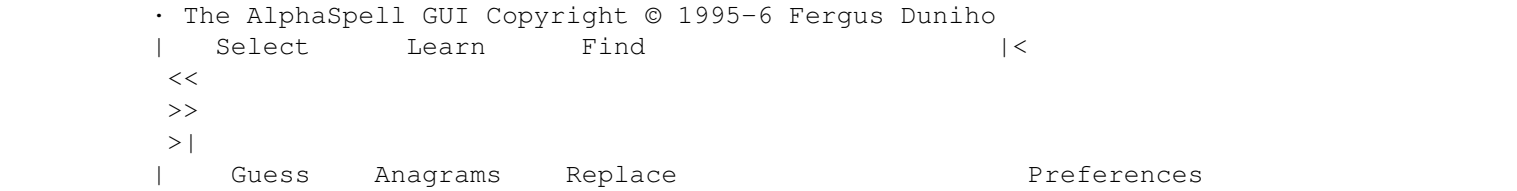
1.4 Using the AlphaSpell GUI

The AlphaSpell GUI has three different windows.

- The main window
- The prefs window
- The learn window

1.5 The main window

This window is designed to sit unobtrusively above the window that your document is in. That way, you don't have to move it around to see the words you want to check out and maybe change. This window lets you check your document for misspellings by letting you search for and change the words that AlphaSpell didn't find when it spell checked your document.



1.6 The Select Button

This button displays a listview of all the words that AlphaSpell didn't find when it spell checked your document. If you select a word from the listview, it shows up in both of the string gadgets, and the listview disappears. If you click on the close button of the listview, it goes away without changing anything.

1.7 The Learn Button

This button stores the word in the top string gadget to a list of words that you can save to your user dictionary when you're finished spell checking. It will put a lower case word in a list of lower case words, and it will put a mixed case word in a list of mixed case words. The Learn button is mainly for learning correct words that AlphaSpell didn't find in any of its dictionaries. But you can also add any word you want by entering it into the top string gadget before clicking on the Learn button.

1.8 The Find Button

This button searches through the text for the next (or first) occurrence of the word in the string gadget to its right. If that word has been newly changed, it searches for the first occurrence of the word. Otherwise, it searches for the next occurrence of the word.

1.9 The String Gadget for the Find String

This string gadget contains the string that you want to search through the text for. This will typically be one of the words that AlphaSpell didn't find, and you can move through the words in this list with << and >> buttons. You can change the value of this gadget with the Select button or by typing in a new word. You can learn the word in this gadget with the Learn button.

1.10 The << Button

This button moves you backwards through the list of unfound words. If you're at the beginning of the list, it moves you to the end.

1.11 The >> Button

This button moves you forward through the list of unfound words. If you're at the end of the list, it moves you to the beginning.

1.12 The |< Button

This button moves you to the first word in the list of unfound words.

1.13 The >| Button

This button moves you to the last word in the list of unfound words.

1.14 The Guess Button

This button has AlphaSpell guess at what the word in the bottom string gadget is supposed to be. It displays its guesses in a listview. When you select a word from it, the listview goes away, and the word shows up in the bottom string gadget. If you click on the listview's close button, the listview goes away, and nothing is changed. How AlphaSpell guesses a word is determined by the value of the edit distance, which you can set in the Preferences window.

1.15 The Anagrams Button

This works much like the Guess button, except that it finds anagrams. An anagram of a word is a word with exactly the same letters. For example, "live," "evil," and "vile" are anagrams.

1.16 The Replace Button

When you've found a word with the Find button, you can change it to the string in the lower string gadget by clicking on this button.

1.17 The String Gadget for the Replace String

This string gadget is supposed to hold the string that you want to replace the Find string with. You can change the value of this string gadget with the Guess button, the Anagrams button, or by typing in it. Also, it will change to the Find string whenever you move through the list of unfound words. This is to ease guessing of each unfound word, as well as to easily let you edit any changes you want to it.

1.18 The Prefs Button

This button opens up the Preferences window.

1.19 The Preferences Window

In this window, you can select your preference settings.

These are:

Dictionary Drawer

The path AlphaSpell searches for dictionaries in.

Dictionaries

A list or wildcard pattern indicating the dictionaries that AlphaSpell uses for spell checking and for guessing.

User Dictionary

The stem name for your user dictionary. This is so it knows where to send words that you want to learn. If you want the user dictionary used for spell checking and for guessing, you include it among the other dictionaries for those purposes.

Keyfile

This field should contain the full name of AlphaSpell's keyfile. AlphaSpell's keyfile is required for setting the maximum edit distance higher than two. It is also required for compressing dictionaries when AlphaSpell writes them. If you don't have the keyfile, the value in this field doesn't matter. You can get the keyfile by registering AlphaSpell.

Maximum Edit Distance

You select the edit distance with the slider gadget. This value determines how AlphaSpell guesses words. If it's set to 0, AlphaSpell finds words that phonetically resemble the target word. Otherwise, it finds any word whose edit distance from the target word is less than or equal to the value set for the edit distance. The edit distance is the minimum number of deletions, insertions, and transpositions it takes to change one word into another. Using ed(x,y) to mean the edit distance between the words x and y, the following theorems are true:

```
ed(x,y) == 0 if and only if x == y
ed(x,y) == ed(y,x)
ed(x,y) <= ed(x,z) + ed(y,z)
```

To save your preferences for future use, click on the SAVE button. To use your preference settings temporarily, without changing them permanently, click on the USE button. To cancel any changes you make, click on the CANCEL button or the CLOSE gadget.

1.20 The Learn Window

When you click on the CLOSE gadget once you're all finished spell checking, the Learn window will pop up if you selected any words for learning. It will display mixed case words in one listview and lower case words in the other. Each listview has a REMOVE WORD button for removing words from it. The mixed case listview has a MOVE WORD button for moving words from it to the lower case listview. When a word is moved from the mixed case list, it will be converted to lowercase. When you're satisfied with the words you want to add to your user dictionary, click on the SAVE WORDS button. If you decide not to save any words, click on the CLOSE gadget.

1.21 Acknowledgments

The AlphaSpell GUI was made possible by:

```
Varexx               Copyright © 1995-6 Andrew Cook
Arexxport.library    Copyright © 1995 Andrew Cook
rexxtricks.library   Copyright © 1995 Jürgen Kohrmeyer
reqtools.library     Copyright © 1991-4 Nico François and 1995 Magnus Holmgren
rexxreqtools.library Copyright © 1992-1994 Rafael D'Halleweyn.
AlphaSpell V         Copyright © 1995 Fergus Duniho.
```

Its design was influenced by Dirk Holtwick's MUISpell.

1.22 Requirements

Besides what comes with this archive, the AlphaSpell GUI requires AlphaSpell, reqtools.library, rexxreqtools.library, and Varexx v1.6. All of these are available on the Aminet:

AlphaSpell - /text/edit/ASpell.lha
Varexx - /util/rexx/varexx.lha

AlphaSpell is also available from the official AlphaSpell support site on BitNova.

1.23 About the Author

I am a 28 year old graduate student in the Philosophy Department at the University of Rochester. This semester, I am teaching a course on the nature of evil, and next Summer I will be teaching a course on computer ethics. I am the author of AlphaSpell, which this GUI is meant for use with. I have also written the XDME Excelsior Suite and a personality indicator known as the DDLI.

If you're on the Web, check out my homepage:

http://www.ling.rochester.edu/~duniho/index.html

Also check out the AlphaSpell support page:

http://www.ling.rochester.edu/~duniho/AlphaSpell.html

It contains links to AlphaSpell, its GUI's, and its dictionaries, as well as to stuff used by the AlphaSpell GUI. Anything I release of my own will show up here before it shows up on the Aminet. Registered users will be emailed about updates as soon as they're available.

If you have comments, questions, or suggestions, email them to:

fdnh@troi.cc.rochester.edu

1.24 History

v1.6 (13 March 1996)

Fixed a bug introduced in v1.5. The scripts accidently called some library functions before opening the function libraries.

Added scripts for several new text editors: AmokEd, Annotate, BlacksEditor, DME, SkoEd, Textra, TJM_DME, TKed, and TurboText.

Fixed bugs in the GoldEd script.

Modularized portions of the script, thereby making it a bit easier to convert one script to a script for another editor.

v1.5 (11 March 1996)

Bug Fixes in WordComp, which handles whole word searching.

Added script for Emacs.

Adapted the scripts for asynchronous execution. Recommended use:

Emacs: (shell-command "runback rx ASpell.elx")
XDME: execute (runback rx ASpell.xdme)

v1.4 (10 March 1996)

Bug Fix - The slider gadget in the Preferences window now works properly when you name your keyfile something other than S:Alpha-Key.

Some vestigial code removed. Things tidied up a bit.

Documentation on how to adapt the ARexx script for other text editors.

First (|<) and Last (>|) gadgets added to main requester.

v1.3 (5 March 1996)

Bug Fix - Whole word searching now works properly in the XDME version.

Added WordComp function, which is used by the XDME version, and may be used by other versions. It makes whole word searching easier for text editors without that capability built in.

v1.2 (12 February 1996, unreleased)

The gadget text, the screen text, and the window text can now be localized for different languages. WB 2.1 is not required for this.

The listviews in the Learn window are automatically updated.

It makes sure you aren't using an early version of Varexx.

Bug Fix - The XDME version properly searches for strings with

```
            apostrophes.

    Bug Fix - "Select" now works as it was supposed to.

    Bug Fix - Functions that read the replacement string no longer read the
              wrong value, as they would sometimes do after it had been
              read once.

    The XDME version no longer requires the XDME Excelsior Suite.

v1.1 (29 November 1995)

    Added public screen support
    Added keyfile field to preferences window
    Added screen titles
    Touched it up here and there

v1.0 (25 November 1995)

    Initial release
```

1.25 AlphaSpell Support

```
    AlphaSpell has an official support site on BitNova.

                  ____________________________________

                              BitNova
                        Telnet: bitnova.com
                       Phone : (510)581-0600
                      Web   : www.bitnova.com
                      FTP   : ftp.bitnova.com
                  ____________________________________

    You can also visit the AlphaSpell homepage:

    http://www.ling.rochester.edu/~duniho/AlphaSpell.html

    This page includes links to the latest versions of AlphaSpell, this
GUI, and the dictionaries. The newest stuff is always available here before it
is available on the Aminet.
```

1.26 A Note to Authors of Text Editors

```
        If you would like the AlphaSpell GUI to work smoothly with your
text editor, you should make sure that you includes ARexx commands
functionally equivalent to the following pseudo code examples:

FIND WORD/K WHOLE/S CASE/S FIRST/S

    This command searches for a word, and it includes options for whole
    word searching, for case sensitive searching, and for searching for the
```

first instance of the word in the document.

Whole word searching can be left out if you include the ability for an ARexx script to read the current word. For example, XDME lacks a whole word search mode, but ASpell.xdme includes a routine for whole word searching. The advantage of doing it in the script is that it uses the same rules as AlphaSpell does to recognize a whole word.

REPLACE OLDWORD/K NEWWORD/K CASE/S

This command replaces the OLDWORD with the NEWWORD. The CASE switch tells it to give the NEWWORD the same case as the OLDWORD. For example, REPLACE cat Dog CASE would replace "cat" with "dog."

It is important for this command to work with the current word, because the GUI does interactive search and replace. First it finds the word, then it gives you the option of changing it. A command that just replaces the next instance of a word will not do.

DELCHAR N/A
INSERT TEXT/K

These commands will do in place of a REPLACE command. The first one deletes N characters. The second inserts the given text into the document. It is probably best that no word wrapping goes on when using such functions from the script.

SAVEFILE TO/A

This command saves the contents of the current buffer to the named file. If it changes the name of the file, you will also need the ability to find out the current file name, and the ability to change the file name back to what it was after saving the buffer as a temporary file.

READ VAR/K

This command stores the value of one of the editor's own variables into RESULT. Useful variables include a failure flag (which tells whether the last command failed), the file name, and the current word.

To find out how functions such as these would work in a script, read How to adapt the AlphaSpell GUI script for other text editors.

1.27 How to adapt the AlphaSpell GUI script for other text editors

To adapt one of the ASpell ARexx scripts for your editor, all you need to change are five functions at the end of the script: FindWord(), ReplaceWord(), SaveTemp(), GetEditPort(), GetScreen(). You should set EDITPORT to the name of your text editor's ARexx port. Is is easiest to adapt ASpell.ed, ASpell.elx, or ASpell.xdme. ASpell.ged has some extra code in it, which you would have to delete.

1.28 FindWord()

This function is used for finding a word within a document. The first thing it should do is read the value of the target gadget. It does this with the lines:

```
read target
wrd = RESULT
```

These lines should be the same no matter what editor you use. The variable wrd holds the string that you want your editor to search for.

The next thing to do is check the value of arg(1). If its value is 0, FindWord should search for the first occurrence of wrd within your document. You can have FindWord do this by moving to the top before it begins its search, or you can use a find instruction that can be instructed to search for the first word. FindWord() should return 1. Here's an example using pseudo code:

```
ADDRESS
IF arg(1) = 0 THEN DO
    "FIND" wrd "CASE WHOLE FIRST"
END
ELSE DO
    "FIND" wrd "CASE WHOLE"
END
ADDRESS
RETURN 1
```

Notice that the searching done here is case sensitive. This just makes things easier. Also note that it does whole word searching. This is because you are always searching for whole words, and you don't want it to stop every time it finds a small word as a substring of a larger word. If your editor lacks a whole word search mode, here's another example:

```
ADDRESS
IF arg(1) = 0 THEN DO
    TOP
    FIRST
END

DO FOREVER
    "FIND" wrd "CASE"
    READ fail
    IF RESULT > 0 THEN DO
        TOP
        FIRST
        LEAVE
    END
    READ currentword
    cword = RESULT
    IF WordComp(cword, wrd, 1) = 1 THEN LEAVE
END
ADDRESS
RETURN 1
```

This example uses a command for moving to the top of the document (TOP), a command for moving the beginning of a line (FIRST), and a command for reading information from the program (READ).

The variable fail in this example contains a nonzero value if the last command failed, or a 0 if it didn't. This is to check whether the FIND command found another word. The FindWord() in ASpell.xdme works something like this. If you can't tell whether a command has failed, you can check whether the cursor has moved by comparing the old and new positions of the cursor. This is what FindWord() in ASpell.ed does.

The variable currentword contains the current word in the document. This word is compared to the word you're searching for, using the WordComp() function, which is already included in the script. The third argument in WordComp tells it to start comparing at the beginning of cword. This is required when the first argument passed to WordComp is a single word. For a real working example of this method, take a look at FindWord() in ASpell.xdme.

You can also give WordComp a whole line for its first argument. If you do, you need to pass the X position of the cursor as the third argument. For a real working example of this method, take a look at FindWord() in ASpell.ed.

1.29 ReplaceWord()

This function replaces the current word with the word in the replacement string gadget. It begins by reading the value of this gadget:

```
read replacement
newword = RESULT
```

Depending on how your replace command works, you may need to read the value of the target gadget. The pseudo code examples here require it. Take a look at ASpell.xdme and ASpell.elx for examples that don't.

```
read target
wrd = RESULT
```

Here's an example using the REPLACE command:

```
ADDRESS
"REPLACE" wrd newword
ADDRESS
RETURN
```

Here's an example using DELCHAR and INSERT:

```
ADDRESS
"DELCHAR" Length(wrd)
"INSERT" newword
ADDRESS
RETURN
```

1.30 SaveTemp()

This function saves the current document as a temporary file. It uses the file name that the script already stored in the variable tempfile. Here's an example in pseudo code:

```
ADDRESS
"SAVEFILE TO" tempfile
ADDRESS
```

It is important that your SaveTemp() function does not change the name of your document. With some text editors, you may have to read the file name, save the file, then change the file name back. Here's an example:

```
ADDRESS
READ filename
oldname = RESULT
"SAVEFILE TO" tempfile
"CHANGENAME" oldname
ADDRESS
```

Take a look at ASpell.ed and ASpell.ged for real working examples that do this.

1.31 GetEditPort()

This function returns the name of the ARexx port to use for communicating with the text editor. If your editor doesn't use a named port, it should return the empty string. Generally, it checks whether the current port belongs to your editor. If it doesn't belong to it, it checks whether a port to your text editor is open. If it is, it returns the name of that port. If not, it exits from the script. Here's an example:

```
GetEditPort:
IF Abbrev(Address(), "FREXXED.") = 1 THEN RETURN Address()
IF ~SHOWLIST("P", "FREXXED.1") THEN DO
    CALL rtezrequest "FREXXED.1 unavailable", "_Abort", "Missing Port:"
    EXIT
END
RETURN "FREXXED.1"
```

With some editors, you may want to enforce asynchronous operation, because the editor has some problems when it runs the script directly. I had to do that with XDME:

```
GetEditPort:
IF Abbrev(Address(), "XDME.") = 1 THEN DO
    CALL rtezrequest "execute (run rx ASpell.xdme)", "_Abort", "Run this script  ↩
       asynchronously: ", rttags
    EXIT
END
IF ~SHOWLIST("P", "XDME.1") THEN DO
    CALL rtezrequest "XDME.1 unavailable", "_Abort", "Error:", rttags
```

```
    EXIT
END
RETURN "XDME.1"
```

1.32 GetScreen()

This function returns the name of the screen your editor is operating on. Most editors use the Workbench screen, and all you need for them is the following:

```
GetScreen: PROCEDURE
RETURN GETDEFAULTPUBSCREEN()
```

Some editors use their own custom screens. If your editor has some way of telling you the screen name, you can use that. For example, BlacksEditor lets you do this:

```
GetScreen: PROCEDURE
"GetScreenInfo"
info = result
screen = Word(info, Words(info))
RETURN Substr(screen, 2, Length(screen)-2)
```

Note that you do not need to enclose this with ADDRESS commands. This function is called while your text editor's port is the current port.

If your editor lacks this feature, you can just name the port:

```
GetScreen: PROCEDURE
RETURN "SkoEd"
```

1.33 How to Register AlphaSpell

Although this GUI is FREEWARE, AlphaSpell itself is SHAREWARE. You can register AlphaSpell for $20.00 in American currency. You can register with cash, check, or credit card. To register with cash or check, send $20.00 in American currency, or a $20.00 check drawn from an American bank, to:

```
Fergus Duniho
1095 Genesee St.
Rochester, NY 14611-4148
```

To register with your Master Card, Visa, Discover, or American Express card, you have to register on-line through BitNova. You can register by telneting to BitNova directly or through BitNova's World Wide Web Site. BitNova is accessible through the AlphaSpell web page:

```
http://www.ling.rochester.edu/~duniho/AlphaSpell.html
```

1.34 Supported Editors

I have made ARexx scripts for several different text editors, but they are not all equal. Some text editors lack the features the GUI needs to work smoothly with the editor. Here are the text editors I have scripts for, grouped from BEST to WORST, with each group ordered alphabetically. Please note that these are appraisals of how well each editor works with the AlphaSpell GUI, not appraisals of the text editors themselves. I don't want to offend the authors of any of these products.

BEST: Ed, FrexxEd, GoldEd, GNU Emacs, TJM_DME, TurboText, and XDME

GOOD: BlacksEditor

ADEQUATE: Annotate and Textra

POOR: AmokEd, DME, and SkoEd

WORST: TKEd

Those that work best with the GUI perform whole word searching, run asynchronously, put the GUI on the same screen as the editor, and operate smoothly. Among these, FrexxEd, GoldEd, and GNU Emacs use their own built-in routines for whole word searching. That makes searching faster for these editors, but there is occasionally a disparity between what AlphaSpell recognizes as a word and what the editor recognizes as a word. But the problem rarely arises, and the asynchronous operation of the GUI let's you handle it when it does. All the rest, as well as BlacksEditor and SkoEd, perform whole word searching with the help of ARexx. That makes it a bit slower, but that's compensated by the ARexx routine recognizing words in the same manner as AlphaSpell does.

BlacksEditor works well with the AlphaSpell GUI, but it won't let you save a temporary file without saving the file. Annotate and Textra are only adequate for the GUI's purposes, because they don't have features sufficient for the implementation of whole word searching. AmokEd and DME share the same problem. Plus, they have difficulty indicating which word was just found when you search for a word. SkoEd works poorly with the GUI, because the GUI cannot appear on its custom screen. TKEd has the same problem as SkoEd. Plus it won't save a temporary file. So, unless it can be rewritten, it can't actually be used with the AlphaSpell GUI.

I've included scripts for editors that don't cooperate well with the GUI in the hope that others might improve them, or that the authors might give their editors the features that they need to work well with the AlphaSpell GUI. I urge the authors of some of these editors, as well as the authors of other editors, to read my note to text editor authors.