Manual

Thomas Aglassinger

COLLABORATORS			
	TITLE : Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Thomas Aglassinger	August 25, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Manual

1.1 Eiffel/Sofa Mode for GoldEd Studio 6

```
Eiffel/Sofa Mode for GoldEd Studio 6

About

    Copyright        - Freeware
    Requirements     - What else you need
    Installation     - Installer script rulez

Usage

    Syntax parser    - Say it with colors
    Scanner          - Who's there?
    Toolbar          - Only one mouse-click away
    Templates        - Lazy is beautiful
    Mouse'n'browse   - Can I click it?

Miscellaneous

    Support          - Where to get updates
    Advanced topics  - Some internals
    Known problems   - Watch out
    Future           - Things to come
    History          - Sofa, so good


Copyright 1999 Thomas Aglassinger and others, see file  forum.txt
```

1.2 Requirements

```
Most of the ARexx scripts included  use  the  "rexxdossupport.library",
Copyright    ©    Harmut    Goebel    and    available    from    Aminet    in
util/rexx/rexxdossupport.lha.

The Sofa distribution of the freeware SmallEiffel compiler is available
```

from http://www.owlnet.net/amiga/sofa/.

GoldEd Studio is a commercial product and Copyright © Dietmar Eilert. For more information refer to http://members.tripod.com/golded/.

1.3 Installation

To install the whole package, use the included Installer script.

If you also want to use the scanner from outside the toolbar, you have to update the scanner configuration manually. To do that, select "Search/Show function list". Click on "Mode" and then on "+" to add a new scanner. Assign "GOLDED:etc/scanner/eiffel" to the filetype "*.e".

1.4 Syntax Parser

The syntax parser highlights syntax elements in different colors.

Right now it simply utilizes Dietmar Eilert's generic.parser. Actually, I already wrote more Eiffel-specific custom parser. Unfortunately, the parser interface of GoldEd changed from 6.1 to 6.2, and I'm too lazy to update my parser in the near future.

For your confusion, this manual most of the time refers to my custom parser, in particlar when talking about bugs and future enhancements.

1.5 Scanner

The scanner extracts names of features declared in a class. It takes into account all types (functions, procedures, attributes and constants) and all available features (including those exported to {NONE}).

You can activate it from the Toolbar, by selecting the menu item "Search/Function list..." or by pressing Amiga-/.

(Technically speaking, the scanner takes into account lines with an indention level of three blanks or one tab. It then expects the line to either end with "is" or contain a colon (:)).

1.6 Toolbar

The functionality provided by the toolbar depend on the view mode the source code is in:

All modes

Make

Starts GoldEd command ‘PROJECT MAKE’.

You can change the ‘make’ macro in in the GoldEd dialog "Filetype/Projects/Details".

Open class

Find file for class of the word the cursor currently is positioned over using ‘finder’. Then load it in new window.

Note: Control-double-clicking a class name has the same effect.

Open short form

Display short form of the class of the word the cursor currently is positioned over in new window.

Note: Shift-double-clicking a class name has the same effect.

Help

Views this manual.

View as class
View as short form

Changes the view mode. The image for the current view mode is always ghosted.

Only in Eiffel mode

Feature list

Invoke the scanner to view a list of all features declared in the current source code. Features inherited from other classes are not shown. (Use the short form for that.)

Reformat

Reformats the current source code editor using SmallEiffel’s ‘pretty’.

Note that different when calling ‘pretty’ from the CLI, no ".bak" file is created as GoldEd offers more sophisticated backup and undo facilities. This avoids clutter in the source directory.

Create new class

Ask for name of new class, create its file in the current directory. Then fill it with a template and load it into a new window.

The class name must consist of letters, digits and underscores (_). Case does not matter, the automatically script uses lower case for the filename and upper case in the class source. The

".e" suffix also is appended automatically.

See Advanced topics on modifying the template to fit your personal preferences.

1.7 Mouse'n'Browse

Using qualifiers when double-clicking the source code, you can browse classes without having to leave the editor. This gives you a simple version of a seamless environment without a dedicated class browser.

```
Control-double-click = Open class
Shift-double-click   = Open short form
```

1.8 Templates

You can use the following templates to speed up typing certain structures:

```
dE (debug instruction)
   debug("")... end

fR (loop)
   from... until... loop... end

iF (conditional)
   if... then... else... end

iN (multi-branch)
   inspect... when... else... end

iS (routine declaration)
   is... require... local... do... ensure... end;
```

Note: Unlike in most other environments, you don't need any qualifier key like Alt or Control to use templates. This is possible because the Eiffel naming conventions forbid an lower case letter immediately preceeding an upper case letter.

1.9 Support

New versions will be uploaded to Aminet. Look for text/edit/envSOF*.lha

You can contact the author at the following e-mail address:

Thomas Aglassinger <agi@sbox.tu-graz.ac.at>

1.10 Advanced Topics

New class template file

The file "GoldEd:add-ons/eiffel/new_class.e" contains the template for new classes created from the toolbar . You can change it if you like, but remember to bump the version number to something huge (e.g. 50.0) so that it is not overwritten by the installer when updating the environment.

You can use the following placeholders:

```
%/CLASS/   insert name of the new class
%/CURSOR/  position where the cursor should go to after loading
%/DELETE/  marks line to be deleted (useful for comments)
```

New syntax parser keyword file

Most parts of the syntax parser are hard-coded, meaning that it needs a change in its source code and a recompilation to make the Eiffel source look different.

The only exception of this are keywords. The parser reads them from an external file you can specify in "Filetype settings/Syntax Highlighting/Details/Arguments". The structure of this file is as follows:

- every line contains exactly one keyword
- lines beginning with "--" are ignored
- empty lines are ignored

Furthermore, the whole file has to be sorted. White space at the beginning or end of a line is not allowed. For example:

```
-- some keywords
Precursor
True
end
if
then
```

If the keyword file does not exist or not conform to the above rules, the parser views an error messages and refuses to render anything. For some examples keyword files, take a look at "golded:add-ons/eiffel/syntax/".

(In theory, the parser could easily remove white space and sort the keywords by itself. In practice, this is a pain in the arse because it is written in C. As this feature is rarely needed, I considered the inferior usability acceptable.)

1.11 Known Problems

Problems with the syntax parser

The second line of the excerpt

```
print("string ends here"
      %string does not continue"); -- syntax error
```

is rendered as manifest string, though it isn't. Fixing this would need a context parser, which means more work. As this is not proper Eiffel anyway, the compiler points out the problem later.

Thus I don't plan to ever fix this.

Problems with the scanner

GoldEd's reference functions don't work in any useful way because they support only a global name space, but Eiffel has an own name space for every class.

I haven't found a way yet to make the scanner work with the quasi-Eiffel outputted by 'short'. The main problem is that a scanner can see only one line of the source code at a time and does not have any context. Consider the following source excerpt:

```
class HUGO
creation
   make
feature
   make is
     ...
```

To avoid that 'make' shows up twice, it has to check for the " is" at the end of the line. This works nice in the full source. In the short form however, the interface looks like this:

```
class interface HUGO
creation
   make
feature(s) from HUGO
   make
   ...
```

Finally, some indexing lines and invariants might show up as attribute declarations. Consider the following example:

```
indexing
   hinz: kunz
      -- indexing field consisting of only one word
class HUGO
   ...
invariant
   valid_state: valid_state
      -- boolean query without parameters
   ...
```

Here, 'hinz' and 'valid_state' show up in the feature list.

All these issues can only be addressed by an improved scanner

concept of GoldEd.

1.12 Future

This is a list of ideas that might be implemented in future versions.

· Make syntax parser recognize routine declarations.

· Spell check comments (I always wanted that in every syntax parser)

· Make SmallEiffel use ARexx for sending error messages to editor.

Most on the things mentioned below depend on my laziness and how a possible SmallEiffel movement develops on Amiga. So don't hold your breath.

1.13 History

Version 2.0, 11-Oct-1999

- · Updated for GoldEd Studio 6.
- · Fixed bug in syntax parser: also removed "false" from list of keywords.
- · Added Mouse'n'browse
- · Added detection of attributes and constants to scanner
- · Added support for syntax parser to read keywords from external file.
- · Renamed 'finder.ged' to 'open_class.ged'
- · Renamed 'short.ged' to 'open_short.ged'
- · Renamed 'pretty.ged' to 'reformat.ged'
- · Changed 'open_short.ged' so that is caches short forms in a relative file "sofa/short/<class>.e". This was necessary because the online creation of short forms is too slow to be useful in practice.
- · Changed 'open_short.ged' so that the created short form is read-only.
- · Changed 'reformat.ged' so that it automatically removes the *.bak created by 'pretty'. I found the backups cluttering the source directory annoying, especially because GoldEd supports more sophisticated backup and undo facilities anyway.
- · Changed copyright to Eiffel Forum Freeware License and included full source code.
- · Removed "execute.rexx" and "console.rexx" from the archive. Changed (open_class|open_short|reformat).ged so that they won't need them anymore. Now errors cause a new console to pop up. This sucks. But I think it sucks less than not seeing the errors at all. The only reasonable solution to this is a message browser. Which is not included with Sofa yet. This sucks, but hopefully might change in future.
- · Removed support for GoldEd 4. But you can still download an older version that provides the basic functionality of this environment

for GoldEd 4 from aminet:text/edit/GedEiffel.lha.

Version 1.2, 4-Mar-1999

- Fixed bugs in (finder|pretty|short).ged, which all were still referring to an old internal version of "execute.rexx" not being part of the distribution.
- Fixed bugs in scanner:
  - frozen is skipped
  - infix and prefix include succeeding operator name
  - comments ending with " is" are ignored
- Fixed bugs in syntax parser:
  - Added syntax class "Reserved word" for Result, Unique, True etc
  - Added keyword not
- Documented that one can shift-double-click a class name to load it.
- Added screenshot (yeah, yeah, yeah!)

Version 1.1, 25-Jan-1999

- Added routine feature scanner
- Added toolbar
- Added various ARexx scripts to invoke most SmallEiffel tools from the toolbar
- Added templates
- Added SmallEiffel console
- Added Installer script
- Improved syntax parser. It now is a cached parser and supports a lot more different syntax elements.
- Fixed "bugs" in Eiffel example shown in syntax parser configuration requester

Version 1.0, 15-Jan-1999

- Initial release featuring syntax parser