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 GedEiffel - A SmallEiffel Mode for GoldEd

```
GedEiffel - A SmallEiffel Mode for GoldEd

About

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

Usage

    Syntax parser  - Say it with colors
    Scanner        - Who's there?
    Console        - Show what's going on
    Toolbar        - Only one mouse-click away
    Templates      - Lazy is beautiful

Miscellaneous

    Support        - No support
    Known bugs     - Watch out
    History        - What happened so far


Copyright © 1999 Thomas Aglassinger <agi@sbox.tu-graz.ac.at>
```

1.2 Copyright

using GedEiffel.

1.3 Requirements

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

The SmallEiffel compiler for Amiga is available from Aminet in dev/lang/SmallEiffel*.lha. For more information about it refer to the SmallEiffel Homepage at http://www.loria.fr/SmallEiffel/.

Naturally, you will also need GoldEd, Copyright © Dietmar Eilert.

1.4 Installation

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

If you do not trust the script, you can perform the following steps manually:

· Copy "syntax/eiffel.parser" to GoldEd:syntax/
· Copy "scanner/eiffel" to GoldEd:scanner/
· MakeDir GoldEd:toolbar/eiffel
· Copy "toolbar/eiffel/#?" to GoldEd:toolbar/eiffel/
· MakeDir GoldEd:tools/eiffel/arexx
· Copy "Manual.guide" to GoldEd/tools/eiffel/
· Copy "tools/eiffel/arexx/#?" to GoldEd:tools/eiffel/arexx
· Copy "presets/#?" to GoldEd:presets/

If you are using GoldEd 5, wait for the next release.

1.5 Syntax Parser

The syntax parser highlights syntax elements in different colors. It conforms to the rules of chapter 26 of the book Object Oriented Software Construction (2nd ed., 1997). A PDF version of this chapter should also be available from http://www.eiffel.com/.

Comment
: Beginning with two hyphens (--) and lasting until the end of the line.

Manifest string
: Used for text enclosed in double quotes ("). Multi-line strings ending with a percent sign (%) and continuing in the next line after another percent sign are also supported.

  Additionally, character constants enclosed in single quotes (') are rendered in the same color.

Keyword
Used for keywords of the language that play a strictly syntactic role, like if, feature or invariant.

Reserved word
Used for reserved words that carry an associated semantics, such as Current, Result, True and Unique.

(Note: The rule for some of these have changed since 1992, when Eiffel: The Language was published. For example, True was initially written true.)

Type name
Any identifier that may be used as a type, or part of a type, should be written all in upper case. This includes:

- Class names, in all possible uses
- Formal generic parameters, such as G in LIST[G].
- Basic types like BOOLEAN, INTEGER and BIT.

Constant name
Constant attributes should be written with an initial capital letter, with the rest in lower case, as in

- Red, Green, Blue: INTEGER is Unique
- Something_to_stay_that_way: REAL

Error or flaw
Used for some things that are possibly wrong:

- Unterminated manifest strings
- Names not conforming to any convention, like in DontUseThis.

However, this is no complete check. You code might still contain various syntax errors.

1.6 Scanner

The scanner extracts names of routine features.

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

(Technically speaking, it searches for lines that end with " is" and assumes the first word to be the name of the routine.)

1.7 Console

The idea behind the SmallEiffel console

Many buttons from the Toolbar invoke normal CLI commands like ‘make’

and ‘pretty’. Though GoldEd allows such commands to open a new console to show their output, this is not very practical. Quickly one finds himself closing the console all the time.

The SmallEiffel mode uses a different mechanisms: all commands write their output to the same console, named SmallEiffel Console. Its window title is SMALLEIFFEL.1. Basically, you do not have to bother about that because everything is set up properly during installation.

The only important thing to keep in mind is that the SmallEiffel console is output only. That means, although you can apparently enter commands into it, they will never be executed.

Specifying an alternative console window dimension

If you do not like the window dimension of the console, you can specify your own by setting the environment variable "env:Console/SMALLEIFFEL.1" with a proper console specification, for example

```
makedir env:Console
setenv Console/SMALLEIFFEL.1 "CON:0/0/400/200/SMALLEIFFEL.1/SCREEN**/AUTO/ ↩
   CLOSE/WAIT/INACTIVE"
```

The "SCREEN**" will be descaped to "SCREEN*", which refers to the frontmost screen (usually "Workbench" or "GOLDED.1"). For more details on console specifications, refer to the AmigaDOS manual or check the guide that comes with KingCON (available from Aminet).

This is all you have to know for just using the console. Stop reading, unless you want to get confused.

How it works inside

For those how want to know what going in "inside", here’s a more detailed description: The whole effect described above is achieved by two ARexx scripts:

Console.rexx kind-of is a CLI with an ARexx-port. It simply tries to execute every message as a CLI command. However, after executing a command, the console does not quit and close its window, but simply waits for the next command.

Execute.rexx is used to send such a command to the SmallEiffel console.

This might sound more complicated than it actually is. Consider the following CLI invocation:

```
run <>nil: rx >CON: golded:tools/eiffel/arexx/console.rexx Port=SMALLEIFFEL ↩
   .1
```

This simply starts a console offering an ARexx port named ‘SMALLEIFFEL.1’. Because of the ‘run’ command, it runs as an own process. Because of the redirection to CON:, it opens an own window for the new console. Theoretically, you can now make it execute a command like seen below:

```
rx "address SMALLEIFFEL.1 'echo hello'"
```

This executes the CLI command 'echo hello' in the SmallEiffel Console, and also shows its output in it: the word "hello".

However, this is not the recommended way. Instead, use the following:

```
rx golded:tools/eiffel/arexx/execute Port=SMALLEIFFEL.1 echo hello
```

The main reason for that is that the above script always takes care that the output console has the same current directory as the sending console. Furthermore, it will even invoke console.rexx if no console is already running at the requested port. And it takes care of the little magic needed to read the Console/SMALLEIFFEL.1 environment variable.

1.8 Toolbar

The Eiffel toolbar offers the following functions:

Make
: Starts 'make'. You have to provide a Makefile in the current directory.

Indent pretty
: Reformats the current source code in the editor using SmallEiffel's 'pretty'.

  If 'pretty' fails because the code contains syntax errors, it is left unchanged. In that case, compile it for more details about these errors.

  A backup is created with the same filename but the extension ".bak" in the directory where the original source code is located.

Find class
: Find and load source file for class of the word the cursor currently is positioned over using 'finder'.

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

View short form
: Display short form of the class of the word the cursor currently is positioned over.

  (This can be considered some kind of online help. Unfortunately, the awful speed of 'short' renders it pretty useless in practice.)

Browse classes
: In a future version, this will show the class browser.

View list of routines
: Invoke the scanner to display a list of all routine features.

Help
: Views this help.

Close console
   Close the SmallEiffel console. This is useful if you want to have more space for typing and do not intend to use any of the above functions for some time.

   The next command activated from the toolbar will automatically start a new console when needed.

1.9 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;

1.10 Support

This package is not supported anymore as GoldEd 4 is outdated. For a supported version for GoldEd 5, download text/edit/envSOF*.lha from Aminet.

Nevertheless, you can contact the author at the following e-mail address:

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

1.11 Known Bugs

· For some reason, SmallEiffel error messages do not show up in the console when it was started from a make tool which was started from the toolbar. This renders the current implementation of the make button more or less useless.

· The syntax parser renders the second line of

   print("string ends here"

```
        %string does not continue"); -- syntax error
```

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.

- The syntax parser is very slow when closing the window because all parser chunks are allocated using AllocVec(). Using memory pools would probably speed this up.

1.12 Future

No future.

1.13 History

Version 1.3, 9-Nov-1999

- This is just an update of the documentation.
- As GoldEd 5 is released, there is no point in continuing supporting GoldEd 4. However, GoldEd 4 is free and GoldEd 5 is not, so I decided to leave this archive as it is, but without any support.
- Download aminet:text/edit/envSOF*.lha for a version that supports GoldEd 5.

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