E code generation

COLLABORATORS			
	TITLE : E code generation		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 26, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

E code generation

1.1 Introduction

```
*****************************************************************************

                              GenCodeE (v1.3)
                    E code generation module for MUIBuilder
                            Update of 10 March 1994
                    © Copyright 1993, 1994, Lionel Vintenat

*****************************************************************************

 Introduction
 The classic mode
 The environment mode
 The generated code
 The error messages
 The examples
 Bugs
 History
 Future
 Distribution
 The author
```

1.2 Introduction

GenCodeE has 2 working modes : the classic mode and the environment mode.

You will find the classic mode familiar if you have used GenCodeE v1.0 which was provided with MUIBuilder v1.0. Indeed, this classic mode works as GenCodeE v1.0 did (so its name), but with localization support added.

The environment mode is the "great" novelty of GenCodeE v1.3. Indeed, in this mode, GenCodeE will directly modify your E source file to add or update some MUI code.Its use is of course trickier than the use of the classic mode, but infinitely more pleasant, avoiding heavy cut/copy/paste.

The selection of GenCodeE working mode is done by the "Environment" checkmark in the code generation window of MUIBuilder.

The "Locale" checkmark indicates if you want a localized code or not. Its action is the same for both working modes. On the other hand, the "Declarations" and "Code" checmarks don't act at all in the same way according to the working mode. So read carefully the corresponding paragraphs.

1.3 The classic mode

In this mode, GenCodeE writes in the file given in the "Code" string gadget the MUI source for the application or the selected object. Warning : if this file already existed, it is erased without any confirmation request.

If the "Declarations" checkmark is activated, the generated source will contain the variable declarations, and their initialization values for the selected object (or all the variables and their initialization values if you have requested the whole application).
If the "Code" checkmark is activated, the generated source will contain the code for the selected object (or all the code if you have requested the whole application).
You can mix these 2 gadgets to get only the declarations, or only the code, or both in the generated source.

Be careful, in the classic mode, GenCodeE only produces source code to manage the GUI you are developping. Contrary to GenCodeE v1.0, it doesn't generate any environment code like for instance the opening of the muimaster library or the classical waiting loop of an MUI application.
So, I join to this archive 2 E source skeletons to give you a base :
- Application.e for an MUI application without localization
- Application+Locale.e for an MUI application with localization

For the part specific to localization of Application+Locale.e, I was inspired by the generated code of FlexCat. So refer to this progam for more details. Otherwise, it's an excellent program which is worth seeing (its author, Jochen Wiedmann, can be reached on internet at the followind adress : wiedmann@mailserv.zdv.uni-tuebingen.de).
These source files are only examples. So fell free to modify them in order to adpat them to your specific needs.
Note that the doMethod function which is in these 2 files, was given to me by Wouter van Oortmerssen (thanks Wouter !), the Amiga E author.

1.4 The environment mode

In this mode, GenCodeE will try to modify the file given in the "Code" string gadget, to add or update the MUI source of the application or of the selected object.

To achieve this, your source file must obey some rules because GenCodeE isn't an artificial intelligence program, and it can't know what it has to change if nobody helps it !
To be more precise, GenCodeE looks for 2 specific places in your source file :
- where you put your global variable declarations
- the procedure where the variable initializations and the code must be put

in

Indeed, in the environment mode, GenCodeE generates a specific procedure for the creation of each particular object, as for the application. So, all the variable declarations used by the GUI must be global, in order to be reached by each one of these procedures.
In order that GenCodeE spots these variables in your source file, they must be placed on a line which begins by /*MUIB*/. In fact, for each generation, GenCodeE deletes all the lines which begins with /*MUIB*/ and replaces them by the new ones. Be careful, these declarations must follow each other without interruption. GenCodeE will stop its replacing as soon as it meets a line which doesn't begin by /*MUIB*/.
So all you have to do, is to place before the 1st generation, one line which only contains /*MUIB*/, at the place where you declare your global variables. After, GenCodeE will do everything for you.

In the environment mode, GenCodeE put the variable initializations and the code of the GUI in a procedure specific to each object and to the application. For the application, this procedure is named create_app() and for an object with the label ObjLabel, it's named create_ObjLabel().
So to generate code, GenCodeE starts by searching a line which begins with "PROC create_app()" or "PROC create_ObjLabel()" according to the case, then it looks for the final ENDPROC of this procedure. After, it deletes all the contents of this procedure before inserting inside the new variables initializations and the code.
So all you have to do, is to place before the 1st generation, one line which only contains "PROC create_app()" or "PROC create_ObjLabel()" according to the case, followed by another line which only contains "ENDPROC". After, GenCodeE will do everything for you.

As the influence of the "Declaration" and "Code" checkmarks on the generated source is a bit complex, look at this array which clearly presents the things :

	"Declarations" ON	"Declarations" OFF
"Code" ON	initializations+code	only code
"Code" OFF	only declarations	nothing !

Let's take an example which mirrors 99% of the cases. For your GUI, you want to create it without stopping on the way, but you also want to access to one particular element (let's take a list of label my_list) for a specific need. So you add to your source file :
- one line which only contains /*MUIB*/ where you declare your global variables
- the empty procedure create_app()
- the empty procedure create_my_list()

Then, for the creation or the update of the variables, you click on the "Appli Code" button with the "Declarations" checkmark activated and the "Code" checkmark desactivated. For the creation or the update of the initializations and of the code inside your procedures, you activate both "Declarations" and "Code" checkmarks, then you click on the "Appli Code" button or on the "Object Code" button according to the case. And each time, your source file will be correctly modified by GenCodeE.
Note that if you only activate the "Declaration" checkmark and not the "Code" one, and then if you click on the "Object Code" button, GenCodeE will

replace all the global variable declarations by these ones which are only related to the selected object. This is rarely what you want, because it is all the variables of the application (and not only these ones which are relatives to one object) which are interesting.

A last advice : make some tries with this working mode before using it really. As it directly modifies your source file (and so very precious data), it's careful to well understand the way it works before any using on one of your source files.

1.5 The generated code

As MUI-Builder generates generic code which is later interpreted by modules, E generated code is identical in its content to C generated code. So, refer to language C paragraph for all that is related to generated variables, notification and localization. The only important difference is the MUI_TRUE constant which must be defined to the value 1 and which is used instead of TRUE (which equals -1) for MUI, because MUI doesn't always recognize value 1 like true.

The only difficult point is related to the strings you enter using MUI-Builder. The general rule is to enter them as if you want to use them with C language, with one exception: unlike what was stated before, the character '"' can be entered without a '\' (it was a problem only C).
The E code generation module will translate it correctly. More precisely, this module recognizes:

- \r replaced by \b
- \n, \t et \e leaved unchanged
- \0oo where oo is an octal number : be careful, the \ must be followed by a 0, itself followed by 2 figures among 0 and 8
- \xhh where hh is an hexadecimal number : be careful, the \ must be followed by a x (case sensitive), itself followed by 2 correct symbols (figure or letter among a and f, or A and F)
- ' replaced by \a

Be careful - if the syntax of \0oo or the syntax of \xhh isn't correct, the E generation code module won't print any error but the result may not be the one you expect!

If E code generation module meets \033 or \x1B (decimal value 27, namely the ESC code), it will replace it by \e. If on the other hand it meets \0oo or \xhh with a decimal value different from 27, it will produce a string not under the form 'string', but under the form ["s", "t", "r", "i", "n", "g", 0]:CHAR because there isn't equivalent things in E language to \0oo and to \xhh of the C language. But the 2 forms are totally equivalent.

Finally, the generated source file is full of macros identical to the C language ones. To use those macros, you will need an annex preprocessor because Amiga E doesn't support macro replacment for the moment. So, MUI-Builder is provided with an extra archive, Mac2E.lha (available alone on aminet too). You will find in this archive all that you need to use generated source files, especially with a preprocessor specialized in macro replacment designed for Amiga E. Refer to this archive for more details.

1.6 The error messages

When an error happens, GenCodeE prints an explaining requester :
- "Out of memory !" -> GenCodeE needs more memory !
- "I/O problem !" -> GenCodeE can't find, or read, or write a file
- "Incorrect source format !" -> GenCodeE can't find in your source file /*MUIB*/ or create_app() or create_ObjLabel() or ENDPROC depending on your selection
- "You must generate a label for the object !" -> you have requested the code of a specific object in the environment mode without generating a label for it

1.7 The examples

All the provided examples were made in the environment mode, either from Application.e for the non-localized examples, or from Application+Locale.e for the localized examples.

For the non-localized examples, there isn't anything special to say. I followed the standard generation method, described in the paragraph about the environment mode. I simply added some lines to open and to close windows.

For the localized examples, I did in addition a search/replace (case sensitive !) for the word "application" to replace it by the name of the application, then I added with a cut/paste some code generated by FlexCat (once again, if you don't know it, try it quickly !). Only Click needed in addition to modify a bit the code generated by FlexCat to change 2 "\033" by "\e" in a E string.
Finally, for each localized example, I provided, in addition to the source file and to the executable file , the files xxxx.cd, xxxx.ct (French version), xxxx.catalog (French version) and xxxx_cat.e (the code generated by FlexCat for this catalog).

1.8 Bugs

None known.

1.9 History

```
Version 1.0 :  - 1st distributed version with MUIBuilder v1.0
Version 1.1 :  - a little bug fixed (GenCodeE sometimes forgot to indent the
               begining of the code)
               - version never distributed
Version 1.2 :  - localization support added
               - version never distributed
Version 1.3 :  - environment mode added
               - 2nd version distributed with MUIBuilder v1.1
```

1.10 Future

The ideal for such a code generation module is to be able to rely on separated compilation. Like this, the GUI code updates are only local to a file, which is only accessed by the module. This is easier for everbody.

Unfortunately, Amiga E doesn't yet support this feature. So GenCodeE may progress a lot (but for the better ;-)) in the same way to MUIBuilder of course, but also in the same way to the future implementation (by Amiga E itself, or by an extra tool) of separated compilation.

1.11 Distribution

GenCodeE and all the other files in the E language directory can't be distributed separated of the MUIBuilder archive without my authorization. Also, all these files are copyright by the author, and none of them can be modified without my permission.

I cannot be held responsible for the use of this program and any damages that it may cause: use it at your own risk!

1.12 The author

You can reach me by mail at:

- my student address, up to and including July 1994

  Lionel Vintenat
  appartement 21
  11 rue François Oulié
  31500 TOULOUSE
  FRANCE

- my family address:

  Lionel Vintenat
  3 impasse Boileau
  Lotissement Les Termes
  87270 COUZEIX
  FRANCE

Write to me at my student address until July 1994 as I'm there much more often than at my family address.

You can also reach me on the INTERNET. My e-mail address is vintenat@irit.fr. I would prefer that you contact me by e-mail rather than mail. I will reply to all questions that are sent to me by e-mail, but don't expect a reply by mail (I am very lazy when it comes to picking up a pen...).