builder

COLLABORATORS			
	TITLE : builder		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 18, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 builder **1**

1.1 MUIBuilder documentation . . . 1
1.2 COPYRIGHT . . . 1
1.3 INTRODUCTION . . . 2
1.4 MUI . . . 2
1.5 Advantages . . . 3
1.6 Using MUI-Builder . . . 3
1.7 General Principles . . . 4
1.8 Objects . . . 4
1.9 Context saving . . . 5
1.10 poids . . . 6
1.11 application . . . 6
1.12 fenetre . . . 6
1.13 temporary list . . . 7
1.14 Group . . . 7
1.15 bouton . . . 8
1.16 liste . . . 8
1.17 dirlist . . . 9
1.18 String gadget . . . 9
1.19 label . . . 10
1.20 cycle . . . 10
1.21 radio . . . 10
1.22 image . . . 10
1.23 Space . . . 11
1.24 checkmark . . . 11
1.25 slider . . . 11
1.26 Gauge . . . 11
1.27 scale . . . 12
1.28 text . . . 12
1.29 Proportionnal Gadget . . . 12

1.30 Special Characters 13
1.31 code 13
1.32 options 14
1.33 Application Code 15
1.34 Object Code 15
1.35 Remove a Label 15
1.36 Add Label 15
1.37 C Language 15
1.38 E Language 16
1.39 guide 17
1.40 Registering 17
1.41 Future 18
1.42 Greetings 18

Chapter 1

builder

1.1 MUIBuilder documentation

```
                                                MUI-BUILDER
            V1.0

          Written by Eric Totel
             in 1993
```


Introduction

Advantages of MUIBuilder

Use of MUIBuilder

Code Generation

AmigaGuide Generation

Future improvements

Copyright

Registering

Greetings

1.2 COPYRIGHT

You can contact me at :

E-Mail : totel@laas.fr

or

Eric Totel
5 rue Riquet
31000 Toulouse

1.3 INTRODUCTION

Thanks for trying MUI-Builder !!!

You'll learn to use a tool I hope user-friendly,
but nevertheless far from perfection.
Please, feel free to send me all your ideas and appreciations about
this software ... so it will evolve in the way your want.

With MUI-Builder, you'll be able to write @{ " MUI " link BUILD-2 }
applications without neither writing lines of source nor knowing
anything about @{ " MUI " link BUILD-2 } functions' syntax (even if
this one is eventually quite simple).

MUI-Builder's aim is to let you create your graphic interface
without technical problem, and with no more effort than thinking
about your final goal.

At the beginnning, I only wrote this program for my own needs
and for learning the use of MUI, this wonderful tool written by
Stefan Stuntz whom I want to thank here.

I hope you'll find this program as usefull as I've already found it.

1.4 MUI

This application uses

MUI - MagicUserInterface

MUI is a system to generate and maintain graphical user interfaces. With the aid of a preferences program, the user of an application has the ability to customize the outfit according to his personal taste.

MUI is distributed as shareware. To obtain a complete package containing lots of examples and more information about registration please look for a file called "muiXXusr.lha" (XX means the latest version number) on your local bulletin boards or on public domain disks.

If you want to register directly, feel free to send

DM 20.- or US$ 15.-

to

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

1.5 Advantages

Many people will wonder about the usefullness of this program, assuming that MUI programing is very easy !

Here come all the advantages that i and all the beta-testers have found in using MUI-Builder :

1. MUI-Builder is simple way to quickly learn the principles of MUI programing, by reading the generated source code.

2. MUI-Builder is a sort of MUI interpreter. You are able to test the GUI look of your application and to see how the result will really look like.

3. MUI-Builder offers to the user a great flexibility in source code generating. You will be able to select precisely which object code you want to generate, with or without the IDCMP loop, or the declarations and initializations.

4. You can finally build the inline help of your application, directly from the application builder.

1.6 Using MUI-Builder

The principles on which MUI-Builder is used can be classified in ↩ the
following headings :

General Principles

Objects

Code Generation

Context Saving
To get some help any time in the program, simply hit 'HELP'.

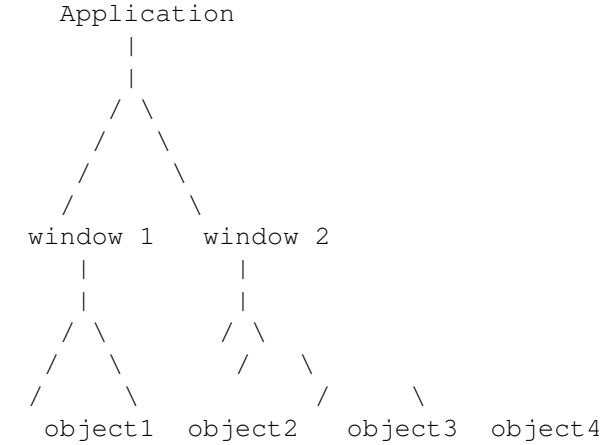
1.7 General Principles

MUI-Builder works on the same principles than MUI, and the 2
programs are strictly linked.

So, my first advice is to read MUI documentation to
better understand the way of working.

MUI-Builder allows you to create the complete interface of an application,
i.e. all the windows of this application.

In fact, each application is a dependency graph of the interface objects
which form it. By this way, an application with 2 windows, each one
compound of 2 objects, can be seen as the following tree :



MUI-Builder is aimed at letting you create this tree only
using a simple, user-friendly, graphic interface (at least, I hope so).

1.8 Objects

MUI-Builder's objects are the followings :

Application

Window

Group

Button

List

Dirlist

String

Label

Cycle

Radio

Image

Space

CheckMark

Slider

Gauge

Scale

Text

Prop
Note that an object which has already been created can be modified by double-clicking on its name in a list where it appears.

Some objects have an
'Weight'
attribute that
can be modified.

1.9 Context saving

The SAVE and LOAD buttons will respectively save from and load to a file the interface of the application you're creating.

This way, you'll be able either to go on with a work you've broken off, or to modify the interface of an old application.

Note that you are able to just drop an icon of MUIBuilder SaveFile in the main window to load a file.

1.10 poids

The 'Weight' attribute is very interesting :
it will allow you to precisely define the relatives sizes of the differents objects.

Every object has a default weight of 100.

If you wish a group had a double size than another one,
you will give it a double weight.
This way, you can avoid button as wide as a half of the screen.

A null weight make the object keep its minimal size whatever happens.

1.11 application

Your application is the root node of the dependency tree representing your complete graphic interface

By clicking the 'Appli' button in the main window of MUI-Builder,
you can set :

- The 'Base' of the application i.e. the name of the AREXX server of your application
- the author's name
- the version
- the copyright text
- a description of your application

It can have only one sort of child objects : the windows
.

1.12 fenetre

You all already know what a window is ! But the problem is learning making one with MUI-Builder ..

The MUI-Builder 'Window attributes' window has three distinct parts :

1. The window's attributes (its label in the code and its name for the title bar).
2. A text bar where you can read the nature of the link between the selected objects.

3. Finally, the set of the sons, grandson, grand-grandson ...
of the window. They are classified in 2 groups :

3.1 Left list :
Groups which related to the window (including the main
group or root group, which is the unique direct son of
the window).

3.2 Middle list :
The sons of the selected
group
(in the left list).

3.3 Right list:
a temporary objects list (see
tmp list
).

Groups
are the basics elements where all the other
objects
will be placed. Every time you'll create an object, you'll have to declare it as son of a group by selecting this group.

1.13 temporary list

In this list you can put objects you want to move from a group to another one, or from a window to another one.

1.14 Group

When you create a new window, it already has a group as son : This is the root group.

Then, you'll make you window by attaching
objects
to this root group.

Some of these sons can be groups, and so on.

In the end, you'll have to define the group's attributes :

- Horizontal : The objects of the group will be placed horizontally.

- PageMode : The group will show only one of this sons at the same time.

- SameHeight : The all sons of the group will have the same height.

```
  - SameWidth  :  The all sons of the group will have the
      same width.

  - SameSize   :  The all sons of the group will have the
      same size.

  - Virtual    :  The group will be virtual.

  - Title      :  The group will have a title.

  - Columns    :  Format the group in columns
      Enter the number of columns in the
      string gadget

  - Rows       :  Format the group in rows
      Enter the number of rows in the
      string gadget

  - Spacing    :  Horizontal : enables to control horizontal
      spacing between the objects

      Vertical   : enables to control vertical
      spacing between the objects

      You must specify the value of the spacing
      in the associated string gadget.
```

1.15 bouton

```
To completely define a button, you must specify :

  - its label

  - The text that will appear in it

  - its corresponding shortkey

you can specify its @{ " Weight " LINK Poids } too.
```

1.16 liste

```
To completely define a list, you must specify :

  - its label

  - its @{ " Weight " LINK Poids }

  - the type of the list :

    - standard
```

```
    - a floattext list, that allows to show texts

    - a list of volumes ( volumes + assigns )

Note that there is an other type of list called @{ " DirList " LINK DirList } .
This type of list can be created by clicking on its button in
the 'Object Choice' window.
```

1.17 dirlist

```
The Directory list shows the files and directories in the
selected directory.

The possible options are :

- the @{ " Weight " LINK Poids }

- 'Drawers Only'  : shows ONLY directories

- 'Files Only'    : shows ONLY files

- 'MultiSelection'  : enables a multiselection of the files
        in the list

- 'Reject Icons'  : don't show the '.info' files

- 'Sort High-Low' : reverse sorting order

- 'Sort Type'   : choice of the sort key ( Name, Date, Size )

The string gadget called 'dir' must contain the name of the
directory you want to list.
```

1.18 String gadget

```
To completely define a string gadget, you must specify :

  - its title ( that you can remove )

  - its label

  - the initial content of the string gadget

  - a string containing all the letters accepted by the gadget

  - a string containing all the letters refused by the gadget

  - the maximal lenght of the string

It's possible to define the @{ " Weight " LINK Poids } too.
```

1.19 label

To completely define a label gadget, you must specify :

- its label
- the text that appears on screen
- the @{ " Weight " LINK Poids }

1.20 cycle

To completely define a cycle, you must specify :

- the list of its entries
- its label
- the @{ " Weight " LINK Poids }

1.21 radio

To completely define a cycle, you must specify :

- the buttons' list
- its label
- the @{ " Weight " LINK Poids }

1.22 image

Before any other thing, you must choose an image by clicking
on it.

Possible choices are :

```
- 'Free Vertical'  : the image will resize vertically.
- 'Free Horizontal': the image will resize horizontally.
- 'Input Mode'     : the user will be able to select the image.
- 'Fix Height'     : set the image height to a constant specified
          in the associated string gadget.
          This makes 'Free horizontal' useless.
- 'Fix Width'    : set the image width to a constant specified
          in the associated string gadget.
          This makes 'Free vertical' useless.
```

As with many objects, you must specify the label.

1.23 Space

This object allows you to insert a space between two other objects.
So the window will be resizable.

1.24 checkmark

You can specify if a title must be written or not before the CheckMark
(as with strings or sliders).

If you want a title, you can specify it in the 'title' string gadget

Don't forget to specify a label.

1.25 slider

You can specify :

- if the current level must be displayed ('Slider Quiet')
- if the slider must have a title or not

Then you must specify :

- the maximum value
- the minimum value
- the initial value

You can also set :

- the title
- the label

1.26 Gauge

To define a gauge, you must set :

- its orientation (horizontal or vertical)

- if you want to set the height of the gauge to a constant value
  that must be specified in the associated String Gadget.
  This is especially used with horizontal gauges.

- if you want to set the width of the gauge to a constant value
  that must be specified in the associated String Gadget.
  This is especially used with vertical gauges.

- if its value must be 'divided'

- its maximum value

- its label

1.27 scale

A scale gadget must be used with a Gauge gadget to display a graduation beside it.

You only have to specify its orientation.

1.28 text

To define a Text Gadget, you must specify :

- 'Text_SetMax' : the maximum size of the gadget will be its initial size.
- 'Text_SetMin' : the minimum size of the gadget will be its initial size.
- 'Backgrounds' : test and you'll see !
- 'Frames' : test and you'll see !

The text must be written in the associated string gadget and can contain every special character what you can find in a C source code.

Don't forget the label.

1.29 Proportionnal Gadget

Set the following options :

- 'Horizontal Prop' : the gadget must be horizontal
- 'Fix Width' : set the width to a constant specified in the associated string gadget.
- 'Fix Height' : set the height to a constant specified in the associated string gadget.

Then you should specify :

- the number of entries
- the number of the first entry
- the number of visible entries

Don't forget the label.

1.30 Special Characters

```
In EVERY text of your interface, you can insert the following
special characters :

- \n    newline
- \r    carriage return
- \t    tabulation
- \e    escape
- \   the backslash caracter \
- \"    a double quote
- \xNN    the character of ascii code NN  ( in hexadecimal )
- \nnn    the character of ascii code nnn ( in octal )
- \c    c if c is any other caractere

Here are some examples :

\033b   to print a bold text
\033n   to come back to normal text
\0338   to display a white text
\033c   to center the text
\033l   to justify left
\033r   to justify right
...

Note that you have to type '\"' instead of '"' when you want a
double-quote. Otherwise, you'll get errors when compiling.
```

1.31 code

```
                When you click on the 'Code' button of the MUI-Builder main  ↩
                   window,
you run the generation of the source code.

MUI-Builder will then verify you don't have used twice the same
label for two different objects.

Otherwise you'll see two lists :

  - one containing the names of all objects in you application
  - one containing the names of all labels that will be
    generated in your source code.

You will be able to control very precisely the way MUI-Builder
will generate the labels in your code. If you don't care with
this option : MUI-Builder will do everything automatically for you !

Before generating the source code, you'll have to define the
               options
              .

Then you will you the following buttons :
```

```
                App Code

                Object Code

                Remove Label

                Add Label
```
 This software actually create a generic code which is more a description of the program than a real compilable source.

After this generic code was created (temporary file in 'T:'), MUIBuilder executes one of the code-generation modules (located in the 'modules' directory) which uses the temporary file.

Actually only C and E languages are available.
In the future Assembly language will be supported.

If you feel good enough (!!!) to program a module for you favorite language : feel free to contact me !!! I'll explain to you how to use the generic code to create the source of the alien language !!!

Thanks to the configuration pannel ('Config' button), you are able to choose the language you want to use. Actually, available choices are :

```
  @{ " C Language " link Langage_C }
  @{ " E Language " link Langage_E }
```

1.32 options

Three options are available :

- 'Declarations' : to obtain the declarations and initializations in your source

- 'Environment' : select it if you want to generate code for includes, event loop, procedure declaration ...

- 'Code' : select it if you want to generate the MUI code.

Note that each option may be selected independently from the others. You can also create exactly the part of the code you desire ...

Example :

You've created a window and you want to add a button in your code : select only the 'Code' option and create the button code ! After inserting this text in the program, you'll need the declaration of the object button in your source : select 'Declaration' ... and insert the generated text directly where you want in your source !

1.33 Application Code

This button enables to generate the source code for the whole application.

1.34 Object Code

This button enables to generate the source code of an object you previously chose in the 'Objects labels' list.

1.35 Remove a Label

By selecting this button, you deactivate the code generation for the object selected in the 'Generated Labels' list.

MUI-Builder automatically knows if it has to generate (or not) the label of each object. For example it is often useless to keep a pointer variable to a MUI-Group, unless you want to dynamically add objects to this group during the execution of the program.

This button (and the
Add Label
button) allows
to change MUI-Builder standard definitions on objects.

1.36 Add Label

By clicking on this button, you tell MUI-Builder to generate the label of the selected object.

See
Remove Label
.

1.37 C Language

You should notice the following features :

- Some objects use auxiliary variables. The name of these variables is 'STR_"variable_name"'.

- NO notifycation is made from MUI-Builder. So, if you want to quickly test the program, at least open a window before the event loop (set(WI_window, MUIA_Window_Open, TRUE)). If you only inclide this line, you must exit the program using the 'Exchange' commodity.

The generated code uses a header file called 'code.h'.
You should find it in the MUI-Builder archive.
This file is provided by Stefan Stuntz in the MUI-Package, and
will allow the compilation with most of the C-compilers.

1.38 E Language

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 node for all that is related to generated variables and notification. Differences are syntaxical. These are the points of the code generation specific to E code :

- the generated file always begin with 'OPT OSVERSION=37' to ensure that kickstart 2.0 or more is present
- some E included files are called with the MODULE instruction : these files are especially necessary for the doMethod function (see below)
- the MUI_TRUE constant is defined to the value 1 : this one must be used instead of TRUE (which equals -1) for MUI, because MUI doesn't always recognize value 1 like true
- the call to init() specific to C language is replaced by a simple opening of the muimaster library : if this opening fails, the program ends returning 100
- a doMethod function is added after the main procedure : this one correspond to the DoMethod function of the amiga.lib library which is unsuable with Amiga E, and which using syntax is (conformable to the one of the amiga.lib) doMethod(object_pointer, [param1, param2, ...]).

Notice that the generated doMethod function was given to me by Wouter van Oortmerssen (thanks Wouter !) the Amiga E author.

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 that was said before, the character " can be entered without \ (it was a problem only in C language). It's the E code generating module which will translate. More precisely, this module recognize :

- \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 produced 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 replacing 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 replacing and designed for Amiga E. So refer to this archive for more details.

Let's finish those explainations by the traditional copyright :
- the E generation code module (GenCodeE) can't separated of the MUI-Builder archive
- GenCodeE stay under copyright of the author and so can't be modified without my agreement
- GenCodeE is provided without any warranty of any kind : you use it at your own risk !

Lionel Vintenat

1.39 guide

You will endly be able to make your inline help, directly from your favorite interface builder (MUI-Builder of course !!!).

You are able to attach a help text to each MUI-object created by MUI-Builder.

Then MUI-Builder will automatically create a hypertext documentation in AmigaGuide Format (that you will be able to view by clicking the 'View' button).

As with code generation, you are able to only create one part of the documentation to insert it directly in your previously written documentation.

You can edit the text of the documentation :

- the main text with the 'App Node' button
- a window text with the 'Window Node' button
- an object text with the 'Object Node' button

You are able to create the documentation :

- for the application with the 'Generate whole Doc' Button
- for a window with 'Generate win Doc'
- for an object with 'Generate obj Doc'

By double-clicking on an object or a window name, you can edit the title of the associated help-text, and view it.
(same as the 'Edit' buttons).

1.40 Registering

Nothing's easier !
This program is GiftWare !!!

This means you are absolutely NOT forced to send me some money
to go on using this program.

But, if you really like this soft, if you find it really usefull,
then you are allowed (eh eh eh !!) to send me about 15 $US or
50FF to the following address :

  Eric Totel
  5 rue Riquet
  31000 Toulouse
  France

If you didn't enjoy enough MUI-Builder to send me some money,
please send me a postcard, a letter, an email (totel@laas.fr)
to encourage me, give me some ideas or remarks ... and so on ...
I need traductions for the documentation, and superb icons for
program ... if you made one of these things, please send them !!!

Don't hesitate at all : I absolutely want to know if further
developments of MUI-Builder will be really usefull and appreciated.

1.41 Future

Future improvements would probably be :

- use of the locale library in MUI-Builder itself and in its
  generated source code.

- the possibility to exchange data with text editors using AREXX
  port to allow you to directly insert MUI-Builder generated source
  code in your program.

- notification.

- all the interesting ideas you could suggest me.

1.42 Greetings

I'd like to thank all the people who helped me and who
contributed to MUI_Builder :

- Lionel Vintenat for the idea of the generic code and
  the implementation of the 'E language' module.

- Gael Marziou , Pierre Carrette, and Pascal Pensa for their
  carefull tests and suggestions (creating the inline help
  was only one of them).

- Pascal Rabier for helping me to translate the documentation
  from French to English.

- All the beta-testers for their bugs reports.