builder

COLLABORATORS

	TITLE : builder		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 26, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

builder

1.1 MUIBuilder documentation

```
         MUI-BUILDER
             V1.1

           Written by Eric Totel
              in 1994
```

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 !!!

MUI-Builder is a tool that I hope you find to be user-friendly.
It is, however, far from perfection.

Please feel free to send me all your ideas and opinions about
this software so it may evolve in the way you'd like.

With MUI-Builder, you'll be able to write @{ " MUI " link BUILD-2 }
applications without writing lines of source code nor knowing
anything about @{ " MUI " link BUILD-2 } functions' syntax (even
though MUI is simple in itself).

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

At the beginnning, I wrote this program only for my own needs
and for learning how to use MUI. MUI is a wonderful tool
written by Stefan Stuntz

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

1.4 MUI

This application uses

MUI - MagicUserInterface

(c) Copyright 1993 by Stefan Stuntz

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 usefulness of this program, as MUI is simple to program in the first place.

The advantages that I and all the beta-testers have found in using MUI-Builder follow:

1. MUI-Builder is a 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 look of your application's GUI and see how the result will really look.

3. MUI-Builder offers the user a great flexibility in source code generation. 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.

5. It is the very first builder that enables an easy localization !

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 as MUI, and the two programs are intricately intertwined.

My first advice is to read MUI documentation to better understand how it works.

MUI-Builder allows you to create the complete interface of an application. For example, all the windows of this application were create with MUI-Builder.

Each application is made up of a hierarchy of interface objects. As an example, the following diagram depicts an application with two windows, each containing two objects:

```
      Application
          |
          |
         / \
        /   \
       /     \
      /       \
    window 1   window 2
       |         |
       |         |
      / \       / \
     /   \     /   \
    /     \        /     \
     object1  object2   object3  object4
```

MUI-Builder is aimed at letting you create this tree simply by 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 Configuration

```
All choices will be saved in an environment variable (ENV:MUIBuider.env)
which is creating when you are installing MB.

There are the following choices :

- Ghost windows : to see ONLY one window of MUIBuilder at a time.
  ( very useful for slow computers, and for people who dislike
  to have too many windows on the screen ! )

- Icons : select it if you want an icon with the save file

- Requests : select it if you want all the warning requesters

- Code : chose your generator module

- Editor : You must specify here the name of your favorite
  editor. This editor should be run in a synchronous way !
  ( i.e. not as a background task ).

  the following editors should work without any
  problems :

  - Vim (v2.0) : with the '-x' option
  - CygnusEd   : with the '-keepio' option
  - GoldEd     : with the 'STICKY' option
  - TurboText  : with the 'WAIT' option
  - Ed       : default editor

- GetString : If you enter a string here, it will be the default
  name of the 'GetString' function which is used to localize
  your program. If this String is EMPTY, then MUIBuilder will
  create a name using the application name ( 'GetApplicationNameString' ).
```

1.10 Context saving

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

Using these buttons, you can continue with a previously unfinished
application, or modify an already existing application.

Not that the MUIBuilder main window is an AppWindow.  You can
therefore load an MUIBuilder save file by dropping its
respective icon on the window.
```

1.11 Weight

The 'Weight' attribute allows you to precisely define the relative sizes of the differents objects with respect to each other.

Every object has a default weight of 100, which gives each object an equal share of the window space.

Modifying this value will change the size of one group or object with respect to another group or object. For example, if you have two objects and wish to make the first object three times as large as the second, you would assign the first object a weight of 75, and the second a weight of 25.

Using this method, you can avoid having gadgets larger than they may become.

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

1.12 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 object: the window .

1.13 Window

You all already know what a window is! But the problem is learning how to make 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, a list of the children, grand children,
      and so on of the window. They are classified into two
      groups:

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

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

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

   Groups  are the basics elements to which all other  objects
  will be placed. Every time you create an object you will have
  to declare it as a child of a specific group.  You may specify
  this group by selecting it in the list gadget.
```

1.14 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.
```

1.15 Group

```
  When you create a new window, it already has a group default child.
  This child object is the root group.

  You define a window by attaching children to this root group.
  Note that other groups may also be children to the root group.

  You must define the attributes for each group:

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

    - Register   :  The group will show only one of its
        children at a time.
        After selecting this attribute, you must
        go to the ’Register’ entries page to
        chose the names of the pages of the register.

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

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

    - SameSize   :  The all children 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 you to control horizontal
        spacing between the objects

        Vertical   : enables you to control vertical
        spacing between the objects

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

1.16 Button

```
  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.
```

1.17 List

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

    - its label

    - its @{ " Weight " LINK Poids }

    - if 'double click' is enable or not

    - if the multiselection is enable

    - 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 a @{ " DirList " LINK DirList }.
  This type of list can be created by clicking on its button in
  the 'Object Choice' window.
```

1.18 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.
```

1.19 String gadget

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

  - its title ( that you can replace with your own )

  - 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 length of the string

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

1.20 label

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

  - its label

  - the text that appears on screen

  - the @{ " Weight " LINK Poids }
```

1.21 cycle

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

  - the list of its entries

  - its label

  - the @{ " Weight " LINK Poids }
```

1.22 radio

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

  - the buttons' list

  - its label

  - the @{ " Weight " LINK Poids }
```

1.23 image

```
To define an image for this gadget, you must select the image
that you wish to be displayed.

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 other objects, you must specify the label.
```

1.24 Space

```
This object allows you to insert a space between two other objects
so the window can be resized.
```

1.25 checkmark

```
You may specify whether the title will be placed to the left or
the right of the checkmark.

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

Don’t forget to specify a label.
```

1.26 slider

```
You can specify:

- if the current level must be displayed ( ’Slider Quiet’ )
- if the slider must be displayed in the reverse mode
- 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
```

1.27 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
    it 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
    it 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
```

1.28 scale

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

You only have to specify its orientation.
```

1.29 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.
```

1.30 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.
```

1.31 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.
```

1.32 code

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

MUI-Builder will verify that you haven't used the same
label for two different objects twice.

If everything goes well, 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 about
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 see the following buttons :

   App Code
   Object Code
   Remove Label
   Add Label
```

```
The software actually creates generic code, which is more a
description of the program than a real compilable source.

After this generic code is 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.  The available
choices are :

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

1.33 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.

  - 'Locale'   : to generate localize code.
         ( see  Locale  )

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 it will 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 !
```

1.34 Application Code

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

1.35 Object Code

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

1.36 Remove a Label

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

MUI-Builder automatically knows if it has to generate
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.

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

1.37 Add Label

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

See Remove Label .

1.38 Locale

When you select the ’Locale’ Option in the code-generation window,
you chose to generate localized source code.

In fact, all strings and shortcut will be replaced by a call
to a ’GetString’ function (which should return a locale string
from your catalog file).

A lot of program are able to generate this function for you
(Catcomp and Flexcat for examples).

The name of the ’GetString’ function can be choosed either in
the preferences window, or in the code-generation window.
(see @{ "Configuration"link Configuration })

If you don’t know how to use these features, please take a look
at the examples included in the MUIBuilder archive.

```
The catalog description file can be generated by clicking on
the @{ "Catalog" link Catalogue } button.
```

1.39 Catalog

```
The name of the '.cd' file must be given before generating
the Catalog.

Here is a summarize of what is necessary to know about
the catalog description file ( 'xxxx.cd' file )

You'll find in this file all the strings of your program
in the default language.

Thanks to that file, you will be able to create automatically
( with Catcomp or Flexcat ) :

  - the translation file ( 'xxxxx.ct' file )
    in which you'll find the strings in a foreign language.

  - the program source file corresponding to your favorite
    language.
( please, take a look at these programs for more details )

The catalog description file uses the following agreement
for the key shorcuts :

  _e Exemple

means : Exemple
```

1.40 C Language

```
You should notice the following features :

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

- When you generate localized source code, the following line
  will appear at the beginning of the file ( if your catalog
  file is called 'MUIBuilderStrings.cd' :
  '#include MUIBuilderStrings.h'

There are now TWO modules for C code generation :

  - @{ "GenCodeC_Old" link GenCodeC_Old } : similar to the
    generator of MUIBuiderV1.0

  - @{ "GenCodeC" link GenCodeC } : a new type of generator
    which will enable you to generate separate GUI source file !

- NO notification 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 include this line, you must exit the program
    using the 'Exchange' commodity.
```

1.41 gencodec_old

```
  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.
```

1.42 gencodec

```
  This generator is the most interessant, i think !!!

  Thanks to this generation module, you will be able to create
  a separated GUI source file .

  It copies at the beginning of the file the 'C-Header' file which
  could be found in the 'Modules' directory. So you should be able
  to customize this file to your own needs !

  The generator will create TWO files :

  - a '.c' file including two functions :
    - CreateObject  :  returns a pointer to a structure
    - DisposeObject :  frees memory

  - a '.h' file : you must include it in your program. It contains
    the structure and functions declarations for your MUInterface !

  You will use these file in the followin way :

  #include <libraries/mui.h>

  #include <clib/muimaster_protos.h>
  #include <clib/alib_protos.h>
  #include <clib/dos_protos.h>
  #include <clib/exec_protos.h>

  #include <pragmas/muimaster_pragmas.h>
  #include <pragmas/exec_pragmas.h>

  #include "essai.h"      /* ---- MUIBuilder ---- */

  struct Library * MUIMasterBase;

  main()
  {
    struct ObjApp * app;
    BOOL  running = TRUE;
    ULONG signal;
    extern struct Objapp * CreateSmall_Example( void );

    Init();       /* Initialisations  */
            /* Library Openings   */

    app = CreateApp();    /* ---- MUIBuilder ---- */

            /* Put here your notifications  */

    set( app->WI_main, MUIA_Window_Open, TRUE );   /* Window opening */

    while (running)     /* IDCMP loop */
        {
                switch (DoMethod(app->app,MUIM_Application_Input,&signal))
                {
                case MUIV_Application_ReturnID_Quit:
                        running = FALSE;
                        break;
                }
      if (signal) Wait(signal);
        }
    DisposeApp(app);    /* ---- MUIBuilder ---- */
  }
```

1.43 E Language

```
Please look at the separated documentation.

Thanks.
```

1.44 guide

```
You will 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 create only one part of
the documentation for insertion directly into  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 ).
```

1.45 Registering

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

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

  But, if you really like this software or find it really usefull,
  then you are may send about 15 $US or 50FF to the following address:

    Eric Totel
    5 rue Riquet
    31000 Toulouse
    France

  If you didn't enjoy MUI-Builder enough 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 translations for the documentation, and superb icons for the
  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.
```

1.46 Future

```
  Future improvements would probably be :

  - the notification will be implemented directly using MUIBuilder

  - 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.

  - all the interesting ideas you could suggest me. ( I WANT IDEAS !!! )
```

1.47 Greetings

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

- Pierre Carrette for his fantastic help !

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

- Gael Marziou , Pascal Pensa, Christian Brandel
  Daniel Murrel  for their carefull tests and suggestions

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

- Christian Brandel for the german documentation, and the german
  catalog.

- Tobias Ferber for his superb MagicWB icon !!! ( no,no ... not superb :  ↩
   FANTASTIC !!! :-) )

- Michal Kara for his czech translation of the catalog, and the
  implementation of the ’Assembly language’ module.
```