builder

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

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

builder

1.1 builder.guide

```
                                                MUI-BUILDER
            V1.0

          Ecris par Eric Totel
              en 1993
```


1.2 COPYRIGHT

Vous pouvez me contacter à :

E-Mail : totel@laas.fr

or Eric Totel
5 rue Riquet
31000 Toulouse

MUI-Builder est Giftware.

Ce programme peut être librement distribué, tant que personne ne tire aucun bénéfice de cette distribution. Tout autre type de vente ne peut en aucun cas être effectuée sans l'autorisation de l'auteur.
MUI-Builder peut être inclus dans des collections de logiciels du domaine public, tant que les conditions ci-dessus restent vérifiées.

L'auteur dénie ici toute responsabilité quant à d'éventuelles dégradations de données occasionnées par l'utilisation de MUI-Builder. Le logiciel est délivré 'tel quel', sans aucune garantie d'aucune sorte.

1.3 INTRODUCTION

Merci d'avoir récupéré MUI-Builder !

Vous avez maintenant entre vos mains un outil qui, je l'espère devrait vous être très utile.

MUI-Builder est un outil qui va vous permettre d'écrire des Applications @{ " MUI " link MUI }, sans avoir à taper des lignes et des lignes de code, et sans avoir à connaitre la syntaxe (somme toute relativement simple) de @{ " MUI " link MUI } .

Grâce à MUI-Builder, vous allez enfin pouvoir créér vos interfaces graphiques sans la moindre arrière pensée, et sans plus d'effort que de réfléchir à ce que vous voulez réaliser.

Le but premier de ce programme était uniquement de satisfaire à mes besoins (en créant un outil qui devrait être intégré dans @{ " MUI " link MUI } ↩
)
et d'apprendre à utiliser cet outil fantastique que nous a réalisé Stefan Stuntz (que je remercie d'ailleurs ici).

J'espère que ce soft vous sera aussi utile qu'il me l'a déja été jusqu'à présent.

1.4 MUI

Cette application utilise

MUI - MagicUserInterface

MUI est un système pour générer et maintenir des interfaces graphiques. Avec l'aide d'un programme de préférences, l'utilisateur d'une application a la possibilité d'en modifier l'apparence exterieure pour qu'elle s'adapte à ses goûts.

MUI est distribué en tant que shareware. Pour obtenir un package contenant de nombreux exemples et plus d'informations en ce qui concerne l'enregistrement procurez-vous un fichier appelé "muiXXusr.lha" (où XX représente le numéro de version le plus récent) sur votre BBS local ou sur des disks du domaine public.

Si vous voulez vous enregistrer directement, vous pouvez envoyer

DM 20.- ou US$ 15.-

à

Stefan Stuntz
Eduard-Spranger-Straße 7
80935 München
GERMANY

1.5 avantages

Beaucoup se demanderont quel est l'intérêt d'un tel programme compte tenu de la simplicité d'utilisation de MUI.

A tous ces gens j'expose ici les avantages que moi et tous ceux qui ont eu l'amabilite de tester MUI-Builder ont pu remarquer :

1. MUI-Builder est, pour les débutants, LE moyen de maîtriser RAPIDEMENT MUI, en regardant le code généré par le Builder. Son utilisation en fait presque un didacticiel destiné à vous guider dans la découverte de MUI.

2. MUI-Builder se comporte, pour ceux qui maîtrisent déjà MUI, comme une sorte d'interpréteur du langage associé à MUI. Vous pouvez directement tester le look de votre interface graphique pour connaître à l'avance le résultat. Tous ceux qui ont utilisés régulièrement MUI se sont aperçus en effet de certaines limitations (en particulier en ce qui concerne le resizing et la taille fixe de certains objets qui affectent parfois le look de l'interface).

3. MUI-Builder offre une grande flexibilité dans la génération du code (chaque objet peut être généré indépendamment des autres, avec ou sans les déclarations et initialisations des variables associées aux objets MUI).

4. Il est enfin possible, depuis un interface builder de construire l'aide en ligne correspondant à l'interface graphique. Ce qui

est tout à fait nouveau (à ma connaissance) sur l'Amiga.

1.6 Utilisation

Les principes d'utilisation de MUI-Builder peuvent être classés de la manière suivante:

Principes généraux

Les Objets

La sauvegarde du contexte courant

Pour obtenir de l'aide à n'importe quel moment dans le programme ↩ , pressez tout simplement la touche 'HELP'.

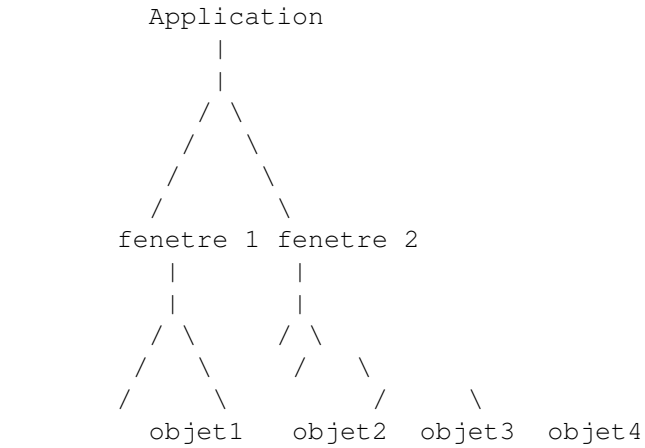
1.7 Principes généraux

MUI-Builder conserve le même type d'idée de conception de l'interface que MUI lui-même (auquel il est d'ailleurs très lié).

Je ne peux donc que vous conseiller de vous referer à la documentation de MUI, pour mieux appréhender le principe.

MUI-Builder permet de créer l'interface complète d'une application, et, par la même occasion, de créer toutes les fenêtres attachées à l'application.

En fait, chaque application est un arbre de dépendance des objets d'interface qui la compose. Ainsi, une application comportant deux fenêtres, qui contiennent chacune un objet quelconque, pourra être schématiquement représentée par l'arbre suivant:



L'idée de MUI-Builder est de vous permettre de créer cet arbre

en utilisant une interface graphique (que j'espère suffisamment
simple et conviviale).

1.8 Les Objets

MUI-Builder permet de créer tous les objets que vous voudrez ↩
bien
attacher à votre application. Ces objets peuvent être classés
comme suit :

Application

Fenetre

Groupe

Bouton

Liste

Dirlist

Chaine

Label

Cycle

Radio

Image

Espace

CheckMark

Slider

Jauge

Scale

Texte

Prop

Il est à noter qu'un objet, outre le fait d'être créé, peut
également être modifié. Pour cela, il suffit de double-cliquer
sur son nom dans une liste où il apparaît.

Pour certains objets, vous pourrez modifier l'attribut 'Weight'
(
poids
en Francais)

1.9 La sauvegarde du contexte

Les boutons SAVE et LOAD vous permettront respectivement de sauvegarder et récupérer l'interface de l'application en cours de création.

De cette manière vous serez en mesure de poursuivre un travail interrompu, ou bien de restaurer une ancienne application dont vous voulez modifier l'interface.

Il est à remarquer que la fenêtre principale de MUIBuilder est une appwindow : vous pouvez donc amener une icone sur la liste de la fenêtre principale pour lire un fichier !

1.10 poids

Le poids est l'un des attributs les plus intéressants des objets : il va vous permettre de définir de manière fine les tailles relatives des objets les uns par rapport aux autres.

Tous les objets ont un poids par défaut de 100 .

Si vous désirez par exemple qu'un groupe ait une taille double de son groupe voisin, vous lui affecterez un poids double. De cette manière, on peut éviter de se retrouver avec des boutons de taille phénoménale !!!

De plus un poids de zéro indique que l'objet doit garder (dans tous les cas de figure) sa taille minimale.

1.11 application

L'application est en fait le noeud racine de l'arbre de dépendance constitutant l'interface graphique complète.

En appuyant sur le bouton 'Appli' dans la fenêtre principale de MUI-Builder, vous pouvez déterminer :

- la 'base' de votre application, c'est-à-dire le nom du serveur AREXX de votre application.
- le nom de l'auteur
- le titre de votre application
- la version

- un texte relatif au copyright

- une description de votre application

L'application ne peut avoir qu'un seul type de fils, à savoir les

Fenetres
.

1.12 fenetre

Vous savez tous déjà ce qu'est une fenêtre ! Notre probleme va être
d'apprendre à la construire grâce à MUI-Builder ...

La fenêtre de création d'une fenêtre se divise en trois parties
distinctes :

1. Les attributs de la fenêtre (le label qui apparaitra
   dans le code généré, et le titre qui apparaîtra dans
   la barre de titre de la fenêtre lorsque vous
   l'afficherez).

2. Une barre de texte ou apparaitra les liens de parentés
   entre les objets que vous allez sélectionner.

3. Enfin l'ensemble des fils, petits fils, petits petits ...
   fils de la fenêtre qui sont disposés en trois groupes :

   3.1 Dans la liste de gauche :
   L'ensemble des groupes ayant un lien de parenté avec
   la fenêtre (Y compris le groupe principal, aussi
   appelé groupe_racine qui est le seul fils direct
   de la fenêtre).

   3.2 Dans la liste du milieu:
   L'ensemble des fils du
   groupe
   sélectionné dans la
   fenêtre de gauche.

   3.3 Une liste d'objets 'temporaire' (voir
   tmp list
).

Les
groupes
constituent les éléments de base
dans lesquels vont être intégrés tous les autres
objets
. Ainsi,
à chaque fois que vous ajouterez un objet, vous devrez le déclarer
comme étant le fils du groupe que vous aurez précédemment
sélectionné.

1.13 liste temporaire

Dans cette liste vous pouvez amener des objets, afin de les passer
d'un groupe à l'autre, ou d'une fenêtre à l'autre.

1.14 groupe

Lorsque vous créez une nouvelle fenêtre, un groupe lui est déja
attaché : le groupe principal, encore nommé 'groupe racine'.

Vous allez ensuite construire votre fenêtre en attachant des
des
objets
à ce groupe racine.

Parmi les fils pourront, bien entendu, exister d'autres groupes
qui, eux-mêmes, contiendront d'autres fils ... etc ...

Enfin, vous devrez, pour chaque groupe, définir l'ensemble de ses
attributs :

- Horizontal : pour que le groupe ait une disposition
  horizontale

- PageMode : Le groupe ne pourra 'voir' qu'un seul
  de ses fils à la fois.

- SameHeight : Tous les fils du groupe auront la même
  hauteur

- SameWidth : Tous les fils du groupe auront la même
  largeur

- SameSize : Tous les fils du groupe auront la même
  taille

- Virtual : Le groupe sera un groupe virtuel

- Title : Le groupe sera surmonté d'un titre que
  vous devez donner dans la barre 'title'

- Weight : Permet de définir le @{ "poids" LINK Poids} du groupe

- Columns : affiche le groupe sous forme de
  colonnes (le nombre de colonnes devant
  être entré dans la chaine 'number')
  Cet attribut permet d'aligner des éléments
  dans une fenêtre.

- Rows : affiche le groupe sous forme de
  lignes (le nombre de lignes devant
  être rentré dans la chaine 'number')

- Spacing : Horizontal : permet de definir l'espacement

horizontal entre les objets

Vertical : permet de definir l'espacement
vertical entre les objets

L'espacement est a rentrer dans la chaine
correspondante.

1.15 bouton

Pour définir complètement un bouton, il suffit de donner :

- Le label que portera le bouton dans le programme
- Le texte qui apparaîtra dans le bouton
- Le raccourci clavier qui sera supporté par le bouton

Vous pouvez également donner le @{ "poids" LINK Poids } du bouton,
grâce au slider 'Weight'.

1.16 liste

Pour définir complètement une liste, il suffit de donner :

- Son label dans le programme C généré.
- Son @{ "poids" LINK Poids }.
- Le type de la liste, qui peut être :
  - une liste standard
  - une liste de floattext, permettant d'afficher
    un texte banal.
  - une liste de 'Volumes', qui contient tous les
    assigns et volumes de votre configuration.

Il est à remarquer qu'il existe un autre type de liste, appelé
@{ " DirList " LINK DirList } .
Ce type de liste peut être créé indépendamment en cliquant
sur son bouton dans la fenêtre 'Object Choice'.

1.17 dirlist

La liste de directory affiche à l'écran, les fichiers et
répertoires présents dans le répertoire spécifié.

Les options possibles pour cet objet sont les suivantes :

```
- le poids ( cf @{ " Poids " link Poids } )

- 'Drawers Only'  : à sélectionner si vous ne voulez QUE les
        répertoires
- 'Files Only'    : à sélectionner si vous ne voulez QUE les
        fichiers
- 'MultiSelection'  : autorise une multi-sélection dans la liste
- 'Reject Icons'  : n'affiche pas les '.info'
- 'Sort High-Low' : inversion de l'ordre de tri
- 'Sort Type'   : Choix du critère de tri (Nom, date, taille)

La chaine 'dir' doit contenir le répertoire de lecture pour la
DirList.
```

1.18 chaine

```
Le gadget Chaine est défini par les donnees suivantes :

  - Le titre qui apparaîtra directement à gauche du gadget
    string ... ( et que vous pouvez enlever )

  - le label de la variable associée au gadget chaine

  - le contenu initial du gadget chaîne lors de l'affichage

  - une chaîne de caractères contenant les caractères acceptés
    par votre gadget chaine.

  - une chaîne de caractères contenant les caractères refusés
    par votre gadget string.

  - la longueur maximale de la chaîne qui devra être acceptée

On peut également redéfinir le @{ " poids " LINK Poids } de la chaine.
```

1.19 label

```
Pour définir complètement un label, il faut donner :

  - son label dans le programme

  - le contenu du texte constituant le label

  - on peut également redéfinir son @{ "poids" LINK Poids }.
```

1.20 cycle

Pour définir complètement un cycle, il faut donner :

- la liste des entrées du cycle
- le label du gadget cycle dans le programme
- on peut redéfinir le @{ "poids" LINK Poids } du cycle

1.21 radio

Pour définir complètement des boutons radios, il faut :

- la liste des boutons radios
- le label du gadget radio dans le programme
- on peut également redéfinir le @{ "poids" LINK Poids }

1.22 image

Avant toute chose vous devez cliquer sur une des images pour
sélectionner celle qui vous intéresse.

Les choix à effectuer sont les suivants :

- 'Free Vertical' : l'image se redimensionnera dans le sens vertical
- 'Free Horizontal': l'image se redimensionnera dans le sens horizontal
- 'Input Mode' : Votre image pourra être sélectionnée par
  l'utilisateur.
- 'Fix Height' : Détermine la hauteur de l'image une fois pour
  toutes (indiquée dans le gadget string correspon-
  dant). Rend inutile 'Free Vertical' lorsqu'il
  est activé.
- 'Fix Width' : Détermine la largeur de l'image une fois pour
  toutes (indiquée dans le gadget string correspon-
  dant). Rend inutile 'Free Horizontal'

Comme pour la plupart des objets, il faut donner le label de la
variable qui sera générée dans le programme.

1.23 espace

Cet objet sert uniquement à insérer un espace entre deux
autres objets, afin de permettre le redimensionnement de
la fenêtre.

1.24 checkmark

Vous pouvez, pour les CheckMarks (comme pour les chaines et les slider) indiquer si un titre doit être écrit ou non juste devant le CheckMark.

Si ce titre doit apparaître, vous devez l'indiquer dans la chaine 'Title'.

... et bien sûr il faut une fois de plus donner le label de l'objet.

1.25 slider

Pour le slider, il faut spécifier :

- si le niveau courant doit être affiché ('Slider Quiet')
- si un titre doit apparaître devant le slider

Il faut alors rentrer :

- la valeur maximale qui peut être atteinte
- la valeur minimale
- la valeur initiale

vous pouvez également remplir :

- le titre qui peut éventuellement apparaître devant le slider
- le label de l'objet

1.26 jauge

Pour définir une jauge, il faut donner :

- le sens de la jauge (horizontale ou verticale)
- si l'on veut fixer la hauteur de la jauge (utilisé en particulier pour les jauges horizontales) Cette hauteur doit être spécifiée dans le gadget String correspondant.
- si l'on veut fixer la largeur de la jauge (utilisé en particulier pour les jauges verticales) Cette largeur doit être spécifiée dans le gadget String correspondant.
- si son contenu va être 'divisé'
- le maximum que peut atteindre la jauge
- Le label de la jauge dans le programme généré

1.27 scale

Le gadget Scale doit être utilisé en conjonction avec le gadget Jauge. Son but est de dessiner une graduation sous ce dernier.

Le seul élément à donner à la création de ce gadget est le sens de positionnement (horizontal et vertical).

1.28 text

Les gadgets textes peuvent être définis grâce aux attributs ↩ suivants :

- 'Text_SetMax' : la taille maximale du gadget texte est la taille initiale du gadget.
- 'Text_SetMin' : la taille minimale du gadget texte est la taille initiale du gadget.
- 'Backgrounds' : testez et vous verrez !
- 'Frames' : idem

Le texte doit être rentré dans le gadget chaîne correspondant et peut comporter tous les caractères spéciaux que l'on peut en particulier trouver dans des sources C.

Vous devrez également rentrer le nom de label du gadget.

1.29 Gadget Proportionnel

Les options à choisir sont les suivantes :

- 'Horizontal Prop' : indique si le gadget doit être horizontal
- 'Fix Width' : Largeur fixe (à indiquer dans le gadget string correspondant)
- 'Fix Height' : Hauteur fixe (à indiquer dans le gadget string correspondant)

Ensuite, vous devez indiquer :

- le nombre d'entrées pour le gadget proportionnel
- le numéro de la 1ère entrée
- le nombre d'entrées visibles

Et il faut bien sûr donner au programme le label de l'objet.

1.30 Caractères Spéciaux

dans les textes que vous rentrez (TOUS les textes qui apparaissent
à l'écran et non pas seulement les gadgets texte) ,
vous pouvez insérer des caractères spéciaux :

- \n   nouvelle ligne
- \r   retour chariot
- \t   tabulation
- \e   escape
- \   le caractere anti-slash
- \"   un guillemet
- \xNN   le caractere de code ascii NN (en hexadecimal)
- \nnn   le caractere de code ascii nnn (en octal)
- \c   c si c est n'importe quel autre caractere

Voici quelques exemples d'utilisation :

\033b   pour un texte en gras
\033n   pour revenir à un texte normal
\0338   pour mettre le texte en blanc
\033c   pour centrer le texte
\033l   pour justifier le texte à gauche
\033r   pour justifier le texte à droite

...

Il est à remarquer qu'il faut rentrer '\"' et non '"' lorsque vous
voulez un guillemet. Sinon vous aurez des problèmes dans le
code généré à la compilation.

1.31 code

Lorsque vous appuyez sur le bouton 'Code' de la fenêtre ↩
principale
de MUI-Builder, vous lancez la génération du code de l'interface
graphique.

MUI-Builder va alors vérifier que vous n'avez pas défini plusieurs
fois la même variable, puis va vous prévenir si c'est le cas.

Sinon, vous allez vous retrouver devant la liste, d'une part de
tous les objets que vous avez créés, d'autre part de tous les
noms de variables qui seront générés par le programme.

Ces deux listes vont vous permettre de controler de manière très
précises les variables qui vont être générées par le programme.
Si vous n'en voyez pas l'intérêt : ne vous en faites pas, MUI-Builder
s'occupe de tout pour vous, sans que vous n'ayez rien à y faire.

Vous devez, avant de générer le code, définir les
options
Les boutons disponibles dans cette fenetre sont :

App Code

```
                Object Code

                Remove Label

                Add Label
                 Le programme génère actuellement un fichier-programme 'générique ↩
                    '
qui correspond plus à une description d'un programme qu'à
un source réel, d'ailleurs ...

Lorsque ce source est généré ( fichier temporaire dans T: ),
MUIBuilder appelle l'un des modules de création de source
( actuellement un module de génération de code C, et un de code E )
qui va utiliser le fichier temporaire pour créer le source dans
le langage cible.

Par conséquent, MUIBuilder devrait pouvoir, dans l'avenir,
générer du code source dans tous les langages où MUI est utilisable.

Le module de génération de l'assembleur est en cours de realisation

Par contre : si VOUS désirez un langage bien spécifique, et que
vous vous sentez capable de créér le générateur de code pour
ce langage bien particulier, CONTACTEZ-MOI !!!! : je vous
expliquerai dans le détail ce qui constitue le code générique et
comment vous y prendre pour la création automatique de votre
code.

Grâce au panneau de configuration ( bouton 'Config' ) vous pouvez
déterminer le type de langage dans lequel sera généré le code.
Pour l'instant, sont disponibles :

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

1.32 options

```
Trois options vous sont proposées :

  - 'Déclarations' : si cette option est activée, vous
        obtiendrez la déclarations des variables
        et leur initialisation dans le code généré.

  - 'Environnement': détermine si vous voulez ou non tout ce
        qui concerne les includes, la boucle de
        gestion des évènements, la déclaration
        de la procédure ... etc ...

  - 'Code'   : indique que vous voulez la génération du
        code MUI.

Il est à remarquer que chacune de ces options peut être validée
```

complètement indépendamment des autres ... permettant ainsi de
générer exactement la partie de code que l'on veut.

Prenons un exemple bien concret :

Supposons que vous ayez créé une fenêtre et que vous désiriez
rajouter un bouton dans votre code : vous sélectionnerez alors
uniquement 'Code' et génèrerez le code pour le bouton uniquement.
Puis apres avoir inséré ce texte dans votre programe, vous
aurez besoin de rajouter la déclaration uniquement : sélectionnez
alors uniquement 'Declaration' ... et vous insérez directement le
code généré à l'endroit adequat dans le texte de votre
programme.

1.33 code-application

Ce bouton vous permet de générer le code pour toute l'application.

Bien sur, le code généré dépend des options que vous aurez choisies.

1.34 code-object

Ce bouton vous permet de générer le code de l'objet sélectionné dans
la liste 'Objects labels'.

Bien sur, le code généré dépend des options que vous aurez choisies.

1.35 remove-label

Vous pouvez grâce à ce bouton, indiquer au programme de ne pas
générer le nom de la variable que vous avez sélectionnée dans la
liste 'Generated Labels'.

MUI-Builder sait automatiquement si, oui ou non, il doit générer
un nom de variable. Ainsi, il est par exemple inutile (dans la
majorité des cas) de garder une variable liée à un groupe MUI, à
moins d'avoir besoin de rajouter dynamiquement des objets dans
ce groupe durant l'exécution du programme.

Ce bouton, ainsi que le bouton
Add Label
permettent
de changer les définitions standards de MUI-Builder sur certains
objets.

1.36 add-label

Vous pouvez, grâce à ce bouton, indiquer au programme de générer une variable pour un objet donné.

Voir également
Remove Label

1.37 Langage C

Un certain nombre de choses doivent être remarquées à propos du code généré :

- Certains objets nécessitent des variables auxiliaires, autres que celles qui les définissent directement. Ainsi, un texte doit référer à une chaîne de caractères pour définir son contenu. De la même manière, une list de type FloatText doit également pointer sur une chaîne de caractère.
  Ces variables auxiliaires sont générées et initialisées par le programme et portent, pour les chaines de caractères le nom 'STR_"label_de_l'objet"'.

- AUCUNE notification n'est effectuée dans le programme généré. Vous devez donc, si vous voulez tester rapidement le programme, au moins ouvrir une fenêtre avant la boucle évènementielle.
  (ce qui s'effectue avec set(WI_window, MUIA_Window_Open, TRUE))
  Si vous faites uniquement cette modification, vous devez sortir du programme en utilisant la commodite 'Exchange'.

Le code généré utilise un fichier 'header' nommé 'code.h', livré avec l'archive de MUI-Builder. Ce fichier, extrait des sources livrés avec MUI, devrait normalement vous permettre de compiler les sources générés par MUI-Builder avec les compilateurs les plus répandus.

1.38 Langage E

Comme MUI-Builder génére du code générique qui est ensuite interprété par des modules, le code E généré est identique dans son contenu au code C généré. Reportez vous donc au paragraphe sur le langage C pour tout ce qui concerne les variables générées et la notification. Les différences sont d'ordre syntaxique. Voici les points de la génération spécifiques au code E :

- le fichier généré commence toujours par 'OPT OSVERSION=37' pour assurer que les ROM 2.0 ou plus sont présentes
- une série de 'fichiers include E' sont appelés via l'instruction MODULE : ces fichiers sont en particulier nécessaires pour la fonction doMethod (voir plus loin)
- la constante MUI_TRUE est définie à 1 : celle-ci doit être utilisée à la place de TRUE (qui vaut -1) pour MUI, car celui-ci ne reconnait pas toujours la valeur -1 comme vrai
- l'appel de init() en C a été remplacé par une simple ouverture de la muimaster.library : si son ouverture est impossible, le programme rend la main en retournant 100

- une fonction doMethod est ajoutée à la suite de la procédure main : celle-ci correspond à la fonction DoMethod de la amiga.lib inutilisable avec AmigaE, et dont la syntaxe d'utilisation (conforme à l'originale de la amiga.lib) est doMethod(pointeur_objet, [param1, param2, ...])
Notez que la fonction doMethod généré m'a été donnée par Wouter van Oortmerssen (merci Wouter !) l'auteur d'Amiga E.

Le seul point délicat concerne les chaines entrées dans MUI-Builder. La règle générale est de les entrer comme si vous vouliez les utiliser en C à une exception près : contrairement à ce qu'il est dit avant, le caractère " peut être entré tel quel (ce n'était génant que pour le C). C'est le module de génération de code E qui s'occupera de la traduction. Plus exactement ce module reconnait :
- \r qui est remplacé par \b
- \n, \t et \e laissés tels quels
- \0oo où oo est un nombre en octal : attention le \ doit être obligatoirement suivi par un 0, suivi lui-même de 2 chiffres entre 0 et 8
- \xhh où hh est un nombre héxadécimal : attention le \ doit être obligatoirement suivi par un x en minuscule, suivi lui-même de 2 symbôles corrects (chiffre ou lettre entre a et f, ou A et F)
- le caractère ' est remplacé par \a

Attention, si la syntaxe de \0oo ou de \xhh n'est pas respectée, le module de génération du code E ne signalera pas d'erreur mais le résultat ne sera surement pas celui qui était attendu !
Si le module de génération de code E rencontre \033 ou \x1B (valeur 27 en décimal, c'est à dire le code de ESC), il le remplacera par \e. Si par contre, il rencontre un \0oo ou un \xhh de valeur décimal différente de 27, il produira une chaine E non pas sous la forme 'chaine', mais sous la forme ["c", "h", "a", "i", "n", "e", 0]:CHAR car il n'y a pas d'équivalent en E aux \0oo et aux \xhh du C. Mais les 2 formes sont totalement équivalentes.

Enfin, le fichier source produit contient plein de macros identiques à celles du C. Pour utiliser ces macros, vous aurez besoin d'un préprocesseur annexe car Amiga E ne le permet pas à l'origine. C'est pour cela qu'est jointe à cette archive une autre archive, Mac2E.lha (également disponible seule sur aminet). Vous trouverez dans cette archive tout ce qu'il faut pour exploiter les fichiers sources générés, avec en particulier un préprocesseur spécialisé dans le remplacement des macros et destiné à Amiga E. Reportez-vous donc à cette archive pour plus de détails.

Finissons ces explications par le traditionnel copyright :
- le module de génération de code E (GenCodeE) ne peut-être séparé de l'archive de MUI-Builder
- GenCodeE reste sous copyright de l'auteur et ne peut donc être modifié sans mon accord
- je dégage toute responsabilité quant à l'utilisation de ce programme et les dommages qu'il pourrait causer : vous l'utilisez à vos risques et périls !

Lionel Vintenat

1.39 guide

Grâce à MUI-Builder, plus personne ne pourra plus avoir de raisons valables pour réaliser ce que l'on trouve sur toutes les autres machines (y compris les PC, ce qui n'est pas peu dire !!!), à savoir une aide en ligne, une documentation hypertexte de votre programme.
En effet, vous pouvez attacher à tout objet MUI créé depuis MUI-Builder un petit texte d'aide.
Ensuite, MUI-Builder génèrera automatiquement une documentation hypertexte au format AmigaGuide (que vous pourrez même admirer depuis MUI-Builder grâce au bouton 'View').

Au même titre que la génération de code, vous pouvez ne générer qu'une partie de la documentation pour l'insérer directement dans une documentation déjà écrite.
Ainsi, si vous effectuez des modifications dans la documentation générée par MUI-Builder, elle ne seront pas perdues : vous ne générerez que la partie du texte concernant les éléments de l'interface que vous avez modifiés, et vous les insérerez dans le document.

Vous pouvez éditer le texte de la documentation
- le texte principal avec 'App Node'
- le texte pour une fenêtre avec 'Window Node'
- le texte d'un objet avec 'Object Node'

Vous pouvez générer la documentation :
- pour toute l'application avec 'Generate whole Doc'
- pour une fenêtre avec 'Generate win Doc'
- pour un objet avec 'Generate obj Doc'

En double-cliquant sur un nom d'objet ou de fenêtre, vous pouvez éditer le titre du texte d'aide correspondant, et voir le texte attaché à l'objet. (même chose avec les deux boutons 'Edit').

1.40 L'enregistrement

Rien de plus simple que de s'enregistrer !

MUI-Builder est GiftWare.

Ce qui signifie que vous n'êtes absolument
pas tenu de verser quelque chose à l'auteur pour continuer à utiliser ce logiciel.

Si vous aimez ce logiciel, qu'il vous est ou vous a été particulièrement utile, alors vous pouvez m'envoyer un modeste don (de l'ordre de 50FF ou 15$US) à l'adresse suivante :

Eric Totel
5 rue Riquet
31000 Toulouse
France

A tous ceux qui ne se sentent pas assez philantrope pour envoyer ce genre de don (les étudiants fauchés par exemple !),

je suggère de m'envoyer une simple lettre, une
carte postale, un mail (adresse : totel@laas.fr), pour
m'encourager, me donner vos impressions sur le logiciel, vos
idées ... etc ...

N'HESITEZ surtout PAS : je veux absolument savoir si MUI-Builder
est utilisé ou apprécié, de manière à savoir s'il est vraiment
utile d'en continuer le développement.

D'autre part, si vous avez traduit la documentation dans
votre propre langue (ou une autre qui n'est pas fournie dans
l'archive actuelle de MUIBuilder), ou bien avez dessiné un
superbe icone pour ce programme (comparé à l'horreur qui vous
est livrée ici) ... alors (s'il vous plait) envoyez moi ce
que vous avez fait de toute urgence !!!!

1.41 Futur

Parmi les améliorations futures de MUI-Builder, il est prévu
d'implémenter :

- La localisation, de MUI-Builder lui-même, ainsi que la
  localisation du code généré par MUI-Builder

- La possibilité d'échanger des données avec des éditeurs de
  texte par l'intermédiaire du port AREXX, afin d'insérer
  directement du code généré par MUI-Builder dans votre
  programme.

- Toutes les modifications que vous auriez envie de voir apporter
  à MUI-Builder (n'hésitez surtout PAS à me faire parvenir TOUTES
  vos idées et remarques, de manière à ce que le programme
  évolue suivant vos besoins).

1.42 Remerciements

Je voudrais remercier ici tous ceux qui m'ont aidé et ont
contribué à la réalisation de MUI-Builder :

- Lionel Vintenat pour l'idée du code générique et la réalisation
  du générateur de code pour le langage E.

- Pascal Rabier pour le mal qu'il s'est donné à traduire la
  documentation en Anglais.

- Gael Marziou, Pierre Carrette, et Pascal Pensa pour les tests
  attentifs qu'ils ont réalisés, et pour toutes les (nombreuses)
  idées qu'ils m'ont suggéré (la réalisation de la création d'une
  aide en ligne n'étant que l'une d'entre elles).

- Tous les autres testeurs pour les bugs qu'ils ont réussi à
  découvrir.