in

COLLABORATORS			
	TITLE : in		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 8, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

in

1.1 main

```
                  ================================================================
>>>>>> AutoGui.gc by D. Keletsekis <<<<<<<<<<<<<<<<<<<<<<<<<<<<
================================================================
A Gui4Cli gui for the automatic creation of guis for programs,
based on their Command Line Arguments.
================================================================


               Introduction
               What it is..

               Instalation
               Where to put it..


               Usage
               How to do it..

               Prefs
               Change its parameters..

               CLI arguments
               What those /N/S things mean..

               Editing
               How to move gadgets about..

               Gui4Cli
               All the available commands..


               Author
               It wasn't his fault!..

================================================================
16/2/2000 - dck@hol.gr - http://users.hol.gr/~dck/gcmain.htm
================================================================
```

1.2 author

```
 ------------------------------------------------------
 This program is WhateverWare(TM).
 To use it, you must own or have access to a computer
 that can run it. Otherwise I'm comming after you!
 ------------------------------------------------------

 The Author of this mess is :

 Dimitris C. Keletsekis
 14 King George str.,
 Athens 10674,
 Greece

 Email : dck@hol.gr

 This software is provided as-is. Use it at your own risk.
 No warranties are made or implied.

 ------------------------------------------------------
```

1.3 gc_commands

```
 Gui4Cli commands
 ================================================================

 (This is a listing of all Gui4Cli commands and events)

 Parser Commands :

NewFile         NewFileName
TextFile        FileName

 --------------------------- GLOBAL COMMANDS ------------------------


WinBig          L T W H Title
WinSmall        L T W H
WinType         MASK (Close|Drag|Zoom|Depth|Borderless|Backdrop|RIGHT|BOTTOM)
WinOut          ConsoleSpecification
WinOnWin        GuiName LeftOffset TopOffset
WinOnMouse      LeftOffset TopOffset
Screen          PublicScreenName
WinFont         FontName Size  UL|BD|IT(Mask)
WinBackground   SOLID|PATTERN|ICON|IMAGE APen|Name BPen
UseTopaz
NoFontSense
VarPath         VariableSearchPath
```

```
 ResInfo        FontHeight ScreenWidth ScreenHeight
 ShareMenu      GuiFile


  ----------------------------- EVENTS -----------------------------

 xButton        L T W H Title
 xCheckBox      L T W H Title Variable OnText OffText ON|OFF
 xVSlider       L T W H Title Variable Min Max Current ShowStr
 xHSlider       L T W H Title Variable Min Max Current ShowStr
 xTextIn        L T W H Title Variable StartingText Bufflength
 xCycler        L T W H Title Variable
 xRadio         L T W H Variable Spacing
 xArea          L T W H COMP|BOX|NONE
 xPalette       L T W H

 xListview      L T W H Title Variable File|Dir Offset NUM|TXT|MULTI|DIR
 LV Hooks       LVDirHook HookID, LVHook HookID

 xMemu          Menu Item SubItem Shortcut
 xIcon          L T IconName (no .info)
 xAppMenu       AppMenuName Variable ONOFF
 xAppIcon       L T IconName Title Variable ON|OFF
 xAppWindow     Variable

 xPipe          PipeFileName ON|OFF
 xTimer         TIME|SINGLE|REPEAT Time|Interval  ON|OFF
 xNotify        File|Dir ON|OFF
 xHotKey        KeyCombination ON|OFF

 xRoutine       RoutineName
 xOnHelp        ON|OFF|AUTO
 xOnKey         Letter|#KeyValue
 xOnReturn      LaunchID
 xOnJump        Variable
 System Events  xOnLoad, xOnOpen, xOnClose, xOnQuit
 Other Events   xOnActive, xOnFail, xOnDiskIn etc..


  -------------------------- GRAPHIC Events ------------------------

 Graphics       The following commands :
    BOX      L T W H  IN|OUT  BUTTON|RIDGE|ICONDROP
    CTEXT    L T Text FontName size FGpen BGpen UL|BD|IT|EMBOSS|SIZE(mask)
    LINE     L T L T ColorNo
    SQUARE   L T W H ColorNo FILL|NOFILL
    CIRCLE   centerL centerT xradius yradius ColorNo FILL|NOFILL
    ICON     L T IconName (no info)

 Text           L T W H Text Length BOX|NOBOX
 Gauge          L T W H IN|OUT BUTTON|RIDGE|ICONDROP APEN BPEN PERCENT
 xTextBox       L T W H Title Text


  --------------------- ATTRIBUTES & Event MODIFIERS -----------------

Gadget Modifiers :
```

```
 GadID          IDNumber
 GadHelp        HelpText
 GadFont        FontName  FontSize  MASK(Underline|Bold|Italics)
 GadTitle       ABOVE|BELOW|LEFT|RIGHT
 GadKey         Letter (or #ASCII value)
 GadTxt         LEFT|CENTER|RIGHT
 LVDirHook      HookID
 Local          Variables/var/var...

 Attr           AttributeName Value (IMPORTANT)


  --------------------------- EVENT COMMANDS ---------------------------


Controlling Gadgets :

 SetGad         GuiFile GadIDs ON|OFF|SHOW|HIDE  (Arexx capable)
 Update         GuiFile GadID Value   (Arexx capable)
 ChangeArg      GuiFile GadID ArgNumber NewValue
 ChangeGad      GuiFile GadID L T W H Title
 ReDraw         GuiFile
 PartReDraw     GuiFile L T W H
 GadReDraw      GuiFile LeftGad Top Right Bottom Offset
 ChangeIcon     GuiFile GadID L T NewIconName
 SetAttr        GuiFile/GadID AttributeName Value
 SetGadValues   GuiFile

Control Statements :

 If/ElseIf/Else/Endif/And.. Argument Operator Argument
 IfExists/Else/EndIf..       SYSTEM  Name|~Name
 While/EndWhile/And/Or      Argument Operator Argument
 Mark/Goto                  MarkName
 Gosub/Return               GuiName RoutineName  (ARexx capable)
 DoCase/Case/Break/EndCase  (DoCase) Argument - Case Operator Argument
 Stop

          --- All Commands below this line are ARexx capable ----

 Quit

DOS Commands :

 Run, CLI       CommandLine
 SendRexx       PortName  CommandLine
 Wait            SYSTEM  Name|~Name TimeOut
 MakeDir        DirName
 Assign         Device: Path|REMOVE
 Rename         OldFile NewFile
 Launch         LaunchID CommandLine

 FailAt         ErrorNumber

Recursive commands :
```

```
 Copy      FileName (with wild characters) Destination
 Delete    FileName (with wild characters)
 Action    COPY|COPYNEW|MOVE|DELETE|SIZE|PROTECT|CLI File/Dir Destination
 LVAction  COPY|COPYNEW|MOVE|DELETE|SIZE|PROTECT|CLI Destination

Note : DOS and Recursive Commands always set the  $$RetCode

Handling GUIs :

 Load/Open...  GuiLoad GuiFullPathName - GuiOpen/GuiClose/GuiQuit GuiName
 GuiRename     OldGuiName NewGuiName
 Status
 Info          GUI|GADGET|PALETTE|IMAGE Guiname|Guiname/GadID|ImageAlias

Handling Variables :

 SetVar        Variable String (or var = string)
 DelVar        Variable
 AppVar        Variable Text
 CutVar        SourceVar CUT|COPY CHAR|WORD|LINE Amount DestinationVar
 Counter       Variable INC|DEC Amount
 Append        File String
 JoinFile      Path File Variable
 ParseVar      Variable
 CalcVar       ResultVar  Argument operator Argument
 ReadVar       FileName Start Length Variable
 SearchVar     Variable String CI|CS FIRST|NEXT
 RepVar        Variable OldString NewString CI|CS

 Extract       FromVar ITEM ToVar

 ListView Commands  :

 LVUse         GuiFile GadID
 LVDel         LineNumber
 LVPut         NewText
 LVChange      NewFromFile
 LVSort        ASC|DSC|%FieldName
 LVFind        String
 LVAdd         String
 LVInsert      (Before)LineNumber String
 LVClear
 LVSave        FileName
 LVMove        +-Offset|#LineNumber
 LVGo          first|next|prev|last|#LineNumber
 LVSearch      string CI|CS First|Next
 LVRep         OldString NewString CI|CS
 LVMode        NUM|TXT|MULTI|DIR
 LVClip        CUT|COPY lines|-1 ADD|PASTE|INSERT Gui ID
 LVSwitch      Gui ID

 LVMulti       First|Next|On|Off|All|None|Show
 LVDir         Parent|Root|Disks|All|None|Refresh|NoRefresh|#DirName

 DataBase ListView Commands  :

 DBSum         ALL|SELECTED|UNSELECTED %FieldName ResultVar
```

```
 RecSort       %FieldName

Multimedia commands :

 SetColor      GuiFile ColorNumber R G B
 Palette       LOAD|SAVE|SET|GET|REMAP Src Dest
 Speak         Text
 Images
    LOADIMAGE    ImageFile Alias ScreenName|NoRemap
    FREEIMAGE    Alias
    IMAGE        Left Top Alias
    CHANGEIMAGE  GuiFile GadID Left Top Alias
 Sound Effects
    LOADSOUND FileName Alias
    FREESOUND Alias
    PLAYSOUND Alias
    SETSOUND  Alias VOLUME/SPEED value

Various Commands :

 SetScreen     GuiFile ScreenName
 GuiScreen     GuiFile FRONT|BACK
 GuiWindow     GuiFile ON|BIG|SMALL|FRONT|BACK|WAIT|RESUME
 SetWinTitle   GuiFile NewTitle
SetScreenTitle GuiFile NewTitle
 ReqFile       L T W H Title SAVE|LOAD|MULTI|DIR Variable DirName
 CD            NewDirectoryName
 Delay         Ticks
 EZReq         Text Choices Variable
 Say           Text
 Set           [parameter]  [value]
 SetStack      StackSize
 MakeScreen    ScreenName Depth|(W/H/D/Mode) Title
 KillScreen    ScreenName
 TTGet         FullPath/IconName (without ".info")
 BreakTask     TaskName CDEF(signals)
 Flash
 MoveScreen    GuiName/#ScreenName X Y
 Workbench     Open/Close
 SetPointer    GuiName  #Image/DEFAULT/HIDE
```

1.4 installation

```
  Installation
  ============================================================

  - Copy the AutoGui directory anywhere..

  The Gui4Cli version included here is version 3.8.3
  If you have Gui4Cli installed, and your version is older,
  copy it over..
```

```
Otherwise..

- Copy the binaries "Gui" and "Gui4Cli" to your C: directory.
  This is the minimum Gui4Cli installation.

The full version of Gui4Cli can be found on Aminet or at:
- http://users.hol.gr/~dck/gcmain.htm
```

1.5 introduction

```
Introduction
============================================================

The Amiga OS has a very usefull funtion, ReadArgs(), which
will read in CLI arguments according to a given template,
something like: "FILE/A,OPTION/N/A". This function is used by
most programs to get their arguments when launched from a
shell. Usually, if you type a program name followd by a "?"
(>c:List ?) you'll get a list of all the program's arguments.
That's ReadArgs() at work..

AutoGui enables you to create a fully functioning gui, based
on these command line arguments.

An argument can be a string (in this case a TextIn gadget will
be used) or a "/N" number (a TextIn gadget allowing only numbers),
or a "/S" switch (a CheckBox).

AutoGui will take these facts, do some calculations and create a
Gui4Cli script containing all the relevant gadgets, positioned
in a sensible way. TextIn gadgets will also have a file request
button next to them.

The gui will be fully functioning and you will be able to run it
right off.

Gui4Cli allows you to visually edit the gui while it is running,
so you can move the gadgets around to fix anything you don't
like. You can also change the code and add/change its behaviour
in any way. The Gui4Cli languange is powerfull and you can do
a great many things with it (like add comodities, pipes, file
notifications, make screens and many many other things..)
```

1.6 prefs

Preferences

The Prefs gui allows to set some global parameters that will be used in making the gui. You don't have to change anything here.. the defaults should be good enough for most.

This is what the gadgets do:

- "Gui Width"
  Is the.. gui width.. What more can I say..

- "Window Margins"
  is the distance of titles etc from the window border.

- "Title Margins"
  is the distance (in pixels) that will be used in calculating how much room to leave for the gadget titles. If the titles fall on the window borders or on other gadgets, you should increase this..

- "Gadget distance"
  This is the vertical distance that gadgets will have from each other.

- "Box Attributes"
  The gui has 2 decorative boxes, unless you also choose the "Pipe: output" in which case it will have 3 boxes (one more in the middle) The look of these boxes can be changed by changing the strings contained in the 3 Text gadgets - this is
  what they mean
  ..

- "Program Stack"
  This is the amount of stack to give a program when launching it. The default value is 4k and you'll seldom have to change it, but some programs require more.

- "Add Pipe: Output"
  Will allocate and open a PIPE: device and will redirect the command's output into it. A TextBox gadget will be added in the middle of the gui, displaying the last line of the output. This is useful for commands that have a single line, or slow output (like format or something..). For a command like "list" where you get lots of fast output, this is useless..

- "Font"
  You can choose a global window font here that will be used for all titles and gadgets. Choose the font size (inside the font dir). If this field is left blank the gui will use the users prefered screen font (which is best). If you set a font, keep in mind that if other people use your gui they may not have your fonts..

- "Titles"
  Here you may choose a style for the gadget titles

1.7 readargs

```
Command Line Argument Template
==============================================================

(This is an edited extract from the ReadArgs() function autodocs..)

ReadArgs() parses the commandline according to a template that is
passed to it.  This specifies the different command-line options and
their types.  A template consists of a list of options.  Options are
named in "full" names where possible (for example, "Quick" instead of
"Q").  Abbreviations can also be specified by using "abbrev=option"
(for example, "Q=Quick").

Options in the template are separated by commas.
Options can be followed by modifiers, which specify things such as
the type of the option.  Modifiers are specified by following the
option with a '/' and a single character modifier.  Multiple modifiers
can be specified by using multiple '/'s.  Valid modifiers are:

/S - Switch.  This is considered a boolean variable, and will be
     set if the option name appears in the command-line.

/K - Keyword.  This means that the option will not be filled unless
     the keyword appears.  For example if the template is "Name/K",
     then unless "Name=<string>" or "Name <string>" appears in the
     command line, Name will not be filled.

/N - Number.  This parameter is considered a decimal number, and will
     be converted by ReadArgs.  If an invalid number is specified,
     an error will be returned.

/T - Toggle.  This is similar to a switch, but when specified causes
     the boolean value to "toggle".  Similar to /S.

/A - Required.  This keyword must be given a value during command-line
     processing, or an error is returned.

/F - Rest of line.  If this is specified, the entire rest of the line
     is taken as the parameter for the option, even if other option
     keywords appear in it.

/M - Multiple strings.  This means the argument will take any number
     of strings, returning them as an array of strings.  Any arguments
     not considered to be part of another option will be added to this
     option.  Only one /M should be specified in a template.  Example:
     for a template "Dir/M,All/S" the command-line "foo bar all qwe"
     will set the boolean "all", and return an array consisting of
     "foo", "bar", and "qwe".

     There is an interaction between /M parameters and /A parameters.
     If there are unfilled /A parameters after parsing, it will grab
```

```
     strings from the end of a previous /M parameter list to fill the
     /A's.  This is used for things like Copy ("From/A/M,To/A").
```

1.8 tbstyle

```
* xTEXTBOX style:

apen/bpen/bgpen/border/recess

- apen, bpen : these are the pens that will be used to draw the
               title text. "bpen" will be used as shadow/outline

- bgpen      : this is the background color of the textbox. You
               can give a -1 if you do not want a background.

- border     : NONE, PLAIN, BUTTON, RIDGE or ICONDROP
               (and I'm not going to explain what each one is..)

- recess     : "IN" to have the border recessed, OUT otherwise.

Example :
>  2/1/3/BUTTON/OUT
```

1.9 usage

```
                 AutoGui Usage
================================================================

* Getting the Arguments:

The first thing to do in making a gui is see what arguments
it should handle. To do that, we need to know the command or
program it should run so as to get its arguments.

When you load the gui you will be prompted to choose a
command. You should choose a CLI command or other program
that uses the ReadArgs template
               ReadArgs template
               - most do..

AutoGui will run the command like: "Run command ? >T:Temp"
If all goes well, the command's template will be stored into
the file T:Temp and AutoGui will load it from there into the
gui's listview. You may get error reports here, since many
commands complain heavily if no arguments are provided.
Others may do nothing at all..

You may, at any time, also click on "NEW.." and load another
command. The current arguments will be cleared and the new
command will be run again to get its arguments & list them.
```

* What about if the command does not behave ?

If AutoGui can not extract the CLI template from the program,
you can manually enter it, by:

- Choosing the command to run, with "COM"

- Adding the arguments one by one, with "ADD"

Note that with a little imagination you can use this method
to make custom guis for programs that don't use the standard
ReadArgs template (like unix progs etc)

* Editing the Arguments:

Once you have a list of arguments you can re-arrange or edit them.

Use "UP" or "DOWN" to re-arrange the arguments. AutoGui will
use the arguments as it finds them, working out the positioning
as it goes along. A string gadget will take up the full
width of the window, while you can have many checkboxes or
number gadgets in one line, so move them about as you see
fit to get the look you want. Note that after the gui is created
you can
visually edit
it and move or resize the gadgets all
over the place..

You can also "EDIT" the arguments, changing them, but be
carefull that the program accepts them.

Some arguments may have two forms: "CS=CASESENSITIVE/K".
In this case AutoGui will throw away the "CS" and only
use the "CASESENSITIVE" part. You may want to edit that..

* Controlling the look of the gui:

The
Prefs..
button will open another gui and let you adjust
some of the parameters AutoGui will use in creating your gui.

* Ta Daaa!..:

You are now set.. Click on "CREATE!" and be amazed..

The gui that will pop up is fully functioning, writen in the
proper style, and ready to run. You may use it, or edit it in
any way..

Then hit "NEW.." and make another..

1.10 visualedit

```
VISUAL EDITING
==============================================================

Gui4Cli GUIs can be edited while they are running. This is done
by using the CONTROL key together with the mouse :


   Moving Gadgets :

Press CONTROL-MouseClick on a Gadget or Graphic to Select it.
You will see that an outline of the gadget is drawn. You can
now let go of the CONTROL button and move the gadget outline
around in your window.

When you are satisfied with the new position, just click the
mouse and the gadget/graphic will be redrawn to this new position.

You can also use CONTROL-H instead of CONTROL-MouseClick to
select a gadget, and CONTROL-H again to place it where you
want.

This comes in handy if :
(a) you are running programs like CycleToMenu etc which may
interfere with your mouse clicks, or
(b) you are trying to paste some gadget over another gadget
in which case GadTools may eat-up the mouse clicks.

There is also a GRID available, which makes lining up the gadgets much
easier. By default the grid size is 1 which means "no grid".
You can set the grid size to any size you want, with "SET GRID 5"
(5 is a good size..) - or through the Prefs gui.


   Resizing Gadgets :

You can resize a gadget by clicking on it's bottom right corner.

Note that some gadgets (such as ICONs, Images, xICONs and
CTEXT) can not be resized.

Also note that to resize a listview you have to click on the
listview's bottom right corner - *not* on the arrow buttons
(unless you use control-h).

Since LVs adjust their size automatically to show as many
lines as possible, they may be a little difficult to select
correctly for resizing..


   Resizing the window :
```

You can enlarge or reduce the window size by resizing the
window while holding down the CONTROL key. In this case, the
window is resized, while the gadget sizes/positions remain the
same.

Cloning gadgets :

After selecting a gadget you can "clone" it (i.e. make a copy) by
using the CONTROL-J shortcut. In this case a copy of the gadget will
be created and drawn where the mouse is at.

Note that *only* the gadget information is copied.
The gadgets modifiers and it's commands are *not* copied.

Inter-GUI Cloning :

After selecting a gadget you can also place it in another
Gui4Cli window. In this case a copy of the gadget will be created
and drawn into the window where you clicked.

This enables you to make new guis by copying gadgets from other
guis - in effect, a gui editor.

Here also, *only* the gadget information is copied.

Deleting gadgets :

You can delete a gadget or graphic by selecting it and pressing
the DELETE key.

Saving your GUIs :

Once you have made the GUI of your dreams, you can save it by
pressing CONTROL-G. A simple requester will ask you if you
want to save the gui.

IMPORTANT notes on saving :

- NEVER - load a gui, then change it's file by manually editing
  it and then edit & save the gui. To keep all your file notes
  etc in tact, Gui4Cli remembers the line numbers of the gadgets
  when the file is loaded, and then, when saving, it goes and
  changes only these lines, leaving everything else untouched.

  So if you have meanwhile changed the gui manually the gadget
  line numbers will have changed and ... well you're looking
  for trouble, that's what!

  Same will happen if you visually edited the file before and
  added 1 or more xCYCLER or xRADIO type gadgets. Since these need
  extra lines to describe the fields they'll have, it throws the
  whole numbering scheme off if you try to save the gui again
  without having reloaded it first.

Hitting CONTROL-R to reload the gui (if you have changed it manually or have added cycler/radio gads) *before* starting to visually edit it, will reload and thereby refresh the correct line numbers etc.

Gui4Cli will check and tell you if it doesn't find the edited gadget where it should be in the file.

- BE VERY CAREFUL when saving multi-gui files, if you are trying to edit more than one of the file's guis at the same time. Reloading the gui will *not* refresh the gadget line numbers of the other guis, since only the active gui is reloaded..

  If you must edit many guis of a multi-gui file at the same time, then reload *all* the file's guis everytime you save any one of them.

- If you have deleted existing gadgets, then the lines of the original GUI describing the gadget and all it's attached commands will be commented (i.e. a ; will be added in front)

  This will be done for all the lines following the gadget, until the next gadget is declared, or the end of file is reached.

- If you have created new gadgets, these will be added at the bottom of the GUI file. If it's a multi-gui file, then at the end of the given gui within the file.

- Any RESIZE_BIG/RESIZE_SMALL commands that the gui file may contain will be commented out and the gui will be saved at it's current size.

Quirks :

- Circles are selected by clicking on their lower right quarter.

- xTEXTIN gadgets and Boxes (even filled ones) are selected by clicking on their border. You have to click *exactly* on a pixel on their border.

- TextIn gads (and maybe some others) will give you a little trouble if you try to paste them over themselves (i.e. move them a tiny bit. Try to pick them up from their edges..

1.11 index

Guide INDEX :

Installation

Introduction

Prefs

ReadArgs

TBStyle

Usage

VisualEdit