KEY_TO_TRITON

Philipp Lonke <phips@scout.franken.de>

COLLABORATORS

	TITLE : KEY_TO_TRITON	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Philipp Lonke <phips@scout.franken.de>	October 23, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

KEY_TO_TRITON

1.1 The key to TRITON programming in Blitz2

```
                                        TRITON step by step
                          --------------------

      A small key to TRITON programming using the interfaces written
                 by Philipp Lonke <phips@scout.franken.de>



                 Introduction
                     What's all about.

                 Needful things
                     What should be known before

                 Installation
                     How to get started

                 Programming example
                     "Hello world" in
                TRITON

                 FAQ
                     Frequently asked questions

                 Addings
                     What's more to say?

                 The future
                     What could come...

                 Thanks and legals...
                     The end!

                 For advanced
                      How to create smaller execs


                 Changes
```

1.2 Introduction

First of all, forget everything about GUI programming in Blitz2. It is not the same creating a GadTool-GUI or a TRITON GUI.

Read this documentation carefully, so you really understand the difference.

I wish to thank a lot Rupert "HelpApp" Henson, without his help this conversion could not been finished. He had the trick how to do the Project-TagList and helped me converting the macros. Now this trick is obsolete due to the Taglistlib from D.C.J. Pink.

Go back

1.3 Some useful definitions

When speaking of TRITON, there are to major terms:

an application: That is your program. The informations (application tags) are for use in the TRITON Preferences Editor (ShareWare)

a project : In fact, that's your GUI. For every window you use, you create a new project.

Second, every string passed to any TRITON library function has to be passed using Null(s$) because these functions need the adress of a null-terminated string.

NEVER use an ID of zero for anything (window, button etc.)

If you ever break your program with the debugger when the window was already open, do *NOT* end it with the debugger, instead continue the program and close the window. Otherwise TRITON will not close the window what causes confusion when you restart your

program!

Go back

1.4 How to install TRITON on your Blitz2 system

First of all: Before you can use this package, you have to get at least the
TRITON developer archive from aminet/dev/gui. Take also a look at
the TRITON user archive in aminet/util/libs. It contains the very good
TRITON preference editor.

IF YOU DON'T HAVE THESE ARCHIVES, THIS PACKAGE IS WORTHLESS TO YOU!

If you are not sure which library-IDs are free on your system, get the
program "ViewLibs" or the new library manager "LibMan". Both should be
avaiable on any
Blitz2-ftp-site
. Just type "ViewLibs"
in the shell, it'll show up all libraries with their IDs. Using LibMan,
start it and change view to "Sort by ID".

When you have the developer archive, you have to
convert
the
file "triton/developer/fd/triton_lib.fd" with FDConvert. This program you should find in
your Blitz2:BBTools-drawer or on any
Blitz2-ftp-site
.

Put the converted library in "blitzlibs:amigalibs"

Then, you should compile the "taglistlib.bb2" file to
"TagListlib.obj" and put this into "blitzlibs:userlibs". Take
care, that the library ID of 10 is free. If not, change the
constant #Taglistlib from 10 to the id that fits to your system. And
remember, before you compile the library, include "blitzlibs:libmacs.res"
in the resident file list of the compiler options menu (Amiga-o)

To make sure that you check for a free ID, I changed the libray ID in the
source code to "xx" which will produce an error if you compile it. Just
exchange the "xx" to your favourite free library ID.

Now delete your old "deflibs" file and create a new one with
MakeDefLibs, or, if you use Blitz2.1, run
LibMan
.

Reload Blitz2 and try typing "TR_OpenProject_" (without quotes) and
press the HELP-Key on your keyboard. Do the same typing
"InitTagList" (without quotes) and press the help key again.

```
If it turns to your tokencolor and a short helptext appears in your titlebar,
then you did everything right.

To be able to use the TRITON include-file "triton.bb2", you have
to save it first with Blitz2, so that it will be tokenised!
```

1.5 Converting 'triton_lib.fd' with FDConv

```
Start a shell.
cd to the appropriate directory where the triton_lib.fd file is.
Type "FDConv triton_lib.fd"
FDConv asks now for the library name.
Type "triton.library"
It'll then suggest a library ID.
Ignore this suggestion and type a free ID in, e.g. 12
The converted library will automatically be added to the
Blitzlibs:userlibs/ directory.

Take care that the IDs of TRITON.library and the taglistlib are
different!!!!!
```

1.6 How to program a TRITON user interface?

```
                  The use and syntax of the commands of the taglistlibrary can  ↩
                     easily
  be taken from the library source.
  The use and syntax of the commands of TRITON.library can be taken
  from the autodoc file which comes with the TRITON developer archive.

  Remember that you are using system library calls. You get all the
  keywords from the TRITON archive's autodoc file, just add an under-
  score to them. You mustn't write i.e. TR_OpenProject but instead
  TR_OpenProject_

If you plan programs with more than 5 taglists, don't forget to
increase the amount in 'compiler options' !


  Now we can start:

  Of course, first come the includes (remember amigalibs.res!) and
  the most important variable initialisation.

      INCLUDE "blitz2:bbincludes/libraries/TRITON.bb2"    ; or wherever you
                                                          ; put it.
      InitTaglist 2,10                    ; making place for our application
      InitTagList 1,20                    ; making place for 20 tags
                                          ; in the first taglist

      Use TagList 2                       ; and use the app. taglist
```

Now, we could tell
TRITON
, that we launch a new application, our program. Therefore, TRITON wants to know something about our program for its prefs program.

The tags you can use, are:

```
#TRCA_Name
#TRCA_LongName
#TRCA_Info
#TRCA_Version
#TRCA_Release
#TRCA_Date
```

And it looks like this in your code:

```
AddTags #TRCA_Name,Null("TritonTemplate")
AddTags #TRCA_LongName,Null("TritonTemplate")
AddTags #TRCA_Info,Null("Looks like a template")
AddTags #TAG_END,0
```

Now we open our application. Remember that this variable MUST be long!

```
application.l=TR_CreateApp_ (TagList)
```

Before we continue, we have to check that nothing has happened.

```
if (application)

    ... code ...
```

Now we create our GUI. First, we want to do a window with two buttons. So we use the prepared Taglist:

```
Use TagList 1
```

Therefore we use the macros. For the window, the most important ones are:

```
!WindowID{id}   - the ID must not be zero!!!

!WindowPosition{
    positionflag
    }

!WindowTitle{Null("Window title")}
    as you remember, strings and text must always
    be passed with the Null(string) command!!

!WindowFlags{
    flag1|flag2|flag3|...
    }
```

If you create your Taglist with the TagListLib, then remember that
the last Tag must be #TAG_END,0

Now we come to our Buttons. All Gadget are grouped in
        TRITON
        .
There are two major kind of Groups: horizontal and vertical
aligned groups. Depending on if you want your gadgets
horizontal or vertikal aligned :) you must choose between
these groups. Of course, they can be mixed. So you can make
4 Buttons in 2 horizontal groups and put these groups into a
vertical group. Every group must be ended with the !EndGroup
macro.

```
    Groups : !HorizGroup, !VertGroup
                (arrangement of buttons: look into the include file!
                 A Group must always end with the macro !EndGroup)

    Buttons: !Button{Null("T_ext"),id}
                (Shortcuts are marked by an underscore in front of the Key ↩
                   )
```

The code looks like this:

```
    AddTags !VertGroupA
    AddTags     !Space
    AddTags     !HorizGroupA
    AddTags         !Space
    AddTags         !Button{Null("_Save"),12}
    AddTags         !Button{Null("_Cancel"),15}
    AddTags         !Space
    AddTags     !EndGroup
    AddTags     !Space
    AddTags !EndGroup
    AddTags #TAG_END,0
```

You should type your code always structured to keep the context
in mind. It’s easier to overview ;)

Now we can open our window, i.e. our project which MUST also be
a variable of long!

```
    project.l=TR_OpenProject_(application,TagList)
```

Now comes the real program:

```
    if (project)   ; only if no error occured

        close_me.b=False      ; let the window open

        while NOT close_me    ; and as long as it’s open
            TR_Wait(application,0)     ;wait for a message

            *trmsg.TR_Message=TR_GetMsg_(application)
```

```
                        ; what does the user do??

            while (*trmsg)   ; as long as it's valid

                if (*trmsg\trm_Project=project)    ; it's for our window

                  select *trmsg\trm_Class       ; which message?

                    case #TRMS_CLOSEWINDOW
                        close_me=True

                    case #TRMS_ACTION           ; a button was triggered

                      select *trmsg\trm_ID      ; which one?

                        case 1
                            ; button 1
                            ; code

                        case 2
                            ; button 2
                            ; code

                      end select

                    case #TRMS_NEWVALUE     ; check for i.e. checkboxes
                                            ; and some other gadgets which
                                            ; return this message instead
                                            ; of #TRMS_ACTION. You need it ↩
                                                only if
                                            ; you have such a gad in your  ↩
                                               GUI.
                  end select
                endif

                TR_ReplyMsg_ *trmsg     ; always reply to a msg as
                                        ; fast as possible!

                *trmsg=TR_GetMsg_ (application) ; and get the next

            wend
        wend

      TR_CloseProject_ project         ; close our window
    else
        NPrint "Unable to create project"       ; if it failed
    endif

    TR_DeleteApp_ application          ; and tell TRITON that
                                        ; our program was terminated
else

    nprint "unable to create application"   ; if it failed

endif

  Free Taglist 1              ; give the taglists memory free
```

```
     end


To see all the macros and constants, take a look at the Blitz
include file "TRITON.bb2"or at the C include file "triton.h"

To see how to do other gadgets (some send TRMS_NewValue messages!) and what
more functions the TRITON library offers, take a look at the autodoc file,
the demolistings or ask
            me
             directly.

The use of QuickHelp is shown in TOOLMANAGER1a.bb2, also the
use of Blitz2-Lists for the
            ListView
                A special case is TRITON s
            Easyrequester
            , which
replaces the system's requester.



             Go back
```

1.7 TRITON ListViews

```
You can use Blitz2-Lists for TRITON's Listviews which makes them easy
to use.

Here the source taken out from {i}TOOLMANAGER1.bb2



; ... start code snipped ....

NEWTYPE .LVItem

  num.w
  text$

End NEWTYPE

Dim List LVNodes.LVItem(9)

InitTagList 1,200

If AddItem(LVNodes())
  LVNodes()\text="2024View"
  If AddItem(LVNodes())
    LVNodes()\text="Add to archive"
    If AddItem(LVNodes())
      LVNodes()\text="Deletetool"
      If AddItem(LVNodes())
```

```
          LVNodes()\text="Edit text"
          If AddItem(LVNodes())
            LVNodes()\text="Env"
            If AddItem(LVNodes())
              LVNodes()\text="Exchange"
              If AddItem(LVNodes())
                LVNodes()\text="Multiview"
              EndIf
            EndIf
          EndIf
        EndIf
      EndIf
    EndIf
  EndIf

  ResetList LVNodes()

  ; ... application tags snipped ...

  Use TagList 1

  ; ... rest of gui snipped ....

  AddTags   !HorizGroupAC
  AddTags     !Space
  AddTags     !VertGroupAC
  AddTags       !CenteredTextID{Null("Object List"),2}
  AddTags       !Space
  AddTags       !ListSSCN{&LVNodes(0)-36,2,0,0}       ; important!! &LVNodes(0) ↩
     -36
  AddTags     !EndGroup
```

1.8 Positionflags

```
                          possible Flags are:

              #TRWP_DEFAULT
              #TRWP_BELOWTITLEBAR
              #TRWP_CENTERTOP
              #TRWP_TOPLEFTSCREEN
              #TRWP_CENTERSCREEN
              #TRWP_CENTERDISPLAY
              #TRWP_MOUSEPOINTER
              #TRWP_ABOVECOORDS
              #TRWP_BELOWCOORDS


             Go back
```

1.9 Windowflags

```
                    the flags are combined with "OR" or "|". Possible
flags are:

#TRWF_BACKDROP
#TRWF_NODRAGBAR
#TRWF_NODEPTHGADGET
#TRWF_NOCLOSEGADGET
#TRWF_NOACTIVATE
#TRWF_NOESCCLOSE
#TRWF_NOPSCRFALLBACK
#TRWF_NOZIPGADGET
#TRWF_ZIPCENTERTOP
#TRWF_NOMINTEXTWIDTH
#TRWF_NOSIZEGADGET
#TRWF_NOFONTFALLBACK
#TRWF_NODELZIP
#TRWF_SIMPLEREFRESH
#TRWF_ZIPTOCURRENTPOS
#TRWF_APPWINDOW
#TRWF_ACTIVATESTRGAD
#TRWF_HELP
#TRWF_SYSTEMACTION


 Go back
```

1.10 TRITON's easyrequester

```
The function TR_EasyRequest_ (*app,body,gads,tags) creates a requester
as the Blitz-Function Request does.

body : Null("This is the requester text")
gads : Null("Ok|Try again|Quit")
tags : you can use the following tags:

       #TREZ_ReqPos,position   : use the #TRWP_ tags for
                                positioning the requester
       #TREZ_LockProject,bool  : should the requester
                                lock the project? True or
                                false, so you needn't use
                                TR_LockProject_ every time
       #TREZ_Return
       #TREZ_Title,Null("Title")
       #TREZ_Activate,bool
```

1.11 Frequently asked questions

```
                Q: How can I change the contents of a listview?

A: That's very simple. Do it this way:
```

```
   TR_SetAttribute_ *project,id,0,NOT 0

   ; code to change list contents

   TR_SetAttribute_ *project,id,0,&List
```

Q: I have a ReturnOK-Button and a string gadget in my window. Everytime something is entered into the string gad and confirmed by return, the button is triggered. How can I avoid this?

A: Use the macro !StringGadgetNR instead of !_StringGadget.

Q: I want to use a fixed width font, how to do it?

A: Very easy: When you do your windowtags, just add this line:

```
   AddTags #TRWI_FixedWidthFontAttr,font
```

where font is initialised as font.TextAttr=Null("name.font"),size

If you just want to use the system's fixed width font, then you can go on and use the !FW... macros without setting the #TRWI_Fixed... constant.

Q: How do I get the string of a string-gadget?

A: You have to peek$ to the pointer.

```
   *text=TR_GetAttribute(*project,stringID,0)
   text$=peek$(*text)
```

Go back

1.12 Some final words

I think you got now the difference between a GadTools (or, worse, a Blitz) GUI and TRITON. But from now on you just don't need to care about fontsensitivity and calculating positions of gadgets.

You should only take the above example as a model to program a TRITON GUI. I have to admit that I didn't change the other demo listings to this way, so just take them to see how to create other gadgets or layouts. Take care of these rules:

a) use goto and/or gosub rarely in your program. This is a not very good style which should be avoided as often as possible. Blitz2 offers many possibilities for it: statements and functions.

```
    b) Every macro that has the same name as a Blitz2-Keyword
       begins with an Underscore. So the original macro
       !StringGadget{...} is named !_StringGadget{...}. If a
       TRITON macro turns yellow in your TED, just put a "_"
       in front :)

    c) Before getting a message, use TR_Wait_ *app,otherbits

    d) Always check that your project/application was opened!

    e) Reply every Message you get from TRITON.

    f) To program, use the macros - they are the easiest way
       to create a TRITON GUI. To explain all macros would
       exceed this little docu, but just take a look at either
       the Blitz2-Includes or the original C-Includes (in
       "TRITON/developer/includes/libraries"). The name of all
       constants and macros should be self-explaining. Try them
       out!

       You really should take a deep look at the Blitz2-include
       file TRITON.bb2, just to see what kinds of gadgets you
       can create and what the macro names are for!
       And you should also take a even deeper look at the autodoc
       file in TRITONs developer archive.

But always remember: Due to the TRITON Preferences Editor, the
user can not only change the look of the gadgets but also place
your window(s) on any screen he likes. So do never use fixed
coords! In fact, you do not need to size your window - the user
can change it and it will be safed in ENV: and ENVARC:, so with
every startup of your program, the window opens in the same
dimensions and coords where the user closed it last.

About this Guidefile

You should keep in mind, that this file does some system calls to show you the
include file and some sources. If you change the path or the location of this
guidefile, these calls may end in an error.



            Go back

            Authors adress
```

1.13 Contact the author

```
                If you have any suggestions, ideas or if you just need a ↩
                   little
help, contact me

    via eMail:  phips@scout.franken.de
```

in the BlitzBasic
Mailing list
via SnailMail: please understand that you can't reach me by SnailMail. Use eMail instead.

If you really don't reach me, just write to the programmer of

TRITON
, Stefan Zeiger (adress in the orig. docu!), he surely knows where I am and how you reach me - he's nearly a neighbor of mine :)

Go back

1.14 The end and the future

So I wish you happy blitzing with
TRITON
and hope to see some of your programs which use TRITON. I'd really appreciate, if you send me just a few words when you finished a program that uses TRITON, so we are able to create a TRITON-Applications-list as exists for MUI.

I included a little program called "memo" to this package - it can only be started via CLI and pops up a TRITON requester with your text in it. You need two arguments!

```
memo "Hello World!" "Remember to write Philipp!!"
```

This program could be used with CyberCron, DCron etc. for reminding.

Go back

1.15 Thanks and more go to...

Thanks go to these people (in no order):

- Rupert Henson, who helped me very much creating the macros
- D.C.J. Pink for his TagListLib
- of course to Stefan Zeiger for
  TRITON
- and to ACID Software for BlitzBasic2
- ~Irena, my girlfriend for everything
- Michael Bergmann for nice calls, interesting discussions about computers in any (im-)possible corner of the world
- ~James Savage for his trying to get TRITON to work :-)

```
        · Graham Kennedy for some nice advices for TRITON
        ·~and all the other I forgot...

    BlitzBasic2 is (c) by ACID Software     <acid@iconz.co.nz>
    TagListLib.bb2 is (c) by D.C.J. Pink   <danpink@danpink.demon.co.uk>
    Triton is (c) by Stefan Zeiger          <s.zeiger@laren.rhein-main.de>

    Legal stuff: THIS PACKAGE IS PUBLIC DOMAIN.I TAKE NO GUARANTEE FOR
        ANYTHING THAT HAPPENS TO YOU AND/OR YOUR MACHINE BY USING IT.
        BUT IF YOU CHANGE THE CODE PLEASE SEND ME A COPY OF IT SO I
        CAN ALWAYS BE ABLE TO UPDATE IT IN FUTURE RELEASES!


    keep on blizzing,
            phips@scout.franken.de
```

Go back

1.16 About TRITON

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

                                    Triton

                            An object oriented GUI creation system.

                                  (c) 1993-1995 Stefan Zeiger

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

Triton is an object oriented GUI creation system for AmigaOS. Triton makes it much easier to create good looking graphical user interfaces (GUIs) than GadTools, BOOPSI or other systems. Complicated things like resizability of windows or a fully font sensitive gadget layout are handled entirely by Triton.

Furthermore Triton GUIs can be configured by means of a Preferences editor, including e.g. a screen and a window manager for most comfortable GUI management.

There is a mailing list for discussions and questions about Triton. If you have any problems with Triton or simply want to get in touch with other developers who are using Triton, you can subscribe to the list.

In that case, send EMail to majordomo@mail.im.net with any subject and the line 'subscribe triton' in the body of your message. If you want the list mail to be sent to a different EMail address (and *only* if you want this), please use 'subscribe triton a.different@email.address' instead (after replacing 'a.different@email.address' with the address to send the mail to of

course).

In order to unsubscribe from the list, simply follow the above rules, replacing ‘subscribe’ by ‘unsubscribe’.

If you need more help, send mail to majordomo@mail.im.net with a line ‘help’ in the body.

Go back

1.17 About BlitzBasic2

```
BlitzBasic 2.1 is (c) by Acid Software
LibMan is (c) BlitzBasic Distribution Köln and
          written by Peter Eisenlohr

Subscribe to the BlitzBasic-Mailing-list, if you like:

     To: majordomo@helsinki.fi
     Subject: any
     Body:

     subscribe blitz-list

BlitzBasic-FTP sites are:

    x2ftp.oulu.fi/pub/amgiga/prog/blitz
    acid.nz.com/acid/blitz
```

1.18 Changes since release 1.0

```
I eliminated some typos. If you still find some, write me immediately!

I tried to fix the macro !ListROC{} but couldn’t finde the bug. Blitz
still reports the macro being to long. Sorry that you can’t use it, you
just have to do it "by hand".

Release 1.1

The macro !ListROC{} works now.
```

1.19 For advanced programmers

```
There is a way to make your executables a lot smaller when using
TRITON.

First, you don’t have to use Null() to pass strings to TRITON. You
could type &string$ instead as the Blitz2-Strings are all
```

```
zero-terminated. I recommend Blitz2 v1.9 at least to be sure that
this will work!
Remember then, that the strings must not be changed until they have
been used by TRITON, which is usually after TR_CreateApp_

You can also use a label reference to pass the strings. Remember,
that if you use Functions/Statements, you have to create new
labels for each Function/Statement!.

To get a view how these tricks work in reality, read the source
of the listing "memo2.bb2" which is written and commented by
Daniel Pink.
```