Triton

An object oriented GUI creation system

Release OpenTriton 2.0

1 Introduction

1.1 Introduction

Triton is a standard Amiga shared runtime library. Triton makes it much easier to create good looking graphical user interfaces (GUIs) than GadTools, BOOPSI or other systems.

The most important features are:

- Object oriented system
- Automatically font sensitive, font adaptive
- Automatic keyboard shortcuts for default window actions
- *Really* easy to use
- Beautiful customizable AmigaOS3.x look
- Comes as a freely distributable shared library
- Size! It's the Triton among the minnows of GUI creation systems ;)
- Easy installation: For a quick installation simply put triton.library into '`Libs:`' or the current directory.
- Resizability of windows wherever applicable
- A Preferences editor which allows you to customize the look and feel of all Triton GUIs
- No installation required. Simply putting triton.library into the current directory is enough for a minimum installation.

By using Triton you don't have to worry about otherwise very time-consuming things like font-sensitivity and resizability of your windows. What is even more important is that you can easily change your user interfaces later without having to rearrange display objects. Simply add an object to a group and the whole GUI will adapt to make room for it.

1.2 Copyright

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Exception to the above: The icons used in the Triton package were created and are copyrighted by Michael-Wolfgang Hohmann (mickh@iM.Net) and may not be used in other projects without his written permission. The agreement to use these icons for the Triton distribution has been given.

Note: Registered trademarks are not explicitly marked as such in this manual. If a name is not marked as a trademark, this does not imply that it is free.

1.3 Installation

First of all, Triton requires at least OS2.04. If you're still running 1.2 or 1.3, you have to upgrade to OS2.04 or better. OS1.3 is *obsolete* and I will not support it. You certainly won't regret upgrading to OS3.1.

The user relevant parts of Triton (i.e. the contents of the user archive) can be installed comfortably using the supplied installer scripts. In order to save space, the required '`Installer`' program is not included in the Triton archive. '`Installer`' is included in the AmigaOS system software beginning with OS release 3.0 and can also be found as a freely distributable archive on many BBSs and ftp sites.

If you're running OS2.0x, use the '`Install/Workbench_2.0`' installer script. If you're running OS2.1 or better, use the script for your preferred language from the '`Install/Workbench_2.1+`' drawer. Note that not all installation scripts are fully localized. Unlocalized strings will be displayed in English.

Two versions of triton.library are included in this distribution. The one in libs37 requires OS2.04 or better, the one in libs39 requires OS3.0 or better and is optimized for OS3.0+, so that it is shorter and will run slightly faster than the OS2.04 version on an OS3.0+ system. The OS3.0+ version cannot be used with OS2.0/2.1. Beginning with Triton release 2.0 the OS3.0+ version of the library also requires an 68020 or better processor and is optimized for a 68030. I assume that most Amiga users who are still interested in new software and use OS3.0 or better do also have at least a 68020 processor. Note that the V37 version will behave exactly like the V39 version on V39+ systems. It is only a bit larger and slower than the native version.

1.4 Usage

The usage of a Triton GUI should be pretty clear. All windows have two additional keyboard shortcuts (if not explicitly disabled by the application):

- 'Esc' will simulate the 'close window' gadget.
- 'Del' will simulate the 'resize window' gadget.

Furthermore, various objects react on some standard keyboard shortcuts:

- In palette, slider, scroller and similar gadgets with an up/down facility you can use the shifted shortcut to decrease the value.
- Listviews can be controlled with either the cursor keys (`Shift`ed and `Alt`ernate shortcuts are also available) or the numeric keypad (including the numeric arrow keys, `Home`, `End`, `PgUp` and `PgDn`). The programmer may choose to disable these shortcuts, mainly in order to make the primary listview accessable via the cursor keys and the secondary listview via the numeric keypad.
- Within string gadgets, `Tab` and `Shift Tab` will activate the previous/next string gadget as usual. You can access all keyboard shortcuts by pressing them together with the right Amiga key (`RAmiga`). If no such shortcut exists, Triton will instead try to interpret the key you pressed as a menu shortcut. You may use the cursor up and down keys normally from within string gadgets to control a listview.
- You can cancel a button which has been pressed down with a keyboard shortcut by pressing down either `Shift` key before releasing the shortcut. This is similar to releasing the mouse button outside the borders of a button.

For information on how to use Triton as a programmer for developing Triton applications, see the developer documentation which is included in the Triton developer archive in the directory '`Developer`'.

2 Preferences editor

2.1 Introduction

The Triton Preferences Editor allows you to customize the look and feel of applications which are using the Triton GUI creation system.

The Triton Preferences Editor allows you to change basically two kinds of settings:

1. Global settings for all Triton applications (e.g. public screens).
2. Private application settings.

Note that the listed global application `<<Global>>` belongs to the second type, too. If an application is created, it will inherit the global application's settings. Thus all applications without own settings will use the global settings. As soon as you change any setting of an application, it will become independant and will not inherit future changes of the global application. If you want to delete an application's private settings and make it use the global settings again (including inheriting future changes to them), select the menu item `Edit/Reset To Global`. This system may sound a bit complicated to you, but it is quite easy to handle.

2.2 Usage

Main window menu `Project`:

The Triton Preferences Editor can be installed by hand by simply dragging it into the 'Prefs' drawer of your boot partition ('`sys:Prefs`'). Then start the editor and confirm that you want to install the Preferences System. It may be removed again later by selecting 'Project/Remove' in the editor's main window's menu.

Main window menu `Edit`:

This menu contains all standard items plus `Reset to Global` (see Section 2.1 [Preferences editor - Introduction], page 5). All changes made from within this menu do only apply to the currently selected application.

Main window gadgets:

`Save`, `Use` and `Cancel` should be self-explaining. They apply to all modified settings of all applications. `Test` will restart the Preferences editor using the current settings without saving anything.

`Info` will give you all information available about the selected application. This includes name, version, compilation date and so on.

`Delete` will delete an application, i.e. its settings and information files. You have to restart the application to make it appear in the Triton Preferences Editor again. If you just want to delete an application's private settings, use `Edit/Reset to Global`.

The gadget `System` opens the system window which contains all global settings. All other gadgets open windows for private application settings.

`Frames` window:

The frames window allows you to configure the look of frames and groups. This should be mostly self-explaining.

`Images` window:

You can change some default Triton images (e.g. window backgrounds) in this window.

With `Use <Return> arrow in buttons` you can decide how buttons which can be activated by pressing `<Return>` will be highlighted. If you check the gadget, these buttons will have an arrow on the right side. By default they will have their label displayed in **bold** text instead.

`Pens` window:

In this window you can change some Triton pens. Especially important are `Halfshine` and `Halfshadow` which are used for rendering frames in `XEN 1`, `XEN 2` and `NeXT` look.

Also listed among these pens are the foregrounds and backgrounds of string gadgets. From a logical point of view the backgrounds should rather be images than pens but due to limitations of AmigaOS this is technically not possible.

`Windows` window:

This window contains the Triton window manager. The list shows all windows of the selected application which have been opened since the last reboot or are saved permanently to '`envarc:Triton`'. The windows are displayed by their titles (if available; otherwise the ID number in square brackets). If you close a Triton window, Triton will remember its last dimensions

in a file in '`env:Triton`', so that it will have the same dimensions when you reopen it. If you want to keep the window dimensions permanently, so that they are always available (unless you change them later on), select the appropriate window and click on `Snapshot`. This system works just like the window snapshotting of the Amiga Workbench. If you want Triton to "forget" a custom window position and reposition the window according to the original positioning information (e.g. centering the window on the screen after choosing a new screen resolution), you can `UnSnapshot` the window. The gadgets `Snapshot all` and `UnSnapshot all` will (un)snapshot all windows of the currently selected application. The gadget `Info` will bring up a requester containing some information about the currently selected window.

`System` window:

The system window contains all global settings. The most important one of those is the public screen manager. You can add, delete and edit public screens. Double click on a screen or select `Use` and the name of the screen will be copied into the above string gadget. This is the screen for the currently selected application. You may also type in any other screen name (either of an existing public screen or of a public screen which is managed by the Triton public screen manager).

When Triton opens a window, it will first look if the specified screen exists and use it in this case. Otherwise it will try to create it if an entry for this screen can be found in the screen manager. If this doesn't work either, Triton will fall back onto the default public screen. You can test how a screen will look by selecting `Test`. The screen will be opened and can be closed again by closing the information window which is displayed in it. Note that the screen will not be made public when testing.

The screen manager settings override any specifications made by the programmer. Setting an application's screen to `<<Program default>>` will make it use the screen specified by the programmer. However only public and workbench screen windows can be promoted using the screen manager. Custom screen windows will always open on the custom screen specified in the application.

Except for the public screen manager, the `System` window currently contains only one switch: `Simple refresh`. Select it to make all windows use `Simple` instead of `Smart` refresh. This will require less memory but make the refresh slower und worse looking (at least with the standard AGA graphics - I haven't tried it with a graphics board).

3 Odds & Ends

3.1 Astronomy lesson

Triton - A moon of Neptune
(by Michael Berg)

Triton is a quaint little moon, in that it is one of the very few moons to be known to have retrograde orbits around their host. Triton is believed to be a 3-6000 km large world of mostly liquid nitrogen oceans, perhaps with a thin methane atmosphere as well, but noone knows for sure since either of the Voyager twins ever got close enough to take convincing snapshots.

Neptune itself is also quite strange in that it has an almost 90 degree tilted inclination, so that it "rolls" through space, unlike Earth and the other planets (minus Venus), which all spin like tops in their orbits. Noone can tell for sure why Neptune has such an eccentric inclination, but it has been suggested that it was caused by a gigantic collision of some sort (probably with a large asteroid). In the process Neptune also lost its third moon, which at the time was none other than Pluto. The collision accelerated Pluto to the point where it had enough momentum to actually leave its orbit and shoot into space on its own. As it did so it probably passed below the Roche' radius of Neptune, causing it to break up in two or more bodies under the influence of the massive tidal forces of Neptune. This would certainly account for Charon, which is in itself too large to have been "captured" by an object as small as Pluto.

Okay, end of today's astronomy lesson :-)

3.2 Acknowledgments

Thanks must go to (in more or less random order):

- The Triton 2.0 beta-testers:

  Magnus Holmgren, Nils Sjoholm, Volker Stolz, Patrick Ohly, Jürgen Kohrmeyer, Carsten Raufuß, Matthias Andree, Heikki Linnakangas, Stefan 'eau' Schulz, Michael Bergmann, Klaus Melchior, Philipp Lonke, Tom Eicher, Uffe Holst, Marco Frischkorn
- The Triton 1.x beta-testers:

  Nico François, Carsten Raufuß, Michael Berg, Magnus Holmgren, Marco Frischkorn, Patrick Ohly, Daniel Schrod and everyone else I have forgotten
- The catalog translators:

  Philipp Lonke for the French translation and the BlitzBasic interface Magnus Holmgren for the Swedish translation and the DICE interface Frank Verheyen for the Dutch translation and the AmigaE interface

- The authors of developer support material:

  Philipp Lonke (BlitzBasic), Magnus Holmgren (DICE), Frank Verheyen (AmigaE), Gunther Nikl (gcc), Klaus Melchior (DICE), Peter Fröhlich (AmigaOberon), Volker Stolz (AmigaOberon), Oskar Liljeblad (Assembler), Stefan Schulz (M2Amiga), Sotirios Pappas (KickPascal/MaxonPascal),
- Jürgen Kohrmeyer for writing `TritonRexx`
- Kai Iske for his BOOPSI class sources
- Michael Berg for the 'Astronomy Lesson'
- Tom Eicher and Everett M. Greene for proof-reading the documentation
- Michael 'Mick' Hohmann for the icons
- The people at iM.Net for running the Triton mailing list
- Everyone I've forgotten

I have used the following software products for creating Triton:

- The C sources were compiled with *SAS/C 6.56*. With SAS/C it was possible to create Triton without a single line of assembly language code.
- The sources and the documentation were edited with *GNU Emacs 18.59* on my Amiga and *XEmacs 19.13* on my Linux PC.
- The documentation was written in Texinfo and formatted using *PasTeX* and *makeinfo*.
- Shell commands were entered in *KingCON* and *xterm* windows ;-)

3.3 Support

If you have suggestions or remarks about Triton, or if you find any bugs, please let me know.

Contacting the author:

- **EMail:**
  szeiger@usa.net
- **Mail:**
  Stefan Zeiger
  Seligenstädter Weg 24
  D-63796 Kahl
  Germany
- **Voice:**
  +49-6188-900712

You can also find this information on my WWW home page at `http://home.pages.de/~szeiger/`.

3.4 Updates

If you have access to the World Wide Web (WWW), the easiest way of obtaining a new version of Triton or getting information about Triton is the Triton Home Page at `http://home.pages.de/~szeiger/triton.html`. You can follow links from this page to AmiNet ftp sites that carry the latest Triton archives.

Glossary

'BOOPSI' Basic Object Oriented Programming System for Intuition. See the BOOPSI chapter of the RKM ("ROM Kernel Manual") volume "Libraries" for details.

'DICE' Dillon's Integrated C Environment. A C development system for AmigaOS.

'Endless loop'
See 'Loop, endless'.

'FD software'
Freely distributable software, including 'Public Domain (PD)', 'Freeware', 'Giftware', 'Shareware', etc.

'GCC' 'GNU C': The C compiler of the "Free Software Foundation". This compiler and its development system is freely distributable and a port is also available for AmigaOS.

'GUI' Graphical User Interface.

'Loop, endless'
See 'Endless loop'.

'SAS/C' A commercial C development system for AmigaOS.

Index

Table of Contents