Tko – oo cget/configure and oo widgets

- History

- Implementation
  - Tcl and C implementation
  - Creation of new widgets

- Usage in simulator (2015..)
  - Build process
  - Demo with tko::graph (adopted from rbc::graph) and tko::path (adopted from tkpath)

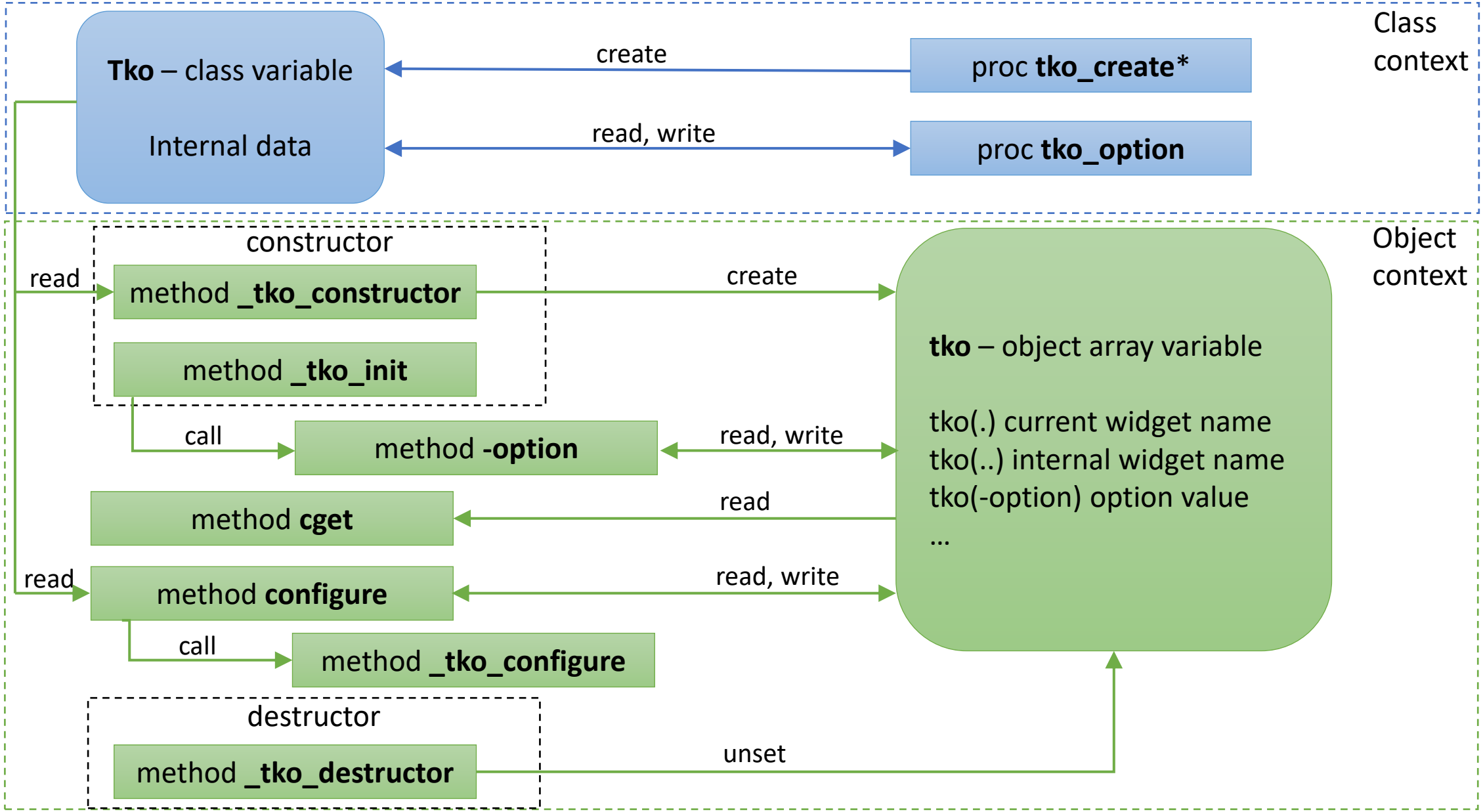
- Further ideas

René Zaumseil <r.zaumseil@freenet.de> @ EuroTcl 2024

History

- Tip 510 Add Rbc and Tkpath widgets to Tk (2018)
  - Branch at https://core.tcl-lang.org/tk/timeline?r=tip-510
  - Presentation at https://ssl.webpack.de/www.eurotcl.eu/presentations/EuroTcl2018-Zaumseil-WorkingOnTk.pdf
- Tip 556 Add oo like widgets to Tk (2019)
  - Branch at https://core.tcl-lang.org/tk/timeline?r=tip-556
  - Presentation at https://ssl.webpack.de/www.eurotcl.eu/presentations/EuroTcl2019-Zaumseil-oo_class_widgets.pdf
- Package tko
  - Repository at https://chiselapp.com/user/rene/repository/tko
  - 2020 start from tip 556
  - 2023 change to tcl implementation (tko.tcl)
  - 2024 tcl/tk9 ready
    - C changes
      - obc, objv arguments (int -> Tcl_Size)
      - Tk_ConfigureWidget()
    - Changes in application code
      - source –encoding cp1252
      - open & fconfigure –encoding cp1252

Implementation – overview



Implementation – creation functions

	tko_create class	tko_create superclass	tko_create widget	tko_create superwidget
mixin ::tko::mixin	x	x	x	x
variable tko	x	x	x	x
method _tko_configure	x		x	
Internal class variable Tko	x	x	x	x
self unexport create createWithNamespace destroy new			x	x
method unknown ?args?			x	x
superclass –append tkoclass|tkowidget		x		x

tko::widget tkowidget widget ?usepatterns? ?ignorelist?

- tkowidget – name of new widget command, p.e. mywidget
- widget – name of widget command to wrap, p.e. ttk::button
- usepattern – list of option patterns to wrap, p.e. {-class –t*}
- ignorelist – list of options to ignore, p.e. {-textvariable –takefocus}

Implementation – option definition

- tko_option

  Return current option definitions from internal Tko class variable

- tko_option –option

  Return definition of given option from internal Tko class variable

- tko_option –option –synonym

  Define –option for usage in cget / configure calls. Only tko(-synonym) exists.

- tko_option –option dbname dbclass default body

  Create new tk-like -option. If dbname is not empty then use dbname and dbclass to initialize this option from the option database. The value of body will be used to define the –option method. This method will be called in the my _tko_init call in the constructor and in the configure method.

- tko_option –option dbname dbclass default body startbody

  Create new tk-like -option. If dbname is not empty then use dbname and dbclass to initialize this option from the option database. The value of startbody will be used to define the –option method and this method will be called in the my _tko_init call in the constructor and in the configure method.

  In the my _tkoinit call the body will be used to redefine the –option method. Using "error readonly" as body will create readonly options.

Implementation – C coding

- tkoWidget.h
  - enum Tko_WidgetOptionType {..} – common option types
  - struct Tko_WidgetOptionDefine {..} – definition of options
  - struct Tko_Widget {..} – common widget data used in objects
- tkoWidget.c
  - int Tko_WidgetClassDefine(..) – Create a new widget class
  - int Tko_WidgetCreate(..) – Create new widget object (my _tko_constructor in C)
  - int Tko_WidgetOptionInit(..) – Call "my _tko_init classname".
  - void Tko_WidgetDestroy(..) – Delete widget window, command and ressources
  - ClientData Tko_WidgetClientData(..) – Return pointer to widget client data
  - Tcl_Obj *Tko_WidgetOptionGet(..) – Return TclObj value of option or NULL if widget is destroyed
  - Tcl_Obj *Tko_WidgetOptionSet(..) – Set option value
- tkoFrame.c
  - oo version of frame, labelframe and toplevel
- tkoGraph.c
  - tko::graph widget (oo version of rbc::graph)
- tkoPath.c
  - tko::path widget (oo version of tkpath)

Implementation – new widget

```
# example code from tko_dialog.tcl
set ::tko::dialog toplevel
::oo::class create ::tko::dialog {
    ::tko_createwidget
    if {[catch {private variable _tko}]} {variable _tko}
}
# readonly option
::oo::define ::tko::dialog ::tko_option -class class Class TkoDialog {error readonly} {}
# normal option
::oo::define ::tko::dialog ::tko_option -title title Title {} {
    if {$::tko::dialog eq {toplevel}} {
        wm title $tko(.) $tko(-title)
    } else {
        $_tko(.h) configure -text $tko(-title)
    }
}
#
::oo::define ::tko::dialog constructor {args} {
    my _tko_constructor $args $::tko::dialog {-class $tko(-class)}
    #...
    my _tko_init
}
#
::oo::define ::tko::dialog destructor {
    my _tko_destructor
}
# widget method
::oo::define ::tko::dialog method deactivate {{returnvalue {}}} {
     #..
}
```

Implementation – widget inheritance

```
# example code from tko_dialogbox.tcl
::oo::class create ::tko::dialogbox {
    ::tko_createsuperwidget ::tko::dialog
    if {[catch {private variable _tko}]} {variable _tko}
}
# normal option
::oo::define ::tko::dialogbox ::tko_option -message message Message {} {}
#
::oo::define ::tko::dialogbox constructor {args} {
    next -class TkoDialogbox {*}$args
    #..
    my _tko_init
}
# widget method
::oo::define ::tko::dialogbox method deactivate {{returnvalue {}}} {
    next $returnvalue
    #..
}
```

oo widgets – performance

command	create	cget -width	configure -width	configure -width 100	destroy
::ooclass	6.8	1.4	1.3	1.2	2.1
::tkoclass	26.4	2.1	13.5	5.8	3.4
::frame	4.6	0.8	1.0	11.9	175.4
::ttk::frame	184.3	0.9	2.1	5.8	48.0
::tko::frame	87.9	3.0	11.9	29.5	178.0
::snit::frame	93.0	32.2	37.8	184.7	190.2
::wrapframe	453.2	3.0	16.7	28.8	188.5
::wrapframe0	258.9	2.2	15.7	6.1	173.9
::labelframe	5.3	0.8	1.2	14.8	146.9
::ttk::labelframe	191.6	0.9	9.3	8.1	45.3
::tko::labelframe	95.8	2.9	12.1	33.9	182.2
::snit::labelframe	94.7	32.9	38.5	210.4	200.6
::wraplabelframe	506.6	3.1	17.5	32.5	193.2
::wraplabelframe0	262.3	2.1	16.3	6.1	176.4
::toplevel	6.2	0.8	1.1	14.0	467.5
::tko::toplevel	100.0	3.0	11.9	32.1	461.3
::snit::toplevel	96.5	32.4	39.2	210.3	498.7
::wraptoplevel	501.9	3.2	17.1	32.3	491.2
::wraptoplevel0	265.4	2.2	15.8	6.2	487.4
::tko::path	130.6	3.2	11.9	41.1	169.8
::tko::graph	2153.9	3.8	13.4	56.8	64.5

```
# widget creation code
::tko::widget ::wrapframe frame
::tko::widget ::wrapframe0 frame {-width}
::tko::widget ::wraplabelframe labelframe
::tko::widget ::wraplabelframe0 labelframe {-width}
::tko::widget ::wraptoplevel toplevel
::tko::widget ::wraptoplevel0 toplevel {-width}
::oo::class create ::ooclass {
    method configure {args} {}
    method cget {args} {}
    self method unknown {args} {
        tailcall ::ooclass create $args}
}
::oo::class create ::tkoclass {
    tko_createclass
    tko_option -width {} {} 0 {}
    constructor {args} {
        my _tko_constructor $args; my _tko_init}
    self method unknown {args} {
        tailcall ::tkoclass create $args}
}
::snit::widget ::snit::frame {
    hulltype ::frame
    delegate option * to hull
}
::snit::widget ::snit::labelframe {
    hulltype ::labelframe
    delegate option * to hull
}
::snit::widget ::snit::toplevel {
    hulltype ::toplevel
    delegate option * to hull
}
```

Usage in simulator

- Build executable
  - build.bat unpack and install msys/mingw and call zipkit.sh local
  - zipkit.sh from http://chiselapp.com/user/rene/repository/zipkit
  - Build tcl/tk with **–disable-shared**
  - Build other extensions
  - Copy results in app.vfs directory
  - Run in app.vfs directory: **zip –r –q .../wish.exe ***

Live demonstration

Further ideas

- tko::path: new pdf method to get complete pdf content instead of itempdf calls
- tko::graph: change Tk_ConfigureWidget with Tk_SetOptions
- tko::graph: pdf output
- Sdl support (p.e. sdltk in androwish)

Questions?