René Zaumseil r.zaumseil@freenet.de

Tcl tip: https://core.tcl.tk/tips/doc/trunk/tip/510.md

Tk oo::class like widgets

Functionality

Widget methods

Widget creation in C and Tcl

Examples

Related Tips

Open issues

Functionality

- Widgets as oo::class’s, can be used as superclass
  - tko::widget -- internal base class, used in C widget creation
  - tko::frame, tko::labelframe, tko::toplevel -- same functionality as tk widgets
  - graph -- rbc::graph widget
  - path -- tkpath widget
- Same syntax as tk/ttk widgets
  - Widget creation: **widgetName** *pathName ?options?*
  - Widget command: *pathName* **method** *args*
- cget/configure methods
- Unknown method for new widget classes (in $::tko::unknown)
- Dynamic options at class and object level
- C interface

Widget methods

- Already existing functionality:

```
cget –option
configure ?-option value …?
```

- New option related enhancements:

```
configure optionadd –synonym -option
configure optionadd –option dbname dbclass ?default? ?flags?
configure optiondel -option
configure optionhide ?-option …?
configure optionshow ?-option …?
configure optionvar
```

- Initialize options after widget creation (internal):

```
configure init
```

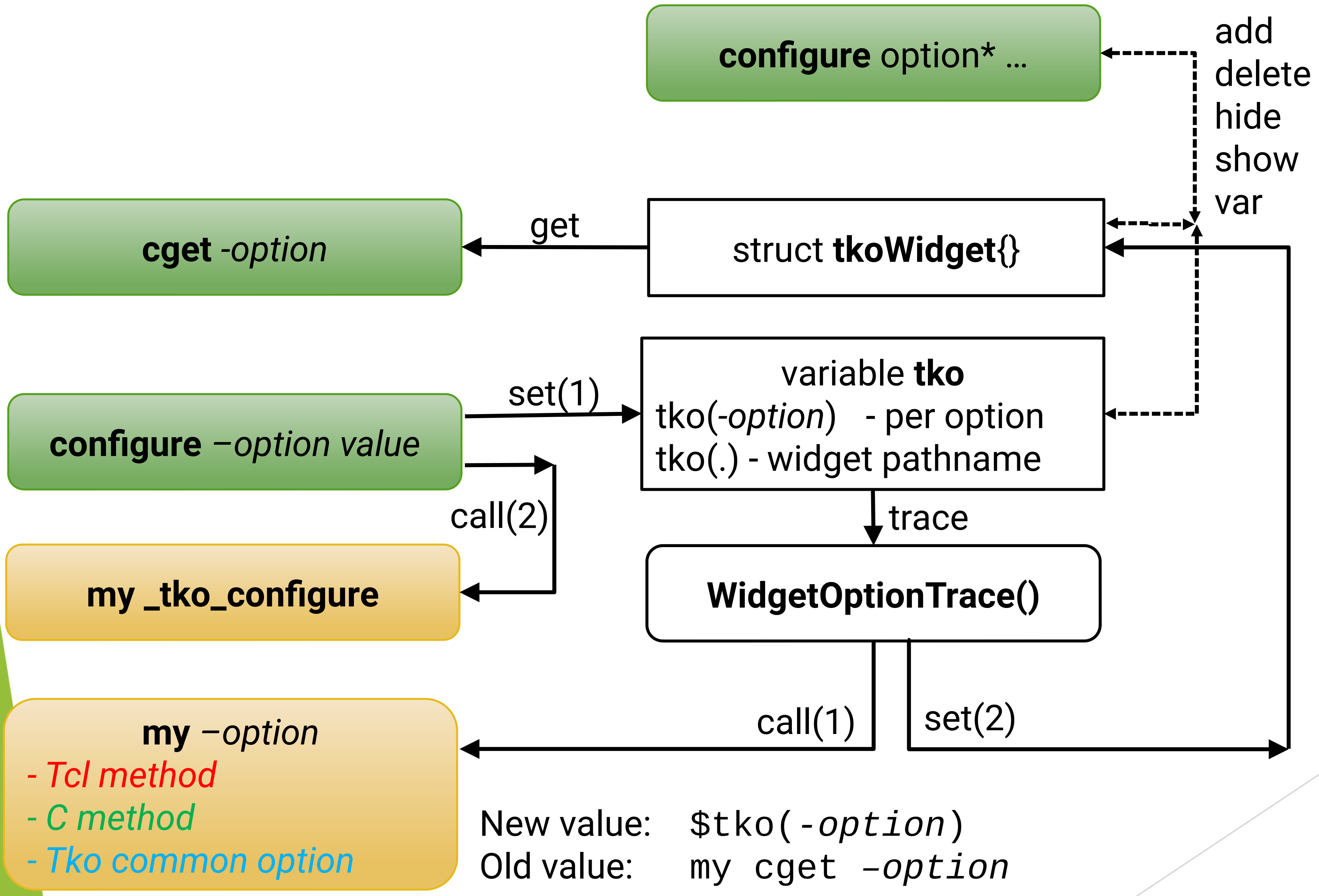
- One method per option, called after option changes:

```
-option
```

- Constructor method, will be called from unknown method provided in $::tko::unknown

```
constructor optionlist arglist
```

Widget methods working



New value: `$tko(-option)`
Old value: `my cget -option`

Widget option typedefs

- Available common option setting types

```
typedef enum tkoWidgetOptionType {
    TKO_SET_CLASS = 1,     /* (Tcl_Obj **)address */
…
    TKO_SET_JUSTIFY /* (Tk_Justify *)address */
} tkoWidgetOptionType;
```

- Option definition structure

```
typedef struct tkoWidgetOptionDefine {
  const char *option;    /* Name of option. Starts with "-" minus sign */
  const char *dbname;    /* Option DB name or synonym option if dbclass is NULL */
  const char *dbclass;   /* Option DB class name or NULL for synonym options. */
  const char *defvalue;  /* Default value. */
  int flags;             /* bit array of TKO_OPTION_* values to configure option
*/
  Tcl_Obj *optionPtr;    /* tko internally used, always init with NULL! */
  const char *proc;      /* If not NULL it is the body of the new -option method
*/
  Tcl_MethodCallProc *method;/* If not NULL it is the C-function name to call */
  tkoWidgetOptionType type;/* option type used in common option set method */
  Tcl_ObjectMetadataType *meta;/* meta data used in common option set method */
  int offset;            /* offset in meta data struct */
} tkoWidgetOptionDefine;
#define TKO_OPTION_READONLY 0x1 /* option is only setable at creation time */
```

Widget method and option data

- Class methods: constructor, destructor, public …, delimeter, private …, delimeter

```
static Tcl_MethodType frameMethods[] = {
    {TCL_OO_METHOD_VERSION_CURRENT, NULL, FrameConstructorFrame, NULL, NULL},
    {TCL_OO_METHOD_VERSION_CURRENT, NULL, FrameDestructor, NULL, NULL},
    {-1, NULL, NULL, NULL, NULL},
    {TCL_OO_METHOD_VERSION_CURRENT, "_tko_configure", FrameMethod_tko_configure,
            NULL, NULL},
    {-1, NULL, NULL, NULL, NULL}};
```

- Class options: processed fifo, -class should be first!

```
static tkoWidgetOptionDefine labelframeOptions[] = {
  {"-class", "class", "Class", "TkoLabelframe", NULL,
   NULL, TKO_OPTION_READONLY, TKO_SET_CLASS, NULL, 0},
…
  {"-borderwidth", "borderWidth", "BorderWidth", DEF_FRAME_BORDER_WIDTH, 0,NULL,
   NULL, NULL, TKO_SET_PIXEL, &frameMeta, offsetof(tkoFrame, borderWidth)},
…
  {"-bd" , "-borderwidth", NULL, NULL, 0, NULL, NULL, NULL,0,NULL,0},
…
  {"-labelwidget", "labelWidget", "LabelWidget", "",0, NULL,
   NULL, FrameMethod_labelwidget, 0, NULL, 0},
…
  {NULL,NULL,NULL,NULL,NULL,NULL,0,0,NULL,0}};
```

Tko widget interface functions

- Add methods and option at class initialization

```
int TkoWidgetClassDefine(Tcl_Interp *interp,
      Tcl_Class clazz, Tcl_Obj *classname,
      const Tcl_MethodType *methods,
      tkoWidgetOptionDefine *options);
```

- Return internal Tk_Window, If NULL no window exists. If *Tk_Window is NULL window is deleted

```
Tk_Window *TkoWidgetWindow(Tcl_Object object);
```

- Return global name of tko() option array or NULL if it not exists

```
Tcl_Obj *TkoWidgetOptionVar(Tcl_Object object);
```

- Return current value of given option or NULL if it not exists

```
Tcl_Obj *TkoWidgetOptionGet(Tcl_Interp *interp,
      Tcl_Object object, Tcl_Obj *option);
```

C widget class creation

(1) Create new class

```
static const char *initScript =
     "::oo::class create ::tko::frame {"
     "  superclass ::tko::widget;"
     "  variable tko;"
     "  {*}$::tko::unknown }";
   Tcl_GlobalEval(interp, initScript);
```

(2) Add class methods and options

```
if((object=Tcl_GetObjectFromObj(interp, TkoObj.tko_frame)) == NULL
   || (clazz=Tcl_GetObjectAsClass(object)) == NULL) {
      return TCL_ERROR;
   }
   if(TkoWidgetClassDefine(interp, clazz,
      Tcl_GetObjectName(interp, object),
      frameMethods, frameOptions) != TCL_OK) {
      return TCL_ERROR;
   }
```

C widget constructor

```
static Tcl_ObjectMetadataType pathMeta ={
    TCL_OO_METADATA_VERSION_CURRENT,
     "PathMeta",
     PathMetaDelete,
     NULL;
```

(1) Check correct calling

```
if((object = Tcl_ObjectContextObject(context)) == NULL) return
TCL_ERROR;
    skip = Tcl_ObjectContextSkippedArgs(context);
    /* Check objv[] arguments: ... optionlist arglist */
     if(objc - skip != 2) return TCL_ERROR
```

(2) Create and initialize widget structure

```
path = (TkPathCanvas *) ckalloc(sizeof(TkPathCanvas));
```

(3) Set object meta data

```
Tcl_ObjectSetMetadata(object,&pathMeta,(ClientData) path);
```

(4) Insert own options in parameter optionlist

(5) Call next constructor

```
Tcl_ObjectContextInvokeNext(interp,context,objc,objv,skip);
```

(6) Get and check internal Tk_Window

```
path->win = TkoWidgetWindow(object);
    if(path->win == NULL || *(path->win) == NULL) return TCL_ERROR;
```

Widget creation

oo::class create <widget>

superclass ..
variable tko
method unknown {args} ;# <= $tko::unknown
method –*option* {}
method _tko_configure {}

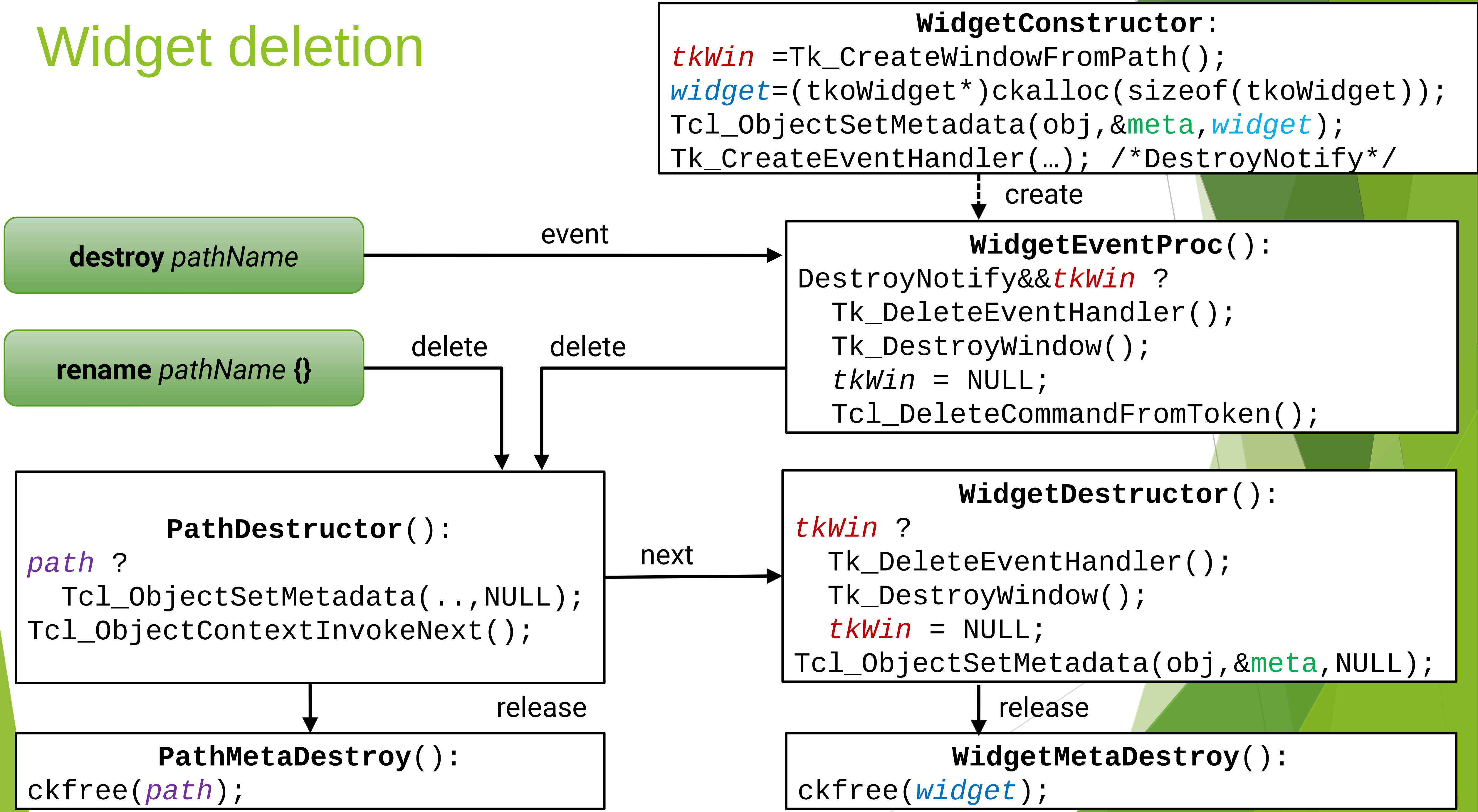
<widget> *pathName*

constructor {optlist arglist} ;# see optionadd
destructor

```
              WidgetConstructor:
tkWin =Tk_CreateWindowFromPath();
widget=(tkoWidget*)ckalloc(sizeof(tkoWidget));
Tcl_ObjectSetMetadata(obj,&meta,widget);
WidgetOptionAdd(..); /*for each option*/
Tk_CreateEventHandler(..); /*DestroyNotify*/
Tcl_TraceVar2(..); /*write trace on variable tko*/
```

```
           WidgetOptionAdd:
- Add option
- Set option default value in tko array
- Call –option method when readonly
```

Widget deletion



destroy *pathName*

rename *pathName* **{}**

```
WidgetConstructor:
tkWin =Tk_CreateWindowFromPath();
widget=(tkoWidget*)ckalloc(sizeof(tkoWidget));
Tcl_ObjectSetMetadata(obj,&meta,widget);
Tk_CreateEventHandler(…); /*DestroyNotify*/
```

```
WidgetEventProc():
DestroyNotify&&tkWin ?
  Tk_DeleteEventHandler();
  Tk_DestroyWindow();
  tkWin = NULL;
  Tcl_DeleteCommandFromToken();
```

```
PathDestructor():
path ?
  Tcl_ObjectSetMetadata(..,NULL);
Tcl_ObjectContextInvokeNext();
```

```
WidgetDestructor():
tkWin ?
  Tk_DeleteEventHandler();
  Tk_DestroyWindow();
  tkWin = NULL;
Tcl_ObjectSetMetadata(obj,&meta,NULL);
```

```
PathMetaDestroy():
ckfree(path);
```

```
WidgetMetaDestroy():
ckfree(widget);
```

Widget ressources

```
proc Do {script} {set i 0; format %7.1f [lindex [time $script 1000] 0]}
proc Test {cmd} {set ret [Do "$cmd .\[incr i\]"]
  append ret [Do {.[incr i] cget -bg}]
  append ret [Do {.[incr i] cget -width}]
  append ret [Do {.[incr i] configure -bg red}]
  append ret [Do {.[incr i] configure -width 100}]
  append ret [Do {.[incr i] configure}]
  append ret [Do {destroy .[incr i]}]
}
```

Test <cmd>	create	cget		configure			destroy
frame	3.5	0.6	0.6	0.6	0.6	9.0	165.2
tko::frame	225.5	1.1	0.8	2.6	2.7	10.0	58.4
labelframe	4.2	0.6	0.6	0.7	0.7	11.9	166.2
tko::labelframe	264.0	1.3	0.8	3.4	3.1	12.9	76.6
toplevel	5.2	0.6	0.6	0.6	0.7	10.4	466.7
tko::toplevel	387.2	1.4	0.8	3.1	2.9	11.4	147.9

Memory:	1000 * ::frame	= 0.6 MB
	1000 * ::tko::frame	= 5 MB

Tcl widget extending

- Add class methods

```
oo::define tko::frame method classmethod {} {puts =class}
  tko::frame .f
  .f classmethod ;# ▯ =class
  oo::define tko::frame deletemethod classmethod
  .f classmethod ;# ▯ error
```

- Add object methods

```
oo::objdefine .f method objmethod {} {puts =object}
.f objmethod ;# ▯ =object
tko::frame .f1
.f1 objmethod ;# ▯ error
```

Tcl widget class creation

- Create own widget class with new options

```
oo::class create myframe {
    superclass ::tko::frame; variable tko; {*}$::tko::unknown
    constructor {optlist arglist} {
      next {{-opt opt OPT opt1} {-readonly ro RO ro1 1} {-o –opt}} $arglist
    }
    destructor {puts DES; next}
    method –opt {} {puts [my cget –opt]->$tko(-opt)}
    method –readonly {} {puts never}
    method –background {} {if {$tko(-background) eq {red} error; next}
}
```

- Testing

```
myframe .f
.f configure –o v2 ;# ▯ v1->v2
.f cget –readonly ;# ▯ ro1
.f configure –readonly ro2 ;# ▯ error
.f configure –background red ;# ▯ error
.f configure optionhide {*}[.f configure optionshow] ;# ▯
.f configure optionshow –class –relief ;# ▯ -class –relief
.f configure optiondel –relief ;# ▯ -class
destroy .f ;# ▯ DES
```

Tcl widgets object options

- Create oo::class widget object

```
tko::frame .f
```

- Add normal object option

```
 oo::objdefine .f method -opt {} {
    variable tko
    puts [my cget -opt]->$tko(-opt)
  }
  .f configure optionadd -opt opt OPT v1
  .f configure -opt v2 ;#  v1->v2
```

- Add readonly object option

```
oo::objdefine .f method -readonly {} {puts -opt}
.f configure optionadd -readonly ro Ro ro1 1
.f cget -readonly ;#  ro1
.f configure -readonly ro2 ;#  error
```

- Remove object options

```
.f configure optiondel -opt
.f configure -opt;#  -error
.f configure optiondel -readonly
.f configure -readonly;#  -error
```

Related Tips

- Tip 369: Widget cargo command

```
# at class level
oo::define tko::widget variable cargo
oo::define tko::widget method cargo {mode args} {
  switch -- $mode {
    set {dict set cargo {*}$args}
    unset {dict unset cargo {*}$args}
    get {dict get $cargo {*}$args}
  }
}
# at object level
oo::objdefine .w variable cargo
oo::objdefine .w method cargo {mode args} {
  my variable cargo
  switch -- $mode {
    set {dict set cargo {*}$args}
    unset {dict unset cargo {*}$args}
    get {dict get $cargo {*}$args}
  }
}
```

- Tip 349: New «cargo» option for tk widgets
- Tip 180: Add a Megawidget Support Core Package

Open issues

- Tip 510 Open issues for path/graph widget problems
  - Mac implementation
  - Platform usage (SDL, GDI+ Cairo) and configure support
  - Change old code (see Discussion topics)
- Split tip 510 in tko class and graph+path parts?
- Tko syntax
  - `configure option* …`
  - `configure init`
  - Class option definition
  - Component handling?
- Related issues
  - Handling of unique abbreviations of oo methods (oo::class Donal?)
  - oo::class cget/configure
  - Other option database (sqlite3, themes)
  - Using fossil md format for man pages

Questions?

- René Zaumseil r.zaumseil@freenet.de
- Tcl tip: https://core.tcl.tk/tips/doc/trunk/tip/510.md