Tk mini-tutorial - mbl@lri.fr

Widgets - Les widgets de Tk forment une hiérarchie dont la racine est nommée ".". Chaque widget a nom qui est son chemin d'accès dans la hiérarchie. Par exemple le chemin ".menubar.edit.menu.col désigne l'item "coller" du menu "menu" associé au bouton "edit" de la barre de menus "menubar". P créer un widget, il faut que son widget parent ait été créé préalablement. Le widget racine "." est créé lancement de wish. A l'exception des menus et des widgets de type "toplevel", un widget est touj géométriquement inclus dans son parent.

Création des widgets - Pour chaque type de widget il existe une commande Tk permettant de crée nouveau widget de ce type. Par exemple :

```
button .ok -text OK -command {puts "d'accord"}
```

crée un widget de type bouton et de nom ".ok". "-text" et "-command" sont des *options de configura* du widget, qui spécifient respectivement la chaîne de caractères qui apparait dans le bouton et le script qui sera exécuté lorsque l'on activera le bouton.

Il est recommandé d'éviter l'usage des noms complets des widgets pour faciliter l'évolution e réutilisation du code. Pour cela, on passe souvent en paramètre à une procédure qui crée des widget nom du widget parent. Par exemple :

```
proc CréerOkCancel {top} {
        button $top.ok -text OK ...
        button $top.cancel -text Cancel ...
        . . .
}
```

Une fois créé, un widget n'est pas immédiatement visible à l'écran : il faut le confier à un *gestionnair géométrie*. Tk a trois gestionnaires de géométrie qui sont accessibles par les commandes place, pack et (voir en page 5). La façon la plus simple de rendre un widget visible est d'utiliser le packer :

```
pack .ok
```

Comme la commande de création d'un widget retourne toujours le nom du widget, on peut écrire de fa plus concise :

```
pack [button .ok -text OK -command {puts "d'accord"}]
```

Manipulation des widgets - Une fois créé, un widget peut être modifié de deux façons. D'une par nom du widget est utilisable comme une commande Tcl. Cela permet de manipuler le widget dans un s "orienté-objet" par des commandes de la forme :

```
nom-du-widget méthode arguments
```

Par exemple :

```
.ok configure -command {puts "coucou"}
set nom [.ok cget -text]
.ok invoke
```

La méthode "configure" permet de changer des options de configuration après la création du widget. méthode "cget" permet de récupérer la valeur d'une option de configuration. La méthode "invoke" valide pour un bouton et permet d'activer un bouton par programme. L'effet est le même que si l'utilisat cliquait sur le bouton.

D'autres commandes de manipulation suivent la syntaxe traditionnelle de Tcl :

destroy w	détruire un widget (et son sous-arbre)
raise w / lower w	changer l'ordre de superposition des widgets

La commande winfo permet d'obtenir des informations générales sur les widgets :

winfo children w	retourne la liste des noms des fils du widget w
winfo exist w	indique si le widget w existe (a été créé et mais pas encore détruit)
winfo parent w	retourne le nom du parent de w
winfo name w	retourne le dernier composant du nom de w
winfo x |y | width | height w	retourne la position/taille de w

Options générales - Les options de configurations sont utilisées lors de la création d'un widget et d les méthodes configure et cget d'un widget. Beaucoup d'options de configuration sont communes à grand nombre de widgets. Voici les principales :

-bg couleur	couleur de fond du widget. Une couleur peut être spécifiée par un nom (red, green, ...) ou par une valeur RGB hexa : #rrggbb
-fg couleur	couleur du contenu du widget (par exemple son texte)
-bd nn	épaisseur du bord (doit être non nul pour que -relief ait un effet)
-relief raised | sunken | flat | ridge | solid | groove	aspect 3D du cadre
-font police	police de caractères pour l'affichage du texte dans un widget. Une police est créée par la commande Tk font.

Boutons - commandes label, button, checkbutton, radiobutton

Les widgets de la famille bouton ont en commun d'afficher un texte (-text, -textvariable) ou une imag image). Ils peuvent être actifs ou inactifs (-state). Les labels sont passifs, les buttons permettent déclencher une action (-command), les checkbuttons et les radiobuttons affichent un état (-variable, -va -onvalue, -offvalue). Voici les principales options :

-text label	texte affiché dans le bouton
-textvariable var	variable active associée au texte du bouton (voir ci-dessous)
-image img	image (créée par la commande image) affichée dans le bouton
-state normal | active | disabled	état du widget. Si normal, le widget est activable, si active il est "enfoncé", si disabled il ne peut être activé (grisé).
-command cmd	commande à exécuter lorsque le bouton est activé
-variable var	variable active associée à l'état du checkbutton ou radiobutton
-value val	valeur à affecter à la variable lorsqu'un radiobutton est sélectionné
-onvalue val	valeur à affecter à la variable lorsqu'un checkbutton est "on"
-offvalue val	valeur à affecter à la variable lorsqu'un checkbutton est "off"

L'option -textvariable permet de spécifier le nom d'une variable ; Tk fait en sorte que le texte affiché d le bouton soit toujours la valeur de cette variable. Par exemple :

```
set undoText "Impossible d'annuler"
pack [button .undo -textvariable undoText]
```

Le bouton apparaît avec le texte "Impossible d'annuler". Si l'on exécute plus tard :

```
set undoText "annuler copier"
```

le texte du bouton affiche maintenant "Annuler copier".

Pour les checkbuttons et radiobuttons, on peut établir également une variable active qui refléte l'état bouton :

```
checkbutton .rectoVerso -text "Recto-verso" -variable rectoVerso -onvalue 1 -offvalue 0
```

Si l'utilisateur change l'état du bouton, la variable rectoVerso prend la valeur 0 ou 1. Inversement, si affecte une valeur 1 ou 0 à la variable rectoVerso, le checkbutton reflète immédiatement la nouvelle vale

Pour les radiobuttons, on spécifie qu'un ensemble de radiobuttons fonctionnent ensemble en les liant même variable :

```
radiobutton $box.left -text Gauche -variable align -value left
radiobutton $box.center -text Centre -variable align -value center
radiobutton $box.right -text Droite -variable align -value right
set align center
```

De la même façon que pour les checkbuttons, lorsque l'on active un radiobutton, la valeur qui lui associée par -value est affectée à la variable spécifiée par -variable. Si l'on change la valeur de la varia le radiobutton dont la valeur associée correspond à cette nouvelle valeur est activé et les autres sont re dans l'état normal.

Note importante : lorsque l'on spécifie un script par l'option -command, il y a deux substitutions : celle a lieu au moment de l'évaluation de la commande contenant l'option, et celle qui a lieu à chaque fois qu script est exécuté :

```
button .ok -text OK -command "puts $toto"
```

affichera la valeur qu'avait toto lors de la création du bouton, tandis que

```
button .ok -text OK -command {puts $toto}
```

affichera la valeur de toto au moment où l'on cliquera le bouton.

Listes - commande listbox

Une liste permet d'afficher un ensemble d'items texte. Les options spécifiques sont :

-width w	largeur (en caractères) de la boîte
-height h	hauteur (en nombre d'items) de la boîte
-selectmode single | browse | multiple | extended	mode de sélection

Les items d'un liste sont manipulés par un ensemble de méthodes du widget liste. Les items sont repé par un *index* qui peut être le numéro de l'item (à partir de 0) ou le mot-clé "end" pour le dernier item. principales méthodes sont les suivantes :

$list insert idx elem ...	insère un ou plusieurs éléments avant l'index idx. Si idx vaut "end", les éléments sont ajoutés à la fin
$list get idx	rétourne le texte de l'item d'index idx
$list delete i1 i2	enlève les items d'index compris entre i1 et i2
$list curselection	retourne la liste des numéros des items sélectionnés

Lorsqu'il y a plus d'items que la hauteur de la listbox ne permet d'en afficher, on peut associer une b de défilement à la liste (voir ci-dessous).

Barres de défilement - commande scrollbar

Les barres de défilement sont destinées à être couplées à des widgets pouvant faire défiler leur conte notamment les listbox, text et canvas. Les principales options d'une scrollbar sont :

-orient hor | ver	orientation de la barre
-command cmd	commande exécutée lorsque la valeur de la barre change

Le couplage d'une barre avec (par exemple) une listbox s'effectue en utilisant des méthodes et des opti spécifiques de la scrollbar et du widget scrollé. Exemple :

```
lisbox .l -yscrollcommand {.s set}
scrollbar .s -command {.l yview}
```

L'option -yscrollcommand de la listbox permet de mettre à jour la scrollbar lorsque le contenu de la list change, en invoquant la métode set de la scrollbar. Réciproquement, l'option -command de la scroll permet de faire défiler le contenu de la listbox lorsque l'on agit sur la scrollbar, en invoquant la méth yview de la listbox.

Potentiomètres - commande scale

Un potentiomètre permet de spécifier une valeur dans un intervalle. Les principales options spécifiq sont les suivantes :

-from f / -to t	spécifier les bornes de l'intervalle
-showvalue on|off	contrôler l'affichage de la valeur courante du potentiomètre
-label txt	nom du potentiomètre
-variable var	nom de la variable active associée au potentiomètre
-command script	script exécuté lorque la valeur du potentiomètre change. Avant d'appeler le script, on y ajoute la valeur courante du scale.

La valeur courante d'un potentiomètre peut être accédée par les méthodes set et get.

Cadres et fenêtres principales - commandes frame, toplevel, wm

Toutes les commandes de création de widget ci-dessus créent des widget "atomiques", c'est-à-dire qui contiennent pas d'autres widgets. Il existe deux types de widgets dont l'unique but est de contenir d'au widgets : les *frames*, qui servent à grouper des widgets, et les *toplevels*, qui permettent de créer nouvelles principales de l'application. Le widget ".", créé à l'initialisation de wish, est un widget tople La principale option spécifique d'un widget toplevel est -menu, qui permet de spécifier le menu associé fenêtre toplevel. Selon les plateformes, ce menu apparaît dans la fenêtre elle-même ou dans la barre menus principale.

La commande wm permet de manipuler les widgets toplevel. Par exemple :

wm title top titre	spécifie le titre qui apparaît dans la bannière de la fenêtre
wm [de]iconify top	iconifie / dé-iconifie une fenêtre
wm withdraw top	ferme une fenêtre (sans détruire son contenu)

D'autre part la commande winfo permet de récupérer le toplevel associé à un widget :

winfo toplevel w	retourne le nom de la fenêtre toplevel contenant le widget w.

Menus - commandes menu, menubutton, tk_optionMenu, tk_popup

Tk permet de gérer plusieurs types de menus : menus déroulants, menus d'options, menus contextu menus détachables. Pour les menus déroulant et contextuels, il faut créer un menubutton qui est un bou spécial permettant de faire apparaître le menu. Les options spécifiques d'un menubutton incluent ce d'un button (notamment -text, -textvariable, -image) ainsi que les options suivantes :

-menu m	nom du menu associé au menubutton. Le nom du menu doit être un descendant du nom du menubutton
-indicatoron on|off	indique s'il s'agit d'un menu déroulant (off) ou d'options (on)
-direction above | below | left | right | flush	indique la position relative du menu par rapport au menubutton

Pour créer le menu lui même, on utilise la commande menu après avoir créé le menubutton :

```
menubutton .edit -text Edition -menu .edit.menu
menu .edit.menu
```

Puis l'on crée les items du menu avec la méthode add du menu. Par exemple :

```
.edit.menu add command -label Couper -command {Cut}
```

La méthode add du menu a la syntaxe suivante :

```
menu add type -option valeur ...
```

"type" est le type de l'entrée et peut avoir l'une des valeurs suivantes :

command	item de type commande (le cas le plus fréquent)
separator	ligne de séparation pour grouper les items
radiobutton	bouton radio
checkbutton	case à cocher
cascade	sous-menu

Les options valides pour chaque type d'item sont similaires à celles des boutons, notamment -ima -state, -variable, -onvalue, -offvalue, -value, -menu (pour les cascade). Par contre le texte d'un item spécifié par l'option -label et non pas -text, et l'option -textvariable n'est pas disponible.

D'autres méthodes permettent de manipuler les items du menu. Chaque item est repéré par un indice peut être spécifié par son numéro, par le mot-clé "end" ou par une chaîne de caractère correspondan label de l'item. Les principales méthodes des menus sont les suivantes :

menu delete i1 i2	détruit les items d'indice i1 à i2
menu insert i type ...	similaire à add mais insère l'item avant l'item d'indice i
menu invoke i	invoque la commande de l'item d'indice i
menu entrycget i -option	retourne la valeur d'une option de l'item d'indice i
menu entryconfigure i -option value ...	change la valeur des options de l'item d'indice i

La commande tk_optionMenu permet de créer un menu d'options :

```
tk_optionMenu w var val1 val2 ...
```

"w" est le nom du menubutton créé par la commande ; "var" est le nom de la variable active qui contien valeur courante du menu ; "val1 val2 ..." sont les valeurs possibles qui apparaissent dans le menu. commande retourne le nom du menu afin de pouvoir configurer ses options.

Principales configurations des menus :

Barre de menus	Créer le menu correspondant à la barre de menus. Chaque item de ce menu une cascade associée à un menu de la barre de menus. Associer la barre de me à une fenêtre toplevel par son option -menu.
Menus déroulants	Créer un menubutton et son menu associé. Le menu doit être un sous-widget menubutton, et doit être spécifié par l'opion -menu de celui-ci.
Menus pop-up	Créer un menu et le faire apparaître à la position de la souris par la comma tk_popup x y.
Menus d'options	Créer un menubutton et son menu par la commande tk_optionMenu.
Menus détachables	Les menus Tk sont détachables par défaut. Selon les plateformes, la premi entrée du menu permet de le détacher, ou bien il suffit de "tirer" le menu. On p détacher un menu explicitement en invoquant son premier item (celui d'indice

Placement des widgets - commandes place, pack, grid

Une fois créé, un widget doit être confié à un gestionnaire de géométrie pour qu'il puisse être re visible. La syntaxe générale d'un gestionnaire de géométrie est la suivante :

geo w -option valeur ...	place le widget w selon un ensemble d'options
geo forget w	fait disparaître le widget w (sans le détruire)

Le gestionnaire **place** permet de placer les widgets en fixant leur taille et leur position de façon absolue relative au widget parent. Les principales options sont :

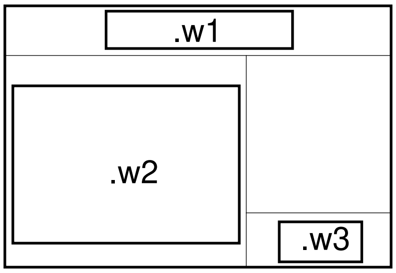
-x val / -y val / -width val / -height val	taille et position absolues du widget
-relx v / -rely v / -relwidth v / -relheight v	taille et position relatives du widget
-in master	nom du widget dans lequel se fait le placement. Par défaut il s'agit du widget parent, sinon il faut que ce soit un descendant de son parent.

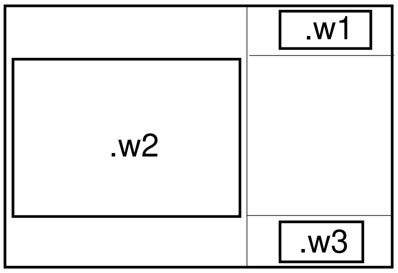
Les valeurs relatives sont des valeurs réelles comprises entre 0 et 1 qui indiquent une proportion rapport à la taille du widget maître.

Le gestionnaire **pack** permet de placer les widgets les uns par rapport aux autres. Le principe est le suiv : chaque widget est placé successivemet dans l'espace disponible de son widget maître (par défaut parent), et occupe une "tranche" de cet espace spécifiée par l'option -side. Comme le montre l'exemple dessous, le résultat dépend de l'ordre de placement des widgets :

```
pack .w1 -side top
pack .w2 -side left
pack .w3 -side bottom
```

```
pack .w2 -side left
pack .w1 -side top
pack .w3 -side bottom
```





Les principales options du gestionnaire pack sont les suivantes :

-side top | left | right | bottom	côté du placement dans l'espace restant.
-fill x | y | both | none	indique si le widget doit occuper tout l'espace disponible
-expand on | off	indique si l'espace réservé au widget doit suivre les changements de taille du widget parent
-padx x / -pady y	taille de la marge à réserver autour du widget
-anchor dir	point d'ancrage du widget dans son espace libre. "dir" est center ou l'un des points cardinaux : n, s, e, w, ne, etc.
-in master	nom du widget dans lequel se fait le placement. Par défaut il s'agit du widget parent, sinon il faut que ce soit un descendant de son parent.

Le gestionnaire **grid** permet de placer les widgets dans une grille. Les principales options sont :

-column c / -row r	coordonéees de la case dans laquelle placer le widget.
-columnspan c / -rowspan r	nombre de cases utilisées pour placer le widget (défaut 1).
-padx x / -pady y	taille de la marge à réserver autour du widget
-sticky sides	placement du widget dans sa case. "sides" est une combinaison de n, s, e, w.
-in master	nom du widget dans lequel se fait le placement. Par défaut il s'agit du widget parent, sinon il faut que ce soit un descendant de son parent.

On peut également spécifier des options s'appliquant à une colonne ou une ligne entière :

```
grid columnconfigure | rowconfigure w index -option valeur ...
```

Les options disponibles sont :

-minsize s	taille minimale de la ligne ou de la colonne
-weight w	contrôle la proportion de l'espace libre affecté à cette ligne ou colonne lorsque la taille du widget parent change.

Gestion des événements - commandes bind, bindtags, focus, grab, tkwait, after

Lorsqu'un événement se produit, Tk détermine le widget qui le reçoit. Pour les événements souris, il s' typiquement du widget sous le curseur tandis que pour les événements clavier, il s'agit du widget spéc par la commande focus. A chaque widget est associé un ensemble de *"tags"* ou *étiquettes* que l'on contrôler par la commande bindtags. A chaque tag on peut associer par la commande bind un ensemble *liaisons* (*"bindings"*) entre un nom d'événement et un script Tcl. Lorsqu'un événement <e> se pro dans le widget w, Tk recherche, pour chaque tag t du widget w, les liaisons dont l'événement corresp à <e>, puis exécute, en séquence, les scripts associés à ces liaisons. Exemple :

```
pack [frame .essai]
bind .essai <ButtonPress> {puts click}   # écrire "click" quand on clique sur .essai
```

Chaque widget a par défaut 4 tags : son nom, le nom de sa classe (par exemple Button pour les wid créés par la commande button), le nom du widget toplevel dont il est un descendant, et le tag prédé "all". On peut modifier cette liste par la commande bindtags :

bindtags w	retourne la liste des tags du widget w
bindtags w list	définit la liste des tags du widget w

Pour ajouter le tag monTag en tête de la liste des tags du widget $w, on peut écrire :

```
bindtags $w [concat monTag [bindtags $w]]
```

Une liaison est spécifiée avec la commande bind de la façon suivante :

```
bind tag event script
```

"tag" est une chaîne de caractères quelconque définissant une étiquette ; "event" est une descript d'événement de la forme suivante : <mod-type-detail>

mod combinaison éventuellement vide des valeurs suivantes :
Control, Shift, Lock, Meta, Alt (modificateurs du clavier)
Button1, Button2, Button3 (modificateurs de la souris)
Double, Triple (clicks multiples)

type type d'événements. Les principaux types sont :
ButtonPress, ButtonRelease, Motion, Enter, Leave (souris)
Key, KeyPress, KeyRelease (clavier)

détail information complémentaire : numéro du bouton pour ButtonPress,
code de la touche pour Key, etc.

"script" est un script Tcl qui sera exécuté lorsque la liaison sera activée. Des substitutions spéciales ont dans ce script avant son exécution :

%W	nom du widget dans lequel a eu lieu l'événement
%x %y	position de la souris dans le widget au moment de l'événement
%X %Y	position de la souris par rapport à l'écran au moment de l'événement
%b	numéro du bouton pour ButtonPress et ButtonRelease
%A	code ASCII de la touche pour Key, KeyPress et KeyRelease
%K	code de la touche pour Key, KeyPress et KeyRelease
%%	le caractère %

La commande focus permet de spécifier le widget qui reçoit les événements clavier :

focus	retourne le nom du widget qui reçoit les événements clavier
focus w	définit le widget qui reçoit dorénavant les événements clavier

Les commandes grab et tkwait permettent de gérer des interactions modales :

grab set [-global] w	spécifie qu'à partir de dorénavant, seuls les widgets du sous-arbre de w reçoivent les événements. Si -global est spécifié, le "grab" s'applique à toutes les applications actives.
grab reset w	annule le "grab" spécifié par "grab set".
tkwait variable var	traiter les événements jusqu'à ce que la valeur de var ait changé
tkwait window w	traiter les événements jusqu'à ce que le widget w soit détruit

La commande after permet de déclencher l'exécution d'un script après un délai fixé :

after ms | idle script	programme l'exécution du script apès ms millisecondes ou dès que la file d'évenements est vide et retourne un identificateur id
after cancel id	annule un script programmé avec la commandes ci-dessus
update idletasks	force la mise à jour de l'affichage et l'exécution des "idle scripts"