Référence des commandes Tcl
Tcl/Tk, Apprentissage et Référence

http://www.vuibert.fr/

Table des matières

Ce chapitre de référence fournit une documentation aussi complète que possible concernant toutes les commandes de base de Tk. Tout comme le chapitre précédent qui recense les commandes de base de Tcl, cette documentation est établie à partir de la référence officielle qui accompagne la distribution de la version 8.4 de Tcl/Tk.

Les conventions d'écriture des commandes, sous-commandes, options requises ou optionnelles etc. sont les mêmes que dans la Référence de Tcl. Les widgets possèdent des options dites standard parce que partagées par beaucoup d'entres eux et des options dites spécifiques qui leur appartiennent en propre et n'ont pas de signification pour d'autres widgets. Les options standard sont simplement mentionnées dans les sections consacrées à un widget particulier : elles sont toutes rassemblées et expliquées dans la section appelée *Options standard* qui commence à la page 142. Les options spécifiques sont expliquées avec les widgets qu'elles concernent.

On ajoutera les conventions suivantes propres à Tk concernant la manière de désigner la valeur de certaines options :

Désignation des unités d'écran Coordonnées et distances sont exprimées en unités d'écran qui sont soit des nombres entiers désignant un nombre de pixels d'écran, soit des nombres en virgule flottante suivis d'une lettre symbolisant l'unité utilisée : **m** pour millimètres, **c** pour centimètres, **i** pour pouces (*inches*), **p** pour points d'imprimeur (équivalents à 1/72 pouce). L'axe horizontal est orienté vers la droite et l'axe vertical vers le bas.

Désignation des couleurs Les couleurs peuvent être désignées de deux manières différentes :

- par le nom d'une couleur prédéfinie : on en trouvera la liste dans l'annexe *Les couleurs de Tcl/Tk* du livre.
- par la spécification numérique en nombres hexadécimaux des intensités de rouge, vert et bleu. Selon que l'écran autorise une profondeur de 4-bits, 8-bits, 12-bits ou 16-bits, le format sera :

```
#RVB
#RRVVBB
#RRRVVVBBB
#RRRRVVVVBBBB
```

Désignation des ancres Les valeurs possibles pour les ancres de positionnement sont *n, ne, e, se, s, sw, w, nw* ou *center*.

Désignation des reliefs Les valeurs possibles pour le relief d'un widget sont *flat*, *raised*, *sunken*, *groove*, *solid* et *ridge*.

Désignation des curseurs Les curseurs sont désignés par leur nom. Les valeurs possibles sont listées dans la partie Apprentissage du livre dans le chapitre *Contrôle de l'interface avec Tk*.

Désignation des bitmaps Les bitmaps peuvent être désignés de deux manières différentes :

- par le nom d'une image prédéfinie : *error*, *gray12*, *gray25*, *gray50*, *gray75*, *hourglass*, *info*, *questhead*, *question* ou *warning*.
- par le nom d'un fichier contenant un bitmap valide au format X11 standard. Le nom du fichier est précédé d'un arobase @ : @*NomDeFichier*.

Désignation des polices Les trois systèmes permettant de désigner une police de caractères avec ses attributs ont été expliqués à la section **??** p.**??**.

Désignation des pointillés Il y a deux syntaxes possibles pour désigner des motifs de pointillés et de tiretés. Elles sont expliquées à la section *Motifs pointillés* p.34.

Désignation des classe visuelle Une classe visuelle d'écran peut être désignée soit par une des valeurs *directcolor*, *grayscale*, *greyscale*, *pseudocolor*, *staticcolor*, *staticgray*, *staticgrey*, *truecolor*, *best* ou *default*, soit par le nom d'une autre fenêtre dont la classe visuelle sera adoptée.

Désignation des valeurs booléennes Les conventions pour désigner des valeurs booléennes sont les mêmes qu'avec Tcl :

- *0*, *false*, *no* et *off* pour faux.
- *1*, *true*, *yes* et *on* pour vrai.

bell

Fait entendre un bip sonore.

Syntaxe

bell ?-**displayof** *fenêtre*? ?-**nice**?

Description

Cette commande fait entendre le bip interne du moniteur correspondant à la fenêtre *fenêtre* et renvoie une chaîne vide. Si l'option **-displayof** est omise, le moniteur de la fenêtre principale de l'application est utilisé par défaut. La commande utilise les réglages courants de l'écran (qui peuvent être modifiés, sur une plateforme Unix, au moyen de l'utilitaire xset par exemple).

Si l'option **-nice** n'est pas spécifiée, cette commande réinitialise l'économiseur d'écran, ce qui a pour effet de rendre l'écran visible à nouveau. Il faut noter toutefois que certains économiseurs d'écran peuvent ignorer cette commande.

bind

Permet de déclencher des scripts Tcl à partir d'événements d'interface.

Syntaxe

binindentbind *widget séquence*
bind *widget séquence script*
bind *widget séquence* +*script*

Introduction

La commande **bind** associe des scripts Tcl à des événements d'interface. Si les trois arguments sont spécifiés, **bind** fera en sorte que le script Tcl *script* soit évalué dès que le ou les événements spécifiés par l'argument *séquence* se produisent dans la ou les fenêtres identifiées par l'argument *widget*.

Si *script* est préfixé par un signe +, il est ajouté à toute liaison déjà existante pour *séquence*; sinon *script* remplace une liaison qui existe déjà. Si *script* est une chaîne vide, la liaison courante pour *séquence* est détruite, laissant *séquence* non liée. Dans tous les cas où un argument *script* est fourni, **bind** renvoie une chaîne vide. d *widget*

Si l'argument *séquence* est spécifié sans un *script*, le script actuellement lié à *séquence* est renvoyé, ou une chaîne vide est renvoyée s'il n'y a aucune liaison pour *séquence*. Si ni *séquence* ni *script* ne sont spécifiés, la valeur de retour est une liste dont les éléments sont toutes les séquences pour lesquelles il existe des liaisons pour *widget*.

L'argument *widget* détermine à quelles fenêtres la liaison s'applique. Si l'argument *widget* commence par un point, comme dans `.a.b.c`, il doit être le nom de chemin d'une fenêtre; autrement, il peut s'agir d'une chaîne quelconque. Chaque fenêtre a une liste de widgets associés et une liaison s'applique à une fenêtre particulière si son widget figure parmi celles qui sont spécifiées pour la fenêtre. Bien que la commande **bindtags** puisse être utilisée pour assigner un jeu arbitraire de widgets à une fenêtre, les widgets adoptent les comportements par défaut suivants:

Si un widget est le nom d'une fenêtre interne, la liaison s'applique à cette fenêtre.

Si le widget est le nom d'une fenêtre de premier niveau, la liaison s'applique à cette fenêtre ainsi qu'à toutes les fenêtres internes.

Si le widget est le nom d'une classe de composants, comme *Button*, la liaison s'applique à tous les composants de cette classe;

Si l'argument *widget* a la valeur *all*, la liaison s'applique à toutes les fenêtres de l'application.

Motifs d'événements

L'argument *séquence* spécifie une série de un ou plusieurs motifs d'événements, avec des espaces optionnels entre les motifs. Chaque motif d'événement peut prendre trois formes différentes :

- un caractère ASCII imprimable autre qu'une espace ou bien le symbole <. Cette forme de motif correspond à un événement KeyPress pour ce caractère particulier.
- la seconde forme est plus générale. Elle adopte la syntaxe suivante :

  <*modificateurs-type-détail*>

  Le motif est entouré de crochets angulaires < > entourant zéro ou plusieurs modificateurs, un type d'événement et une information supplémentaire appelée *détail* qui identifie un bouton ou une touche symbolique particuliers. Chacun de ces champs peut être omis mais il doit y avoir au moins un type ou un détail. Les champs sont séparés soit par des espaces, soit par des tirets.
- la troisième forme est employée pour permettre à l'utilisateur de spécifier des événements que l'on qualifie d'événements virtuels. Elle adopte la syntaxe suivante : <<*nom*>>

Le motif d'événement virtuel est entouré de doubles crochets angulaires << >> à l'intérieur desquels on indique le nom de l'événement virtuel. On ne peut pas combiner de modificateurs, tels que **Shift** ou **Control**, avec un événement virtuel. Les liaisons à un événement virtuel peuvent être créées avant que l'événement virtuel soit défini, et si la définition d'un événement virtuel change dynamiquement, toutes les fenêtres qui lui sont liées répondront immédiatement à la nouvelle définition.

Modificateurs

Les modificateurs peuvent avoir les valeurs suivantes :

Alt	Méta, M
Button1, B1	Mod1, M1
Button2, B2	Mod2, M2
Button3, B3	Mod3, M3
Button4, B4	Mod4, M4
Button5, B5	Mod5, M5
Control	Quadruple
Double	Shift
Lock	Triple

Lorsque deux valeurs sont indiquées (comme Button1 et B1), elles désignent deux notations différentes pour un même modificateur. Les significations devraient être évidentes dans le cadre d'un système X : Button1 signifie que le premier bouton de la souris est enfoncé lorsque l'événement se produit.

Pour qu'une liaison corresponde à un événement donné, les modificateurs dans l'événement doivent inclure tous ceux qui ont été spécifiés dans le motif de l'événement. Un événement peut aussi comporter des modificateurs supplémentaires non spécifiés dans la liaison. Par exemple, si le premier bouton de la souris est enfoncé en même temps que les touches Majuscule et Contrôle, le motif `<Control-Button-1>` correspondra à l'événement mais pas `<Mod1-Button-1>`. Si aucun modificateur n'est spécifié, toute combinaison de modificateurs peut figurer dans l'événement.

Le modificateur *Méta* ou *M* se réfère à l'un quelconque des modificateurs de **M1** à *M5* associés à la touche « méta » sur le clavier (de valeur symbolique *Méta_R* ou *Méta_L*). S'il n'y a pas de touches méta, ou si elles ne sont associées à aucun modificateur, *Méta* et *M* ne correspondront à aucun événement. De la même manière, le modificateur *Alt* se réfère à l'un quelconque des modificateurs associés à la touche « alt » sur le clavier (de valeur symbolique *Alt_R* ou *Alt_L*).

Les modificateurs *Double*, *Triple* et *Quadruple* sont une commodité pour spécifier des clics répétés de souris ou d'autres événements répétés. Ils font qu'un motif d'événement particulier puisse être répété 2, 3 ou 4 fois, et posent une condition de temps et d'espace dans la séquence : pour qu'une séquence d'événements corresponde au motif *Double*, *Triple* ou *Quadruple*, tous les événements doivent se produire à des instants rapprochés et quasiment sans que la souris ne bouge.

Types d'événements

Le champ *type* mentionné dans la seconde syntaxe de description des motifs d'événements peut être l'un des types standard du système X avec quelques abréviations supplémentaires. Quelques types non standard ont été ajoutés pour supporter mieux les plates-formes Windows et Macintosh. Tous les types sont rassemblés dans le tableau suivant :

`<Activate>`	`<Gravity>`
`<ButtonPress>`, `<Button>`	`<KeyPress>`, `<Key>`
`<ButtonRelease>`	`<KeyRelease>`
`<Circulate>`	`<Leave>`
`<Colormap>`	`<Map>`
`<Configure>`	`<Motion>`
`<Deactivate>`	`<MouseWheel>`
`<Destroy>`	`<Property>`
`<Enter>`	`<Reparent>`
`<Expose>`	`<Unmap>`
`<FocusIn>`	`<Visibility>`
`<FocusOut>`	

On notera que `<Button>` et `<ButtonPress>` sont synonymes, de même que `<Key>` et `<KeyPress>`. Certains de ces événements sont spécifiques à certaines plates-formes :

<Activate> et <Deactivate>
: Ces deux événements sont envoyés à toutes les sous-fenêtres d'une fenêtre de premier niveau lorsqu'elles changent d'état. En plus de la notion de fenêtre focalisée, les plates-formes Macintosh et Windows possèdent la notion de fenêtre active : sous MacOS, les composants de la fenêtre active ont une apparence différente de celle des fenêtres désactivées. L'événement <Activate> est envoyé à toutes les sous-fenêtres d'une fenêtre de premier niveau lorsqu'elle passe de l'état activé à l'état désactivé. Il n'y a pas de symbole pourcentage % de substitution (voir plus loin) correspondant à ces événements.

<MouseWheel>
: Certaines souris sur la plate-forme Windows possèdent une roue utilisée pour faire défiler des documents sans manipuler les barres de défilement. Lorsqu'on fait tourner la roue de la souris, le système génère des événements <MouseWheel> que l'application peut intercepter pour provoquer un défilement. Comme les événements <Key>, l'événement est toujours adressé à la fenêtre actuellement focalisée.

  Lorsque l'événement est reçu, on peut utiliser la séquence de substitution %D pour obtenir le champ d'incrément de l'événement qui est une valeur entière mesurant le déplacement de la roue (la valeur minimale est 120). Le signe détermine le sens de déplacement de la roue : s'il est positif le défilement devrait se faire vers le haut, s'il est négatif vers le bas.

La dernière partie d'une spécification de motif d'événement est l'argument *détail*. Dans le cas d'un événement <ButtonPress> ou <ButtonRelease>, il s'agit du numéro du bouton (1-5). Si un numéro de bouton est indiqué, seul un événement sur ce bouton particulier pourra correspondre au motif. Si aucun numéro de bouton n'est indiqué, un événement peut correspondre à n'importe quel bouton. Indiquer un numéro de bouton spécifique est différent d'indiquer un modificateur de bouton ; dans le premier cas, on se réfère à un bouton enfoncé ou relâché tandis que dans le second on se réfère à un autre bouton déjà enfoncé lorsque l'événement se produit. Si un numéro de bouton est indiqué, l'argument *type* peut être omis : ce sera par défaut <ButtonPress>. Par exemple, le spécificateur `<1>` est équivalent à `<ButtonPress-1>`.

Si le type d'événement est <KeyPress> ou <KeyRelease>, l'argument *détail* peut être spécifié sous la forme d'une touche symbolique. Sous système X, les touches symboliques sont des spécifications textuelles pour des touches particulières sur le clavier : on en trouvera la liste complète dans l'annexe *Les touches symboliques* du livre. On peut au besoin utiliser la notation %K décrite ci-dessous pour produire le nom de touche symbolique d'une certaine touche. Si un argument *détail* est donné sous forme de touche symbolique, le champ *type* peut être omis : il vaudra <KeyPress> par défaut. Par exemple, `<Control-comma>` est équivalent à `<Control-KeyPress-comma>`.

Scripts de liaison et substitutions

L'argument *script* de la commande **bind** est un script Tcl, qui sera exécuté dès que la séquence donnée d'événements se produira. L'argument *commande* sera exécuté dans le même interpréteur que celui qui exécute la commande **bind**, et il s'exécutera au niveau global (seules les variables globales lui sont accessibles). Si l'argument *script* contient des signes de pourcentage %, le script ne sera pas exécuté directement. À la place, un nouveau script sera engendré en remplaçant tous les symboles de pourcentage ainsi que le caractère qui les suit par une certaine information provenant de l'événement courant : la combinaison d'un pourcentage avec une lettre constitue une séquence de substitution, le remplacement dépendant de la lettre-clé. De manière générale, la chaîne de remplacement est la valeur décimale d'un champ donné dans l'événement courant. Certaines substitutions ne sont valables que pour certains types d'événements ; lorsqu'elles sont utilisées pour d'autres types d'événements, la valeur substituée est indéfinie.

%%
Remplacé par un simple pourcentage.

%#
Le numéro de la dernière requête client exécutée par le serveur (le champ *serial* de l'événement). Valable pour tous les types d'événements.

%a
Le champ *above* de l'événement, formaté sous forme de nombre hexadécimal. Valable seulement pour les événements <`Configure`>.

%b
Le numéro du bouton qui a été enfoncé ou relâché. Valable seulement pour les événements <`ButtonPress`> et <`ButtonRelease`>.

%c
Le champ *count* de l'événement. Valable seulement pour <`Expose`>.

%d
Le champ *detail* de l'événement. La chaîne **%d** est remplacée par une chaîne identifiant le détail. Pour les événements <`Enter`>, <`Leave`>, <`FocusIn`> et <`FocusOut`>, la chaîne aura une des valeurs suivantes :

NotifyAncestor	*NotifyInferior*
NotifyNonlinearVirtual	*NotifyPointerRoot*
NotifyDetailNone	*NotifyNonlinear*
NotifyPointer	*NotifyVirtual*

Pour les autres événements, la chaîne substituée est indéfinie.

%f
Le champ *focus* de l'événement (0 ou 1). Valable seulement pour les événements <`Enter`> et <`Leave`>.

%h
Le champ *height* de l'événement. Valable seulement pour les événements <`Configure`> et <`Expose`>.

%k
Le champ *keycode* de l'événement. Valable seulement pour les événements <KeyPress> et <KeyRelease>.

%m
Le champ *mode* de l'événement. La chaîne substituée est une des valeurs : *NotifyNormal*, *NotifyGrab*, *NotifyUngrab* ou *NotifyWhileGrabbed*. Valable seulement pour les événements <Enter>, <FocusIn>, <FocusOut> et <Leave>.

%o
Le champ *override_redirect* de l'événement. Valable seulement pour les événements <Map>, <Reparent> et <Configure>.

%p
Le champ *place* de l'événement, substitué par une des valeurs *PlaceOnTop* ou *PlaceOnBottom*. Valable seulement pour les événements <Circulate>.

%s
Le champ *state* de l'événement.
Pour les événements <ButtonPress>, <ButtonRelease>, <Enter>, <KeyPress>, <KeyRelease>, <Leave> et <Motion>, une chaîne décimale est substitutée. Pour <Visibility>, c'est une des valeurs *VisibilityUnobscured*, *VisibilityPartiallyObscured*, et *VisibilityFullyObscured* qui est substituée.

%t
Le champ *time* de l'événement. Valable seulement pour les événements contenant un champ *time*.

%w
Le champ *width* de l'événement. Valable seulement pour les événements <Configure> et <Expose>.

%x
Le champ *x* de l'événement. Valable seulement pour les événements contenant un champ *x*.

%y
Le champ *y* de l'événement. Valable seulement pour les événements contenant un champ *y*.

%A
Substitue le caractère ASCII correspondant à l'événement, ou une chaîne vide si l'événement ne correspond pas à un caractère ASCII (par exemple si la touche majusucle est enfoncée). La fonction XLookupString fait tout le travail de traduction de l'événement en un caractère ASCII. Valable seulement pour les événements <KeyPress> et <KeyRelease>.

%B
Le champ *border_width* de l'événement. Valable seulement pour les événements <Configure>.

%D
Cette substitution renvoie la valeur d'incrément d'un événement <MouseWheel> de roue de souris. Cette valeur représente le nombre d'unités de rotation de la roue.

%E
Le champ *send_event* de l'événement. Valable pour tous les types d'événements.

%K
La touche symbolique correspondant à l'événement, substituée comme chaîne textuelle. Valable seulement pour les événements **KeyPress** et <KeyRelease>.

%N
La touche symbolique correspondant à l'événement, substituée comme nombre décimal. Valable seulement pour <KeyPress> et <KeyRelease>.

%R
L'identificateur de fenêtre *root* de l'événement. Valable seulement pour les événements comportant un champ *root*.

%S
L'identificateur de fenêtre *sous-fenêtre* de l'événement, formaté comme nombre hexadécimal. Valable seulement pour les événements comportant un champ *sous-fenêtre*.

%T
Le champ *type* de l'événement. Valable pour tous les types d'événements.

%W
Le chemin de la fenêtre à laquelle l'événement a été rapporté (le champ *window* de l'événement). Valable pour tous les types d'événements.

%X
Le champ *x_root* de l'événement. Si un gestionnaire de fenêtres racines virtuelles est utilisé, la valeur substituée est l'abscisse correspondante dans la racine virtuelle. Valable seulement pour les événements <ButtonPress>, <ButtonRelease>, <KeyPress>, <KeyRelease>, et <Motion>.

%Y
Le champ *y_root* de l'événement. Si un gestionnaire de fenêtres racines virtuelles est utilisé, la valeur substituée est l'ordonnée correspondante dans la racine virtuelle. Valable seulement pour les événements <ButtonPress>, <ButtonRelease>, <KeyPress>, <KeyRelease>, et <Motion>.

La chaîne de remplacement pour une substitution de symbole pourcentage % est formatée comme une liste Tcl valide. Cela signifie qu'elle sera entourée d'accolades si elle contient des espaces et si des caractères spéciaux comme $ et { peuvent être préfixés par des contre-obliques. Cela garantit que la chaîne sera transmise à l'interpréteur Tcl lorsque le script de la liaison sera évalué. La plupart des remplacements sont des nombres ou des chaînes bien définies comme *Above* par exemple ; pour ces remplacements aucun formatage particulier n'est jamais nécessaire. La situation la plus courante de reformatage se trouve dans la substitution %A. Par exemple, si *script* est l'instruction

```
insert %A
```

et si le caractère tapé est un crochet ouvrant [, le script effectivement exécuté sera

```
insert \[
```

Cela a pour effet que la commande **insert** reçoit la chaîne de remplacement originale (un crochet ouvrant) comme premier argument. Si une contre-oblique n'avait pas été insérée, Tcl n'aurait pas pu analyser cette instruction correctement.

Correspondances multiples

Il est possible que plusieurs liaisons correspondent à un événement X donné. Si les liaisons sont associées à différents arguments *widget*, alors chaque liaison sera exécutée, dans l'ordre. Par défaut, dans l'ordre de précédence, une liaison pour le composant sera exécutée d'abord, suivie par une liaison de classe, une liaison pour le premier niveau, et une liaison **all**. La commande **bindtags** peut être utilisée pour modifier cet ordre pour une fenêtre particulière ou pour associer des balises de liaison supplémentaires à la fenêtre.

Les commandes **continue** et **break** peuvent être utilisées dans un script de liaison pour contrôler le processus des recherches de correspondances. Si **continue** est invoquée, le script de liaison courant est interrompu et Tk poursuit en exécutant les scripts associés à d'autres widgets. Si **continue** est invoquée, le script de liaison courant est interrompu et aucun autre script n'est invoqué pour l'événement.

Si plusieurs liaisons correspondent à un événement particulier et si elles concernent le même *widget*, la liaison la plus spécifique est choisie et son script est évalué. La détermination de la liaison la plus spécifique se fait selon les règles suivantes :

1. un motif d'événement qui spécifie un bouton ou une touche particulière est plus spécifique qu'un motif qui ne le fait pas ;
2. une séquence plus longue, en terme de nombre d'événements qui correspondent, est plus spécifique qu'une séquence plus courte ;
3. si les modificateurs spécifiés dans un motif sont un sous-ensemble des modificateurs d'un autre motif, le motif avec plus de modificateurs est plus spécifique ;
4. un événement virtuel dont le motif physique correspond à la séquence est moins spécifique que le même motif physique qui ne serait pas associé à un événement virtuel ;
5. lorsqu'une séquence correspond à plusieurs événements virtuels, l'un des événements virtuels sera choisi mais l'ordre sera indéfini.

Si des séquences contiennent plusieurs événement, les tests 3 à 5 sont appliqués dans l'ordre de l'événement le plus récent à l'événement le plus ancien dans les séquences. Si ces tests échouent, la séquence la plus récemment enregistrée est retenue.

Si plusieurs événements virtuels sont déclenchés par la même séquence, et si tous sont liés au même widget, seul l'un d'eux sera déclenché; il sera choisi de façon aléatoire :

```
event add <<Paste>> <Control-y>
event add <<Paste>> <Button-2>
event add <<Scroll>> <Button-2>
bind Entry <<Paste>> {puts Paste}
```

```
bind Entry <<Scroll>> {puts Scroll}
```

Si l'utilisateur tape Control-y, la liaison <<Paste>> sera invoquée, mais s'il enfonce le second bouton de la souris alors soit la liaison <<Paste>>, soit la liaison <<Scroll>> sera invoquée sans qu'il soit possible de prédire laquelle.

Séquences multiples d'événements

Lorsqu'une *séquence* spécifiée dans une commande **bind** contient plusieurs motifs d'événements, son script est exécuté dès que les événements récents correspondent à la séquence. Cela signifie par exemple que si le premier bouton de la souris est enfoncé de façon répétée, la séquence `<Double-ButtonPress-1>` correspondra à chaque clic, sauf le premier. Si des événements extérieurs risquant d'empêcher une correspondance avec le motif se produisent au milieu d'une séquence d'événements, ces événements sont ignorés à moins qu'il ne s'agisse de <`ButtonPress`> ou de <`KeyPress`>. Par exemple, `<Double-ButtonPress-1>` correspondra à une série de clics du premier bouton de la souris, même si des événements <`ButtonRelease`> ou <`Motion`> se produisent entre les événements <`ButtonPress`>.

En outre, un événement <`KeyPress`> peut être précédé d'un nombre quelconque d'événements <`KeyPress`> résultant d'une touche modificatrice sans que ces modificateurs n'empêchent une correspondance. Par exemple, la séquence d'événements `aB` correspondra à l'enfoncement de la touche a, le relèvement de la touche a, l'enfoncement de la touche Majuscules puis celui de la touche b : l'enfoncement de la touche Majuscule est ignoré car c'est seulement un modificateur. Enfin, si plusieurs événements <`Motion`> se produisent à la suite, seul le dernier est utilisé pour faire correspondre des séquences.

Erreurs

Si une erreur se produit au cours de l'exécution du script pour une liaison, le mécanisme de **bgerror** se déclenche pour rapporter cette erreur. La commande **bgerror** sera exécutée au niveau global (en dehors du contexte de toute procédure Tcl).

Si un événement ne correspond à aucune liaison existante, il est ignoré. Un événement non lié n'est pas considéré comme une erreur.

bindtags

Détermine quelles liaisons s'appliquent à une fenêtre et dans quel ordre d'évaluation.

Syntaxe

bindtags *fenêtre ?listeBalises?*

Description

Lorsqu'une liaison est créée avec la commande **bind**, elle est associée soit à une fenêtre particulière de la forme `.a.b.c`, soit à un nom de classe tel que **Button**, au mot-clé *all*, ou à n'importe quelle autre chaîne. Toutes ces formes sont appelées balises de liaison. Chaque fenêtre comporte une liste de balises de liaison qui détermine la forme d'exécution des événements pour la fenêtre. Quand un événement se produit dans une fenêtre, il est appliqué à chacune des balises de la fenêtre dans l'ordre : pour chaque balise, la liaison la plus spécifique correspondant à la balise et à l'événement donnés est exécutée (cf. p. 13 pour les règles qui régissent ce mécanisme).

Par défaut, chaque fenêtre possède quatre balises de liaison consistant en le nom de la fenêtre, le nom de classe de la fenêtre, le nom de l'ancêtre de premier niveau le plus proche de la fenêtre et le mot-clé *all*, dans cet ordre. Les fenêtres de premier niveau ont seulement trois balises par défaut car elles sont elles-mêmes le parent de premier niveau.

La commande **bindtags** permet de lire et de modifier les balises de liaison d'une fenêtre.

Si **bindtags** est invoquée avec un seul argument, le jeu courant de balises de liaison pour *fenêtre* est renvoyé sous forme de liste. Si l'argument *listeBalises* est spécifié dans la commande **bindtags**, il doit constituer une liste Tcl valide ; les balises de *fenêtre* sont changées en les éléments de la liste.

Les éléments de *listeBalises* peuvent être des chaînes quelconques ; cependant, toute balise commençant par un point est considérée comme le nom d'une fenêtre ; si aucune fenêtre de ce nom n'existe au moment où l'événement est exécuté, la balise est ignorée pour cet événement. L'ordre des éléments dans *listeBalises* détermine l'ordre dans lequel les scripts de liaison sont exécutés en réponse aux événements. Par exemple, la commande

```
bindtags .b {all . Button .b}
```

renverse l'ordre dans lequel les scripts seront évalués pour un bouton nommé **.b** de telle sorte que les liaisons **all** seront invoquées en premier, puis les liaisons pour le premier niveau de **.b**, puis les liaisons de classe pour finir enfin par les liaisons de **.b**. Si *listeBalises* est une liste vide, les balises de liaison pour *fenêtre* sont réordonnées dans l'ordre par défaut décrit ci-dessus.

La commande **bindtags** peut être utilisée pour introduire des balises supplémentaires quelconques pour une fenêtre ou pour supprimer des balises standard. Par exemple, la commande

```
bindtags .b {.b BoutonSpécial . all}
```

remplace la balise *Button* pour *.b* par *BoutonSpécial*. Cela implique que les liaisons par défaut pour les boutons, associées à la balise *Button*, ne s'appliqueront plus à *.b*, mais que toutes les liaisons associées à *BoutonSpécial* (qui pourrait être une nouvelle classe particulière de boutons) s'appliqueront.

bitmap

Permet de créer et de manipuler des images bichromes.

Syntaxe

image create bitmap *?nom? ?options?*

Description

Les bitmaps, dans la terminologie de Tk, sont des images dont les pixels peuvent être de deux couleurs différentes ou transparents. Une image bitmap est définie au moyen de quatre éléments: une couleur de fond, une couleur de premier plan et deux tableaux de pixels (bitmaps) appelés respectivement *source* et *masque*.

Ces bitmaps spécifient des valeurs pour un tableau rectangulaire de pixels et doivent être tous deux de mêmes dimensions. Pour les pixels du masque qui sont nuls, l'image n'affiche rien, ce qui produit un effet de transparence. Pour les autres pixels, l'image affiche la couleur de premier plan pour les valeurs 1 de la source et la couleur de fond pour les valeurs 0 de la source

Création de bitmaps

Comme toutes les autres images, les images bichromes sont créées au moyen de la commande **image create**. Les images bichromes admettent les options suivantes:

-background *couleur*
: Spécifie une couleur de fond pour l'image au moyen des syntaxes usuelles de Tk pour désigner les couleurs. Si la valeur de l'option est une chaîne vide, les pixels d'arrière-plan seront transparents: ceci est obtenu en utilisant l'image source comme masque et en ignorant les options **-maskdata** et **-maskfile**.

-data *chaîne*
: Spécifie le contenu de la source sous forme de chaîne. Cette chaîne doit se conformer au format X11 pour les images bitmaps (telles qu'on les obtient par exemple avec le programme bitmap d'Unix). Si les deux options **-data** et **-file** sont spécifiées simultanément, c'est l'option **-data** qui l'emporte.

-file *nom*
: L'argument *nom* représente le nom d'un fichier contenant une image bitmap source. Ce fichier doit se conformer au format X11 pour les images bitmaps (comme ceux générés par le programme bitmap d'Unix).

-foreground *couleur*
: Spécifie une couleur de premier plan pour l'image au moyen des syntaxes usuelles de Tk pour désigner les couleurs.

-maskdata *chaîne*
: Spécifie le contenu du masque sous forme de chaîne. Cette chaîne doit se conformer au format X11 pour les images bitmaps (telles qu'on les obtient par

exemple avec le programme bitmap d'Unix). Si les deux options **-maskdata** et **-maskfile** sont spécifiées simultanément, c'est l'option **-maskdata** qui l'emporte.

-maskfile *nom*
L'argument *nom* représente le nom d'un fichier définissant un masque. Ce fichier doit se conformer au format X11 comme pour l'option **-file**.

Commandes spécifiques des bitmaps

Lorsqu'une image bichrome est créée, Tk crée une nouvelle commande ayant le nom *nomImage* qui aura été donné à la nouvelle image. Cette commande peut être utilisée pour invoquer diverses opérations concernant l'image. La forme générale est :

nomImage opération ?arg arg ...?

Les opérations reconnues par les composantes de type *bitmap* sont les suivantes :

cheminBouton **cget** *option*
Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **image create bitmap**.

cheminBouton **configure** *?option? ?valeur option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration du bitmap. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *nomImage*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **image create bitmap**.

button

Permet de créer et de manipuler les boutons.

Syntaxe

button *cheminBouton ?options?*

Description

La commande **button** crée une nouvelle fenêtre (indiquée par l'argument *cheminBouton*) et en fait un composant graphique de type *bouton*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques du bouton telles que la couleur, la police, le texte, le relief initial etc. La commande **button** renvoie la valeur de son argument *cheminBouton*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminBouton*, mais en revanche l'objet parent de *cheminBouton* doit exister effectivement.

Un bouton est un composant graphique qui affiche un texte, une icone bitmap ou une image que l'on appelle *étiquette du bouton*. Si l'étiquette est textuelle, elle est entièrement composée dans une même police ; le texte peut occuper plusieurs lignes à l'écran, soit par utilisation de sauts de lignes, soit parce qu'un paramètre **-wrapLength** a été fixé pour forcer les coupures de ligne. Il est enfin possible de souligner certains caractères du texte avec l'option **-underline**, le plus souvent pour en faire par la suite des raccourcis clavier.

Un bouton peut s'afficher dans trois états différents, selon la valeur de l'option **-state** : son apparence peut être bombée, creusée ou plate. On peut le faire clignoter et il peut invoquer une commande Tcl lorsque le premier bouton de la souris est cliqué dessus. Lorsque l'utilisateur clique sur un bouton, la commande Tcl déclarée au moyen de l'option **-command** est invoquée.

Opérations sur les boutons

Avec l'instruction :
button *cheminBouton ?options?*
la commande **button** crée une nouvelle commande Tcl dont le nom est précisément *cheminBouton*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le bouton. La forme générale est :
cheminBouton opération ?arg arg ...?

Les opérations reconnues par les composantes de type *button* sont les suivantes :

cheminBouton **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **button**.

cheminBouton **configure** *?option? ?valeur option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration du composant graphique. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminBouton*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **button**.

cheminBouton **flash**
Fait clignoter le bouton. Cet effet est obtenu en réaffichant à plusieurs reprises le bouton en faisant alterner les couleurs de son état normal et de son état activé. À la fin du clignotement, le bouton est laissé dans le même état que lorsque la commande a été invoquée. Cette commande est ignorée si l'état du bouton est *désactivé* (*disabled*).

cheminBouton **invoke**
Accomplit exactement ce qui se produirait si un utilisateur cliquait sur le bouton avec la souris : l'état du bouton est commuté et la commande Tcl associée, si elle existe, est invoquée. La valeur de retour est celle de la commande Tcl ou bien une chaîne vide si aucune commande n'est associée au bouton. Cette commande est ignorée si l'état du bouton est *désactivé* (*disabled*).

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les boutons avec les caractéristiques suivantes :

1. un bouton graphique est activé dès que le pointeur de la souris passe au-dessus et désactivé dès qu'il le quitte. Sous Windows, cela ne fonctionne qu'avec le premier bouton de la souris ;
2. le relief d'un bouton graphique prend la forme en creux dès que le premier bouton de la souris est pressé au-dessus de lui et le relief reprend sa forme originale lorsque le bouton de la souris est relâché ;
3. si le premier bouton de la souris est pressé sur un bouton graphique puis relâché toujours au-dessus du bouton, ce bouton graphique est invoqué. En revanche si le pointeur n'est plus sur le bouton graphique lorsque le bouton de la souris est relâché, rien ne se produit ;
4. lorsque le bouton est focalisé par l'application, la barre d'espacement peut être utilisée pour l'invoquer.

Si l'état d'un bouton est *disabled*, aucune des actions mentionnées ne se produit.

Les comportements décrits sont les comportements par défaut et peuvent toujours être modifiés en définissant de nouvelles liaisons pour des composants particuliers ou en redéfinissant les liaisons de classe.

-activebackground	activeBackground	Foreground
-activeforeground	activeForeground	Background
-anchor	anchor	Anchor
-background ou -bg	background	Background
-bitmap	bitmap	Bitmap
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-image	image	Image
-justify	justify	Justify
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus
-text	text	Text
-textvariable	textVariable	Variable
-underline	underline	Underline
-wraplength	wrapLength	WrapLength

TAB. 1 – *Options standard reconnues par les composants de type* button.

Options standard

Les options standard supportées par les composants de type *button* sont rassemblées[1] dans le tableau 1. Ces options sont décrites en détail à la page 142.

Options spécifiques

-command **command** **Command**

Spécifie une commande Tcl à associer avec le bouton. Cette commande est typiquement invoquée lorsque le premier bouton de la souris est relâché au-dessus de la fenêtre du bouton.

-compound **compound** **Compound**

Spécifie si le bouton doit afficher à la fois une image et du texte, et dans ce cas, l'endroit où l'image doit être placée par rapport au texte. Les valeurs acceptables pour cette option sont *bottom*, *center*, *left*, *none*, *right* et *top*. La valeur par défaut

1. Notons que la documentation de Tk mentionne à tort l'option **-repeatdelay** et l'option **-repeatinterval**. Celles-ci ne sont pas supportées par les boutons.

est *none*, ce qui signifie que le bouton affichera ou bien du texte, ou bien une image, en fonction de la valeur des options standard **-image** et **-bitmap**.

Cette option n'est disponible qu'à partir de la version 8.4 de Tcl.

-default **default** **Default**

Spécifie l'un des trois états désignés par les constantes *normal*, *active* ou *disabled* pour l'anneau qui entoure le bouton par défaut. Dans l'état activé (*active*), le bouton est dessiné selon l'apparence spécifique à la plate-forme pour un bouton par défaut. Dans l'état normal, le bouton est dessiné avec l'apparence specifique à la plate-forme pour un bouton qui n'est pas le bouton par défaut, en laissant assez d'espace pour dessiner le bouton par défaut. Les états *normal* et *actif* feront des boutons de même taille. Dans l'état désactivé (*disabled*), le bouton est dessiné comme un bouton ordinaire mais sans laisser d'espace pour l'apparence d'un bouton par défaut : dans ce cas, le bouton pourra être plus petit que dans l'état activé.

-height **height** **Height**

Spécifie la hauteur désirée pour le bouton. Si une image ou un bitmap est affiché dans le bouton, la valeur est mesurée en unités d'écran ; pour du texte elle est mesurée en nombre de lignes de texte. Si cette option n'est pas spécifiée, la hauteur désirée pour le bouton est calculée à partir de la taille de l'image, du bitmap ou du texte qu'il contient.

-overrelief **overRelief** **OverRelief**

Spécifie un relief alternatif pour le bouton à utiliser lorsque le curseur de la souris passe au-dessus. Cette option peut être utilisée pour fabriquer des boutons de barre d'outils en utilisant une configuration telle que ceci :

```
-relief flat -overrelief raised
```

Si la valeur de cette option est la chaîne vide, aucun relief alternatif n'est utilisé lorsque le curseur de la souris passe au-dessus du bouton. La chaîne vide est la valeur par défaut.

-state **state** **State**

Spécifie l'un des trois états désignés par les constantes *normal*, *active* ou *disabled* pour le bouton. Dans l'état normal, le bouton est affiché en utilisant les options **foreground** et **background**. L'état activé est typiquement utilisé lorsque la souris est au-dessus du bouton. Dans l'état activé, le bouton est affiché en utilisant les options **activeForeground** et **activeBackground**. L'état désactivé (*disabled*) signifie que le bouton ne réagit plus : les liaisons par défaut refuseront d'activer l'objet et ignoreront les clics de souris. Dans cet état, les options **disabledForeground** et **background** déterminent comment le bouton doit être affiché.

-width **width** **Width**

Spécifie la largeur désirée pour le bouton. Si une image ou un bitmap est affiché dans le bouton, la valeur est mesurée en unités d'écran ; pour du texte elle est

mesurée en nombre de caractères. Si cette option n'est pas spécifiée, la hauteur désirée pour le bouton est calculée à partir de la taille de l'image, du bitmap ou du texte qu'il contient.

canvas

Permet de créer et de manipuler les canevas.

Syntaxe

canvas *cheminCanevas ?options?*

Introduction

La commande **canvas** crée une nouvelle fenêtre (indiquée par l'argument *cheminCanevas*) et en fait un composant graphique de type *canevas*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques de la case à cocher telles que la couleur, la police, le texte, le relief initial etc. La commande **canvas** renvoie la valeur de son argument *cheminCanevas*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminCanevas*, mais en revanche l'objet parent de *cheminCanevas* doit exister effectivement.

Les canevas permettent de réaliser des objets graphiques structurés. Un canevas affiche un nombre quelconque d'objets graphiques tels que rectangles, cercles, lignes ou texte. Ces éléments peuvent être manipulés et des commandes peuvent leur être associées à la manière de la commande **bind** : par exemple, une commande particulière pourrait être associée à l'événement <Button-1> de telle sorte qu'elle soit exécutée à chaque fois que le premier bouton de la souris est pressé au-dessus de cet élément. Cela signifie que les éléments d'un canevas peuvent avoir des comportements associés et définis au moyen de scripts Tcl.

Liste d'affichage

Les éléments d'un canevas sont ordonnés en vue de leur affichage : le premier élément de la liste sera affiché en premier et ainsi de suite. Un élément qui s'affiche est susceptible de masquer tout ou partie des éléments précédemment affichés, d'où l'importance de l'ordre d'affichage : lorsqu'un nouvel élément est créé, il est placé en fin de liste et s'affiche donc au-dessus des précédents. Il existe des commandes pour modifier l'ordre de cette liste et ré-arranger les divers éléments contenus dans un canevas.

Les objets de type fenêtre font exception à ces règles : le système de fenêtrage exige qu'ils soient toujours affichés au-dessus des autres objets. Pour modifier l'ordre d'empilement des fenêtres, aucune commande spécifique à chacune d'elles n'existe et il faut se servir des commandes globales de Tk **raise** et **lower** (cf. p. 120 et 172).

Identificateurs et balises

Les objets contenus dans un canevas peuvent être désignés soit au moyen d'un numéro d'identification, soit au moyen d'une balise. Chacun possède un numéro d'identification unique qui lui est attribué lors de sa création. Les numéros d'identification ne peuvent être modifiés et ne sont pas réutilisés pendant toute la durée de vie du canevas. Chaque objet possède aussi un nombre quelconque de balises qui lui sont associées : ce sont simplement des chaînes de caractères qui peuvent prendre n'importe quelle forme à l'exception de valeurs numériques : x123 est acceptable mais 123 ne l'est pas. Une même balise peut être associée à plusieurs éléments, ce qui permet de les grouper de diverses façons : on pourrait imaginer d'affecter une balise appelée *selection* à une série d'objets sélectionnés. Les deux mots-clés suivants sont réservés pour servir de balises particulières :

- *all* est réservé pour désigner implicitement tous les éléments d'un canevas.
- *current* est réservé pour désigner l'élément courant, c'est-à-dire l'élément le plus haut placé dont la zone englobe la position du pointeur de la souris. Si la souris n'est pas sur un canevas ou sur un élément d'un canevas, aucun élément ne porte la balise *current*.

Lorsqu'on désigne un élément dans une commande concernant un canevas au moyen d'une valeur numérique, Tk considère qu'il s'agit d'un numéro d'identification désignant un élément unique. Si l'on utilise une valeur non numérique, Tk l'interprète comme une balise et considère qu'il est fait référence à tous les éléments marqués par cette balise. Dans tout ce qui suit, l'abréviation *Balise/ID* indiquera une référence faite par l'une ou l'autre des deux méthodes, étant entendu qu'un numéro d'identification désigne toujours un élément unique tandis qu'une balise est susceptible d'en désigner plusieurs.

Les balises offrent de multiples possibilités de sélection car elles peuvent être utilisées dans des expressions comportant l'un des opérateurs logiques `&&`, `||`, `^` et `!` (resp. ET, OU inclusif, OU exclusif et NON), comme dans les deux expressions équivalentes suivantes :

```
.c find withtag {(a&&!b)||(!a&&b)}
.c find withtag {a^b}
```

qui permettent de désigner les éléments ayant la balise a ou la balise b, mais pas les deux.

Certaines commandes n'opèrent que sur un seul élément à la fois : si une balise correspondant à plusieurs éléments est utilisée, la commande s'appliquera au premier élément dans l'ordre de la liste d'affichage.

Coordonnées

Toutes les coordonnées utilisées dans le cadre d'un canevas sont stockées comme des nombres en virgule flottante. Coordonnées et distances sont exprimées en unités d'écran qui sont des nombres en virgule flottante éventuellement suivis d'une lettre : les lettres **m**, **c**, **i**, **p** correspondent respectivement à des millimètres, des centimètres, des pouces (*inches*) ou des points d'imprimeur (équivalents à 1/72 pouce).

Si aucune lettre n'est précisée, il s'agit de pixels. Comme il est d'usage avec les coordonnées d'écran, l'axe horizontal est orienté vers la droite et l'axe vertical vers le bas.

Transformations

L'origine du système de coordonnées d'un canevas est normalement le coin supérieur gauche de la fenêtre contenant le canevas. On peut translater ce point en utilisant les commandes **xview** et **yview** décrite ci-dessous. La translation est la seule transformation supportée par le système de coordonnées des canevas : aucun autre changement de base n'est possible par rapport au système de coordonnées de la fenêtre (ni changement d'échelle, ni rotation). On peut cependant modifier l'échelle des éléments pris individuellement (mais toutefois pas leur appliquer de rotation).

Indices

Les éléments de type *texte* utilisent la notion d'indice pour spécifier des positions particulières à l'intérieur de l'élément. Il en est de même des éléments de types *lignes* ou *polygones* : pour ceux-ci, les indices représentent des coordonnées et sont censés être toujours des nombres pairs car les coordonnées vont toujours par deux. Les indices peuvent prendre une des formes suivantes :

nombre
: Spécifie un caractère d'un élément de type texte par son indice numérique, où 0 correspond au premier caractère de la chaîne. Pour une ligne ou un polygone, si un nombre impair est utilisé, Tk l'augmente d'une unité pour en faire un nombre pair. Les valeurs négatives sont ajustées à 0 ; les valeurs qui excèdent la longueur de l'objet (en nombre de caractères dans un texte ou en nombre d'éléments dans une liste de coordonnées) sont ajustées à la longueur véritable de l'objet.

end
: Indique le caractère ou la coordonnée situés immédiatement après le dernier objet (caractère ou coordonnée). Cela correspond aussi à la longueur de cet objet en nombre de caractères ou en nombre de coordonnées.

insert
: Indique le caractère situé immédiatement après le curseur d'insertion. Ceci n'a de sens que pour les objets de type *texte*.

sel.first
: Indique le premier caractère d'une sélection. Il y a erreur à utiliser cette forme s'il n'y a pas de sélection dans l'élément.

sel.last
: Indique le caractère qui suit immédiatement le dernier de la sélection. Il y a erreur à utiliser cette forme s'il n'y a pas de sélection dans l'élément.

@*x,y*
: Cette forme se réfère à un caractère ou aux coordonnées d'un point avec x et y

spécifiés dans le système de coordonnées du canevas. Si x et y correpondent à un point situé à l'extérieur de la zone couverte par l'objet texte, Tk considère qu'il s'agit du caractère de la chaîne situé à l'extrêmité la plus proche de ce point.

Motifs pointillés

De nombreux éléments d'un canevas supportent la notion de pointillés et de tiretés. Ceux-ci sont décrits selon deux syntaxes différentes :

- au moyen de listes de valeurs entières : chaque valeur représente un nombre de pixels sur un segment. Seuls les segments impairs sont tracés ; les autres sont transparents et l'alternance des deux crée l'effet de pointillé.
- au moyen d'une liste de symboles pris parmi les cinq caractères suivants : point, virgule, tiret, caractère de soulignement et espace. L'espace est utilisée pour accroître l'espacement entre les autres symboles.

Les exemples suivants indiquent les équivalences entre les deux notations :

```
-dash .    = -dash {2 4}
-dash -    = -dash {6 4}
-dash -.   = -dash {6 4 2 4}
-dash -..  = -dash {6 4 2 4 2 4}
-dash {.}  = -dash {2 8}
-dash ,    = -dash {4 4}
```

Les deux syntaxes présentent une différence notable : la première tient compte de la largeur de ligne courante tandis que la deuxième en est indépendante.

Opérations sur les canevas

Avec l'instruction :

canvas *cheminCanevas ?options?*

la commande **canvas** crée une nouvelle commande Tcl dont le nom est précisément *cheminCanevas*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le canevas. La forme générale est :

cheminCanevas opération ?arg arg ...?

Les opérations reconnues par les canevas sont les suivantes :

cheminCanevas **addtag** *balise spécificateur ?arg arg ...?*

Pour chaque élément correspondant aux contraintes spécifiées par les arguments *spécificateur* et *arg*, cette commande ajoute une balise *balise* à la liste des balises déjà associées à l'élément si *balise* n'est pas déjà dans cette liste. Si aucun élément ne répond aux contraintes, la commande est sans effet. Cette commande renvoie une chaîne vide comme résultat.

Les arguments *spécificateur* et *arg* peuvent prendre une des formes suivantes :

above Balise/ID

Sélectionne l'élément qui suit immédiatement celui indiqué par l'argument *Balise/ID* dans la liste d'affichage. Si *Balise/ID* désigne plusieurs

éléments, c'est le dernier de la liste d'affichage (le plus haut placé) qui est utilisé.

all
Sélectionne tous les éléments dans le canevas.

below Balise/ID
Sélectionne l'élément qui précède immédiatement celui indiqué par l'argument *Balise/ID* dans la liste d'affichage. Si *Balise/ID* désigne plusieurs éléments, c'est le premier de la liste d'affichage (le plus bas placé) qui est utilisé.

closest x y ?dist? ?début?
Sélectionne l'élément le plus proche du point de coordonnées x et y. Si plusieurs éléments répondent à ce critère (c'est-à-dire si deux éléments au moins englobent ce point), c'est le plus haut placé qui est utilisé. Si *dist* est spécifié comme une valeur non négative, on considère que tout élément dont le point s'écarte d'une distance inférieure à *dist* englobe aussi ce point. Cela revient à admettre un halo autour de l'élément. L'argument *début* peut être utilisé pour désigner un élément parmi les plus proches. C'est l'élément le plus haut placé parmi ceux situés en dessous de l'indice *début* dans la liste d'affichage qui est utilisé.

enclosed x1 y1 x2 y2
Sélectionne tous les éléments inclus dans la zone rectangulaire définie par les coordonnées (x_1,y_1) et (x_2,y_2) avec $x_1 \leq y_1$ et $x_2 \leq y_2$.

overlapping x1 y1 x2 y2
Sélectionne tous les éléments qui intersectent la zone rectangulaire définie par les coordonnées (x_1,y_1) et (x_2,y_2) avec $x_1 \leq y_1$ et $x_2 \leq y_2$.

withtag Balise/ID
Sélectionne tous les éléments spécifiés par l'argument *Balise/ID*.

cheminCanevas **bbox** *Balise/ID ?Balise/ID Balise/ID ...?*
Renvoie une liste de quatre nombres décrivant la boîte qui englobe tous les éléments décrits par le ou les arguments *Balise/ID*. S'il n'y a pas d'éléments qui correspondent ou s'ils sont vides, la commande renvoie une chaîne vide. Les deux premiers éléments de cette liste sont l'abscisse et l'ordonnée du coin supérieur gauche de la zone d'écran couverte par l'élément et les deux derniers sont l'abscisse et l'ordonnée du coin inférieur droit. Les valeurs retournées peuvent être légèrement supérieures aux valeurs réelles.

cheminCanevas **bind** *Balise/ID ?séquenceEvnt? ?commande?*
Cette opération associe la commande *commande* à tous les éléments désignés par *Balise/ID* de telle sorte que dès que la suite d'événements représentée par l'argument *séquenceEvnt* se produit, la commande soit invoquée. Cette opération est analogue à la commande **bind** à la différence qu'elle agit sur un élément d'un canevas plutôt que sur un composant graphique entier. Les règles régissant les arguments *séquenceEvnt* et *commande* sont les mêmes que pour **bind** (cf. p. 13).
Si tous les arguments sont spécifiés, une nouvelle liaison est créée remplaçant

toute liaison déjà existante pour cette suite d'événements et cet élément : toutefois si le premier caractère de l'argument *commande* est un signe +, l'argument s'ajoute à la définition de la liaison plutôt que de la remplacer. Dans ce cas, la valeur de retour est une chaîne vide. Si l'argument *commande* est omis, l'opération **bind** renvoie la définition de la *commande* associée à *Balise/ID* et à la *séquenceEvnt*. Si à la fois *commande* et *séquenceEvnt* sont omis, l'opération renvoie la liste de toutes les séquences pour lesquelles des liaisons ont été définies concernant l'élément *Balise/ID*.

Les seuls événements acceptant des liaisons sont ceux relatifs à la souris et au clavier comme *Enter*, *Leave*, *ButtonPress*, *Motion* et *KeyPress* ou bien les événements virtuels. Les événements *Enter* et *Leave* se déclenchent pour un élément lorsque celui-ci devient courant ou cesse de l'être. Les événements de clavier sont dirigés vers l'élément focalisé s'il en existe un (voir l'opération **focus** ci-dessous).

Si un événement virtuel est utilisé dans une liaison, cette liaison se déclenche seulement si l'événement virtuel est défini par un événement de souris ou de clavier sous-jacent. Il peut se produire par ailleurs que plusieurs liaisons correspondent à un événement particulier si l'une a été définie en utilisant le numéro d'identification de l'élément et l'autre en utilisant une de ses balises : dans ce cas,toutes les liaisons correspondantes sont invoquées : une liaison associée à la balise **all** sera déclenchée en premier, puis celles qui correspondent aux balises de l'élément dans l'ordre, enfin celle qui est associée au numéro d'identification. Si plusieurs liaisons correspondent à une *même* balise, seule la plus spécifique est invoquée.

Si des liaisons ont été créées pour le canevas entier au moyen de la commande Tk **bind**, elles s'ajoutent à celles créées pour un élément particulier : les liaisons d'élément sont invoquées avant celles du canevas lui-même.

cheminCanevas **canvasx** *x ?granularité?*

Convertit une abscisse *x* donnée dans le système de coordonnées de la fenêtre en une abscisse de canevas. Si l'argument *granularité* est spécifié, il indique une unité : l'abscisse est arrondie au plus proche multiple de *granularité*.

cheminCanevas **canvasy** *y ?granularité?*

Convertit une ordonnée *y* donnée dans le système de coordonnées de la fenêtre en une ordonnée de canevas. Si l'argument *granularité* est spécifié, il indique une unité : l'ordonnée est arrondie au plus proche multiple de *granularité*.

cheminCanevas **cget** *option*

Renvoie la valeur courante de l'option de configuration désignée par l'argument *option*. Cet argument peut prendre une quelconque des valeurs admises par la commande **canvas**.

cheminCanevas **configure** *?option? ?valeur? ?option valeur ...?*

Permet d'obtenir ou de modifier les options de configuration du canevas. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminCanevas*.

Si l'argument *option* est spécifié sans une valeur correspondante, la commande

renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **canvas**.

cheminCanevas **coords** *Balise/ID ?x0 y0 ...?*
Interroge ou modifie les coordonnées qui définissent un objet du canevas. Si aucune coordonnée n'est spécifiée, cette commande renvoie une liste dont les éléments sont les coordonnées de l'objet désigné par *Balise/ID*. Si des coordonnées sont spécifiées, elles remplacent les coordonnées actuelles de l'objet désigné. Si *Balise/ID* fait référence à plusieurs objets du canevas, seul le premier dans la liste d'affichage est utilisé.

cheminCanevas **create** *type x y ?x y ...? ?option valeur ...?*
Crée un nouvel objet de type *type* dans le canevas *cheminCanevas*. Le format exact des arguments qui suivent **type** dépend du type souhaité : il s'agit en général des coordonnées d'un ou de plusieurs points suivies de spécifications particulières à ce type (voir plus loin les descriptions des types supportés). Cette commande renvoie le numéro d'identification de l'objet créé.

cheminCanevas **dchars** *Balise/ID premier ?dernier?*
Pour chaque élément désigné par *Balise/ID*, cette commande détruit les caractères ou les coordonnées dans l'intervalle spécifié par les arguments *premier* et *dernier* inclus : pour les objets de type texte, les indices désignent des caractères ; pour des lignes et des polygones, ils désignent des coordonnées (voir la section *Indices* à la page 195). Pour les objets qui ne connaissent pas la notion d'indices, la commande **dchars** est simplement ignorée. Si l'argument *dernier* est omis, il est identique à *premier*. Cette commande renvoie une chaîne vide.

cheminCanevas **delete** *?Balise/ID Balise/ID ...?*
Détruit tous les éléments désignés par chaque argument *Balise/ID* et renvoie une chaîne vide.

cheminCanevas **dtag** *Balise/ID ?supprBalise?*
Pour chacun des éléments désignés par *Balise/ID*, détruit la balise spécifiée par *supprBalise* de la liste de celles qui sont associées avec l'élément. Si un élément n'a pas de balise de ce nom, la commande est sans effet. Si l'argument *supprBalise* est omis, il est identique à *Balise/ID*. Cette commande renvoie une chaîne vide.

cheminCanevas **find** *spécificateur ?arg arg ...?*
Cette commande renvoie une liste de tous les éléments correspondant aux contraintes spécifiées par les arguments *spécificateur* et *arg*. Ceux-ci obéissent aux mêmes règles que pour la commande **addtag** décrite ci-dessus. Les éléments sont renvoyés dans l'ordre d'empilement, le plus bas placé venant en premier dans la liste.

cheminCanevas **focus** *?Balise/ID?*
Établit la focalisation sur l'élément du canevas désigné par *Balise/ID* pour recevoir les événements de clavier. Si *Balise/ID* fait référence à plusieurs

éléments, la focalisation est établie sur le premier de la liste d'affichage qui accepte un curseur d'insertion. Si au contraire *Balise/ID* ne fait référence à aucun élément, ou si aucun d'entre eux n'accepte de curseur d'insertion, la commande est sans effet. Si *Balise/ID* est une chaîne vide, plus aucun élément du canevas n'est focalisé. Si enfin *Balise/ID* n'est pas spécifié, la commande renvoie le numéro d'identification de l'élément actuellement focalisé ou une chaîne vide s'il n'y en a aucun.

Une fois focalisé, un élément affiche un curseur d'insertion et tous les événements de clavier lui sont adressés. En fait, pour que cela fonctionne correctement, il faut que le canevas lui-même ait été focalisé par la commande Tk de base appelée aussi **focus**, ce qui n'est pas automatique : on fait en général suivre l'opération **focus** appliquée à un élément d'un canevas d'une commande globale **focus** concernant le canevas lui-même.

cheminCanevas **gettags** *Balise/ID*

Cette commande renvoie une liste de toutes les balises associées à l'élément désigné par l'argument *Balise/ID*. Si *Balise/ID* fait référence à plusieurs éléments, la commande liste les balises du premier élément trouvé, dans l'ordre de la liste d'affichage. Si *Balise/ID* ne correspond à aucun élément, ou s'il s'agit d'un élément qui ne possède pas lui-même de balise, la commande renvoie une chaîne vide.

cheminCanevas **icursor** *Balise/ID index*

Place le curseur d'insertion pour le ou les éléments désignés par *Balise/ID* juste après le caractère d'indice *index*. Si certains des éléments désignés par *Balise/ID* n'acceptent pas de curseur d'insertion, la commande est sans effet. La section *Indices* à la page 195 décrit les diverses formes que peut prendre l'argument *index*.

Le curseur d'insertion ne sera effectivement affiché que si l'élément est focalisé grâce à une instruction **focus**, mais la position du curseur peut être fixée sans que l'élément ne soit focalisé. Cette commande renvoie une chaîne vide.

cheminCanevas **index** *Balise/ID index*

Renvoie l'indice numérique correspondant à l'argument *index* dans *Balise/ID*. La section *Indices* à la page 195 décrit les diverses formes que peut prendre l'argument *index*. La notion d'indice concerne les éléments de type texte, les lignes et les polygones. Si l'argument *Balise/ID* fait référence à plusieurs éléments, c'est le premier dans l'ordre de la liste d'affichage qui est utilisé.

cheminCanevas **insert** *Balise/ID position chaîne*

Pour chacun des éléments désignés par *Balise/ID*, si l'élément accepte l'insertion de texte ou de coordonnées, l'argument *chaîne* est inséré juste avant la position spécifiée par l'argument *position*. Pour des éléments de type texte, *chaîne* représente des caractères ; pour des lignes ou des polygones, *chaîne* représente des coordonnées. La section *Indices* à la page 195 décrit les diverses formes que peut prendre l'argument *position*. Cette commande renvoie une chaîne vide.

cheminCanevas **itemcget** *Balise/ID option*
Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option* pour l'élément désigné par *Balise/ID*. Cette commande est analogue à **cget** à la différence qu'elle s'applique à un élément de canevas plutôt qu'au canevas entier. L'argument *option* peut prendre une quelconque des valeurs admises par l'opération **create** utilisée pour créer l'élément. Si l'argument *Balise/ID* fait référence à plusieurs éléments, c'est le premier dans l'ordre de la liste d'affichage qui est utilisé.

cheminCanevas **itemconfigure** *Balise/ID ?option? ?valeur? ?option valeur ...?*
Cette commande est analogue à la commande **configure** à la différence qu'elle modifie des options pour un élément particulier désigné par *Balise/ID* plutôt que des options globales du canevas lui-même. Les règles concernant l'argument *option* sont les mêmes que pour la commande **configure**.

cheminCanevas **lower** *Balise/ID ?position?*
Déplace tous les éléments désignés par *Balise/ID* dans la liste d'affichage juste avant l'élément désigné par *position*. Si *Balise/ID* fait référence à plusieurs éléments, tous sont déplacés mais leur ordre relatif reste inchangé. L'argument *position* peut être une balise ou un numéro d'identification. S'il correspond à plusieurs éléments, c'est le premier dans la liste d'affichage qui est utilisé comme destination pour les éléments à déplacer.
Cette commande est sans effet sur les éléments de type fenêtre. Les fenêtres masquent toujours les autres types d'éléments et l'ordre d'empilement des fenêtres est contrôlé par les commandes Tk de base **raise** et **lower** (cf. p. 120 et 172) et non par les commandes de même nom concernant les canevas eux-mêmes qui sont présentées ici. Cette commande renvoie une chaîne vide.

cheminCanevas **move** *Balise/ID x y*
Déplace, dans le système de coordonnées du canevas, tous les éléments désignés par *Balise/ID* par translation de vecteur (x,y). Cette commande renvoie une chaîne vide.

cheminCanevas **postscript** *?option valeur option valeur ...?*
Cette commande génère une représentation en langage Postscript de tout ou partie du canevas. Si l'option **-file** est spécifiée, le code Postscript généré est écrit dans ce fichier et une chaîne vide est renvoyée; sinon le code Postscript est renvoyé comme valeur de retour de cette commande. Si l'interpréteur qui régit le canevas est déclaré sûr (*safe*), cette opération échouera car les interpréteurs sûrs ne sont pas autorisés à écrire des fichiers.
Si l'option **-channel** est spécifiée, l'argument désigne le nom d'un canal déjà ouvert en écriture. Dans ce cas, le code Postscript généré est écrit sur ce canal et celui-ci est laissé ouvert à la fin de l'opération.
Le code généré adopte le format dit *encapsulé* (EPS, *encapsulated Postscript*) et utilise les conventions de structuration dites DSC (*Document Structuring Conventions*) de la version 3.0 de Postscript.
Il faut être attentif au fait que le code Postscript est généré en fonction de ce qui apparaît dessiné à l'écran : si le canevas vient d'être créé mais n'apparaît

pas encore, il aura une taille initiale de 1×1 pixel et le code Postscript risque d'être vide. Pour éviter ce problème, on devra invoquer une commande **update** pour attendre que le canevas ait atteint sa taille définitive ou faire appel aux commandes **-width** et **-height** pour fixer les dimensions de l'aire du canevas à imprimer.
Enfin la commande **postscript** admet des options de type *option valeur* qui permettent de spécifier des paramètres supplémentaires pour la génération du code Postscript. Les options supportées sont les suivantes :

-colormap *nomVar*
: L'argument *nomVar* doit indiquer le nom d'une variable de type tableau qui spécifie les couleurs à utiliser dans le code Postscript. Chaque élément du tableau est une instruction Postscript qui fixe une valeur particulière de couleur. Par exemple :

```
1.0 1.0 0.0 setrgbcolor
```

Lorsqu'il produit le code Postscript, Tk vérifie si un élément du tableau *nomVar* porte le même nom de couleur. Si c'est le cas, Tk utilise la valeur trouvée comme commande Postscript pour fixer la couleur. Si cette option n'a pas été utilisée ou si aucune entrée du tableau ne correspond, Tk utilise les valeurs de rouge, vert et bleu du système X (sous Unix).

-colormode *mode*
: Spécifie la façon de produire l'information de couleur. L'argument *mode* peut prendre une des valeurs symboliques suivantes : *color* (couleurs), *gray* (niveaux de gris) ou *mono* (noir et blanc)

-file *fileName*
: Spécifie le nom d'un fichier dans lequel écrire le code Postscript produit. Si cette option n'est pas spécifiée, le Postscript est renvoyé comme valeur de retour de la commande.

-fontmap *nomVar*
: L'argument *nomVar* doit indiquer le nom d'une variable de type tableau qui spécifie les polices à utiliser dans le code Postscript. Chaque élément du tableau est une liste de deux éléments qui représentent le nom et la taille d'une police Postscript. En produisant des commandes Postscript pour une police particulière, Tk vérifie si un élément du tableau *nomVar* porte le même nom. Si c'est le cas, Tk utilise l'information trouvée, sinon il essaie de deviner au mieux la police à utiliser. Cela ne fonctionne en général qu'avec les polices usuelles telles que Times, Helvetica et Courier et seulement si le nom de la police dans le système X comporte le bon nombre de tirets et la taille de la fonte comme dans :

```
-*-Courier-Bold-R-Normal--*-120-*
```

-height *size*
: Spécifie la hauteur de l'aire du canevas à imprimer. Par défaut, c'est la hauteur du canevas lui-même qui est utilisée.

-pageanchor *ancre*
Spécifie le point de la zone imprimée du canevas à placer sur le point de positionnement de la page (spécifié par les options **-pagex** et **-pagey**). Par exemple, **-pageanchor n** signifie que le centre supérieur de la zone imprimée du canevas devrait correspondre au point de positionnement. L'emplacement par défaut est **center**.

-pageheight *taille*
Spécifie que le code Postscript doit être proportionné à la fois en x et en y de telle sorte que la zone imprimée ait la hauteur *taille* sur la page Postscript. L'argument *taille* est un nombre en virgule flottante suivi d'une lettre précisant l'unité utilisée : les lettres **m**, **c**, **i**, **p** correspondent respectivement à des millimètres, des centimètres, des pouces (*inches*) ou des points d'imprimeur (équivalents à 1/72 pouce). La valeur par défaut est la hauteur de la zone à l'écran. Si à la fois les options **-pageheight** et **-pagewidth** sont spécifiées, c'est le facteur de proportionnalité de **-pagewidth** qui est utilisé (la mise à l'échelle non uniforme n'est pas implémentée).

-pagewidth *taille*
Spécifie que le code Postscript doit être proportionné à la fois en x et y de telle sorte que la zone imprimée ait la largeur *taille* sur la page Postscript. L'argument *taille* a la même forme que pour l'option **-pageheight**. La valeur par défaut est la largeur de la zone à l'écran. Si à la fois les options **-pageheight** et **-pagewidth** sont spécifiées, c'est le facteur de proportionnalité de **-pagewidth** qui est utilisé (la mise à l'échelle non uniforme n'est pas implémentée).

-pagex *position*
L'argument *position* indique l'abscisse du point de positionnement sur la page Postscript en utilisant une des formes usuellement acceptées par l'option **-pageheight**. Cette option s'utilise en conjonction avec les options **-pagey** et **-pageanchor** pour déterminer l'emplacement de la zone à imprimer sur la page Postscript. Par défaut, il s'agit du centre de la page.

-pagey *position*
L'argument *position* indique l'ordonnée du point de positionnement sur la page Postscript en utilisant une des formes acceptées par l'option **-pageheight**. Cette option s'utilise en conjonction avec les options **-pagex** et **-pageanchor** pour déterminer l'emplacement de la zone à imprimer sur la page Postscript. Par défaut, il s'agit du centre de la page.

-rotate *bool*
L'argument *bool* vaut 0 ou 1 et spécifie si la zone à imprimer doit être pivotée de 90 degrés. S'il n'y a pas rotation, l'axe des abscisses de la zone imprimée est la petite dimension de la page (format portrait) ; sinon, il correspond à la grande dimension (format paysage). Par défaut il n'y a pas de rotation.

-width *taille*
Spécifie la largeur de l'aire du canevas à imprimer. Par défaut, c'est la largeur du canevas lui-même qui est utilisée.

-x *position*
Spécifie l'abscisse du bord gauche de la zone du canevas qui doit être imprimée. Celle-ci est exprimée dans le système de coordonnées du canevas. Par défaut, il s'agit de l'abscisse du bord gauche de la fenêtre.

-y *position*
Spécifie l'ordonnée du bord supérieur de la zone du canevas qui doit être imprimée. Celle-ci est exprimée dans le système de coordonnées du canevas. Par défaut, il s'agit de l'ordonnée du bord supérieur de la fenêtre.

cheminCanevas **raise** *Balise/ID ?position?*
Déplace tous les éléments désignés par *Balise/ID* dans la liste d'affichage immédiatement après l'élément désigné par *position*. Si *Balise/ID* fait référence à plusieurs éléments, tous sont déplacés mais leur ordre relatif reste inchangé. L'argument *position* peut être une balise ou un numéro d'identification. S'il correspond à plusieurs éléments, c'est le dernier dans la liste d'affichage (le plus haut placé) qui est utilisé comme destination pour les éléments à déplacer.
Cette commande est sans effet sur les éléments de type fenêtre. Les fenêtres masquent toujours les autres types d'éléments et l'ordre d'empilement des fenêtres est contrôlé par les commandes Tk de base **raise** et **lower** (cf. p. 120 et 172) et non pas par les commandes de même nom concernant les canevas eux-mêmes qui sont présentées ici. Cette commande renvoie une chaîne vide.

cheminCanevas **scale** *Balise/ID xOrigine yOrigine xProp yProp*
Remet à l'échelle tous les l'éléments désignés par *Balise/ID* dans le système de coordonnées du canevas. Les arguments *xOrigine* et *yOrigine* désignent le centre de la transformation et les arguments *xProp* et *yProp* correspondent respectivement aux facteurs de proportionnalité en abscisse et en ordonnée. Cette commande renvoie une chaîne vide.

cheminCanevas **scan** *option args*
Cette commande est utilisée pour parcourir un canevas. Elle admet deux formes différentes en fonction de la sous-commande qu'on lui ajoute :

cheminCanevas **scan mark** *x y*
Enregistre les valeurs x et y et la vue courante du canevas en vue d'une utilisation ultérieure avec une commande **scan dragto**. Cette commande est typiquement associée à une pression du bouton de la souris dans le canevas et x et y sont les coordonnées du pointeur de la souris. Elle renvoie une chaîne vide.

cheminCanevas **scan dragto** *x y ?mult?* .
Cette commande calcule la différence entre les valeurs des arguments x et y et celles des arguments x et y de la dernière commande **scan**

mark exécutée pour ce canevas. Elle renvoie une chaîne vide. La vue du contenu du canevas est alors ajustée d'une valeur multiple de cette différence vers la gauche ou vers la droite. Ce multiple est précisé par l'argument *mult* et vaut 10 par défaut. Cette commande est typiquement associée à des déplacements de la souris à l'intérieur du canevas pour simuler l'effet de déplacement rapide du canevas dans sa fenêtre. On associe fréquemment les deux commandes **scan mark** et **scan dragto** au second bouton de la souris, comme ceci :

```
bind $c <2> "$c scan mark %x %y"
bind $c <B2-Motion> "$c scan dragto %x %y"
```

cheminCanevas **select** *option ?Balise/ID arg?*

Manipule une sélection de diverses manières en fonction de l'argument *option*. La commande peut prendre une des formes décrites ci-dessous. Dans tout ce qui suit l'argument *Balise/ID* désigne un élément qui supporte à la fois l'indexation (cf. la section *Indices* à la page 195) et la sélection.

cheminCanevas **select adjust** *Balise/ID index*

Place la fin de la sélection dans l'élément *Balise/ID* au point le plus proche du caractère d'indice *index* et ajuste cette fin de sélection sur ce caractère inclusivement. L'autre extrêmité de la sélection devient le point d'ancrage pour de futures opérations **select to**. Si la sélection n'est pas actuellement dans l'élément *Balise/ID*, cette commande se comporte de la même manière que la commande **select to** du canevas proprement dit. Renvoie une chaîne vide.

cheminCanevas **select clear**

Efface la sélection si elle est dans ce canevas. Si la sélection n'est pas dans le canevas, la commande est sans effet. Renvoie une chaîne vide.

cheminCanevas **select from** *Balise/ID index*

Fixe le point d'ancrage de la sélection immédiatement avant le caractère d'indice *index* dans l'élément désigné par *Balise/ID*. Cette commande ne modifie pas la sélection ; elle en fixe simplement l'extrêmité pour de futures commandes **select to** et renvoie une chaîne vide.

cheminCanevas **select item**

Renvoie le numéro d'identification de l'élément sélectionné si la sélection est dans un élément de ce canevas. Sinon, une chaîne vide est renvoyée.

cheminCanevas **select to** *Balise/ID index*

Fixe la sélection à l'ensemble des caractères de l'élément *Balise/ID* entre le point d'ancrage et l'indice *index*. La nouvelle sélection inclut le caractère désigné par *index* ; elle inclut le caractère correspondant au point d'ancrage uniquement si l'indice *index* est supérieur ou égal à ce point d'ancrage. Le point d'ancrage est déterminé par la plus récente commande **select adjust** ou **select from** exécutée pour ce canevas. Si le point d'ancrage de la sélection dans ce canevas n'est pas actuellement

dans l'élément *Balise/ID*, il est identique au caractère d'indice *index*. La commande renvoie une chaîne vide.

cheminCanevas **type** *Balise/ID*
Renvoie le type de l'élément désigné par *Balise/ID*, comme par exemple **rectangle** ou **text**. Si *Balise/ID* fait référence à plusieurs éléments, c'est le type du premier élément dans la liste d'affichage qui est renvoyé. Si *Balise/ID* ne désigne aucun élément, une chaîne vide est renvoyée.

cheminCanevas **xview** *?args?*
Cette commande est utilisée obtenir ou modifier la position horizontale de la partie du canevas affichée dans la fenêtre. Elle peut prendre une des formes suivantes :

cheminCanevas **xview**
Renvoie une liste de deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : ces valeurs indiquent horizontalement les proportions de l'élément qui sont visibles ou non dans la fenêtre. Par exemple, si les valeurs renvoyées sont 0,2 et 0,6 cela signifie que 20% de la zone du canevas (telle que définie par l'option **-scrollregion**) sont situés à gauche de la partie visible et 40% à droite, tandis qu'entre les deux la partie visible correspond à 40% du canevas. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de l'option **-xscrollcommand**.

cheminCanevas **xview moveto** *fraction*
Ajuste la vue dans la fenêtre du canevas de telle sorte que la portion désignée en pourcentage de la largeur totale par la fraction *fraction* soit masquée par le bord gauche de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1].

cheminCanevas **xview scroll** *nb unité*
Cette commande décale la vue dans la fenêtre du canevas vers la gauche ou vers la droite selon la valeur donnée aux arguments *nb* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage se fait en unités de l'option **-xscrollincrement** si elle est positive, ou en dixièmes de la largeur de la fenêtre dans le cas contraire. Avec *pages*, l'unité de décalage correspond aux neuf dixièmes de la largeur de la fenêtre. Si *nb* est négatif, la partie située au-delà du bord gauche devient visible ; s'il est positif, la partie au-delà du bord droit devient visible.

cheminCanevas **yview** *? args?*
Cette commande est utilisée pour obtenir ou modifier la position verticale de la partie du canevas affichée dans la fenêtre. Elle peut prendre une des formes suivantes :

cheminCanevas **yview**
Renvoie une liste de deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : ces valeurs indiquent verticalement les proportions de

l'élément qui sont visibles ou non dans la fenêtre. Par exemple si les valeurs renvoyées sont 0,6 et 1,0, cela signifie que les 40% inférieurs de la zone du canevas sont visibles dans la fenêtre. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de l'option **-yscrollcommand**.

cheminCanevas **yview moveto** *fraction*
Ajuste la vue dans la fenêtre du canevas de telle sorte que la portion désignée en pourcentage de la hauteur totale par la fraction *fraction* soit masquée par le bord supérieur de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1].

cheminCanevas **yview scroll** *nombre what*
Cette commande décale la vue dans la fenêtre du canevas vers le haut ou vers le bas selon la valeur donnée aux arguments *nombre* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage se fait en unités de l'option **-yscrollincrement** si elle est positive, ou en dixièmes de la hauteur de la fenêtre dans le cas contraire. Avec *pages*, l'unité de décalage correspond aux neuf dixièmes de la hauteur de la fenêtre. Si *nb* est négatif, la partie située au-delà du bord supérieur devient visible ; s'il est positif, la partie au-delà du bord inférieur devient visible.

Les différents objets de canevas

Les sections qui suivent décrivent les différents types d'éléments qui peuvent être dessinés à l'intérieur d'un canevas[2]. Pour chacun d'eux, on précisera d'une part la forme de la syntaxe qui permet de les créer et d'autre part les options de configuration qui peuvent leur être attribuées avec les opérations **create** et **itemconfigure**. La plupart de ces éléments n'acceptent pas la sélection et l'indexation : pour le moment seuls les textes, les lignes et les polygones supportent ces notions qui sont utilisées par certaines opérations telles que **index** et **insert**.

Options communes des objets

De nombreux éléments partagent un ensemble d'options décrites dans cette section. Les options réellement spécifiques à un élément particulier sont décrites ci-dessous en même temps que les éléments eux-mêmes. Le tableau 2 de la page suivante rassemble les options communes et précise les éléments auxquels on peut les attribuer.

-dash *motif*
-activedash *motif*
-disableddash *motif*
Ces options spécifient les motifs pointillés correspondant respectivement aux états normal, activé et désactivé d'un élément. L'argument *motif* peut prendre

2. Une application peut étendre la panoplie des éléments de canevas par des extensions compilées. On se reportera pour cela à la documentation de la fonction **Tk_CreateItemType**.

	arc	*line*	*oval*	*polygons*	*rectangle*	*text*	*bitmap*	*image*	*window*
-dash	•	•	•	•	•				
-activedash	•	•	•	•	•				
-disableddash	•	•	•	•	•				
-dashoffset	•	•	•	•	•				
-fill	•	•	•	•	•	•			
-activefill	•	•	•	•	•	•			
-disabledfill	•	•	•	•	•	•			
-offset	•	•	•	•	•				
-outline	•		•	•	•				
-activeoutline	•		•	•	•				
-disabledoutline	•		•	•	•				
-outlinestipple	•		•	•	•				
-activeoutlinestipple	•		•	•	•				
-disabledoutlinestipple	•		•	•	•				
-stipple	•		•	•	•	•			
-activestipple	•	•	•	•	•	•			
-disabledstipple	•	•	•	•	•	•			
-state	•	•	•	•	•	•	•	•	•
-tags	•	•	•	•	•	•	•	•	•
-width	•	•	•	•	•				
-activewidth	•	•	•	•	•				
-disabledwidth	•	•	•	•	•				

TAB. 2 – *Options standard des éléments de canevas*

une quelconque des formes acceptées pour la désignation des tiretés (voir au début de ce chapitre). Si ces options sont omises, une ligne pleine est utilisée par défaut pour les contours.

-dashoffset *décalage*

Cette option spécifie le *décalage* initial en pixels dans le motif fourni par l'option **-dash**. **-dashoffset** est ignorée s'il n'y pas de motif **-dash**. L'argument *décalage* peut prendre une quelconque des formes décrites dans la section Coordonnées à la page 32.

-fill *couleur*

-activefill *couleur*

-disabledfill *couleur*

Spécifie la couleur à utiliser pour remplir l'aire de l'élément dans les états normal, activé et désactivé respectivement. L'argument *couleur* peut prendre une quelconque des formes acceptées pour la désignation des couleurs. Si *couleur* est une chaîne vide (c'est la valeur par défaut), l'élément ne sera pas rempli. Pour une ligne, cela spécifie la couleur de la ligne tracée et pour les

textes, la couleur de premier plan des caractères.

-outline *couleur*

-activeoutline *couleur*

-disabledoutline *couleur*

Cette option spécifie la couleur à utiliser pour dessiner le contour de l'élément dans les états normal, activé et désactivé respectivement. L'argument *couleur* peut prendre une quelconque des formes acceptées pour la désignation des couleurs. La valeur par défaut est **black**. Si *couleur* est spécifié comme une chaîne vide, aucun contour n'est tracé pour l'élément.

-offset *décalage*

Spécifie le décalage des pointillés. On exprime la valeur du décalage sous les formes d'une paire de coordonnées *x,y* ou en précisant un point du contour au moyen d'une des valeurs suivantes : **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw** ou **center**. Dans le premier cas, l'origine est celle de la fenêtre au niveau supérieur. Pour le canevas lui-même et ses éléments, l'origine est le point supérieur gauche du canevas mais si l'on place un symbole # devant la paire de coordonnées, on indique que c'est l'origine de la fenêtre qui doit être utilisée à la place. L'option **-offset** pour les pointillés s'applique également aux objets canevas. Pour les lignes et les polygones, on peut aussi spécifier un indice comme argument, qui connecte l'origine des pointillés à l'un des points de la ligne ou du polygone.

-outlinestipple *bitmap*

-activeoutlinestipple *bitmap*

-disabledoutlinestipple *bitmap*

Cette option spécifie les motifs de grisé à utiliser pour tracer le contour d'un élément en fonction des états normaux, activés et désactivés respectivement. L'argument indique le motif à utiliser pour les grisés et peut prendre une quelconque des formes acceptées pour la désignation des bitmaps. Si l'option **-outline** n'a pas été spécifiée, cette option est sans effet.Si *bitmap* est une chaîne vide (valeur par défaut), le contour est une ligne pleine.

-stipple *bitmap*

-activestipple *bitmap*

-disabledstipple *bitmap*

Cette option spécifie les motifs de grisé à utiliser pour remplir l'élément en fonction des états normal, activé et désactivé respectivement. L'argument indique le motif à utiliser pour les grisés et peut prendre une quelconque des formes acceptées pour la désignation des bitmaps. Si l'option **-outline** n'a pas été spécifiée, cette option est sans effet. Si *bitmap* est une chaîne vide (valeur par défaut), le remplissage est uni. Pour les textes, cette option affecte les caractères eux-mêmes.

-state *state*

Cette option permet de modifier pour un élément particulier l'option **-state** en vigueur globalement pour l'ensemble du canevas. Elle peut prendre une des valeurs symboliques suivantes : *normal*, *disabled* ou *hidden*.

-tags *listeBalises*
: Spécifie un ensemble de balises (*tags*) à appliquer à un élément. L'argument *listeBalises* est une liste de noms de balises, qui remplacent toute balise déjà existante pour cet élément. La liste peut être vide.

-width *épaisseur*
-activewidth *épaisseur*
-disabledwidth *épaisseur*
: Spécifie l'épaisseur du contour à dessiner autour de l'élément, en fonction des états normal, activé et désactivé respectivement. L'argument *épaisseur* peut rendre une quelconque des formes décrites à la section Coordonnées à la page 32. Si **-outline** est spécifiée comme une chaîne vide alors cette option est sans effet. La valeur par défaut est 1.0. Pour les arcs, les contours épais sont tracés centrés sur le bord de l'arc.

Arcs

Un arc est une section d'ovale délimitée par deux angles (spécifiés au moyen des options **-start** et **-extent**) et affichée de diverses façons en fonction de l'option **-style**. Les arcs sont créés par une syntaxe de la forme :

cheminCanevas **create arc** *x1 y1 x2 y2 ?option valeur option valeur ...?*

Les arguments *x1*, *y1*, *x2* et *y2* représentent les coordonnées de deux points diamétralement opposés d'un rectangle entourant l'ovale.

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les arcs sont indiquées dans le tableau 2 à la page 46. Les arcs supportent en outre les options suivantes :

-extent *degrés*
: Spécifie la taille du secteur angulaire occupé par l'arc. Cette section est exprimée en degrés dans le sens contraire des aiguilles d'une montre en partant de l'angle spécifié au moyen de l'option **-start**. L'argument *degrés* peut prendre des valeurs négatives. Les valeurs sont ajustées modulo 360°.

-start *degrés*
: Spécifie le début du secteur angulaire occupé par l'arc. La mesure des angles se fait comme avec l'option **-extent**.

-style *type*
: Spécifie la façon de tracer l'arc. Si l'argument *type* est *pieslice* (valeur par défaut), la région consiste en une section du périmètre ovale et deux rayons tracés du centre de l'ovale aux extrêmités de l'arc. Si l'argument *type* est *chord*, la région consiste en une section du périmètre ovale et la corde qui sous-tend cet arc. Si l'argument *type* est *arc*, seule la section de l'ovale est tracée. Dans ce cas, une option **-fill** sera sans effet.

Bitmaps

Les objets de type **bitmap** sont des images bichromes, c'est-à-dire possédant une couleur de fond et une couleur de premier plan. Les bitmaps sont créés par une syntaxe de la forme :

cheminCanevas **create bitmap** *x y ?option valeur option valeur ...?*

Les arguments *x* et *y* spécifient les coordonnées d'un point utilisé pour positionner le bitmap sur l'écran (voir l'option **-anchor**).

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les bitmaps sont indiquées dans le tableau 2 à la page 46. Les bitmaps supportent en outre les options suivantes :

-anchor *position*
: L'argument *position* indique la façon de placer le bitmap par rapport au point de positionnement de l'élément. Il peut prendre une des valeurs usuelles pour les ancres (*n, ne, e, se, s, sw, w, nw* ou *center*). Par exemple, si *position* est *center*, le bitmap est centré sur le point ; si *position* est **n**, le bitmap sera tracé de telle sorte que son point supérieur (centre du bord supérieur) corresponde au point de positionnement. La valeur par défaut est *center*.

-background *couleur*
-activebackground *bitmap*
-disabledbackground *bitmap*
: Spécifie la couleur à utiliser pour chacun des pixels nuls du bitmap en fonction des états normal, activé et désactivé respectivement. L'argument *couleur* peut prendre une quelconque des formes acceptées pour la désignation des couleurs. Si cette option n'est pas spécifiée, ou si elle est spécifiée comme une chaîne vide, rien n'est affiché là où les pixels ont la valeur 0, ce qui crée un effet de transparence.

-bitmap *bitmap*
-activebitmap *bitmap*
-disabledbitmap *bitmap*
: Spécifie les bitmaps à afficher dans l'élément en fonction des états normal, activé et désactivé respectivement. L'argument *bitmap* peut prendre une quelconque des formes acceptées pour la désignation des bitmaps.

-foreground *couleur*
-activeforeground *bitmap*
-disabledforeground *bitmap*
: Spécifie la couleur à utiliser pour chacun des pixels de valeur 1 du bitmap en fonction des états normal, activé et désactivé respectivement. L'argument *couleur* peut prendre une quelconque des formes acceptées pour la désignation des couleurs. La valeur par défaut est *black*.

Images

les objets de type **image** sont utilisés pour afficher des images sur un canevas. Les images sont créées par une syntaxe de la forme :

cheminCanevas **create image** *x y ?option valeur option valeur ...?*

Les arguments *x* et *y* spécifient les coordonnées d'un point utilisé pour positionner l'image sur l'écran (voir l'option **-anchor**).

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les images sont indiquées dans le tableau 2 à la page 46. Les images supportent en outre les options suivantes :

-anchor *position*
: L'argument *position* indique la façon de placer l'image par rapport au point de positionnement de l'élément. Il peut prendre une des valeurs usuelles pour les ancres (*n, ne, e, se, s, sw, w, nw* ou *center*). Par exemple, si *position* est *center*, l'image est centrée sur le point ; si *position* est **n**, l'image sera tracée de telle sorte que son point supérieur (centre du bord supérieur) corresponde au point de positionnement. La valeur par défaut est *center*.

-image *nom*
-activeimage *nom*
-disabledimage *nom*
: Spécifie le nom des images à afficher dans l'élément en fonction des états normal, activé et désactivé respectivement. Les images doivent avoir été créées antérieurement au moyen d'une commande **image create**.

Lignes

Les objets de type **line** apparaisent comme une succession de segments ou de courbes connexes. Les lignes supportent les opérations utilisant la notion d'indexation telles que **dchars**, **index** et **insert**. Elles sont créées par une syntaxe de la forme :

Les arguments de *x1* à *yn* indiquent les coordonnées d'une série de plusieurs points qui servent de points de jonction aux segments.

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les lignes sont indiquées dans le tableau 2 à la page 46. Les lignes supportent en outre les options suivantes :

-arrow *position*
: Cette option indique si des flèches sont à tracer à l'une ou l'autre des extêmités

de la ligne. L'argument *position* peut prendre une des valeurs symboliques suivantes : *none* (pas de flèche), *first* (une flèche à la première extrêmité), *last* (une flèche à la seconde extrêmité), ou *both* (une flèche aux deux extrêmités). La valeur par défaut est *none*.

-arrowshape *forme*
Cette option indique la façon de tracer les flèches. L'argument *forme* doit être une liste de trois éléments dont chacun spécifie une distance exprimée sous l'une des formes expliquées à la section *Coordonnées* à la page 32. Le premier élément est la distance le long de l'axe entre le col et la pointe de la flèche. Le second élément indique la distance le long de la ligne entre les points terminaux de la tête de la flèche et la pointe. Le troisième élément indique l'écart entre ces points terminaux et le bord extérieur de la ligne. Si cette option n'est pas spécifiée, Tk choisit une forme par défaut.

-capstyle *style*
Spécifie la manière de tracer les terminaisons aux extrêmités de la ligne. L'argument *style* peut prendre une des valeurs *butt*, *projecting* ou *round*. Si cette option n'est pas spécifiée, la valeur par défaut est *butt*. Une option **-capstyle** sera ignorée aux endroits où des flèches sont dessinées.

-joinstyle *style*
Spécifie la manière de tracer les jonctions aux sommets de la ligne. L'argument *style* peut prendre une des valeurs *bevel*, *miter* ou *round*. Si cette option n'est pas spécifiée, la valeur par défaut est *miter*. Si la ligne n'a que deux points, cette option est sans effet.

-smooth *bool*
L'argument *bool* est une valeur booléenne. L'option indique si la ligne doit être tracée ou non comme une courbe. Si c'est le cas, le tracé se fait au moyen de splines paraboliques : un spline est dessiné pour les premier et deuxième segments, puis un autre pour les deuxième et troisième et ainsi de suite. Des segments droits peuvent être obtenus au sein d'une courbe en dupliquant les extrêmités désirées.
Depuis la version 8.4 de Tcl, on peut spécifier en argument une méthode de tracé au lieu d'une valeur booléenne. Pour le moment, seule la valeur *bezier* est supportée.

-splinesteps *nb*
Spécifie le degré de lissage désiré pour les courbes : chaque spline sera approximé au moyen de *nb* segments linéaires. Cette option n'est utilisée que si l'option **-smooth** est vraie.

Ovales

Chaque ovale peut avoir un contour, un remplissage ou les deux. Les ovales sont créés par une syntaxe de la forme :

cheminCanevas **create oval** *x1 y1 x2 y2 ?option valeur option valeur ...?*

Les arguments *x1*, *y1*, *x2* et *y2* donnent les coordonnées de deux points diamétralement opposés d'un rectangle entourant l'ovale. Si le rectangle est un carré, l'ovale

obtenu sera un cercle.

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les ovales sont indiquées dans le tableau 2 à la page 46.

Polygones

Les objets de type **polygon** sont des lignes fermées faites de segments linéaires ou courbes et possédant un remplissage. Les polygones supportent les opérations utilisant la notion d'indexation telles que **dchars**, **index** et **insert**. Les polygones sont créés par une syntaxe de la forme :

cheminCanevas **create polygon** *x1 y1 ... xn yn ?option valeur option valeur ...?*

Les arguments de *x1* à *yn* indiquent les coordonnées d'une série de trois points distincts au minimum qui définissent les sommets du polygone. Le dernier point n'a pas à être identique au premier : Tk se charge lui-même de refermer la courbe polygonale.

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les polygones sont indiquées dans le tableau 2 à la page 46. Les polygones supportent en outre les options suivantes :

-joinstyle *style*
: Spécifie la manière de tracer les jonctions aux sommets du polygone. L'argument *style* peut prendre une des valeurs *bevel*, *miter* ou *round*. Si cette option n'est pas spécifiée, la valeur par défaut est *miter*.

-smooth *bool*
: L'argument *bool* est une valeur booléenne. L'option indique si le polygone doit être tracé ou non avec des arêtes courbes. Si c'est le cas le tracé se fait au moyen de splines paraboliques : un spline est dessiné pour les premier et deuxième segments, puis un autre pour les deuxième et troisième et ainsi de suite. Des segments droits peuvent être obtenus au sein d'une courbe en dupliquant les extrêmités désirées.

-splinesteps *nb*
: Spécifie le degré de lissage désiré pour les courbes : chaque spline sera approximé au moyen de *nb* segments linéaires. Cette option n'est utilisée que si l'option **-smooth** est vraie.

Les objets de type polygone se distinguent des rectangles, ovales et arcs par la manière dont ils considèrent les points intérieurs à la région qu'ils définissent. Pour un rectangle, un ovale ou un arc, un point est considéré comme intérieur seulement

s'il est dans la région délimitée et si cette région a un remplissage. Un polygone n'a pas à être rempli pour qu'un point qu'il englobe soit considéré comme intérieur. Cette notion de points intérieurs ou non est utilisée dans des opérations telles que **find closest** ou **find overlapping**.

Rectangles

Les objets de type **rectangle** peuvent avoir un contour, un remplissage ou les deux. Les polygones sont créés par une syntaxe de la forme :

cheminCanevas **create rectangle** *x1 y1 x2 y2 ?option valeur option valeur ...?*

Les arguments *x1*, *y1*, *x2* et *y2* donnent les coordonnées de deux points diamétralement opposés du rectangle. Un rectangle inclut ses bords supérieur et gauche mais pas les bords inférieur et droit.

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les rectangles sont indiquées dans le tableau 2 à la page 46. Il n'y a pas d'options spécifiques.

Textes

Un élément de canevas de type texte permet d'afficher une chaîne de caractères à l'écran à l'intérieur du canevas sur une ou plusieurs lignes. Les éléments de type texte supportent les opérations utilisant la notion d'indexation et de sélection telles que **dchars**, **focus**, **icursor**, **insert** et **select**. Ils sont créés par une syntaxe de la forme :

cheminCanevas **create text** *x y ?option valeur option valeur ...?*

Les arguments *x* et *y* spécifient les coordonnées d'un point utilisé pour positionner le texte sur l'écran.

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces options précisent en particulier la manière dont le texte doit être affiché. Les mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les éléments de type texte sont indiquées dans le tableau 2 à la page 46. Les éléments texte supportent en outre les options suivantes :

-anchor *position*

L'argument *position* indique la façon de placer le texte par rapport au point de positionnement de l'élément. Il peut prendre une des valeurs usuelles pour les ancres (*n, ne, e, se, s, sw, w, nw* ou *center*). Par exemple, si *position* est *center*, le texte est centré sur le point ; si *position* est **n**, le texte sera tracé de telle sorte que le centre du bord supérieur de la région rectangulaire

qu'il occupe corresponde au point de positionnement. La valeur par défaut est *center*.

-font *nomPolice*
Spécifie la police à utiliser. L'argument *nomPolice* peut prendre une quelconque des formes acceptées pour la désignation d'une police. Si cette option n'est pas spécifiée, la valeur par défaut sera une police dépendant du système.

-justify *justif*
Spécifie la façon dont se fait la justification du texte dans le rectangle qui le contient. L'argument *justif* peut prendre une des valeurs symboliques *left*, *right*, ou *center*. Cette option n'a de sens que si le texte occupe plusieurs lignes. La valeur par défaut est *left*.

-text *chaîne*
L'argument *chaîne* indique les caractères à afficher. Des symboles de saut de ligne `\r` peuvent être utilisés. Les caractères inclus dans un objet de type texte peuvent aussi être manipulés au moyen des opérations **insert** ou **delete**. La valeur par défaut est une chaîne vide.

-width *longueurLigne*
Spécifie une longueur maximale de ligne pour le texte. Si cette option vaut 0 (valeur par défaut) le texte ne sera coupé qu'aux éventuels caractères de saut de ligne. Sinon, les coupures nécessaires se font aux espaces entre les mots.

Fenêtres

Les objets de type *window* affichent une fenêtre à un endroit donné sur le canevas. Ils sont créés par une syntaxe de la forme :

cheminCanevas **create window** *x y ?option valeur option valeur ...?*

Les arguments *x* et *y* spécifient les coordonnées d'un point utilisé pour positionner la fenêtre sur l'écran (voir l'option **-anchor**).

Après ces coordonnées, il peut y avoir un nombre quelconque de couples *option-valeur*, chacun permettant de fixer des options de configuration pour cet élément. Ces mêmes paires *option-valeur* peuvent être également invoquées au moyen de la commande **itemconfigure**.

Les options standard supportées par les fenêtres sont indiquées dans le tableau 2 à la page 46. Les fenêtres supportent en outre les options suivantes :

-anchor *position*
L'argument *position* indique la façon de placer la fenêtre par rapport au point de positionnement de l'élément. Il peut prendre une des valeurs usuelles pour les ancres (*n, ne, e, se, s, sw, w, nw* ou *center*). Par exemple, si *position* est *center*, la fenêtre est centrée sur le point ; si *position* est **n**, la fenêtre sera tracée de telle sorte que son point supérieur (centre du bord supérieur) corresponde au point de positionnement. La valeur par défaut est *center*.

-height *pixels*
Spécifie la hauteur de la fenêtre. L'argument *pixels* peut prendre une des formes expliquées à la section *Coordonnées* à la page 32. Si cette option

-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightcolor	Highlightcolor
-highlightthickness	highlightThickness	HighlightThickness
-insertbackground	insertBackground	Foreground
-insertborderwidth	insertBorderWidth	BorderWidth
-insertofftime	insertOffTime	OffTime
-insertontime	insertOnTime	OnTime
-insertwidth	insertWidth	InsertWidth
-relief	relief	Relief
-selectbackground	selectBackground	Foreground
-selectborderwidth	selectBorderWidth	BorderWidth
-selectforeground	selectForeground	Background
-state	state	State
-takefocus	takeFocus	TakeFocus
-xscrollcommand	xScrollCommand	ScrollCommand
-yscrollcommand	yScrollCommand	ScrollCommand

TAB. 3 – *Options standard reconnues par les composants de type* canvas.

n'est pas spécifiée, ou est spécifiée comme une chaîne vide, la fenêtre se voit attribuer la hauteur qu'elle requiert naturellement.

-width *pixels*
Spécifie la largeur de la fenêtre. L'argument *pixels* peut prendre une des formes expliquées à la section *Coordonnées* à la page 32. Si cette option n'est pas spécifiée, ou est spécifiée comme une chaîne vide, la fenêtre se voit attribuer la largeur qu'elle requiert naturellement.

-window *cheminCanevas*
Spécifie la fenêtre à associer à cet élément. La fenêtre spécifiée par *cheminCanevas* doit être soit un descendant du canevas, soit un descendant d'un parent du canevas. Ce ne peut jamais être la fenêtre de premier niveau.

Les fenêtres comportent une limitation par rapport aux autres objets d'un canevas : aucune des autres figures ne peut être tracée au-dessus d'un objet fenêtre. Un tel objet masque toutes les parties d'objets qui l'intersectent, quelle que soit leur position dans l'ordre d'empilement.

Options standard des canevas

Les options standard supportées par les composants de type *canvas* sont rassemblées dans le tableau 3. On ne doit pas les confondre avec les options standard ou spécifiques qui permettent d'affecter tel ou tel type d'objet graphique appartenant à un canevas (arc, rectangle etc.).

Options spécifiques des canevas

Les canevas possèdent, au niveau global, les options spécifiques suivantes :

-closeenough **closeEnough** **CloseEnough**

Spécifie une valeur en virgule flottante indiquant à partir de quelle distance d'un objet le curseur de la souris est considéré comme étant « à l'intérieur » de cet objet. La valeur par défaut est *1.0*.

-confine **confine** **Confine**

Spécifie une valeur booléenne indiquant s'il est permis de dessiner des objets du canevas hors de la région définie par l'option **-scrollRegion**. La valeur par défaut est *true* ce qui signifie en l'occurence que les objets sont confinés à cette région.

-height **height** **Height**

Spécifie la hauteur de fenêtre que le canevas demandera au système de fenêtrage. La valeur peut être spécifiée dans une quelconque des unités admises (voir ci-dessus p 32).

-scrollregion **scrollRegion** **ScrollRegion**

Spécifie une liste de quatre coordonnées décrivant les coordonnées gauche, supérieure, droite et inférieure d'une région rectangulaire. Cette région est utilisée pour manipuler le canevas avec des barres de défilement et est considérée comme la frontière du canevas. Chaque coordonnée peut être spécifiée dans une quelconque des unités admises (voir ci-dessus p. 32).

-state **state** **State**

Spécifie l'un des trois états désignés par les constantes *normal*, *active* ou *disabled* pour le canevas. Les objets individuels du canevas peuvent avoir leur propre option *state* qui supplante l'option globale. De nombreuses options qui modifient l'apparence et l'état d'un élément peuvent être spécifiées séparément.

De nombreuses options peuvent prendre des spécifications séparées de telle sorte que l'apparence de l'élément pourra être différente dans des circonstances différentes. Les options dont le nom commence par *active* contrôlent l'apparence des objets lorsque le pointeur de la souris est au-dessus tandis que les options dont le nom commence par *disabled* contrôlent l'apparence des objets lorsqu'ils sont désactivés. Des objets de canevas désactivés ne répondent plus aux liaisons.

-width **width** **Width**

Spécifie la largeur de fenêtre que le canevas demandera au système de fenêtrage. La valeur peut être spécifiée dans une quelconque des unités admises (voir ci-dessus p 32).

-xscrollincrement xScrollIncrement ScrollIncrement

Spécifie un incrément pour le défilement horizontal. Si la valeur de cette option est positive, la vue horizontale dans la fenêtre sera placée de telle sorte que l'abscisse du bord gauche de la fenêtre dans le canevas soit un multiple entier de *xScrollIncrement*. D'autre part, l'unité de défilement lorsque les flèches gauche et droite de la barre de défilement sont cliquées sera égale à *xScrollIncrement*. Si la valeur est négative ou nulle, les contraintes sont supprimées.

-yscrollincrement yScrollIncrement ScrollIncrement

Spécifie un incrément pour le défilement vertical. Si la valeur de cette option est positive, la vue verticale dans la fenêtre sera placée de telle sorte que l'ordonnée du bord supérieur de la fenêtre dans le canevas soit un multiple entier de *yScrollIncrement*. D'autre part, l'unité de défilement lorsque les flèches haut et bas de la barre de défilement sont cliquées sera égale à *yScrollIncrement*. Si la valeur est négative ou nulle, les contraintes sont supprimées.

checkbutton

Permet de créer et de manipuler les cases à cocher.

Syntaxe

checkbutton *cheminCase ?options?*

Description

La commande **checkbutton** crée une nouvelle fenêtre (indiquée par l'argument *cheminCase*) et en fait un composant graphique de type *case à cocher*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques de la case à cocher telles que la couleur, la police, le texte, le relief initial etc. La commande **checkbutton** renvoie la valeur de son argument *cheminCase*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminCase*, mais en revanche l'objet parent de *cheminCase* doit exister effectivement.

Une case à cocher est un composant graphique qui affiche un petit carré appelé *indicateur*, accompagné d'un texte, d'une icone bitmap ou d'une image que l'on appelle l'*étiquette de la case*. Si l'étiquette est textuelle, elle est entièrement composée dans une même police ; le texte peut occuper plusieurs lignes à l'écran, soit par utilisation de sauts de lignes, soit parce qu'un paramètre **-wrapLength** a été fixé pour forcer les coupures de ligne. Il est enfin possible de souligner certains caractères du texte avec l'option **-underline**, le plus souvent pour en faire par la suite des raccourcis clavier.

Une case à cocher possède tous les comportements d'un simple bouton. Elle peut s'afficher dans trois états différents selon la valeur de l'option **-state**. D'autre part, son apparence peut être bombée, creusée ou plate. On peut la faire clignoter et elle peut invoquer une commande Tcl lorsque le premier bouton de la souris est cliqué sur elle.

En outre, les cases à cocher peuvent être sélectionnées :

- si la case à cocher est sélectionnée, l'indicateur est dessiné avec une apparence particulière et une variable Tcl associée est fixée à une valeur particulière (en général 1). La manière dont l'indicateur est marqué dépend du système d'exploitation : sous Unix, il prend une forme de relief en creux avec une couleur particulière. Sous Windows et MacOS, une marque en forme de x ou de v est ajoutée.
- si la case à cocher n'est pas sélectionnée, l'indicateur est dessiné avec une apparence différente et la variable Tcl associée est fixée à une autre valeur particulière (en général 0). Sous Unix, il prend une forme de relief bombé sans couleur particulière. Sous Windows et MacOS, la petite marque en forme de x ou de v est effacée.

Par défaut, le nom de la variable associée à une case à cocher est le même que celui utilisé pour créer la case. Le nom de la variable, de même que les valeurs qu'elle stocke pour représenter les états sélectionné et désélectionné, peuvent cependant être modifiés au moyen de certaines options.

Des options de configuration peuvent aussi être employées pour modifier l'aspect de l'indicateur. Par défaut, une case à cocher est configurée pour se sélectionner et se désélectionner alternativement à chaque clic de souris. Par ailleurs, les cases sont à l'écoute de leur variable associée : si la valeur de celle-ci est modifiée, la case à cocher se marquera elle-même en conséquence.

Opérations sur les cases à cocher

Avec l'instruction :

checkbutton *cheminCase ?options?*

la commande **checkbutton** crée une nouvelle commande Tcl dont le nom est précisément *cheminCase*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant la case à cocher. La forme générale est :

cheminCase opération ?arg arg ...?

Les opérations reconnues par les composantes de type *checkbutton* sont les suivantes :

cheminCase **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **checkbutton**.

cheminCase **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de la case à cocher. Si l'argument *option* n'est pas spécifié, la commande renvoie la liste de toutes les options disponibles pour *cheminCase*.

Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **checkbutton**.

cheminCase **deselect**
: Désélectionne la case à cocher et fixe la variable associée à la valeur correspondante (en général 0).

cheminCase **flash**
: Fait clignoter la case à cocher. Cet effet est obtenu en réaffichant à plusieurs reprises la case en faisant alterner les couleurs de son état normal et de son état activé. À la fin du clignotement, la case à cocher est laissée dans le même état que lorsque la commande a été invoquée. Cette commande est ignorée si

l'état de la case à cocher est *désactivé* (*disabled*).

cheminCase **invoke**
Accomplit exactement ce qui se produirait si un utilisateur cliquait sur la case à cocher avec la souris : l'état de la case est inversé et la commande Tcl associée, si elle existe, est invoquée. La valeur de retour est celle de la commande Tcl, ou une chaîne vide si aucune commande n'est associée à la case. Cette commande est ignorée si l'état de la case à cocher est *désactivé* (*disabled*).

cheminCase **select**
Sélectionne la case à cocher et fixe la variable associée à la valeur correspondante (en général 1).

cheminCase **toggle**
Inverse l'état de la case en modifiant l'aspect visuel et en mettant à jour la valeur de la variable correspondante.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les cases à cocher avec les caractéristiques suivantes :

1. sur les systèmes Unix, une case à cocher est activée dès que le pointeur de la souris passe au-dessus et désactivée dès qu'il la quitte. Sur les systèmes Mac et Windows, si le premier bouton de la souris est enfoncé, la case est activée dès que le pointeur se trouve dedans et désactivé dès qu'il en ressort ;
2. lorsque le premier bouton de la souris est enfoncé au-dessus d'une case à cocher, l'état est commuté et la commande associée, si elle existe, est invoquée ;
3. lorsque la case à cocher est focalisée par l'application, la barre d'espacement peut être utilisée pour l'invoquer. D'autre part, sous Windows, les touches + et = sélectionnent la case, la touche − la désélectionne.

Si l'état de la case à cocher est *disabled*, aucune des actions mentionnées ne se produit. Les comportements décrits sont les comportements par défaut et peuvent toujours être modifiés en définissant de nouvelles liaisons pour des composants particuliers ou en redéfinissant les liaisons de classe.

Options standard

Les options standard supportées par les composants de type *checkbutton* sont rassemblées dans le tableau 4 de la page suivante. Ces options sont décrites en détail à la page 142.

-activebackground	activeBackground	Foreground
-activeforeground	activeForeground	Background
-anchor	anchor	Anchor
-background ou -bg	background	Background
-bitmap	bitmap	Bitmap
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-image	image	Image
-justify	justify	Justify
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus
-text	text	Text
-textvariable	textVariable	Variable
-underline	underline	Underline
-wraplength	wrapLength	WrapLength

TAB. 4 – *Options standard reconnues par les composants de type* checkbutton.

Options spécifiques

-command **command** **Command**

Spécifie une commande Tcl à associer avec la case. Cette commande est typiquement invoquée lorsque le premier bouton de la souris est relâché au-dessus de la fenêtre du bouton. La variable globale associée (option **-variable**) sera mise à jour avant que la commande soit invoquée.

-height **height** **Height**

Spécifie la hauteur désirée pour cette case. Si une image ou un bitmap est affiché dans la case, la valeur est mesurée en unités d'écran ; pour du texte, elle est mesurée en nombre de lignes de texte. Si cette option n'est pas spécifiée, la hauteur désirée est calculée à partir de la taille de l'image, du bitmap ou du texte contenus.

-indicatoron **indicatorOn** **IndicatorOn**

Spécifie si l'indicateur doit être dessiné ou non. La valeur de cette option doit être une valeur booléenne valide. Si la valeur est *false*, l'option **relief** est ignorée

et le relief du composant est toujours enfoncé (*sunken*) lorsqu'il est sélectionné et relevé (*raised*) dans le cas contraire.

-offvalue **offValue** **Value**

Spécifie une valeur à stocker dans la variable associée à la case chaque fois que ce bouton est désélectionné. La valeur par défaut est 0.

-onvalue **onValue** **Value**

Spécifie une valeur à stocker dans la variable associée à la case chaque fois que ce bouton est sélectionné. La valeur par défaut est 1.

-overrelief **overRelief** **OverRelief**

Spécifie un relief alternatif pour la case à cocher, à utiliser lorsque le curseur de la souris est au-dessus de la case. Cette option peut être utilisée pour fabriquer des boutons de barre d'outils en utilisant la configuration suivante :

```
-relief flat -overrelief raised
```

Si la valeur de cette option est la chaîne vide, aucun relief alternatif n'est utilisé lorsque le curseur de la souris passe au-dessus de la case à cocher. La chaîne vide est la valeur par défaut.

-selectcolor **selectColor** **Background**

Spécifie une couleur de fond à utiliser lorsque la case est sélectionnée. Si **indicatorOn** est *true*, la couleur s'applique à l'indicateur. Sous Windows, cette couleur est utilisée comme arrière-plan pour l'indicateur indépendamment de l'état de la sélection. Si **indicatorOn** est *false*, cette couleur est utilisée comme arrière-plan pour le composant entier, à la place de **-background** ou **-activebackground**, chaque fois qu'il est sélectionné. Si l'option est spécifiée comme une chaîne vide, aucune couleur spéciale n'est utilisée lorsque le composant est sélectionné.

-selectimage **selectImage** **SelectImage**

Spécifie une image à afficher (à la place de l'option **image**) lorsque la case à cocher est sélectionnée. Cette option est ignorée à moins que l'option **image** n'ait été spécifiée.

-state **state** **State**

Spécifie l'un des trois états *normal*, *active*, ou *disabled* pour la case à cocher. Dans l'état normal, la case à cocher est affichée en utilisant les options **foreground** et **background**. L'état activé est typiquement utilisé lorsque la souris est au-dessus de la case à cocher. Dans l'état activé, la case à cocher est affichée en utilisant les options **activeForeground** et **activeBackground**. L'état désactivé (*disabled*) signifie que la case à cocher ne réagit plus : les liaisons par défaut refuseront d'activer l'objet et ignoreront les clics de souris. Dans cet état les options **disabledForeground** et **background** déterminent la façon dont le bouton doit être affiché.

-variable **variable** **Variable**

Spécifie le nom d'une variable globale à fixer pour indiquer si la case est sélectionnée ou non. Par défaut, cette variable porte le nom du composant au sein de son parent (i.e. le dernier élément du nom de chemin de la case).

-width **width** **Width**

Spécifie la largeur désirée pour la case. Si une image ou un bitmap est affiché dans la case, la valeur est mesurée en unités d'écran ; pour du texte, elle est mesurée en nombre de caractères. Si cette option n'est pas spécifiée, la hauteur désirée est calculée à partir de la taille de l'image, du bitmap ou du texte contenus.

clipboard

Permet de manipuler les sélections de Tk.

Syntaxe

clipboard *sous-commande ?arg arg ...?*

Description

Cette commande fournit une interface Tcl au mécanisme de stockage temporaire des sélections. Pour copier les données, il faut tout d'abord appeler la commande **clipboard clear** puis éventuellement ajouter des éléments par des appels à **clipboard append**. Tous les ajouts doivent se faire avant de retourner à la boucle d'événements. Le premier argument de **clipboard** détermine le format des autres arguments ainsi que le comportement de la commande. Les formes admises sont les suivantes :

clipboard clear ?**-displayof** *fenêtre*?
Prend possession de la sélection sur le moniteur de la fenêtre *fenêtre* et supprime son contenu. Par défaut, l'argument *fenêtre* est « . ». La commande renvoie une chaîne vide.

clipboard append ?**-displayof** *fenêtre*? ?**-format** *format*? ?**-type** *type*?**--**? *donnée*
Ajoute la donnée *donnée* à la sélection sur le moniteur de la fenêtre *fenêtre* sous la forme indiquée par l'argument *type* dans la représentation spécifiée par *format* et prend possession de la sélection sur le moniteur.
L'argument *type* spécifie la forme sous laquelle la sélection doit être renvoyée et doit être un nom d'atome tel que STRING ou FILE_NAME. Pour plus de détails, on consultera sur les systèmes Unix la page de manuel consacrée aux conventions ICCC (*Inter-Client Communication Conventions*). Par défaut *type* a la valeur STRING.
L'argument *format* spécifie la représentation à utiliser pour transmettre la sélection au demandeur (la seconde colonne de la table ICCCM) et a par défaut la valeur STRING. Si le format est STRING, la sélection est transmise en caractères 8-bits ASCII. Si le format est ATOM, la donnée *donnée* est divisée en champs séparés par une espace ; chaque champ est converti à sa valeur atomique et c'est cette valeur 32-bits qui est transmise. Pour toute autre valeur de *format*, *donnée* est divisée en champs séparés par une espace et chaque champ est converti en un entier 32-bits : c'est un tableau d'entiers qui est transmis au demandeur. Tous les éléments ajoutés à la sélection avec le même *type* doivent avoir le même *format*.
L'argument *format* n'est nécessaire que pour la compatibilité avec les requêtes de sélection adressées par des clients qui n'utilisent pas Tk. Si le contenu de la sélection est récupéré au moyen de Tk, la valeur est reconvertie en une chaîne et l'argument *format* est inutile. Un double tiret **--** peut être spécifié

pour indiquer clairement la fin des options : l'argument qui suit sera toujours considéré comme une donnée.

Les chaînes passées en argument à **clipboard append** sont concaténées avant conversion : il faut donc prendre soin de bien définir les espacements entre les éléments de la chaîne lorsque ceux-ci comportent également des espaces.

clipboard get ?**-displayof** *fenêtre*? ?**-type**?

Récupère les données de la sélection sur le moniteur de la fenêtre *fenêtre*. Par défaut, l'argument *fenêtre* est « . ». L'argument *type* spécifie la forme sous laquelle les données doivent être renvoyées et doit être un nom d'atome tel que STRING (valeur par défaut) ou FILE_NAME, par exemple. Cette commande est équivalente à

```
selection get -selection CLIPBOARD
```

destroy

Détruit une ou plusieurs fenêtres.

Syntaxe

destroy *?fenêtre fenêtre ...?*

Description

Cette commande détruit les fenêtres spécifiées par les arguments *fenêtre* ainsi que tous leurs descendants. Si une fenêtre *fenêtre* correspondant à « . » est détruite alors c'est l'application entière qui est tuée. Les fenêtres sont détruites dans l'ordre dans lequel elles sont indiquées et, si une erreur se produit au cours de la destruction de l'une d'entre elles, la commande s'interrompt et ne détruit pas les fenêtres restantes. Aucune erreur n'est générée si une fenêtre *fenêtre* n'existe pas.

entry

Permet de créer et de manipuler des champs de saisie

Syntaxe

entry *cheminChamp ?options?*

Description

La commande **entry** crée une nouvelle fenêtre (indiquée par l'argument *cheminChamp*) et en fait un composant graphique de type *champ de saisie*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources, pour configurer certaines caractéristiques du champ de saisie telles que la couleur, la police ou le relief etc. La commande **entry** renvoie la valeur de son argument *cheminChamp*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminChamp*, mais en revanche l'objet parent de *cheminChamp* doit exister effectivement.

Un composant graphique de type *entry* affiche une fenêtre horizontale permettant de saisir une ligne de texte éditable. Les fonctions usuelles d'édition sont disponibles à l'intérieur du champ de saisie. Les champs éditables respectent les règles générales de Tk concernant les composants focalisés par l'application : lorsqu'un champ de saisie est focalisé, un curseur d'insertion apparaît pour indiquer l'endroit d'insertion du prochain caractère tapé au clavier.

Les champs de saisie peuvent contenir des chaînes plus longues que la fenêtre de saisie. Seule une partie de la chaîne est alors visible mais les commandes décrites ci-dessous permettent de se déplacer dans cette chaîne et de modifier la vue. Les champs de saisie utilisent le mécanisme standard de l'option **-xscrollcommand** pour interagir avec les barres de défilement.

Validation

On peut aussi examiner le contenu des champs de saisie à des fins de validation. La validation fonctionne en donnant à l'option **-validateCommand** la valeur d'un script qui sera évalué en fonction de l'option **-validate**. Ce script est un test à effectuer et doit renvoyer les valeurs 1 ou 0 (*true* ou *false*). L'option **-validate** peut prendre une des valeurs suivantes :

none
: Aucune validation ne se produit. C'est la situation par défaut.

focus
: Appelle la commande lorsque le champ de saisie reçoit ou perd la focalisation.

focusin
: Appelle la commande lorsque le champ de saisie reçoit la focalisation.

focusout
Appelle la commande lorsque le champ de saisie perd la focalisation.

key
Appelle la commande lorsque le champ de saisie est édité.

all
Appelle la commande dans toutes les situations énumérées ci-dessus.

Pour écrire le test désigné par l'option **-validateCommand**, on peut se servir de séquences de substitution. Elles se présentent sous la forme d'un symbole pourcentage % suivi d'une lettre-clé (comme avec la commande **bind**). Ces séquences de substitution s'appliquent aussi à l'option **-invalidCommand**. Les séquences possibles sont :

%d
Type de l'action de validation : 1 pour *insert*, 0 pour *delete*, ou -1 pour *focus*, *forced* ou *textvariable*.

%i
Indice de la chaîne de caractères devant être inséré ou détruit, s'il en existe un, -1 dans le cas contraire.

%P
La valeur de l'entrée en cas d'édition. Si l'on configure le champ de saisie pour avoir une nouvelle variable *textvariable*, ce sera la valeur de cette nouvelle variable.

%s
La valeur courante du champ avant édition.

%S
La chaîne de caractères devant être insérée ou détruite, s'il en existe un, {} dans le cas contraire.

%v
Le type de validation actuellement fixé.

%V
Le type de validation ayant déclenché la fonction de rappel (*callback*) : *key, focusin, focusout, forced*.

%W
Le nom du champ de saisie.

En général, les options **-textVariable** et **-validateCommand** cohabitent mal. Si l'option **-textVariable** est utilisée uniquement pour lire des saisies, il n'y aura pas de problème. Le risque de conflit se produit lorsqu'on essaie d'attribuer à la variable désignée par **-textVariable** une valeur que la commande de **-validateCommand** refuse : dans ce cas, l'option **-validateCommand** prend la valeur *none* et la commande spécifiée par l'option **-invalidCommand** n'est pas invoquée. Il en va de même si une erreur se produit au cours de l'évaluation du test spécifié dans l'option **-validateCommand**.

Une erreur est générée principalement lorsque les commandes spécifiées par **-validateCommand** ou **-invalidCommand** provoquent elles-mêmes une erreur lorsqu'elles sont invoquées, ou si le test ne renvoie pas une valeur booléenne (0 ou 1). L'option **-validate** prend aussi la valeur *none* si l'on essaie d'éditer le champ de saisie depuis les commandes définies avec **-validateCommand** ou **-invalidCommand**. De telles éditions écraseront la valeur qui avait été éditée. Si l'on veut vraiment éditer le champ de saisie (par exemple pour lui donner la valeur {}) pendant la validation sans que l'option **-validate** ne soit modifiée, une astuce consiste à inclure l'instruction suivante :

```
after idle {\%W config -validate \%v}
```

dans les procédures spécifiées par **-validateCommand** ou **-invalidCommand**. Il est également recommandé de ne pas attribuer de valeur à une variable associée définie par l'option **-textVariable** au cours d'une validation, car la valeur du champ de saisie ne serait plus synchronisée avec celle de la variable.

Avec l'instruction :

entry *cheminChamp ?options?*

la commande **entry** crée une nouvelle commande Tcl dont le nom est précisément *cheminChamp*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le champ de saisie. La forme générale est :

cheminChamp opération ?arg arg ...?

Indices

De nombreuses opérations prennent comme argument un ou plusieurs indices spécifiant des caractères particuliers dans la chaîne d'entrée. Ces indices peuvent prendre une des valeurs ou des constantes symboliques suivantes :

nombre
: Spécifie un caractère par son indice numérique, où 0 correspond au premier caractère de la chaîne.

anchor
: Indique le point d'ancrage de la sélection que l'on fixe au moyen des opérations **select from** et **select adjust**.

end
: Indique le caractère situé immédiatement après le dernier caractère de la chaîne. Cela correspond aussi à la longueur de cette chaîne en nombre de caractères.

insert
: Indique le caractère situé immédiatement après le curseur d'insertion.

sel.first
: Indique le premier caractère de la sélection. Il y a erreur à utiliser cette forme si la sélection n'est pas dans la fenêtre du champ.

sel.last
: Indique le caractère situé immédiatement après le dernier de la sélection. Il y a erreur à utiliser cette forme si la sélection n'est pas dans la fenêtre du champ.

@*nombre*
: Dans cette forme, *nombre* est considéré comme une abscisse en nombre de pixels dans la fenêtre du champ. Le caractère placé à cette abscisse est utilisé.

On peut abréger les valeurs ci-dessus comme par exemple **e** ou **sel.f**. En général, les indices hors limites sont arrondis à la valeur légale la plus proche.

Opérations sur les champs de saisie

Les opérations reconnues par les composantes de type *champ de saisie* sont les suivantes :

cheminChamp **bbox** *index*
: Renvoie une liste de quatre nombres décrivant la boîte qui délimite le caractère d'indice *index*. Les deux premiers éléments de cette liste sont l'abscisse et l'ordonnée (mesurées en pixels relativement au champ de saisie lui-même) du coin supérieur gauche de la zone d'écran couverte par le caractère et les deux derniers sont la largeur et la hauteur du caractère. Cette boîte peut faire référence à une zone située hors de la région visible du champ de saisie.

cheminChamp **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **entry**.

cheminChamp **configure** *?option??valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration du champ de saisie. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminChamp*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **entry**.

cheminChamp **delete** *premier?dernier?*
: Détruit tous les caractères de l'entrée dont les indices se trouvent entre les indices *premier* et *dernier* inclus. Si l'argument *dernier* est omis, il est identique à *premier*+1 ce qui revient à supprimer un seul caractère. Cette commande renvoie une chaîne vide.

cheminChamp **get**
: Renvoie la chaîne contenue dans le champ de saisie.

cheminChamp **icursor** *index*
: Place le curseur d'insertion immédiatement avant le caractère d'indice *index*. Renvoie une chaîne vide.

cheminChamp **index** *index*
Renvoie l'indice numérique correspondant à l'argument *index* lorsque celui-ci est désigné par une des constantes symboliques mentionnées ci-dessus.

cheminChamp **insert** *index chaîne*
Insère les caractères de *chaîne* immédiatement avant le caractère d'indice *index*. Elle renvoie une chaîne vide.

cheminChamp **scan** *option args*
Cette commande est utilisée pour parcourir les entrées. Elle prend deux formes en fonction de l'argument *option* :

cheminChamp **scan mark** *x*
Enregistre la valeur *x* et la vue courante du champ de saisie en vue d'une utilisation ultérieure avec une commande **scan dragto**. Cette commande est typiquement associée à une pression du bouton de la souris dans le champ de saisie. Elle renvoie une chaîne vide.

cheminChamp **scan dragto** *x*
Cette commande calcule la différence entre la valeur de l'argument *x* et celle de l'argument *x* de la dernière commande **scan mark** exécutée pour ce champ de saisie. Elle renvoie une chaîne vide. La vue du contenu du champ est alors ajustée de 10 fois la valeur de cette différence vers la gauche ou vers la droite. Cette commande est typiquement associée à des déplacements de la souris à l'intérieur du champ de saisie pour simuler l'effet de déplacement rapide du contenu du champ. On associe fréquemment les deux commandes **scan mark** et **scan dragto** au second bouton de la souris comme ceci :

```
bind $c <2> "$c scan mark %x %y"
bind $c <B2-Motion> "$c scan dragto %x %y"
```

cheminChamp **selection** *option arg*
Cette commande est utilisée pour ajuster la sélection à l'intérieur du champ. Elle peut prendre plusieurs formes en fonction de l'argument *option* :

cheminChamp **selection adjust** *index*
Détermine l'extrêmité de la sélection qui se trouve le plus près du caractère d'indice *index* et ajuste cette extrêmité à la position du caractère et pas au-delà. L'autre extrêmité devient le point d'ancrage pour de futures opérations **select to**. En l'absence de sélection dans le champ, une nouvelle sélection est faite entre le caractère d'indice *index* et le plus récent point d'ancrage. Cette commande renvoie une chaîne vide.

cheminChamp **selection clear**
Efface la sélection si celle-ci se trouve actuellement dans le champ de saisie. Sinon, la commande est sans effet. Cette commande renvoie une chaîne vide.

cheminChamp **selection from** *index*
Fixe le point d'ancrage de la sélection immédiatement avant le caractère d'indice *index*. Ne modifie pas la sélection. Cette commande renvoie une chaîne vide.

cheminChamp **selection present**
Renvoie 1 si des caractères sont sélectionnés dans le champ, 0 sinon.

cheminChamp **selection range** *début fin*
Sélectionne les caractères situés entre les indices *début* et *fin*. Si *fin* est antérieur ou identique à *début*, la sélection est effacée.

cheminChamp **selection to** *index*
Si l'indice *index* est situé avant le point d'ancrage, la commande sélectionne les caractères depuis l'indice *index* jusqu'au point d'ancrage exclu. Si l'indice *index* correspond au point d'ancrage, rien ne se produit. Si l'indice *index* se trouve après le point d'ancrage, la commande sélectionne les caractères depuis le point d'ancrage jusqu'au caractère d'indice *index* exclu. Le point d'ancrage est déterminé par la plus récente opération **select from** ou **select adjust** effectuée sur ce champ de saisie. Si la sélection n'est pas dans le champ, une nouvelle sélection est créée en utilisant le plus récent point d'ancrage spécifié pour ce champ. Cette commande renvoie une chaîne vide.

cheminChamp **validate**
Cette commande est utilisée pour forcer une évaluation de la fonction définie par l'option **-validateCommand** indépendamment des conditions spécifiées par l'option **-validate**. Ceci est obtenu en fixant temporairement l'option **-validate** à la valeur **all**. Elle renvoie 0 ou 1.

cheminChamp **xview***args*
Cette commande est utilisée pour interroger et modifier la position horizontale du texte dans la fenêtre du champ de saisie. Elle peut prendre une des formes suivantes :

cheminChamp **xview**
Renvoie une liste contenant deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : ces valeurs indiquent les proportions de l'entrée qui sont visibles ou non dans la fenêtre. Par exemple si les valeurs renvoyées sont 0,2 et 0,6 cela signifie que 20% du texte est situé à gauche de la partie visible et 40% à droite, tandis qu'entre les deux la partie visible correspond à 40% du texte. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de la commande spécifiée par l'option **-xscrollcommand**.

cheminChamp **xview** *index*
Ajuste la vue dans la fenêtre du champ de telle sorte que le caractère d'indice *index* apparaisse au bord gauche de cette fenêtre.

cheminChamp **xview moveto** *fraction*
Ajuste la vue dans la fenêtre du champ de telle sorte que le caractère

situé à la position désignée en pourcentage de la largeur totale par la fraction *fraction* soit à gauche du bord de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1].

cheminChamp **xview scroll** *nb unité*
Cette commande décale la vue dans la fenêtre du champ vers la gauche ou vers la droite selon la valeur donnée aux arguments *nb* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage est d'environ *nb* caractères. Avec *pages* le décalage est de *nb* contenus d'écran.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les champs de saisie avec les caractéristiques ci-dessous. Dans tout ce qui suit un *mot* fait référence à toute séquence contiguë de lettres, chiffres et caractères de soulignement :

1. Cliquer le premier bouton de la souris positionne le curseur immédiatement avant le caractère placé sous le pointeur, focalise le champ et efface toute sélection faite dans ce champ. Glisser avec le premier bouton de la souris enfoncé étend une sélection depuis le curseur jusqu'au caractère situé sous le pointeur de la souris.
2. Double-cliquer avec le premier bouton de la souris sélectionne le mot situé sous la souris et positionne le curseur au début du mot. Glisser avec le premier bouton de la souris enfoncé après un double clic étend une sélection constituée de mots entiers.
3. Triple-cliquer avec le premier bouton de la souris sélectionne tout le texte du champ et positionne le curseur avant le premier caractère.
4. Les extrêmités de la sélection peuvent être ajustées en glissant avec le premier bouton de la souris enfoncé tout en appuyant sur la touche Majuscules. Cela ajuste l'extrêmité qui était la plus proche du pointeur lorsque le bouton de la souris a été pressé. Si le bouton a été double-cliqué, l'ajustement se fait par mots entiers.
5. Cliquer le premier bouton de la souris avec la touche Contrôle enfoncée positionne le curseur dans le champ sans affecter la sélection.
6. Tout caractère imprimable normal est inséré à la position du curseur.
7. La vue dans le champ peut être ajustée en utilisant le second bouton de la souris et en déplaçant celle-ci sans relâcher le bouton. Si le second bouton est simplement cliqué, la sélection courante est insérée à la position du pointeur.
8. Si la souris est déplacée hors du champ vers la gauche ou vers la droite avec le premier bouton enfoncé, le champ défilera automatiquement pour faire apparaître une plus grande partie du texte qui se trouvait hors vue.
9. Les touches Gauche et Droite déplacent le curseur d'un caractère vers la gauche ou vers la droite. Elles effacent également toute sélection dans le champ et positionnent le point d'ancrage. Si en même temps la touche Majuscule est enfoncée, le curseur se déplace et la sélection est étendue au nouveau

caractère. Les touches Contrôle-Gauche et Contrôle-Droite déplacent le curseur par mots et étendent la sélection. Les touches Contrôle-b et Contrôle-f se comportent comme Gauche et Droite, respectivement. Méta-b et Méta-f se comportent comme Contrôle-Gauche et Contrôle-Droite, respectivement.

10. Les touches Home ou Contrôle-a déplacent le curseur au début du champ et effacent toute sélection éventuelle dans le champ. Majuscule-Home déplace le curseur au début du champ et étend la sélection jusqu'à ce point.
11. Les touches Fin ou Contrôle-e déplacent le curseur à la fin du champ et effacent toute sélection éventuelle dans le champ. Majuscule-Fin déplace le curseur à la fin du champ et étend la sélection jusqu'à ce point.
12. Les touches Select et Contrôle-Espace fixent le point d'ancrage de la sélection à la position du curseur. Elles n'affectent pas la sélection courante. Les touches Majuscule-Select et Contrôle-Majuscule-Espace ajustent la sélection à la position courante du curseur, sélectionnant depuis le point d'ancrage jusqu'au curseur s'il n'y avait pas de sélection antérieurement.
13. Contrôle-/ sélectionne tout le texte du champ.
14. Contrôle-\ efface toute sélection du champ.
15. Les touches F16 ou Méta-w copient une sélection éventuelle.
16. Les touches F20 ou Contrôle-w copient une sélection éventuelle puis l'effacent.
17. Les touches F18 ou Contrôle-y copient le contenu de la sélection dans le champ à la position du curseur.
18. La touche Suppression (*Delete*) efface une sélection éventuelle dans le champ. S'il n'y a pas de sélection, elle supprime le caractère situé à droite du curseur.
19. Les touches Retour-Arrière et Contrôle-h effacent une sélection éventuelle dans le champ. S'il n'y a pas de sélection, elles suppriment le caractère situé à gauche du curseur.
20. Contrôle-d efface le caractère à droite du curseur.
21. Méta-d efface le mot à droite du curseur.
22. Contrôle-k détruit tous les caractères à droite du curseur.
23. Contrôle-t inverse l'ordre des deux caractères situés à droite du curseur.

Si un champ de saisie est désactivé avec l'option **-state**, il est toujours possible d'ajuster la vue de ce champ ou d'en sélectionner du texte mais aucun curseur n'est disponible et aucune modification n'est permise.

Les comportements décrits sont les comportements par défaut et peuvent être modifiés en définissant de nouvelles liaisons pour des entrées particulières ou en redéfinissant les liaisons de classe.

Options standard

Les options standard supportées par les composants de type *entry* sont rassemblées dans le tableau 5 de la page suivante. Ces options sont décrites en détail à la page 142.

-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-curseur	curseur	Cursor
-exportselection	exportSelection	ExportSelection
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-insertbackground	insertBackground	Foreground
-insertborderwidth	insertBorderWidth	BorderWidth
-insertofftime	insertOffTime	OffTime
-insertontime	insertOnTime	OnTime
-insertwidth	insertWidth	InsertWidth
-justify	justify	Justify
-relief	relief	Relief
-selectbackground	selectBackground	Foreground
-selectborderwidth	selectBorderWidth	BorderWidth
-selectforeground	selectForeground	Background
-takefocus	takeFocus	TakeFocus
-textvariable	textVariable	Variable
-xscrollcommand	xScrollCommand	ScrollCommand

TAB. 5 – *Options standard reconnues par les composants de type* entry.

Options spécifiques

-disabledbackground disabledBackground DisabledBackground

Spécifie la couleur d'arrière-plan à utiliser lorsque le champ est désactivé. Si cette option est la chaîne vide, la couleur d'arrière-plan normale est utilisée.

-disabledforeground disabledForeground DisabledForeground

Spécifie la couleur de premier plan à utiliser lorsque le champ est désactivé. Si cette option est la chaîne vide, la couleur de premier plan normale est utilisée.

-invalidcommand invalidCommand InvalidCommand

Spécifie un script à évaluer lorsque l'option **-validateCommand** renvoie la valeur 0. Fixer cette option à {} désactive cette fonctionnalité (c'est le comportement par défaut). Une utilisation courante de cette option consiste à la fixer à la valeur *bell*. Voir la section *Validation* à la page 67 pour plus d'informations.

-readonlybackground readonlyBackground ReadonlyBackground

Spécifie la couleur d'arrière-plan à utiliser lorsque le champ est en lecture seule. Si cette option est la chaîne vide, la couleur d'arrière-plan normale est utilisée.

-show show Show

Si cette option est spécifiée, le contenu véritable du champ n'est pas affiché dans la fenêtre. À la place, chaque caractère dans la valeur du champ sera remplacé par le premier caractère figurant dans la valeur de cette option, comme par exemple un astérisque $*$. Ceci est utile, par exemple, lorsque le champ est utilisé pour saisir un mot de passe. Pour des raisons de sécurité, si des caractères du champ sont sélectionnés et copiés ailleurs, l'information copiée sera ce qui est affiché et non pas le contenu véritable du champ.

-state state State

Spécifie l'un des trois états désignés par les constantes *normal*, *readonly* ou *disabled* pour le champ. Si le champ est en lecture seule (*readonly*), sa valeur ne peut être modifiée et aucun curseur d'insertion n'est affiché, même si le champ est focalisé ; en revanche, le contenu du champ peut être sélectionné. Si le champ est désactivé (*disabled*), sa valeur ne peut être modifiée, aucun curseur d'insertion n'est affiché et le contenu du champ ne peut être sélectionné ; le champ peut néanmoins être affiché dans une autre couleur selon les valeurs des options **disabledForeground** et **disabledBackground**.

-validate validate Validate

Spécifie le mode de validation. L'option peut prendre une des valeurs *none*, *focus*, *focusin*, *focusout*, *key* ou *all*. La valeur par défaut est *none*. Si l'on souhaite une validation, il faut explicitement déclarer le mode à utiliser. Voir la section *Validation* à la page 67.

-validatecommand validateCommand ValidateCommand

Spécifie un script à évaluer lorsqu'on veut valider la valeur saisie dans le champ. Fixer cette option à {} désactive cette fonctionnalité (c'est le comportement par défaut). Cette commande doit renvoyer une valeur booléenne valide. Si elle renvoie 0 (ou un équivalent), cela signifie que l'on rejette la nouvelle saisie et le script contenu dans l'option **-invalidCommand** sera évalué s'il a été déclaré. Si elle renvoie 1, la nouvelle édition du champ a lieu. Voir la section *Validation* à la page 67.

-width width Width

Spécifie une valeur entière indiquant la largeur souhaitée pour la fenêtre du champ de saisie, mesurée en taille moyenne de caractères dans la fonte associée à ce champ. Si la valeur est négative ou nulle, l'objet choisit une taille juste suffisante pour contenir son texte courant.

event

Permet de manipuler les événements.

Syntaxe

event *sous-commande ?arg arg ...?*

Description

La commande **event** procure un certain nombre de facilités pour traiter les événements concernant les fenêtres : elle permet la synthèse d'événements ou bien la définition d'événements virtuels. La commande a plusieurs formes différentes, déterminées par le premier argument qui indique une sous-commande. Les formes suivantes sont actuellement supportées :

event add *<< virt >> séquence ?séquence ...?*
Associe l'événement virtuel *virt* à la ou les séquences d'événements spécifiées par les arguments *séquence*, de façon que l'événement virtuel se déclenche dès qu'une des séquences se produit. L'argument *virt* peut être n'importe quelle chaîne et *séquence* peut avoir n'importe laquelle des valeurs autorisées pour l'argument *séquence* de la commande **bind**. Si *virt* est déjà défini, les nouvelles séquences d'événements sont ajoutées à celles qui existent déjà pour l'événement.

event delete *<< virt >> ?séquence séquence ...?*
Détruit chacune des séquences spécifiées par le ou les arguments *séquence* de la liste de celles qui sont associées à l'événement virtuel désigné par *virt*. L'argument *virt* peut être n'importe quelle chaîne et *séquence* peut avoir n'importe laquelle des valeurs autorisées pour l'argument *séquence* de la commande **bind**. Toute séquence *séquence* qui n'est pas associée actuellement à *virt* est ignorée. Si aucun argument *séquence* n'est fourni, toutes les séquences sont supprimées de l'événement virtuel *virt*, ce qui fait qu'il ne se déclenchera plus.

event generate *fenêtre événement ?option valeur option valeur ...?*
Génère un événement de fenêtre et fait en sorte qu'il soit exécuté comme s'il provenait du système de fenêtrage. L'argument *fenêtre* indique le nom du chemin de la fenêtre pour laquelle l'événement sera généré ; il peut également s'agir d'un identificateur (du type de ceux renvoyés par la commande **winfo id**) dès lors qu'il s'adresse à une fenêtre de l'application.
L'argument *événement* fournit une description de base de l'événement, comme par exemple `<Shift-Button-2>` ou <<Paste>>. Si l'argument *fenêtre* est vide cela signifie que l'écran entier est concerné et les coordonnées sont relatives à l'écran.
L'argument *événement* peut avoir une quelconque des formes autorisées pour l'argument *séquence* de la commande **bind** mais il doit consister en un motif simple d'événement et non pas en une séquence.

Les paires *option-valeur* peuvent être utilisées pour spécifier des attributs additionnels pour l'événement, comme les coordonnées x et y décrivant la position de la souris. Si l'option **-when** n'est pas spécifiée, l'événement est traité immédiatement : tous les gestionnaires de l'événement devront terminer avant que la commande **event generate** ne retourne. Si l'option **-when** est spécifiée, elle détermine l'instant d'exécution de l'événement. Certains événements, comme une touche de clavier enfoncée, exigent que la fenêtre soit focalisée pour être traités correctement.

event info?<< *virt* >>?

Renvoie l'information concernant les événements virtuels.

Si l'argument << *virt* >> est omis, la valeur de retour est la liste de tous les événements virtuels qui sont actuellement définis. Si << *virt* >> est spécifié, la valeur de retour est une liste dont les éléments sont les séquences physiques d'événements actuellement définis pour l'événement virtuel indiqué ; si l'événement virtuel n'est pas défini, une chaîne vide est renvoyée.

Champs d'événements

Les options suivantes sont supportées par la commande **event generate**. Elles correspondent aux variables substituables spécifées par un symbole % dans les scripts de liaison de la commande **bind**.

-above *fenêtre*

L'argument *fenêtre* spécifie le champ *above* pour l'événement, soit comme un nom de chemin de fenêtre, soit comme un nombre entier identificateur de fenêtre. Valide pour les événements *Configure*. Correspond à la variable de substitution %a dans les scripts de liaison.

-borderwidth *taille*

L'argument *taille* doit être une distance d'écran ; il spécifie le champ *border_width* pour l'événement. Valide pour les événements *Configure*. Correspond à la variable de substitution %B dans les scripts de liaison.

-button *nombre*

L'argument *nombre* doit être un entier ; il spécifie le champ *detail* pour les événements *ButtonPress* ou *ButtonRelease*, supplantant tout numéro de bouton fourni dans l'argument de base de *événement*. Correspond à la variable de substitution %b dans les scripts de liaison.

-count *nombre*

L'argument *nombre* doit être un entier ; il spécifie le champ *count* pour l'événement. Valide pour les événements *Expose*. Correspond à la variable de substitution %c dans les scripts de liaison.

-delta *nombre*

L'argument *nombre* doit être un entier ; il spécifie le champ *delta* pour l'événement *MouseWheel*. L'argument *delta* se réfère à la direction et à l'amplitude de la rotation effectuée sur la roue de la souris. Cette valeur est calculée non pas comme une distance d'écran mais en unités de déplacement de la roue de

la souris. Typiquement, ces valeurs sont des multiples de 120. Par exemple, 120 décalerait l'objet texte de 4 lignes vers le haut et -240 de 8 lignes vers le bas. Ce champ correspond à la variable de substitution %D dans les scripts de liaison.

-detail *détail*

L'argument *détail* spécifie le champ *detail* pour l'événement et doit être une des valeurs suivantes :

NotifyAncestor	*NotifyPointer*	*NotifyNonlinear*
NotifyNonlinearVirtual	*NotifyInferior*	*NotifyVirtual*
NotifyDetailNone	*NotifyPointerRoot*	

Valide pour les événements *Enter*, *Leave*, *FocusIn* et *FocusOut*. Correspond à la variable de substitution %d dans les scripts de liaison.

-focus *bool*

L'argument *bool* doit être une valeur booléenne ; il spécifie le champ *focus* pour l'événement. Valide pour les événements *Enter* et *Leave*. Correspond à la variable de substitution %f dans les scripts de liaison.

-height *taille*

L'argument *taille* doit être une distance d'écran ; il spécifie le champ *height* pour l'événement. Valide pour les événements *Configure*. Correspond à la variable de substitution %h dans les scripts de liaison.

-keycode *nombre*

L'argument *nombre* doit être un entier ; il spécifie le champ *keycode* pour l'événement. Valide pour les événements *KeyPress* et *KeyRelease*. Correspond à la variable de substitution %k dans les scripts de liaison.

-keysym *nom*

L'argument *nom* doit être le nom d'une touche symbolique valide comme par exemple *g*, *space*, ou *Return* ; la valeur correspondante est utilisée comme champ *keycode* pour l'événement, remplaçant tout détail spécifié dans l'argument de base *événement*. Valide pour les événements *KeyPress* et *KeyRelease*. Correspond à la variable de substitution %K dans les scripts de liaison.

-mode *notify*

L'argument *notify* spécifie le champ *mode* pour l'événement et doit prendre une des valeurs *NotifyNormal*, *NotifyGrab*, *NotifyUngrab*, ou *NotifyWhileGrabbed*. Valide pour les événements *Enter*, *Leave*, *FocusIn* et *FocusOut*. Correspond à la variable de substitution %m dans les scripts de liaison.

-override *bool*

L'argument *bool* doit être une valeur booléenne ; il spécifie le champ *override_redirect* pour l'événement. Valide pour les événements *Map*, *Reparent* et *Configure*. Correspond à la variable de substitution %o dans les scripts de liaison.

-place *où*

L'argument *où* spécifie le champ *place* pour l'événement et doit prendre une

des valeurs *PlaceOnTop* ou *PlaceOnBottom*. Valide pour les événements *Circulate*. Correspond à la variable de substitution %p dans les scripts de liaison.

-root *fenêtre*
L'argument *fenêtre* doit être soit un nom de chemin de fenêtre, soit un nombre entier identificateur de fenêtre ; il spécifie le champ *root* pour l'événement. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Enter*, *Leave* et *Motion*. Correspond à la variable de substitution %R dans les scripts de liaison.

-rootx *coord*
L'argument *coord* doit être une distance d'écran ; il spécifie le champ *x_root* pour l'événement. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Enter*, *Leave* et *Motion*. Correspond à la variable de substitution %X dans les scripts de liaison.

-rooty *coord*
L'argument *coord* doit être une distance d'écran ; il spécifie le champ *y_root* pour l'événement. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Enter*, *Leave* et *Motion*. Correspond à la variable de substitution %Y dans les scripts de liaison.

-sendevent *bool*
L'argument *bool* doit être une valeur booléenne ; il spécifie le champ *send_event* pour l'événement. Valide pour tous les événements. Correspond à la variable de substitution %E dans les scripts de liaison.

-serial *nombre*
L'argument *nombre* doit être un entier ; il spécifie le champ *serial* pour l'événement. Valide pour tous les événements. Correspond à la variable de substitution %# dans les scripts de liaison.

-state *état*
L'argument *état* spécifie le champ *state* pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Enter*, *Leave* et *Motion* et doit être une valeur entière. Pour les événements *Visibility*, il doit prendre une des valeurs *VisibilityUnobscured*, *VisibilityPartiallyObscured* ou *VisibilityFullyObscured*. Cette option remplace tout modificateur tel que Méta ou Contrôle spécifié dans l'événement de base *événement*. Correspond à la variable de substitution %s dans les scripts de liaison.

-subwindow *fenêtre*
L'argument *fenêtre* spécifie le champ *subwindow* pour l'événement, soit comme un nom de chemin de fenêtre, soit comme un nombre entier identificateur de fenêtre. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Enter*, *Leave* et *Motion*. Correspond à la variable de substitution %S dans les scripts de liaison.

-time *entier*
L'argument *integer* doit être une valeur entière ; il spécifie le champ *time* pour l'événement. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Enter*, *Leave*, *Motion*, et *Property*. Correspond à la variable

de substitution %t dans les scripts de liaison.

-warp *bool*
L'argument *bool* doit être une valeur booléenne ; il spécifie si le pointeur de la souris doit être déplacé également à l'occasion d'un transfert de focalisation afin que le pointeur se trouve toujours sur la fenêtre focalisée. Ce comportement est déconseillé par le manuel ICCCM. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease* et *Motion*.

-when *quand*
L'argument *quand* détermine quand l'événement sera exécuté. Il doit prendre une des valeurs suivantes :

now
Exécute l'événement immédiatement, avant que la commande ne retourne. Cela se produit aussi si l'option **-when** est omise.

tail
Place l'événement dans la file d'attente d'événements de Tcl derrière les événements déjà mis dans la file pour cette application.

head
Place l'événement en tête de la file d'attente d'événements de Tcl, de telle sorte qu'il sera traité avant tout autre événement déjà dans la file.

mark
Place l'événement en tête de la file d'attente d'événements de Tcl mais derrière tout autre événement déjà placé dans la file au moyen d'une instruction **-when mark**. Cette option est utile pour générer une série d'événements à exécuter dans l'ordre mais en tête de file.

-width *taille*
L'argument *taille* doit être une distance d'écran ; il spécifie le champ *width* pour l'événement. Valide pour les événements *Configure*. Correspond à la variable de substitution %w dans les scripts de liaison.

-x *coord*
L'argument *coord* doit être une distance d'écran ; il spécifie le champ *x* pour l'événement. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Motion*, *Enter*, *Leave*, *Expose*, *Configure*, *Gravity* et *Reparent*. Correspond à la variable de substitution %x dans les scripts de liaison. Si l'argument *fenêtre* est vide, la coordonnée est relative à l'écran et l'option correspond alors à la variable de substitution %X dans les scripts de liaison.

-y *coord*
L'argument *coord* doit être une distance d'écran ; il spécifie le champ *y* pour l'événement. Valide pour les événements *KeyPress*, *KeyRelease*, *ButtonPress*, *ButtonRelease*, *Motion*, *Enter*, *Leave*, *Expose*, *Configure*, *Gravity* et *Reparent*. Correspond à la variable de substitution %y dans les scripts de liaison. Si l'argument *fenêtre* est vide, la coordonnée est relative à l'écran et l'option correspond alors à la variable de substitution %Y dans les scripts de liaison.

Toutes les options qui ne sont pas spécifiées lorsqu'un événement est généré se voient attribuer la valeur 0, à l'exception de l'option **-serial** qui reçoit le numéro de série du prochain événement du système X.

Exemples d'événements virtuels

Pour qu'une liaison d'événement virtuel se déclenche, deux conditions doivent se produire :

– l'événement virtuel doit être défini avec la commande **event add** ;

– une liaison doit être créée pour l'événement virtuel avec la commande **bind**.

Considérons les définitions d'événements virtuels suivantes :

```
event add <<Paste>> <Contrôle-y>
event add <<Paste>> <Button-2>
event add <<Save>> <Contrôle-X><Contrôle-S>
event add <<Save>> <Shift-F12>
```

Par la commande **bind**, un événement virtuel peut être lié à n'importe quel autre type d'événement de base, comme ceci :

```
bind Entry <<Paste>> {\%W insert [selection get]}
```

Les doubles crochets angulaires sont utilisés pour spécifier qu'un événement virtuel est lié. Si l'utilisateur tape Contrôle-y ou enfonce le second bouton de la souris, ou si un événement virtuel *Paste* est fabriqué avec **event generate**, alors la liaison *Paste* sera invoquée.

Si une liaison virtuelle a exactement la même séquence qu'une certaine liaison physique, la liaison physique sera considérée prioritairement. Considérons l'exemple suivant :

```
event add <<Paste>> <Contrôle-y> <Méta-Contrôle-y>
bind Entry <Contrôle-y> {puts Contrôle-y}
bind Entry <<Paste>> {puts Paste}
```

Lorsque l'utilisateur tape Contrôle-y la liaison `<Contrôle-y>` est invoquée, parce qu'un événement physique est considéré comme plus important qu'un événement virtuel. Néanmoins, quand l'utilisateur tape Méta-Contrôle-y c'est la liaison *Paste* qui est invoquée, car le modificateur **Méta** dans le motif physique associé à l'événement virtuel est plus spécifique que la séquence `<Contrôle-y>` pour l'événement physique.

Des liaisons à un événement virtuel peuvent être créées avant que l'événement virtuel n'existe. En effet, l'événement virtuel n'aura jamais besoin d'être effectivement défini, par exemple, sur des plates-formes sur lesquelles l'événement virtuel spécifique n'aurait pas de signification ou serait impossible à générer.

Si la définition d'un événement virtuel change en cours d'exécution, toutes les fenêtres répondront immédiatement à la nouvelle définition. Dans l'exemple qui précède, si le code suivant est exécuté :

```
bind <Entry> <Contrôle-y> {event}
event add <<Paste>> <Key-F6>
```

le comportement sera modifié sur deux points :

- la liaison dissimulée se révélera ; taper Contrôle-y n'invoquera plus la liaison `<Contrôle-y>`, mais plutôt l'événement virtuel *Paste*.
- enfoncer la touche F6 invoquera aussi désormais la liaison *Paste*.

focus

Gère la focalisation.

Syntaxe

focus
focus *fenêtre*
focus *sous-commande ?arg arg ...?*

Description

La commande **focus** est utilisée pour gérer la focalisation des composants par Tk. À un moment donné, une fenêtre est désignée sur chaque moniteur comme fenêtre focalisée : cela signifie que tous les événements de clavier (touche enfoncée ou touche relâchée) sont adressés à cette fenêtre et pas à une autre. C'est en principe, le gestionnaire de fenêtrage qui gère la focalisation parmi les fenêtres de premier niveau sur le moniteur. Sur certains systèmes par exemple, ce gestionnaire focalise automatiquement la fenêtre qui se trouve sous le pointeur de la souris, qu'elle soit active ou non, tandis que sur d'autres systèmes, la fenêtre ne reçoit la focalisation que lorsque l'utilisateur clique dessus pour l'activer. Le gestionnaire de fenêtres ne gère que les fenêtres de niveau supérieur et laisse le soin à l'application de gérer la focalisation parmi tous les descendants d'une fenêtre de niveau supérieur.

Tk mémorise, pour chaque fenêtre de niveau supérieur, une seule focalisation : lorsque le gestionnaire de fenêtres du système place la focalisation sur une fenêtre particulière, Tk redirige cette focalisation sur le widget de la fenêtre qu'il a mémorisé. Avec Tk, de simples déplacements de la souris ne modifient pas la focalisation : celle-ci ne change que lorsqu'un composant réclame explicitement d'être focalisé ou parce que l'utilisateur a pressé une touche qui conduit à déplacer la focalisation (comme par exemple la touche de tabulation qui permet de passer d'un champ de saisie à un autre).

La procédure Tcl appelée **Tk_focusFollowsMouse** peut être appelée pour créer un modèle implicite de focalisation : elle reconfigure Tk de telle sorte qu'une fenêtre reçoive la focalisation dès que le curseur de la souris se trouve au-dessus. Les procédures Tcl **Tk_focusNext** et **Tk_focusPrev** permettent de contrôler l'ordre de focalisation parmi les fenêtres de premier niveau (cf. 85). Elles sont associées par défaut aux touches Tabulation et Majuscule-Tabulation.

La commande **focus** peut prendre une des formes suivantes :

focus
Renvoie le nom de chemin de la fenêtre focalisée sur l'écran qui contient la fenêtre principale de l'application, ou une chaîne vide si aucune fenêtre de l'application n'est focalisée. En cas d'ambiguïté, on peut spécifier l'écran explicitement au moyen de l'option **-displayof**.

focus *fenêtre*
: Si l'application détient actuellement la focalisation sur l'écran de la fenêtre désignée par l'argument *fenêtre*, cette commande réassigne la focalisation à cette fenêtre et renvoie une chaîne vide. Si l'application elle-même ne détient pas la focalisation sur cet écran, la fenêtre *fenêtre* sera simplement mémorisée comme objet focalisé pour sa fenêtre de premier niveau : la prochaine fois que la fenêtre de premier niveau recevra la focalisation, Tk la transmettra directement à la sous-fenêtre *fenêtre*. Si l'argument *fenêtre* est une chaîne vide, la commande est sans effet.

focus -displayof *fenêtre*
: Renvoie le nom de la fenêtre focalisée sur l'écran contenant la fenêtre désignée par l'argument *fenêtre*. Si la fenêtre focalisée sur cet écran n'appartient pas à l'application, la valeur de retour est une chaîne vide.

focus -force *fenêtre*
: Établit la focalisation sur la fenêtre désignée par l'argument *fenêtre*, même si l'application ne détient pas la focalisation pour cet écran. Cette commande n'est pas orthodoxe : normalement, une application ne doit pas réclamer elle-même la focalisation mais plutôt attendre que le gestionnaire de fenêtres la lui assigne. Si l'argument *fenêtre* est une chaîne vide, la commande est sans effet.

focus -lastfor *fenêtre*
: Renvoie le nom du plus récent widget focalisé parmi tous les widgets appartenant à la même fenêtre de premier niveau que *fenêtre*. Si aucun widget à ce niveau n'a encore été focalisé, ou si le plus récent a été détruit entre-temps, c'est le nom de la fenêtre de premier niveau qui est renvoyé. La valeur de retour est le widget qui sera focalisé la prochaine fois que le serveur de fenêtres assignera la focalisation à la fenêtre de premier niveau.

Procédures Tcl de focalisation

Tk_focusNext *fenêtre*
Tk_focusPrev *fenêtre*
Tk_focusFollowsMouse

Tk_focusNext est une procédure Tcl qui renvoie la fenêtre qui suit immédiatement celle désignée par l'argument *fenêtre* dans l'ordre de focalisation. Cet ordre est déterminé par l'ordre d'empilement des fenêtres et par la structure hiérarchique de la fenêtre. À un même niveau, l'ordre est l'ordre d'empilement, la fenêtre la plus basse venant en premier. Si une fenêtre a des descendants, la fenêtre est visitée en premier, puis tous ses enfants récursivement avant de passer à la fenêtre suivante de même niveau. Les fenêtres de niveau supérieur autres que *fenêtre* ne sont pas prises en compte de telle sorte que la commande **Tk_focusNext** ne renvoie jamais une autre fenêtre de niveau supérieur que *fenêtre*.

Après avoir déterminé la fenêtre suivante, la procédure **Tk_focusNext** examine l'option de cette fenêtre pour voir si elle doit l'écarter : si c'est le cas, elle recom-

mence avec la fenêtre suivante jusqu'à ce qu'elle trouve une fenêtre qui accepte la focalisation.

La procédure **Tk_focusPrev** est analogue à **Tk_focusNext** mais procède dans l'ordre inverse et renvoie la fenêtre précédente dans l'ordre de focalisation.

Enfin la procédure **Tk_focusFollowsMouse** modifie le modèle général de focalisation pour l'application en en faisant un modèle implicite : après l'appel de cette procédure, chaque fois que le pointeur de la souris entre dans une fenêtre, Tk lui accorde immédiatement la focalisation. La commande **focus** peut aussi intervenir pour changer une focalisation mais dès que la souris se déplacera sur une autre fenêtre, la focalisation passera à cette nouvelle fenêtre. Il n'existe pas de procédure pour rétablir le modèle explicite : si cela est nécessaire il faudra détruire manuellement les liaisons créées par **Tk_focusFollowsMouse**.

font

Permet de créer et d'examiner les polices.

Syntaxe

font *sous-commande ?arg arg ...?*

Description

La commande **font** procure un certain nombre de facilités pour manipuler les polices, les nommer, examiner leurs attributs etc. La commande peut prendre plusieurs formes selon la sous-commande qui lui est adjointe. Les formes supportées sont les suivantes :

font actual *police* ?**-displayof** *fenêtre*? *?option?*
: Renvoie l'information concernant les attributs réels obtenus lorsque la police désignée par l'argument *police* est utilisée sur l'écran de *fenêtre* ; les attributs réels obtenus peuvent différer des attributs requis en raison de limitations rencontrées sur certaines plates-formes, comme la disponibilité des familles de police ou de leurs tailles. L'argument *police* est une description obéissant aux règles énoncées à la section *Description de polices* ci-après. Si l'argument *fenêtre* est omis, c'est la fenêtre principale qui est utilisée par défaut. Si l'argument *option* est spécifié, la commande renvoie la valeur de cet attribut particulier ; si *option* est omis, la valeur de retour est une liste de tous les attributs et de leurs valeurs. On se reportera à la section *Options de polices* ci-dessous pour une liste de tous les attributs possibles.

font configure *nomPolice ?option? ?valeur option valeur ...*?
: Permet d'obtenir ou de modifier les attributs souhaités pour la police *nomPolice*. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de tous les attributs et de leurs valeurs.

  Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie la valeur de cet attribut. Si un ou plusieurs couples *option-valeur* sont spécifiés, les attributs en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. Dans ce cas, tous les composants graphiques utilisant cette police seront réaffichés en utilisant les nouveaux attributs.

  On se reportera à la section *Options de polices* ci-dessous pour une liste de tous les attributs possibles.

font create *?nomPolice? ?option valeur ...?*
: Crée une nouvelle police nommée et renvoie son nom. L'argument *nomPolice* spécifie le nom de la police ; s'il est omis, Tk fabrique un nouveau nom de la forme **font** x, où x est une valeur entière. Il peut y avoir un nombre quelconque de paires *option-valeur* spécifiant les attributs souhaités pour la police créée.

La section *Options de polices* ci-dessous indique la liste de tous les attributs possibles.

font delete *nomPolice ?nomPolice ...?*
Supprime les polices nommées spécifiées. Si certains composants graphiques utilisent ces polices, les polices ne seront réellement supprimées que lorsque tous les composants auront été libérés. En attendant, les composants continueront d'être affichés avec les dernières valeurs connues concernant la ou les polices en question. Si une police supprimée est par la suite recréée au moyen d'un appel à la commande **font create**, les composants utiliseront la nouvelle police nommée et se réafficheront en utilisant les nouvelles valeurs des attributs.

font families ?**-displayof** *fenêtre*?
La valeur de retour est la liste des noms (sans distinction de majuscules et de minuscules) de toutes les familles de police existant sur l'écran de la fenêtre désignée par l'argument *fenêtre*. Si l'argument *fenêtre* est omis, la fenêtre principale est utilisée par défaut.

font measure *police* ?**-displayof** *fenêtre*? *texte*
Mesure l'espace qu'occuperait la chaîne *texte* si elle était affichée avec la police *police* dans la fenêtre *fenêtre*. Si l'argument *fenêtre* est omis, la fenêtre principale est utilisée par défaut. La valeur de retour est la largeur totale en pixels de l'argument *texte*, à l'exclusion des corrections introduites par certains caractères. Si la chaîne comporte des caractères de fin de ligne ou des tabulations, ceux-ci ne sont pas substitués ou traités de manière particulière dans le calcul.

font metrics *police* ?**-displayof** *fenêtre*? *?option?*
Renvoie l'information métrique de la police lorsque celle-ci est utilisée sur le moniteur de la fenêtre désignée par l'argument *fenêtre*. Si l'argument *fenêtre* est omis, la fenêtre principale est utilisée par défaut. Si l'argument *option* est spécifié, la commande renvoie la valeur de cette propriété métrique particulière ; s'il est omis, la valeur de retour est une liste de toutes les propriétés métriques et de leurs valeurs. On se reportera à la section *Métriques de polices* ci-dessous pour une liste de toutes les propriétés métriques possibles.

font names
La valeur de retour est une liste de toutes les polices nommées actuellement définies.

Description de polices

Les formats suivants sont acceptés en tant que description de police dans toutes les commandes où un argument *police* doit être spécifié. Les mêmes formes peuvent aussi être utilisées pour spécifier l'option **-font** d'un certain nombre de composants graphiques.

[1] *nomPolice*
Le nom d'une police créée en utilisant la commande **font create**. Lorsqu'un

composant graphique utilise une police par son nom, il est garanti qu'il n'y aura jamais d'erreur dès lors que cette police existe, même si on lui ajoute des attributs erronés. Si la police nommée ne peut être affichée avec les attributs exacts requis, une autre police avec des propriétés approchantes sera substituée automatiquement.

[2] *policeSystème*
Un nom de police spécifique à une certaine plate-forme, interprété par le gestionnaire graphique du système. Cela inclut en particulier, sous le système X, des polices XLFD pour lesquelles un astérisque (∗) permet d'abréger certains champs au milieu du nom.
Les polices système suivantes sont supportées :
- sous système X : tous les noms de polices X valides, y compris ceux qui sont listés par la commande xlsfonts ;
- sous Windows : *system*, *ansi*, *device*, *systemfixed*, *ansifixed*, *oemfixed* ;
- sous MacOS : *system*, *application*.

[3] *famille ?taille? ?style? ?style ...?*
Une liste dont le premier élément est la police souhaitée *famille* et dont le second élément (optionnel) est la taille souhaitée. L'interprétation de l'attribut *taille* obéit aux mêmes règles que celles décrites pour l'option **-size** ci-dessous. Tous les arguments optionnels suivants sont des arguments de style. Les valeurs possibles pour ces arguments sont : *normal*, *bold*, *roman*, *italic*, *underline*, *overstrike*.

[4] *XLFD*
Un nom de police sur système Unix prend la forme complète suivante, dans laquelle les arguments sont séparés par un tiret :
`-foundry-family-weight-slant-setwidth-addstyle-pixel-point-resx-resy-spacing-width-charset-encoding`.
Dans la notation XLFD, on peut utiliser un astérique (∗) pour omettre certains champs : il doit y avoir un astérisque par champ omis, sauf à la fin où un astérisque désigne tous les champs restants s'ils doivent être omis. Tous les champs omis reçoivent des valeurs par défaut. En principe, les tailles sont exprimées en pixels et non en points, bien que de nombreuses applications considèrent que les points sont en fait des pixels dans leurs calculs conduisant à des caractères affichés généralement trop grands.

[5] *option valeur ?option valeur ...?*
Une liste de paires de type *option-valeur* spécifiant les attributs désirés de la police dans le même format que celui utilisé par la commande **font create**.

Lorsqu'un argument *police* de description de police est utilisé, le système tente de l'interpréter selon l'une des cinq règles ci-dessus, dans l'ordre indiqué. Les formes [1] et [2] doivent désigner une police existant sous le nom indiqué ou une police système. Les trois autres cas sont acceptés sur toutes les plates-formes et la police la plus approchante sera finalement utilisée. Si aucune police approchante n'est trouvée, une police par défaut sera substituée, variable selon les plates-formes. Si aucune des cinq formes énumérées ne correspond, une erreur est générée.

Métriques

Les options suivantes sont utilisées par la commande **font metrics** pour obtenir des données spécifiques déterminées lorsque la police est créée. Ces propriétés concernent la police globalement, non chaque caractère individuellement.

-ascent
La plus grande hauteur en pixels d'une lettre de la police au-dessus de la ligne de base, augmentée d'un blanc supplémentaire éventuellement ajouté par le créateur de la police.

-descent
La plus grande profondeur en pixels d'une lettre de la police en dessous de la ligne de base, augmentée d'un blanc supplémentaire éventuellement ajouté par le créateur de la police.

-linespace
Renvoie l'interligne vertical exprimé en pixels, c'est-à-dire la distance entre deux lignes successives utilisant la même police de telle sorte que les caractères des deux lignes ne se chevauchent jamais. C'est en général la somme des valeurs fournies par les options **-ascent** et **-descent**.

-fixed
Renvoie une valeur booléenne permettant de déterminer si la police est à espacement fixe (valeur 1) ou proportionnel (valeur 0). Dans une police à espacement fixe, tous les caractères occupent la même largeur. Les largeurs des caractères de contrôle et de tabulation ou des caractères non-imprimables ne sont pas comptabilisées pour déterminer la valeur de cette option.

Options de polices

Les options suivantes sont supportées sur toutes les plates-formes, et sont utilisées pour construire une police nommée ou pour spécifier une police utilisant le style [5] vu ci-dessus :

-family *nom*
Le nom de la famille de police (sans distinction des majuscules et des minuscules). Tk garantit de supporter les familles de polices standard nommées *Courier*, *Times*, *Helvetica*. La famille de police native la plus proche d'une de ces dernières sera automatiquement substituée au besoin. L'argument *nom* peut aussi être le nom d'une police native sur une plate-forme particulière ; dans ce cas, cependant, les choses fonctionneront comme prévu sur cette plate-forme mais pourraient ne pas fonctionner sur les autres. Si la famille n'est pas spécifiée ou reconnue, une police par défaut (dépendant de la plate-forme) sera choisie.

-overstrike *bool*
Cette valeur booléenne spécifie si les caractères de la police doivent être barrés ou non en leur milieu par une ligne horizontale. La valeur par défaut est *false*.

-size *taille*
La taille désirée pour la police. Si l'argument *taille* est un nombre positif, il est interprété comme une taille en points. Si l'argument *taille* est un nombre négatif, sa valeur absolue est interprétée comme une taille en points. Si une police ne peut pas être affichée à une certaine taille, une taille approchante sera utilisée à la place. Si l'argument *taille* n'est pas spécifié ou s'il est nul, une taille de police par défaut (dépendant de la plate-forme) sera choisie.
Les tailles doivent normalement être spécifiées en points de telle sorte que l'application ait la même échelle à l'écran même si la résolution de l'écran est modifiée ou si le script est exécuté sur une autre plate-forme. Néanmoins, spécifier une taille en pixels peut s'avérer utile dans certaines circonstances, par exemple si un texte doit s'aligner par rapport à une image de taille fixée. La correspondance entre points et pixels est établie au démarrage de l'application, en fonction des propriétés du moniteur installé, mais peut être modifiée au moyen de la commande **tk scaling**.

-slant *inclinaison*
La quantité dont les caractères de la police doivent être inclinés par rapport à la verticale. Les valeurs possibles pour l'argument *inclinaison* sont *roman* et *italic*. L'inclinaison disponible la plus proche sera utilisée. La valeur par défaut est *roman*.

-underline *bool*
Cette valeur booléenne spécifie si les caractères de la police doivent être soulignés ou non. La valeur par défaut est *false*.

-weight *poids*
La graisse (ou le poids) nominale des caractères de la police. Les valeurs usuelles sont *normal* pour une épaisseur normale et *bold* pour une police en gras. Le poids disponible le plus proche sera utilisé. La valeur par défaut est *normal*.

frame

Permet de créer et de manipuler des cadres

Syntaxe

frame *cheminCadre ?options?*

Description

La commande **frame** crée une nouvelle fenêtre (indiquée par l'argument *cheminCadre*) et en fait un composant graphique de type *cadre*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques du cadre telles que la couleur de fond ou le relief. La commande **frame** renvoie la valeur de son argument *cheminCadre*.

Un cadre est un composant graphique simple appelé principalement à servir de conteneur pour réaliser des dispositions complexes de composants dans des fenêtres. Les seules caractéristiques d'un cadre sont sa couleur de fond et sa bordure optionnelle en trois dimensions pour le faire apparaître en creux ou en relief.

Opérations sur les cadres

Avec l'instruction :
frame *cheminCadre ?options?*
la commande **frame** crée une nouvelle commande Tcl dont le nom est précisément *cheminCadre*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le bouton. La forme générale est :

cheminCadre opération ?arg arg ...?

Les opérations reconnues par les composantes de type *frame* sont les suivantes :

cheminCadre **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **frame**.

cheminCadre **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration du cadre. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminCadre*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **frame**.

-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus

TAB. 6 – *Options standard reconnues par les composants de type* frame.

Liaisons par défaut

Lorsqu'un nouveau cadre est créé, il ne possède aucune liaison par défaut. Les cadres n'ont pas pour fonction d'être interactifs.

Options standard

Les options standard supportées par les composants de type *frame* sont rassemblées dans le tableau 6. Ces options sont décrites en détail à la page 142.

Options spécifiques

-background **background** **Background**

Cette option est identique à l'option standard **-background**, excepté que sa valeur peut aussi être spécifiée comme une chaîne vide. Dans ce cas, le composant n'affichera ni fond, ni bordure, et aucune couleur de sa carte de couleurs ne sera utilisée pour le fond ou la bordure.

-class **class** **Class**

Spécifie une classe pour la fenêtre. Cette classe sera utilisée pour rechercher dans la base de données de ressources d'autres options pour la fenêtre ou par la suite pour établir des liaisons (*bindings*). L'option **class** ne peut pas être modifiée au moyen de la commande **configure**.

-colormap **colormap** **Colormap**

Spécifie une carte de couleurs à utiliser pour la fenêtre. La valeur peut être ou bien *new*, auquel cas une nouvelle carte de couleurs est créée pour la fenêtre et ses enfants, ou bien le nom d'une autre fenêtre (qui doit être sur le même écran et avoir les mêmes caractéristiques visuelles que *cheminCadre*), auquel cas la nouvelle fenêtre utilisera la carte de couleurs de la fenêtre spécifiée. Si l'option **colormap** n'est pas spécifiée, la nouvelle fenêtre utilise la même carte de couleurs que son parent. Cette option ne peut pas être modifiée au moyen de la commande **configure**.

-container **container** **Container**

Cette option doit prendre une valeur booléenne. Si la valeur est *true*, cela signifie que cette fenêtre sera utilisée pour contenir une autre application (par exemple, une fenêtre de type *toplevel* pourra être imbriquée au moyen de l'option **-use**). La fenêtre supportera les protocoles appropriés du serveur de fenêtres pour les requêtes concernant les caractéristiques géométriques. La fenêtre ne doit pas avoir d'enfants dans cette application. Cette option ne peut pas être modifiée au moyen de la commande **configure**.

-height **height** **Height**

Spécifie la hauteur désirée pour la fenêtre, en unités d'écran. Si cette option est négative ou nulle, la fenêtre ne requerra aucune dimension.

-visual **visual** **Visual**

Spécifie la classe visuelle souhaitée pour la nouvelle fenêtre. Sa valeur peut être une des constantes symboliques suivantes : *directcolor*, *grayscale*, *greyscale*, *pseudocolor*, *staticcolor*, *staticgray*, *staticgrey*, *truecolor*, *best*, *default*, ou le nom d'une autre fenêtre (cf. p. **??**). Si cette option n'est pas spécifiée, la nouvelle fenêtre utilisera les caractéristiques visuelles de son parent. L'option **visual** ne peut pas être modifiée au moyen de la commande **configure**.

-width **width** **Width**

Spécifie la largeur désirée pour la fenêtre, en unités d'écran. Si cette option est négative ou nulle, la fenêtre ne requerra aucune dimension.

grab

Limite des événements de pointeur et de clavier à une hiérarchie d'objets.

Syntaxe

grab ?**-global**? *fenêtre*
grab *sous-commande ?arg arg...?*

Description

Cette commande implémente de simples saisies d'événements de pointeur de souris et de clavier pour Tk. Le terme *grab* désigne une saisie d'événement : cette commande permet de saisir, c'est-à-dire d'intercepter, des événements pour les diriger sur une fenêtre particulière.

Les saisies de Tk sont différentes de celles décrites dans la documentation de Xlib. Lorsqu'une saisie est installée pour une fenêtre particulière, Tk restreint tous les événements de souris à cette fenêtre et à ses descendants. Dès que le pointeur se trouve dans la hiérarchie d'objets de cette fenêtre, il se comporte exactement comme s'il n'y avait pas de saisie et tous les événements sont rapportés de manière normale. En revanche, si le pointeur se trouve en dehors de la fenêtre, les événements de clics de boutons de la souris ou de déplacement de la souris sont envoyés à la fenêtre et les événements d'entrée et de sortie de la zone de la fenêtre sont simplement ignorés.

La fenêtre prend donc possession du pointeur : les fenêtres extérieures seront insensibles aux événements de souris tant que la saisie sera en vigueur. Les fenêtres de la hiérarchie issue de la fenêtre de saisie peuvent aussi comporter des objets de type *toplevel* qui continueront donc de recevoir, ainsi que tous leurs descendants, les événements de souris.

Deux formes de saisies sont possibles : locales et globales. Une saisie locale affecte seulement l'application opérant la saisie : les événements seront rapportés aux autres applications comme si la saisie n'avait pas eu lieu. Une saisie globale bloque toutes les applications sur l'écran et le gestionnaire de fenêtres ne recevra pas les événements de souris lui non plus. Par défaut les saisies sont locales.

Pendant une saisie locale, les événements de clavier (et non plus de souris) sont reçus normalement : le gestionnaire de fenêtres désigne les applications qui doivent recevoir les événements de clavier (touches enfoncées ou relâchées) et s'ils sont envoyés à l'une des fenêtres de l'application qui a installé la saisie, ils sont redirigés vers le widget qui a la focalisation (cf. la commande *focus* p 84).

Pendant une saisie globale, Tk s'empare du clavier de telle sorte que tous les événements de clavier seront toujours dirigés sur l'application qui a installé la saisie. La commande **focus** est toujours utilisée pour déterminer quelle fenêtre de l'application est focalisée, c'est-à-dire reçoit ces événements de clavier. Cette saisie du clavier est levée dès que la saisie globale est elle-même levée.

Les saisies s'appliquent à un moniteur particulier. Si une application possède des fenêtres sur plusieurs moniteurs à la fois, elle peut installer des saisies distinctes sur chacun des moniteurs. Une saisie sur un moniteur particulier affecte seulement les fenêtres présentes sur ce moniteur.

Il est possible que différentes applications sur un même moniteur possèdent simultanément plusieurs saisies locales, mais une seule application peut installer une saisie globale.

La commande **grab** peut prendre une des formes suivantes :

grab ?**-global**? *fenêtre*
: Cette commande est synonyme de **grab set** qui est décrite ci-dessous.

grab current *?fenêtre?*
: Si l'argument *fenêtre* est spécifié, cette commande renvoie le nom de la fenêtre de saisie courante de cette application pour le moniteur contenant la fenêtre *fenêtre*. La valeur de retour est une chaîne vide s'il n'y a pas de fenêtre de saisie. Si l'argument *fenêtre* est omis, la commande renvoie une liste dont les éléments sont toutes les fenêtres saisies par l'application sur tous les moniteurs. Ce sera une chaîne vide si l'application n'a installé aucune saisie.

grab release *fenêtre*
: Lève la saisie sur *fenêtre* s'il y en a une. La commande renvoie une chaîne vide.

grab set ?**-global**? *fenêtre*
: Installe une saisie sur *fenêtre*. Si l'option **-global** est spécifiée, cette saisie sera globale, sinon elle sera locale. Si une saisie était déjà en vigueur pour cette application sur le moniteur contenant la fenêtre *fenêtre*, elle est automatiquement levée. Si une saisie de même forme (globale ou locale) que la nouvelle saisie requise existe déjà sur la fenêtre, la commande ne fait rien. Cette commande renvoie une chaîne vide.

grab status *fenêtre*
: Renvoie *none* si aucune saisie n'est actuellement installée sur *fenêtre*, *local* si une saisie locale y est installée et *global* en cas de saisie globale.

Précautions

Les saisies installées par la commande **grab** ne font pas bon ménage avec les saisies Xlib. Si des applications essaient de manipuler les mécanismes de saisie de X, les choses risquent de ne pas fonctionner correctement. Si un processus unique gère différentes applications Tk, une seule de ces applications peut installer une saisie locale pour un moniteur donné à un moment donné. Si les applications se trouvent dans des processus distincts, il n'y a pas de restriction.

grid

Gestionnaire de placement pour arranger les composants sur une grille.

Syntaxe

grid *sous-commande arg?arg ...?*

Description

La commande **grid** est utilisée pour interagir avec un gestionnaire de placement dit *grille de placement* qui arrange les composants graphiques en rangées et colonnes à l'intérieur d'une autre fenêtre appelée *fenêtre maître*. Chacun des objets placés est qualifié d'*objet esclave*. La commande **grid** peut prendre plusieurs formes selon l'argument *option* qui lui est adjoint :

grid *esclave?esclave ...??options?*
Si le premier argument de **grid** est un nom de fenêtre (n'importe quelle valeur commençant par un point), la commande est exécutée de la même manière que **grid configure**.

grid bbox *maître?colonne rangée??colonne2 rangée2?*
Sans arguments, cette commande envoie une liste de quatre nombres décrivant la zone d'écran occupée par la grille. Les deux premiers éléments de cette liste sont l'abscisse et l'ordonnée (mesurées en pixels relativement à la boîte elle-même) du coin supérieur gauche de la grille dans la fenêtre maître et les deux derniers sont la largeur et la hauteur de la boîte. Si un seul argument *colonne* ou *rangée* est spécifié dans l'instruction, la boîte délimitant cette cellule est renvoyée, la cellule en haut à gauche étant numérotée à partir de zéro. Si à la fois *colonne* et *rangée* sont spécifiés, la boîte délimitant la zone qui couvre ces rangées et colonnes est renvoyée.

grid columnconfigure *maître index?-option valeur...?*
Permet d'obtenir ou de modifier les propriétés de la colonne d'indice *index* de la fenêtre maître spécifiée par l'argument *maître*. Les options valides sont **-minsize**, **-weight** et **-pad** . Si une ou plusieurs options sont fournies, l'argument *index* peut être donné comme une liste d'indices de colonnes auxquelles les options de configuration doivent s'appliquer. L'option **-minsize** fixe la taille minimum, en unités d'écran, autorisée pour cette colonne. L'option **-weight** est une valeur entière et fixe le poids relatif permettant de répartir un espace excédentaire entre les colonnes. Un poids nul indique que la colonne ne doit pas varier par rapport à la taille requise. Une colonne dont le poids est 2 croît deux fois plus vite qu'une colonne de poids 1 lorsqu'un espacement supplémentaire est alloué. L'option **-pad** spécifie le nombre d'unités d'écran qui seront ajoutées au plus grand objet contenu entièrement dans cette colonne lorsque la grille de placement calcule la taille de la fenêtre maître.

Si un seul nom d'option est spécifié, sans aucune valeur, c'est la valeur courante de cette option qui est renvoyée. Si seuls la fenêtre maître et l'indice sont spécifiés, tous les réglages courants sont renvoyés dans une liste de paires de type *option-valeur*.

grid configure *esclave ?esclave ...? ?options?*

Les arguments sont les noms d'une ou de plusieurs fenêtres esclaves suivis de paires d'arguments qui permettent de spécifier la façon de gérer ces objets esclaves. Les caractères `-`, `x` et `^` peuvent être spécifiés à la place d'un nom de fenêtre pour modifier l'emplacement de l'objet *esclave*, comme il est expliqué dans la section *Placement relatif* ci-dessous. Les options suivantes sont supportées :

-column *n*

Insère l'esclave de telle sorte qu'il occupe la *n*-ième colonne de la grille. Les numéros de colonnes commencent à 0. Si cette option n'est pas fournie, l'esclave est placé immédiatement à droite de l'esclave précédent spécifié dans cet appel à la commande *grille*, ou dans la colonne 0 s'il est le premier objet esclave. On peut utiliser la lettre **x** pour représenter des colonnes vides sur cette rangée de la grille : pour chaque **x** qui précède immédiatement *esclave*, la position de colonne est incrémentée de 1, ce qui la décale d'une unité vers la droite.

-columnspan *n*

Insère l'esclave de telle sorte qu'il occupe *n* colonnes de la grille. La valeur par défaut est une colonne, à moins que le nom de la fenêtre ne soit suivi d'un tiret -, auquel cas le nombre de colonnes couvertes est incrémenté de 1 pour chaque tiret qui suit immédiatement.

-in *autre*

Insère les esclaves dans la fenêtre maître spécifiée par l'argument *autre*. Par défaut, il s'agit de la fenêtre parent du premier esclave.

-ipadx *quantité*

L'argument *quantité* spécifie la quantité d'espace intérieur à laisser horizontalement de chaque côté du ou des esclaves. Cet espace est ajouté à l'intérieur des bords de l'objet esclave. L'argument *quantité* doit être une distance d'écran valide, telle que 2 ou .5c (cf. p. 32). La valeur par défaut est 0.

-ipady *quantité*

L'argument *quantité* spécifie la quantité d'espace intérieur à laisser verticalement en haut et en bas du ou des esclaves. Cet espace est ajouté à l'intérieur des bords de l'objet esclave. La valeur par défaut est 0.

-padx *quantité*

L'argument *quantité* spécifie, en unités d'écran, la quantité d'espace extérieur à laisser horizontalement de chaque côté du ou des esclaves. Cet espace est ajouté à l'extérieur des bords de l'objet esclave. La valeur par défaut est 0.

-pady *quantité*
L'argument *quantité* spécifie, en unités d'écran, la quantité d'espace extérieur à laisser verticalement en haut et en bas du ou des esclaves. Cet espace est ajouté à l'extérieur des bords de l'objet esclave. La valeur par défaut est 0.

-row *n*
Insère l'esclave de telle sorte qu'il occupe la *n*-ième rangée de la grille. Les numéros de rangées commencent à 0. Si cette option n'est pas fournie, l'esclave est placé sur la même rangée que l'esclave précédent spécifié dans cet appel à la commande **grid**, ou dans la première rangée inoccupée s'il est le premier objet esclave.

-rowspan *n*
Insère l'esclave de telle sorte qu'il occupe *n* rangées de la grille. La valeur par défaut est d'une rangée, à moins que le nom de la fenêtre esclave ne soit suivi de caractères ^, auquel cas le nombre de rangées couvertes est incrémenté de 1 pour chaque accent circonflexe qui suit immédiatement.

-sticky *style*
Si une cellule de la grille réservée à un objet esclave est plus grande que les dimensions requises, cette option peut être utilisée pour positionner (ou étirer) l'objet esclave à l'intérieur de cette cellule. L'argument *style* est une chaîne contenant zéro ou plus des caractères **n**, **s**, **e** et **w** pour désigner respectivement les bords nord, sud, est et ouest auxquels l'objet esclave doit rester attaché en cas d'expansion : si à la fois **n** et **s** sont spécifiés (resp. **e** et **w**), l'esclave sera étiré pour remplir la hauteur (resp. la largeur) totale de sa cavité. La chaîne peut également contenir des espaces et des virgules qui seront ignorées. L'option **-sticky** joue le même rôle que la combinaison des options **-anchor** et **-fill** utilisée par le gestionnaire de placement **pack**. La valeur par défaut {} fait que l'esclave est centré dans sa cavité à la taille requise.

Si l'un des esclaves est déjà géré par la grille de placement, toute option non spécifiée conservera sa valeur précédente et ne prendra pas les valeurs par défaut.

grid forget *esclave ?esclave ...?*
Retire chaque objet *esclave* de la grille pour son maître et supprime sa fenêtre. Ces esclaves ne seront plus contrôlés par la grille de placement. Les options de configuration pour cette fenêtre sont oubliées, de telle sorte que si l'esclave est géré à nouveau par la grille, les réglages initiaux seront utilisés.

grid info *esclave*
Renvoie une liste dont les éléments représentent l'état de la configuration actuelle de l'esclave désigné par l'argument *esclave* sous la forme de paires *option-valeur* identiques à celles de la commande **grid configure**. Les deux premiers éléments de la liste sont « **-in** *maître* » où *maître* désigne le maître de cet objet esclave.

grid location *maître x y*
Étant donné des valeurs *x* et *y* en unités d'écran relatives à la fenêtre maître, les numéros de colonne et rangée en ce point (*x*,*y*) est renvoyé. Pour les emplacements qui sont au-dessus ou à gauche de la grille, la valeur -1 est renvoyée.

grid propagate *maître ?bool?*
Si l'argument *bool* a une valeur vraie (comme 1 ou *on*), la propagation est activée pour le maître spécifié par l'argument *maître* qui doit être un nom de fenêtre (voir la section *Propagation géométrique* ci-dessous). C'est la situation par défaut. Si l'argument *bool* a une valeur correspondant à *faux* (comme 0 ou *off*), la propagation est désactivée. Dans les deux cas, une chaîne vide est renvoyée. Si *bool* est omis, la commande renvoie 0 ou 1 selon que la propagation est actuellement activée ou non pour le maître *maître*.

grid rowconfigure *maître index ?-option valeur...?*
Permet d'obtenir ou de modifier les propriétés de la rangée d'indice *index* de la fenêtre maître spécifiée par l'argument *maître*. Les options valides sont **-minsize**, **-weight** et **-pad**. Le fonctionnement et la signification de ces options sont les mêmes qu'avec l'opération **columnconfigure** étudiée ci-dessus (en remplaçant les colonnes par les rangées).

grid remove *esclave ?esclave ...?*
Supprime chacun des objets *esclave* de la grille maître et efface leur fenêtre. Les esclaves ne seront plus gérés par la grille de placement. Néanmoins, les options de configuration pour cette fenêtre sont mémorisées, de telle sorte que si l'esclave est géré à nouveau par la grille de placement, les valeurs précédentes seront retenues.

grid size *maître*
Renvoie la taille de la grille (en colonnes puis rangées) pour la fenêtre *maître*. La taille est déterminée soit par l'objet *esclave* occupant la plus large rangée ou colonne, soit par la plus large colonne ou rangée avec des options **-minsize**, **-weight** ou **-pad** non nulles.

grid slaves *maître ?option valeur?*
Si aucune option n'est fournie, une liste de tous les esclaves dans la fenêtre *maître* est renvoyée, les plus récents venant en premier. L'argument *option* peut être ou bien **-row**, ou bien **-column**, ce qui a pour effet de ne renvoyer que les esclaves de la rangée (ou colonne) spécifiée par l'argument *valeur*.

Placement relatif

La commande **grid** est capable de créer des dispositions sans spécifier les informations de rangées et colonnes pour chaque esclave. Cela permet de réarranger, ajouter, supprimer les esclaves sans fournir de numéros explicites de rangées ou colonnes. Si aucune information de rangée ou de colonne n'est fournie, des valeurs par défaut sont choisies pour les options **-row**, **-rowspan**, **-column** et **-columnspan** en fonction de la disposition en cours dans la grille, la place d'un esclave par rapport aux autres et la présence de symboles `-`, `x` et `^` qui permettent d'augmenter

le nombre de colonnes ou de rangées couvertes par un même esclave.

Chaque symbole `-` augmente vers la gauche le nombre de colonnes couvertes. On peut en spécifier plusieurs successivement. Un tiret ne peut pas suivre un `^` ou un x.

Chaque symbole `x` permet de laisser une colonne vide entre l'esclave à gauche et l'esclave à droite.

Chaque symbole `^` augmente le nombre de rangées couvertes. Le nombre de circonflexes `^` dans une rangée doit correspondre au nombre de colonnes couvertes par l'esclave.

L'algorithme de grille

La grille de placement dispose les esclaves en trois étapes :

- tout d'abord, la taille minimum nécessaire pour intégrer tous les esclaves est calculée et, si la propagation est activée, une demande est adressée à la fenêtre maître pour qu'elle prenne cette taille ;
- la taille requise est comparée à la taille actuelle du maître. Si elles diffèrent, de l'espace est ajouté ou ôté selon les besoins ;
- enfin, chaque esclave est positionné dans ses rangées et colonnes conformément aux réglages de l'option **-sticky**.

Propagation géométrique

La grille de placement calcule normalement la dimension que doit avoir un maître pour satisfaire exactement les besoins de ses esclaves et fixe la largeur et la hauteur du maître à ces valeurs. Cela conduit l'information géométrique à se propager vers le haut dans la hiérarchie des fenêtres jusqu'à une fenêtre de premier niveau, ce qui fait que le sous-arbre entier se dimensionne en fonction des dimensions des fenêtres qui se trouvent le plus bas dans la hiérarchie. On peut néanmoins utiliser la commande **grid propagate** pour désactiver la propagation pour un ou plusieurs maîtres. Cela s'avère utile lorsque, par exemple, on souhaite qu'une fenêtre maître ait une taille fixée d'avance.

Restrictions concernant les fenêtres maîtres

Pour chaque esclave, le maître doit être soit le parent de l'esclave (situation par défaut), soit un collatéral (descendant d'un parent de l'esclave). Cette restriction garantit que l'esclave peut être placé sur n'importe quelle partie visible de son maître, sans danger que l'objet esclave ne soit tronqué par son parent. En outre, dans un même appel à la commande **grid**, tous les esclaves doivent avoir le même maître.

Ordre d'empilement

Si le maître d'un esclave n'est pas son parent, il faut s'assurer que l'esclave est bien au-dessus du maître dans l'ordre d'empilement. Sinon, le maître masquera l'esclave et on aura l'impression que l'esclave n'a pas été placé correctement. La meilleure façon de s'en assurer est de créer la fenêtre maître en premier : la fenêtre la plus récente est toujours la plus haute dans l'ordre d'empilement. Une autre solution consiste à utiliser les commandes **raise** et **lower** pour modifier l'ordre d'empilement du maître ou de l'esclave.

image

Permet de créer et de manipuler des images.

Syntaxe

image *sous-commande ?arg arg ...?*

Description

La commande **image** est utilisée pour créer des images, les supprimer et obtenir des données les concernant. Elle peut prendre plusieurs formes selon l'argument *sous-commande*. Les sous-commandes reconnues sont les suivantes :

image create *type ?nom? ?option valeur ...?*
: Crée une nouvelle image et renvoie son nom. L'argument *type* spécifie le type de l'image, et doit être l'un des types actuellement définis (par exemple **bitmap**). L'argument *nom* spécifie le nom de l'image ; s'il est omis, Tk choisit lui-même un nom de la forme **image***n*, où *n* est un nombre entier. Il peut y avoir un nombre quelconque de paires de la forme *option-valeur* pour spécifier des options de configuration pour l'image. Les options valides sont définies séparément pour chaque type d'image : les sous-commandes **bitmap** et **photo** sont présentées respectivement aux pages 24 et 154.
Si une image d'un certain nom existe déjà, elle sera remplacée par la nouvelle et toutes les instances de cette image seront mises à jour et réaffichées en conséquence.

image delete *?nom nom...?*
: Détruit les images désignées par les arguments *nom* et renvoie une chaîne vide. Si des instances de ces images sont affichées dans certains composants, les images ne seront détruites effectivement que lorsque toutes les instances auront été libérées. Néanmoins, l'association entre les instances et le gestionnaire d'images sera interrompue. Les instances existantes conserveront leurs dimensions mais redessineront leur contenu. Si une image détruite est recréée par un autre appel à la commande **image create**, les instances existantes utiliseront la nouvelle image.

image height *nom*
: Renvoie une valeur décimale donnant la hauteur de l'image *nom* en pixels.

image inuse *nom*
: Renvoie une valeur booléenne indiquant si l'image *nom* est utilisée par un composant graphique.

image names
: Renvoie une liste contenant les noms de toutes les images existantes.

image type *nom*
: Renvoie le type de l'image *nom*.

image types
Renvoie une liste dont les éléments sont tous les types d'images valides.

image width *nom*
Renvoie une valeur décimale donnant la largeur de l'image *nom* en pixels.

Deux types d'images sont prédéfinis dans Tk et sont donc disponibles pour toute application :

bitmap
ce sont des images bichromes, autrement dit représentées au moyen de deux couleurs seulement : une couleur pour les pixels dits de premier plan, une couleur de fond ou pas de couleur du tout pour des effets de transparence. Typiquement, cela concerne les images en noir et blanc. La commande **bitmap** associée est présentée à la page 24.

photo
ce sont des images couleurs utilisant l'effet de *dithering* pour approximer les couleurs sur les écrans dotés de capacités limitées. La commande **photo** associée est présentée à la page 154.

label

Permet de créer et de manipuler des étiquettes ou des champs non éditables.

Syntaxe

label *cheminÉtiquette ?options?*

Description

La commande **label** crée une nouvelle fenêtre (indiquée par l'argument *cheminÉtiquette*) et en fait un composant graphique de type *étiquette*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques de l'étiquette telles que la couleur, la police, le texte ou le relief initial. La commande **label** renvoie la valeur de son argument *cheminÉtiquette*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminÉtiquette*, mais en revanche l'objet parent de *cheminÉtiquette* doit exister effectivement.

Une étiquette est un composant graphique qui affiche un texte, une icone bitmap ou une image. Si l'étiquette est textuelle, elle est entièrement composée dans une même police ; le texte peut occuper plusieurs lignes à l'écran, soit par utilisation de sauts de lignes, soit parce qu'un paramètre **-wrapLength** a été fixé pour forcer les coupures de ligne. Il est enfin possible de souligner certains caractères du texte avec l'option **-underline**.

Une étiquette accepte quelques manipulations simples telles que le changement de leur relief ou de leur texte.

Opérations sur les étiquettes

Avec l'instruction :

label *cheminÉtiquette ?options?*

la commande **label** crée une nouvelle commande Tcl dont le nom est précisément *cheminÉtiquette*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant l'étiquette. La forme générale est :

cheminÉtiquette opération ?arg arg ...?

Les opérations reconnues par les composantes de type *label* sont les suivantes :

cheminÉtiquette **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **label**.

cheminÉtiquette **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de l'étiquette.

-activebackground	activeBackground	Foreground
-activeforeground	activeForeground	Background
-anchor	anchor	Anchor
-background ou -bg	background	Background
-bitmap	bitmap	Bitmap
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-image	image	Image
-justify	justify	Justify
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus
-text	text	Text
-textvariable	textVariable	Variable
-underline	underline	Underline
-wraplength	wrapLength	WrapLength

TAB. 7 – *Options standard reconnues par les composants de type* label.

Si l'argument *option* n'est pas spécifié, la commande renvoie la liste de toutes les options disponibles pour *cheminÉtiquette*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **label**.

Liaisons par défaut

Lorsqu'une nouvelle étiquette est créée, elle ne possède aucune liaison par défaut. Les étiquettes n'ont pas pour fonction d'être interactives.

Options standard

Les options standard supportées par les composants de type *label* sont rassemblées dans le tableau 7. Ces options sont décrites en détail à la page 142.

Options spécifiques

-height **height** **Height**

Spécifie une hauteur pour l'étiquette. Si une image ou un bitmap est affiché dans l'étiquette, la valeur est mesurée en unités d'écran ; pour du texte, elle est mesurée en lignes de texte. Si cette option n'est pas spécifiée, la hauteur de l'étiquette est calculée à partir de la taille de l'image, du bitmap ou du texte qu'elle contient.

-state **state** **State**

Spécifie l'un des trois états *normal*, *active* ou *disabled* pour l'étiquette. Dans l'état normal, l'étiquette est affichée en utilisant les options **foreground** et **background**. Dans l'état activé, l'étiquette est affichée en utilisant les options **activeForeground** et **activeBackground**. Dans l'état désactivé, les options **disabledForeground** et **background** déterminent la façon dont l'étiquette est affichée.

-width **width** **Width**

Spécifie une largeur pour l'étiquette. Si une image ou un bitmap est affiché dans l'étiquette, la valeur est mesurée en unités d'écran ; pour du texte, elle est mesurée en nombre de caractères. Si cette option n'est pas spécifiée, la largeur de l'étiquette est calculée à partir de la taille de l'image, du bitmap ou du texte qu'elle contient.

listbox

Permet de créer et de manipuler les boîtes de listes

Syntaxe

listbox *cheminBoîte ?options?*

Description

La commande **listbox** crée une nouvelle fenêtre (indiquée par l'argument *cheminBoîte*) et en fait un composant graphique de type *boîte de liste*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques de la boîte de liste telles que la couleur, la police, le texte et le relief. La commande **listbox** renvoie la valeur de son argument *cheminBoîte*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminBoîte*, mais en revanche l'objet parent de *cheminBoîte* doit exister effectivement.

Une boîte de liste est un composant graphique qui affiche, ligne par ligne, les éléments d'une liste de chaînes de caractères. Les boîtes de listes permettent de faire défiler leur contenu dans les deux directions en utilisant les options **xScrollCommand** et **yScrollCommand** standard. Au moment de sa création, une boîte de liste ne contient aucun élément. Les éléments sont ajoutés ou supprimés par la suite au moyen d'opérations spécifiques. Par ailleurs un ou plusieurs éléments de la liste peuvent être sélectionnés. Une boîte de liste peut aussi exporter sa sélection (voir l'option **exportSelection** ci-dessous) conformément aux protocoles usuels du système X11 sous Unix : la valeur d'une sélection est le texte des éléments sélectionnés séparés par des symboles de saut de ligne.

Indices

De nombreuses opérations prennent comme argument un ou plusieurs indices spécifiant des éléments particuliers de la chaîne d'entrée. Ces indices peuvent prendre une des valeurs ou des constantes symboliques suivantes :

nombre
: Spécifie un élément par son indice numérique, où 0 correspond au premier élément de la boîte de liste.

active
: Indique l'élément contenant le curseur. Cet élément sera souligné lorsque la boîte de liste se trouvera focalisée et on peut le fixer au moyen de l'opération **activate** expliquée ci-dessous.

anchor
: Indique le point d'ancrage de la sélection que l'on fixe au moyen de l'opération **select anchor**.

end
: Indique la fin de la boîte de liste. En général il s'agit du dernier élément de la boîte de liste, mais pour certaines opérations telles que **index** et **insert**, cela fait référence à l'élément qui suit immédiatement le dernier.

@ *x*,*y*
: Indique l'élément qui couvre le point spécifié (en nombre de pixels) par les coordonnées *x* et *y* dans la boîte de liste. Si aucun élément ne couvre ce point, l'élément le plus proche du point est utilisé.

Dans toutes les sous-commandes qui suivent, les arguments *index*, *premier* et *dernier* désignent des indices d'éléments d'une boîte de liste et doivent être désignés sous une des formes ci-dessus.

Opérations sur les boîtes de listes

Avec l'instruction :
listbox *cheminBoîte ?options?*
la commande **listbox** crée une nouvelle commande Tcl dont le nom est précisément *cheminBoîte*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant la boîte. La forme générale est :
cheminBoîte opération ?arg arg ...?

Les opérations reconnues par les composantes de type *listbox* sont les suivantes :

cheminBoîte **activate** *index*
: Fixe l'élément indiqué par l'argument *index* comme élément actif. Cet élément est souligné lorsque la boîte de liste est focalisée Si *index* est en dehors de l'intervalle des éléments de la boîte de liste alors c'est l'élément qui est activé à la place.

cheminBoîte **bbox** *index*
: Renvoie une liste de quatre nombres décrivant la boîte qui délimite l'élément d'indice *index*. Les deux premiers éléments de cette liste sont l'abscisse et l'ordonnée (mesurées en pixels relativement à la boîte elle-même) du coin supérieur gauche de la zone d'écran couverte par l'élément et les deux derniers sont la largeur et la hauteur du widget. Cette boîte peut faire référence à une zone qui se trouve hors de la région visible du champ de saisie. Si aucune partie de l'élément d'indice *index* n'est visible à l'écran ou si l'indice correspond à un élément qui n'existe pas, la commande renvoie une chaîne vide. Si l'élément est partiellement visible, le résultat correspond à l'élément entier, même si une partie n'est pas visible.

cheminBoîte **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **listbox**.

cheminBoîte **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de la boîte de

liste. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminBoîte*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **listbox**.

cheminBoîte **curselection**
Renvoie une liste contenant les indices numériques de tous les éléments de la boîte de liste actuellement sélectionnés. Si aucun élément n'est sélectionné, la commande renvoie une chaîne vide.

cheminBoîte **delete** *premier ?dernier*?
Détruit tous les éléments de la boîte de liste dont les indices se trouvent entre les indices *premier* et *dernier* inclus. Si l'argument *dernier* est omis, il est identique à *premier*+1 ce qui revient à supprimer un seul élément. Cette commande renvoie une chaîne vide.

cheminBoîte **get** *premier ?dernier?*
Si l'argument *dernier* est omis, cette commande renvoie l'élément de la boîte de liste indiqué par *premier*, ou une chaîne vide si *premier* fait référence à un élément qui n'existe pas. Si l'argument *dernier* est spécifié, la commande renvoie la liste de tous les éléments de la boîte de liste compris entre les indices *premier* et *dernier* inclus.

cheminBoîte **index** *index*
Renvoie l'indice numérique correspondant à l'argument *index* lorsque celui-ci est désigné par une des constantes symboliques mentionnées ci-dessus. Si l'argument *index* est **end**, la valeur de retour est le nombre d'éléments de la boîte de liste (et non pas l'indice du dernier élément).

cheminBoîte **insert** *index ?élément élément ...*?
Insère un ou plusieurs éléments dans la liste immédiatement avant l'élément d'indice *index*. Si l'argument *index* est **end**, alors les nouveaux éléments sont insérés à la fin de la liste. Cette commande renvoie une chaîne vide.

cheminBoîte **itemcget** *index option*
Renvoie la valeur courante de l'option de configuration de l'élément d'indice *index*, spécifiée par l'argument *option*. Celui-ci peut prendre n'importe laquelle des valeurs admises par la commande **listbox itemconfigure**.

cheminBoîte **itemconfigure** *index ?option? ?valeur? ?option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration de l'élément d'indice *index*. Les règles concernant l'argument *option* sont les mêmes que pour la commande **configure**. Les options suivantes sont actuellement supportées pour les éléments (l'argument *couleur* ci-dessous peut prendre une quelconque des formes reconnues pour la désignation des couleurs) :

-background *couleur*
L'argument *couleur* spécifie la couleur de fond à utiliser pour l'affichage de l'élément.

-foreground *couleur*
L'argument *couleur* spécifie la couleur de premier plan à utiliser pour l'affichage de l'élément.

-selectbackground *couleur*
L'argument *couleur* spécifie la couleur de fond à utiliser pour l'affichage de l'élément lorsque celui-ci est sélectionné.

-selectforeground *couleur*
L'argument *couleur* spécifie la couleur de premier plan à utiliser pour l'affichage de l'élément lorsque celui-ci est sélectionné.

cheminBoîte **nearest** *y*
Cette commande renvoie l'indice de l'élément visible le plus proche de l'ordonnée *y* dans la boîte de liste.

cheminBoîte **scan** *option args*
Cette commande est utilisée pour parcourir le contenu des boîtes de listes. Elle peut prendre deux formes selon la valeur de l'argument *option* :

cheminBoîte **scan mark** *x y*
Enregistre les valeurs *x* et *y* et la vue courante de la boîte de liste, en vue d'une utilisation ultérieure avec une commande **scan dragto**. Cette commande est typiquement associée à une pression du bouton de la souris dans la liste. Elle renvoie une chaîne vide.

cheminBoîte **scan dragto** *x y*
Cette commande calcule la différence entre les valeurs des arguments *x* et *y* et celles de ces mêmes arguments lors de la dernière commande **scan mark** exécutée pour cette boîte de liste. Elle renvoie une chaîne vide. La vue du contenu du champ est alors ajustée de 10 fois la valeur de cette différence vers la gauche ou vers la droite. Cette commande est typiquement utilisée pour simuler l'effet de déplacement rapide du contenu de la liste avec la souris. On associe fréquemment les deux commandes **scan mark** et **scan dragto** au second bouton de la souris, comme ceci :

```
bind $c <2> "$c scan mark %x %y"
bind $c <B2-Motion> "$c scan dragto %x %y"
```

cheminBoîte **see** *index*
Ajuste la vue de la boîte de liste de telle sorte que l'élément d'indice *index* devienne visible. Si l'élément est déjà visible, la commande est sans effet. S'il est proche de l'un des bords de la boîte, l'élément est rendu visible sur ce bord, et centré dans la boîte dans le cas contraire.

cheminBoîte **selection** *option arg*
Cette commande est utilisée pour ajuster la sélection dans une boîte de liste. Elle peut prendre plusieurs formes en fonction de l'argument *option* :

cheminBoîte **selection anchor** *index*
Fixe le point d'ancrage de la sélection à l'élément d'indice *index*. Si *index*

fait référence à un élément inexistant, l'élément le plus proche est utilisé. Le point d'ancrage de la sélection est la fin de la sélection déterminée en faisant glisser la souris avec le premier bouton enfoncé. Le mot-clé *anchor* peut être utilisé pour désigner ce point d'ancrage.

cheminBoîte **selection clear** *premier?dernier?*
Si certains éléments entre les indices *premier* et *dernier* inclus sont sélectionnés, cette commande les désélectionne. L'état de la sélection n'est pas modifié pour les éléments situés hors de cet intervalle.

cheminBoîte **selection includes** *index*
Renvoie 1 si l'élément d'indice *index* est sélectionné, 0 sinon.

cheminBoîte **selection set** *premier?dernier?*
Sélectionne tous les éléments dont l'indice est compris entre *premier* et *dernier* inclus sans modifier l'état sélectionné ou désélectionné des éléments situés hors de cet intervalle.

cheminBoîte **size**
Renvoie une chaîne décimale indiquant le nombre total des éléments dans la boîte de liste.

cheminBoîte **xview** *args*
Cette commande est utilisée pour obtenir ou modifier la position horizontale du texte dans la fenêtre. Elle peut prendre une des formes suivantes :

cheminBoîte **xview**
Renvoie une liste de deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : ces valeurs indiquent les proportions de l'élément qui sont visibles ou non dans la fenêtre. Par exemple, si les valeurs renvoyées sont 0,2 et 0,6 cela signifie que 20% du texte sont situés à gauche de la partie visible et 40% à droite, tandis que la partie visible correspond à 40% du texte. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de la commande **xscrollcommand**.

cheminBoîte **xview** *index*
Ajuste la vue dans la fenêtre du champ de telle sorte que l'élément d'indice *index* apparaisse au bord gauche de cette fenêtre.

cheminBoîte **xview moveto***fraction*
Ajuste la vue dans la fenêtre de la liste de telle sorte que la portion de la boîte de liste désignée en pourcentage de la largeur totale par la fraction *fraction* soit masquée par le bord gauche de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1].

cheminBoîte **xview scroll** *nb unité*
Cette commande décale la vue dans la fenêtre de la liste vers la gauche ou vers la droite selon la valeur donnée aux arguments *nb* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage est d'environ *nb* caractères. Avec *pages*, le décalage est de *nb* contenus d'écran. Si *nb* est négatif, les caractères au-delà du bord gauche deviennent visibles ;

s'il est positif, les caractères situés au-delà du bord droit deviennent visibles. Les positions des caractères sont définies au moyen de la largeur du caractère 0.

cheminBoîte **yview** *?args?*
Cette commande est utilisée pour obtenir ou modifier la position verticale du texte dans la fenêtre. Elle peut prendre une des formes suivantes :

cheminBoîte **yview**
Renvoie une liste de deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : le premier donne la position de l'élément au sommet de la fenêtre par rapport à la boîte toute entière (0.5 désigne le milieu de la liste par exemple) ; le second indique la position qui suit immédiatement le dernier élément de la fenêtre relativement à la boîte entière. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de la commande **yscrollcommand**.

cheminBoîte **yview** *index*
Ajuste la vue dans la fenêtre du champ de telle sorte que l'élément d'indice *index* apparaisse au sommet de cette fenêtre.

cheminBoîte **yview moveto** *fraction*
Ajuste la vue dans la fenêtre de la liste de telle sorte que l'élément de la boîte de liste désigné en pourcentage de la hauteur totale par la fraction *fraction* se place au sommet de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1].

cheminBoîte **yview scroll** *nb unité*
Cette commande décale la vue dans la fenêtre de la liste vers le haut ou vers le bas selon la valeur donnée aux arguments *nb* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage est de *nb* lignes. Avec *pages*, le décalage est de *nb* contenus d'écran. Si *nb* est négatif, les éléments précédents deviennent visibles ; s'il est positif, les éléments suivants deviennent visibles.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les boîtes de listes avec les caractéristiques ci-dessous. L'essentiel du comportement d'une boîte de liste est déterminé par l'option **-selectMode** qui spécifie une des quatre manières de traiter une sélection :

– si le mode de sélection est *single* ou *browse*, on peut sélectionner au plus un élément à la fois dans la boîte de liste. Cliquer sur un élément avec le premier bouton de la souris le sélectionne et désélectionne tout autre élément qui pourrait être déjà sélectionné. Avec le mode *browse*, on peut passer d'un élément à l'autre en faisant glisser la souris avec le premier bouton enfoncé tandis qu'en mode *single* on ne peut passer d'un élément à l'autre qu'en cliquant dessus.

– si le mode de sélection est *multiple* ou *extended*, on peut sélectionner n'importe quel nombre d'éléments, pas nécessairement contigus. En mode *multiple*, un clic sur un élément fait alterner son état sélectionné ou désélectionné sans affecter les autres éléments. En mode *extended*, un clic du premier bouton de la souris sélectionne l'élément cliqué, désélectionne tous les autres et fixe le point d'ancrage de la sélection à cet élément ; en faisant glisser la souris avec le bouton enfoncé, on sélectionne tous les éléments depuis celui qui a été cliqué jusqu'à celui qui se trouve sous le pointeur de la souris.

Chaque fois que la sélection change dans la boîte de liste, un événement virtuel <<ListboxSelect>> est généré. On peut ainsi faire une liaison à cet élément pour être averti des modifications de sélection dans la boîte de liste.

Les liaisons par défaut suivantes sont définies :

1. En mode *extended*, l'intervalle sélectionné peut être ajusté en pressant le premier bouton de la souris avec la touche Majuscule enfoncée : la sélection est alors faite des éléments compris entre le point d'ancrage et celui qui se trouve sous le pointeur de la souris. L'extrêmité libre de cette nouvelle sélection peut être déplacée avec le bouton enfoncé.
2. En mode *extended*, presser le premier bouton de la souris avec la touche Contrôle enfoncée a un effet de commutateur : le point d'ancrage devient l'élément situé sous le pointeur de la souris et l'état de la sélection est inversé. L'état de la sélection des autres éléments n'est pas modifié. Si la souris est déplacée avec le premier bouton enfoncé, l'état de la sélection de tous les éléments entre le point d'ancrage et celui qui se trouve sous le pointeur de la souris est ajusté sur celui du point d'ancrage ; celui des autres éléments reste ce qu'il était avant l'opération de commutation.
3. Si la souris sort de la boîte de liste avec le premier bouton enfoncé, la fenêtre défile, rendant visible la partie qui était cachée du côté de la souris. Le défilement se poursuit jusqu'à ce que la souris rentre à nouveau dans la fenêtre ou que le bouton soit relâché ou encore que l'on ait atteint la fin de la liste.
4. Le second bouton de la souris sert à parcourir la liste. S'il est enfoncé et glissé au-dessus de la boîte de liste, le contenu défile rapidement dans la direction de déplacement de la souris.
5. Si les touches Haut ou Bas sont enfoncées, la position du curseur se déplace en conséquence d'un élément dans la liste. Si le mode de sélection est *browse* ou *extended* alors le nouvel élément actif est aussi sélectionné tandis que tous les autres sont désélectionnés. En mode *extended*, le nouvel élément actif devient le point d'ancrage de la sélection.
6. En mode *extended*, les combinaisons de touches Majuscule-Haut et Majuscule-Bas déplacent en conséquence le curseur d'un élément dans la liste et simultanément étendent la sélection à ce nouvel élément.
7. Les touches Gauche et Droite font défiler le contenu de la boîte de liste vers la gauche ou vers la droite de la largeur du caractère 0. Les combinaisons de touches Contrôle-Gauche et Contrôle-Droite font défiler le contenu de la boîte de liste vers la gauche ou vers la droite de la largeur de la fenêtre. Les

combinaisons de touches Contrôle-Précédent et Contrôle-Suivant ont le même effet.

8. Les touches Précédent et Suivant font défiler le contenu de la boîte de liste vers le haut ou vers le bas d'une page, c'est-à-dire de la hauteur de la fenêtre.
9. Les touches Début (Home) et Fin font défiler la boîte de liste horizontalement vers les bords gauche et droit respectivement.
10. La touche Contrôle-Début déplace le curseur au premier élément de la liste, le sélectionne et désélectionne tout autre élément de la liste.
11. La touche Contrôle-Début déplace le curseur au dernier élément de la liste, le sélectionne et désélectionne tout autre élément de la liste.
12. En mode *extended*, la combinaison de touches Contrôle-Majuscule-Début étend la sélection au premier élément de la boîte de liste et Contrôle-Majuscule-Fin au dernier élément.
13. En mode *multiple*, la combinaison de touches Contrôle-Majuscule-Début déplace le curseur au premier élément de la boîte de liste et Contrôle-Majuscule-Fin au dernier élément.
14. Les touches Espacement et Sélection ont pour effet de sélectionner l'élément placé à l'endroit du curseur comme si le premier bouton de la souris avait été pressé sur cet élément.
15. En mode *extended*, Contrôle-Majuscule-Espacement et Majuscule-Sélection étendent la sélection à l'élément actif comme si le premier bouton de la souris avait été pressé sur cet élément avec la touche Majuscule enfoncée.
16. En mode *extended*, la touche Échappement (*Escape*) annule la sélection la plus récente et rétablit la sélection précédente.
17. La combinaison de la touche Contrôle et de la barre oblique (/) sélectionne tous les éléments, sauf en modes *single* et *browse*, où seul l'élément actif est sélectionné tandis que tous les autres sont désélectionnés.
18. La combinaison de la touche Contrôle et de la contre-oblique (\) désélectionne tous les éléments, sauf en mode *browse* où elle est sans effet.
19. La touche F16 (appelée Copie sur de nombreuses stations de travail) ou Méta-w copie la sélection éventuelle de la liste.

Les comportements décrits sont les comportements par défaut et peuvent être modifiés en définissant de nouvelles liaisons pour des boîtes de listes particulières ou en redéfinissant les liaisons de classe.

Options standard

Les options standard supportées par les composants de type *listbox* sont rassemblées dans le tableau 8 à la page 117. Ces options sont décrites en détail à la page 142.

Options spécifiques

-height **height** **Height**

Spécifie la hauteur désirée pour la fenêtre, mesurée en nombre de lignes. Si la valeur est négative ou nulle, la hauteur désirée pour la fenêtre est calculée de façon à être juste assez grande pour contenir tous les éléments de la boîte de liste.

-listvariable **listVariable** **Variable**

Spécifie le nom d'une variable. La valeur de la variable est une liste à afficher à l'intérieur du composant ; si la valeur de la variable change, le composant se met automatiquement à jour pour refléter la nouvelle valeur. Si l'on essaie d'assigner une valeur qui n'est pas une liste valide à l'option **-listvariable** une erreur est générée. En revanche, une tentative de supprimer par une commande **unset** une variable associée à une option **-listvariable** échouera mais ne provoquera pas la levée d'une erreur.

-selectmode **selectMode** **SelectMode**

Spécifie un des styles de manipulation des sélections. La valeur de cette option peut être arbitraire mais les liaisons par défaut attendent une des valeurs suivantes : *single*, *browse*, *multiple* ou *extended*. La valeur par défaut est *browse*.

-setgrid **setGrid** **SetGrid**

Spécifie une valeur booléenne qui détermine si la boîte contrôle la grille de redimensionnement pour sa fenêtre de premier niveau. Cette option est typiquement utilisée pour des composants (tels que textes et boîtes de listes), où l'information possède une taille « naturelle » (la taille d'un caractère) et où il semble logique que les dimensions de la fenêtre soient un multiple entier de cette taille. Ces tailles naturelles de fenêtres forment une grille. Si l'option **setGrid** est fixée à la valeur vraie (*true*) alors le composant communiquera avec le gestionnaire de fenêtres afin que, lorsque l'utilisateur redimensionnera interactivement la fenêtre du composant, les dimensions de celle-ci s'afficheront en unités de grille et la taille sera astreinte à rester un multiple entier de ces unités (cf. la discussion de la commande **wm**, p 251 et suivantes).

-state **state** **State**

Spécifie l'un des deux états *normal* ou *disabled* pour la boîte de liste. Si la boîte de liste est désactivée, aucun élément ne peut être inséré ou détruit, les éléments sont dessinés dans la couleur de l'option **disabledForeground**, et une sélection ne peut pas être modifiée et n'est pas affichée (bien que l'information correspondante soit retenue).

-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-exportselection	exportSelection	ExportSelection
-font	font	Font
-foreground ou -fg	foreground	Foreground
-height	height	Height
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-relief	relief	Relief
-selectbackground	selectBackground	Foreground
-selectborderwidth	selectBorderWidth	BorderWidth
-selectforeground	selectForeground	Background
-state	state	State
-takefocus	takeFocus	TakeFocus
-width	width	Width
-xscrollcommand	xScrollCommand	ScrollCommand
-yscrollcommand	yScrollCommand	ScrollCommand

TAB. 8 – *Options standard reconnues par les composants de type* listbox.

-width **width** **Width**

Spécifie la largeur désirée pour la fenêtre, mesurée en nombre de caractères. Si la fonte n'a pas une largeur fixe, la largeur du caractère 0 est utilisée pour convertir les nombres de caractères en unités d'écran. Si la valeur est négative ou nulle, la largeur de la fenêtre est calculée de façon à être juste assez large pour contenir tous les éléments de la boîte de liste.

loadTk

Charge Tk dans un interpréteur sécurisé.

Syntaxe

::safe::loadTk *esclave* ?**-use** *IdFenêtre*? ?**-display** *nomMoniteur*?

Le système Tk sécurisé, dit *Safe Tk*, est basé sur le système *Safe Tcl* qui procure un mécanisme permettant un accès restreint et médiatisé à l'auto-chargement et aux modules afin de sécuriser les interpréteurs. *Safe Tk* ajoute la possibilité de configurer l'interpréteur pour que les opérations de Tk soient elles-mêmes sécurisées et que celui-ci soit chargé dans un interpréteur Tcl sûr.

Description

La commande **::safe::loadTk** n'est pas une commande Tk de base : il s'agit d'une procédure faisant partie de l'extension *Safe Tk* (définie dans le fichier safetk.tcl inclus dans la bibliothèque standard de Tk). Elle initialise les structures requises dans l'interpréteur sécurisé désigné et charge alors Tk dans cet interpréteur. La commande renvoie le nom de l'interpréteur sécurisé. Si l'option **-use** est spécifiée, la fenêtre désignée par l'identificateur *IdFenêtre* est utilisée pour contenir la racine « . » de l'interpréteur sécurisé; il peut s'agir de n'importe quel identificateur valide (variable selon les plates-formes) et pouvant même faire référence à une fenêtre appartenant à une autre application. Par commodité, si la fenêtre qui sera utilisée est l'une des fenêtres Tk de l'application, on peut se servir de son nom de fenêtre (comme par exemple .x.y) plutôt que de son identificateur.

Lorsque l'option **-use** n'est pas spécifiée, une nouvelle fenêtre de premier niveau est créée pour servir de fenêtre « . » à l'interpréteur sécurisé. Sur le système X11, si l'on veut que la fenêtre utilise un autre moniteur que le moniteur par défaut, on peut le spécifier avec l'option **-display**.

Problèmes de sécurité

L'essentiel des problèmes de sécurité est décrit à la section consacrée à l'extension Tcl *Safe Base* dans la Référence de Tcl. La commande **::safe::loadTk** ajoute la valeur de la variable globale **tk_library** prise dans l'interpréteur maître au chemin d'accès virtuel de l'interpréteur sécurisé afin que l'auto-chargement fonctionne dans l'interpréteur sécurisé.

L'initialisation de Tk est alors sécurisée en ce qu'elle ne se fie plus à l'état de l'interpréteur esclave pour le démarrage. La commande **::safe::loadTk** enregistre le nom de l'esclave pour que, lorsque l'initialisation (c'est-à-dire la fonction **Tk_SafeInit**) est appelée puis appelle à son tour la commande **::safe::InitTk** du maître, elle renvoie un tableau **argv** équivalent aux options souhaitées.

Si l'option **-use** n'est pas utilisée, la nouvelle fenêtre de premier niveau est décorée afin que l'utilisateur soit toujours conscient que l'interface présentée pro-

vient d'un code qui peut ne pas être sûr et puisse aisément détruire l'interpréteur correspondant.

Sous X11, des options **-use** et **-display** conflictuelles peuvent générer une erreur fatale dans le système X.

lower

Modifie l'ordre d'empilement des fenêtres.

Syntaxe

lower *fenêtreA ?fenêtreB?*

Description

Si l'argument *fenêtreB* est omis, la commande fait passer la fenêtre *fenêtreA* en dessous de toutes les fenêtres collatérales dans l'ordre d'empilement.

Si l'argument *fenêtreB* est spécifié, il doit indiquer le nom de chemin d'une fenêtre qui est soit une fenêtre collatérale (descendante du même parent) de la fenêtre *fenêtreA*, soit un descendant d'une fenêtre collatérale. Dans ce cas, la commande **lower** insère la fenêtre *fenêtreA* immédiatement en dessous de celle désignée par *fenêtreB* ou en-dessous du parent de *fenêtreB* si celui-ci est un collatéral de *fenêtreA* (ce qui revient alors à faire remonter la fenêtre dans l'ordre d'empilement !).

menu

Permet de créer et de manipuler des menus.

Syntaxe

menu *cheminMenu ?options?*

Introduction

La commande **menu** crée au premier niveau une nouvelle fenêtre (donnée par l'argument *cheminMenu*) et en fait un composant de type *menu*. Des options complémentaires peuvent être spécifiées aussi bien dans la ligne de commandes que dans la base de données de ressources pour configurer certains aspects du menu tels que la couleur ou la police utilisées. La commande **menu** renvoie son argument *cheminMenu*. Au moment de l'invocation de cette commande, il ne doit pas exister de fenêtre nommée *cheminMenu*, mais le composant parent de *cheminMenu* doit exister effectivement.

Un menu est un composant graphique qui affiche des entrées d'une ligne pouvant être disposées sur une ou plusieurs colonnes. Il existe différents types d'entrées, chacun possédant des propriétés particulières, qui peuvent tous être combinés dans un même menu. Il ne faut pas confondre les entrées de menu avec les composants de type *entry* : c'est le menu dans son ensemble qui est un widget, pas les entrées individuelles. Chaque entrée peut être affichée avec au plus trois champs distincts :

- le champ principal est l'étiquette de l'article de menu qui peut prendre la forme d'une chaîne de texte, d'un bitmap ou d'une image, contrôlés respectivement par les options **-label** , **-bitmap** et **-image**
- si l'option **-accelerator** est spécifiée pour une entrée, un second champ est ajouté à droite de l'étiquette. Il sert en général à indiquer un raccourci, c'est-à-dire une séquence de touches du clavier qui déclenchent la même action que l'article de menu lui-même.
- le troisième champ est un *indicateur*, présent uniquement pour des articles de type *case à cocher* ou *bouton radio*. Il indique si ces derniers sont sélectionnés ou non : il apparaît à gauche de l'étiquette.

Les articles de menu peuvent être désactivés: dans ce cas, les étiquettes et les raccourcis claviers sont grisés: les liaisons de menu empêchent un article désactivé d'être invoqué. Chaque fois qu'une entrée active de menu est modifiée, un événement virtuel <<MenuSelect>> est envoyé au menu. L'article de menu peut alors être sollicité depuis le menu et une action peut être déclenchée en conséquence comme par exemple définir un message d'aide en fonction du contexte pour cette entrée.

Divers types d'entrées

Type commande C'est le type d'entrée le plus courant : lorsqu'une entrée de ce type est invoquée, une commande Tcl associée est exécutée. La commande Tcl est

spécifiée grâce à l'option **-command**.

Type séparateur Il s'agit d'une entrée qui prend la forme d'une ligne horizontale et joue le rôle de séparateur. Une telle entrée ne peut être ni activée, ni invoquée. Aucune action ne lui est associée.

Type case à cocher Une case à cocher dans un menu se comporte comme les composants graphiques de même nom. Lorsqu'elle est invoquée, elle bascule alternativement entre deux états : sélectionné et désélectionné. Si l'entrée est sélectionnée, une certaine valeur se trouve stockée dans une certaine variable : l'une et l'autre sont spécifiées au moyen des options **-onvalue** et **-variable**. Si l'entrée est désélectionnée, une autre valeur, spécifiée par l'option **-offvalue**, est stockée dans la même variable.

Un indicateur est affiché à gauche de l'étiquette de la case à cocher : si l'entrée est sélectionnée, le centre de l'indicateur prend la couleur spécifiée par l'option **-selectcolor**, sinon il prend la couleur de fond définie pour ce menu.

Si une option **-command** est spécifiée pour une case à cocher, la valeur de cette option est évaluée comme une commande Tcl chaque fois que l'entrée est invoquée : cela se produit immédiatement après le basculement à l'état sélectionné.

Type bouton radio Un bouton radio dans un menu se comporte comme les composants graphiques de même nom. Les boutons-radios sont organisés en groupes : dans chaque groupe, un seul bouton radio peut être sélectionné à la fois. Si une entrée particulière est sélectionnée, une certaine valeur est stockée dans une certaine variable : l'une et l'autre sont spécifiées au moyen des options **-onvalue** et **-variable**. Tout autre bouton du même groupe antérieurement sélectionné est désélectionné.

Lorsqu'une entrée se trouve ainsi sélectionnée, toute modification de la variable associée conduit l'entrée à se désélectionner elle-même. Le regroupement des boutons est déterminé par la variable qui leur est associée : deux boutons ayant la même variable associée appartiennent à un même groupe. Un indicateur en forme de losange est placé à gauche de l'étiquette de chaque bouton radio : si l'entrée est sélectionnée, le centre de l'indicateur prend la couleur spécifiée par l'option **-selectcolor**, sinon il prend la couleur de fond définie pour ce menu.

Si une option **-command** est spécifiée pour une entrée de type bouton radio, la valeur de cette option est évaluée comme une commande Tcl chaque fois que l'entrée est invoquée : cela se produit au moment où l'entrée est sélectionnée.

Type cascade Les entrées de type cascade permettent de construire des sous-menus. On leur associe un sous-menu au moyen de l'option **-menu**. La commande de composant **postcascade** sert à installer et désinstaller le menu associé immédiatement à côté de l'entrée de type cascade. Le sous-menu associé doit obligatoirement être un descendant du menu contenant l'entrée de type cascade. Il sera affiché grâce à une commande Tcl **post** de la forme :

```
cheminMenu post x y
```

où *cheminMenu* est le chemin définissant le menu associé et x et y sont les coordonnées du coin supérieur droit de l'entrée cascade dans la fenêtre de base contenant le menu. Sous Unix, le sous-menu sera refermé grâce à une commande Tcl **unpost** de la forme :

```
cheminMenu unpost
```

Sur les autres plates-formes, le code natif prend en charge l'effacement du sous-menu.

Si une option **-command** est spécifiée pour une entrée de type cascade, elle est évaluée comme une commande Tcl dès que l'entrée est invoquée. Cette option n'est pas supportée sous Windows.

Type détachable Une entrée détachable est une ligne pointillée qui apparaît en haut d'un menu et indique que celui-ci peut être détaché. Cette fonctionnalité est activée au moyen de l'option **-tearOff**. À la différence des autres entrées de menu, cette ligne ne peut être créée au moyen de l'opération **add** ou détruite au moyen de l'opération **delete**.

Barre de menu

Tout menu peut être installé comme une barre de menus d'une fenêtre de premier niveau (cf. p. 242). Sur Macintosh, si cette fenêtre se trouve au premier plan, les menus et sous-menus sont en fait installés dans la barre de menu au sommet du moniteur principal ; sous Windows ou Unix, il s'agit d'une barre de menu au sommet de la fenêtre elle-même.

Menus particuliers

Certains menus de la barre de menus sont traités de façon particulière. Sur Macintosh, on peut accéder aux menus Pomme et Aide. Sous Windows, on peut accéder au menu Système de chaque fenêtre. Avec le système X Windows sous Unix, un menu d'aide spécial aligné à droite est fourni. Dans tous ces exemples, les menus doivent être créés en concaténant le nom du menu de type barre de menus avec le nom spécial. Ainsi pour une barre de menus appelée .menubar, sur Macintosh, les menus spéciaux seraient .menubar.apple et .menubar.help ; sous Windows, le menu spécial serait .menubar.system ; sur X Windows, le menu d'aide serait .menubar.help.

Lorsque Tk rencontre un menu Pomme sur Macintosh, il installe le contenu de ce menu comme premiers articles du véritable menu Pomme dès que la fenêtre de l'application à laquelle il est associé vient au premier plan. Les articles définis par Tk sont suivis d'un séparateur puis du reste du menu Pomme.

Lorsque Tk rencontre un menu Aide sur Macintosh, il installe le contenu de ce menu à la fin du véritable menu Aide de MacOS dès que la fenêtre de l'application à laquelle il est associé vient au premier plan. Il faut noter, aussi bien pour le menu Pomme que pour le menu Aide, que seul le texte des articles de menu est affiché :

les autres attributs (images, polices, couleurs etc.) ne le sont pas. Si le menu est détachable, le mot *TearOff* est affiché au lieu d'une ligne pointillée.

Lorsque Tk rencontre un menu Système sous Windows, il installe le contenu de ce menu à la fin du véritable menu Système de la barre de menus à laquelle il est attaché. Ce menu peut être invoqué au moyen de la souris ou en tapant la combinaison `Alt+Spacebar`. En raison de certaines limitations de Windows, les attributs de police, couleur, images, menus détachables etc. ne peuvent être affichés dans le menu Système.

Lorsque Tk rencontre un menu Aide sous X Windows, le menu est installé à droite de la barre de menus et est justifié à droite.

Clones

Lorsqu'un menu est installé comme barre de menus pour la fenêtre de premier niveau ou que le menu est détaché, un clone de ce menu est fabriqué. Ce clone est un composant de type menu à part entière, mais il est descendant du menu original. Tout changement dans la configuration du menu original se reflète dans le clone. En outre, les éventuels sous-menus en cascade sont eux aussi clonés dès que l'on pointe dessus. Tous les clones sont détruits dès que le menu original est détruit ou lorsque la barre de menus ou le menu détaché disparaissent.

Opérations sur les menus

Avec l'instruction :
menu *cheminMenu ?options?*
la commande **menu** crée une nouvelle commande Tcl dont le nom est précisément *cheminMenu*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le menu. La forme générale est :
cheminMenu opération ?arg arg ...?

De nombreuses options prennent comme argument un indicateur spécifiant quel est l'article du menu sur lequel on souhaite opérer. Ces indicateurs sont des indices et peuvent prendre une des valeurs ou des constantes symboliques suivantes :

- un *nombre* spécifie une entrée numériquement : 0 correspond à la première entrée du menu, 1 à la suivante et ainsi de suite.
- *active* désigne l'entrée active courante. Si aucune entrée n'est active alors cette forme est équivalente à *none*.
- *end* désigne l'entrée située au bas du menu. Si le menu ne comporte aucune entrée, cette forme est équivalente à *none*.
- *last* Synonyme de *end*.
- *none* Indique « aucune entrée ». Ceci est utilisé couramment avec l'option *active* pour désactiver toutes les entrées du menu. Dans la plupart des cas, la spécification *none* annihile l'effet d'une commande.

– *@nombre*
Dans cette forme, l'argument *nombre* est traité comme une coordonnée verticale dans la fenêtre du menu : l'entrée la plus proche de cette coordonnée est utilisée.
– *Motif*
Si l'indice ne correspond à aucune des formes vues précédemment, c'est cette forme qui est utilisée. Les entrées du menu sont comparées au motif les unes après les autres en partant du haut et c'est la première correspondance trouvée qui désignera l'entrée à laquelle appliquer la commande. Les règles de comparaison sont celles de la commande **string match**.

Les opérations reconnues par les composantes de type *menu* sont les suivantes :

cheminMenu **activate** *index*
Fait passer l'entrée spécifiée par l'indice *index* à l'état *active* et la réaffiche en utilisant ses couleurs actives. Toute entrée précédemment active est désactivée. Si l'indice est spécifié comme *none* ou si l'entrée désignée est désactivée, le menu se retrouve sans aucune entrée ayant l'état *active*. Cette commande renvoie une chaîne vide.

cheminMenu **add** *type ?option valeur option valeur...?*
Ajoute une nouvelle entrée au bas du menu. Le type de la nouvelle entrée est donné par l'argument *type* et doit être l'un des termes suivants (ou une abréviation unique de ces termes) : *cascade*, *checkbutton*, *command*, *radiobutton* ou *separator*. La commande **add** renvoie une chaîne vide. On peut par ailleurs préciser un certain nombre d'options pour fixer certaines valeurs :

-activebackground *valeur*
Spécifie une couleur de fond à utiliser pour afficher cette entrée lorsqu'elle est active. Si cette option prend pour valeur une chaîne vide (valeur par défaut), alors c'est l'option **activeBackground** du menu entier qui est utilisée. Si la variable **tk_strictMotif** a été fixée, cette option est ignorée et l'option **-background** est utilisée à sa place. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-activeforeground *valeur*
Spécifie une couleur de premier plan à utiliser pour afficher cette entrée lorsqu'elle est active. Si cette option prend pour valeur une chaîne vide (valeur par défaut), alors c'est l'option **activeForeground** du menu entier qui est utilisée. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-accelerator *valeur*
Spécifie une séquence de caractères à afficher à droite de l'entrée de menu. Il s'agit usuellement d'un raccourci clavier qui peut être tapé afin de déclencher la même fonction que l'entrée de menu elle-même. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-background *valeur*
Spécifie une couleur de fond à utiliser pour afficher cette entrée lorsqu'elle est dans son état normal (ni active, ni désactivée). Si cette option prend pour valeur une chaîne vide (valeur par défaut), alors c'est l'option **-background** du menu entier qui est utilisée. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-bitmap *valeur*
Spécifie une image *bitmap* à afficher pour cette entrée de menu à la place de l'étiquette textuelle. Les formes acceptées sont les mêmes que pour tous les bitmaps. Cette option supplante l'option **-label** mais doit être ensuite spécifiée comme une chaîne vide afin de permettre à nouveau d'afficher une étiquette textuelle. Si une option **-image** a été spécifiée, elle supplante l'option **-bitmap**. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-columnbreak *valeur*
Si cette option vaut 1, l'entrée apparaît en haut d'une nouvelle colonne. Si elle vaut 0, l'entrée est placée sous l'entrée précédente.

-command *valeur*
Spécifie la commande Tcl à exécuter lorsqu'une entrée de menu est invoquée. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-font *valeur*
Spécifie la police à utiliser pour écrire l'étiquette et le raccourci clavier de l'entrée de menu. Si cette option prend pour valeur une chaîne vide (valeur par défaut), alors c'est l'option **-font** du menu entier qui est utilisée. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-foreground *valeur*
Spécifie une couleur de premier plan à utiliser pour afficher cette entrée lorsqu'elle est dans son état normal (ni active, ni désactivée). Si cette option prend pour valeur une chaîne vide (valeur par défaut), alors c'est l'option **-foreground** du menu entier qui est utilisée. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-hidemargin *valeur*
Spécifie si les marges standard doivent être dessinées pour cette entrée de menu. Cela sert essentiellement à créer des palettes d'images. La valeur 1 indique que les marges sont cachées et 0 que la marge est utilisée.

-image *valeur*
Spécifie une image à afficher dans le menu à la place de l'étiquette textuelle ou d'une icone *bitmap*. L'image doit avoir été créée antérieurement au moyen d'une commande **image create**. Cette option supplante les options **-label** et **-bitmap** mais doit être ensuite explicitement spécifiée comme une chaîne vide afin de permettre à nouveau d'afficher une étiquette textuelle ou une icone bitmap. Cette option n'est pas dispo-

nible pour les séparateurs et les lignes de menu détachable.

-indicatoron *valeur*
Disponible uniquement pour les entrées de type case à cocher et bouton radio. L'argument *valeur* est une variable booléenne qui détermine si l'indicateur doit être affiché ou non.

-label *valeur*
Spécifie la chaîne représentant le texte de l'entrée de menu. Non disponible pour les séparateurs et les lignes de menu détachable.

-menu *valeur*
Disponible uniquement pour les entrées de type cascade. Spécifie le nom de chemin du sous-menu associé à cette entrée. Le sous-menu doit être un descendant du menu lui-même.

-offvalue *valeur*
Disponible uniquement pour les entrées de type case à cocher. Spécifie la valeur à stocker dans la variable qui est associée à cette entrée lorsque celle-ci est désélectionnée.

-onvalue *valeur*
Disponible uniquement pour les entrées de type case à cocher. Spécifie la valeur à stocker dans la variable qui est associée à cette entrée lorsque celle-ci est sélectionnée.

-selectcolor *valeur*
Disponible uniquement pour les entrées de type case à cocher et bouton radio. Spécifie la couleur à afficher dans l'indicateur lorsque l'entrée est sélectionnée. Si cette option prend pour valeur une chaîne vide (valeur par défaut), alors c'est la valeur de l'option **selectColor** du menu entier qui est utilisée.

-selectimage *valeur*
Disponible uniquement pour les entrées de type case à cocher et bouton radio. Spécifie une image à afficher dans l'entrée de menu (à la place de l'option **-image**) lorsque celle-ci est sélectionnée. L'argument *Valeur* est le nom d'une image qui doit avoir été créée antérieurement au moyen d'une commande **image create**. Cette option est ignorée si jamais l'option **-image** n'a pas été également spécifiée.

-state *valeur*
Spécifie l'un des trois états suivants pour l'entrée : *normal*, *active*, ou *disabled*. Dans son état normal, l'entrée est affichée en utilisant les options **foreground** et **background** (voir ci-dessus). L'état actif est utilisé typiquement lorsque le pointeur de la souris est au-dessus de l'entrée : ce sont alors les options **activeForeground** et **activeBackground** qui sont utilisées. L'état désactivé sert à interdire l'utilisation de l'entrée : les couleurs utilisées dans ce cas correspondent aux options **disabledForeground** pour l'entrée et **background** pour le menu

-underline *valeur*
Spécifie l'indice d'un caractère qui apparaîtra souligné dans le texte de

l'entrée. Cette option est aussi utilisée par les liaisons par défaut. La valeur 0 correspond au premier caractère. Si une icone *bitmap* ou une image sont utilisées pour cette entrée alors l'option est ignorée. Cette option n'est pas disponible pour les séparateurs et les lignes de menu détachable.

-value *valeur*
Disponible uniquement pour les entrées de type bouton radio. Spécifie la valeur à stocker dans la variable associée à cette entrée lorsque celle-ci est sélectionnée. Si une chaîne vide est spécifiée, l'option **-label** est utilisée comme valeur à stocker dans la variable.

-variable *valeur*
Disponible uniquement pour les entrées de type case à cocher et bouton radio. Spécifie le nom d'une variable globale à fixer lorsque l'entrée est sélectionnée. Pour les cases à cocher, la variable est aussi utilisée lorsque l'élément est désélectionné. Pour les boutons-radios, une modification de cette variable conduit l'élément courant à se déselectionner lui-même.

cheminMenu **cget** *option*
Renvoie la valeur courante de l'option de configuration désignée par l'argument *option*. Cet argument peut prendre une quelconque des valeurs admises par la commande **menu**.

cheminMenu **clone** *nouveauCheminMenu ?typeClone?*
Fabrique un clone, sous le nom *nouveauCheminMenu*, du menu courant. Ce clone est un menu à part entière mais tout changement se répercute immédiatement sur le menu original et réciproquement. L'argument *cloneType* peut prendre les valeurs **normal**, **menubar** ou **tearoff**.

cheminMenu **configure** *?option? ?valeur option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration du menu. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminMenu*.

Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option en question. Si un ou plusieurs couples *option-valeur* sont spécifiés, ces options se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. Cet argument peut prendre uniquement les valeurs admises par la commande **menu**.

cheminMenu **delete** *index1 ?index2?*
Détruit toutes les entrées de menu entre les indices *index1* et *index2* inclus. Si l'argument *index2* est omis, il est identique à *index1*. Les tentatives de détruire une entrée de type *détachable* sont ignorées : pour ce faire, il faut modifier l'option **tearOff**.

cheminMenu **entrycget** *index option*
Renvoie la valeur courante de l'option de configuration désignée par l'argument *option* pour l'entrée désignée par l'argument *index*. L'argument *option* peut prendre une quelconque des valeurs admises par la commande **add**.

cheminMenu **entryconfigure** *index ?options?*
Cette commande est analogue à la commande **configure**, si ce n'est qu'elle s'applique aux options d'une entrée particulière alors que **configure** concerne les options du menu dans son entier. L'argument *options* peut prendre une quelconque des valeurs admises par la commande **add**. Si cet argument est omis, la commande renvoie une liste décrivant toutes les options courantes de l'entrée d'indice *index*.

cheminMenu **index** *index*
Renvoie l'indice numérique correspondant à l'argument *index* ou au mot *none* si cet argument avait été spécifié comme tel.

cheminMenu **insert** *index type ?option valeur option valeur ...?*
Cette commande insère une nouvelle entrée de menu immédiatement avant l'entrée désignée par l'indice *index*. Les options *type*, *option* et *valeur* ont la même signification que pour la commande **add**. Il n'est évidemment pas possible d'insérer d'entrée avant une entrée de type *détachable*.

cheminMenu **invoke** *index*
Invoque l'action correspondant à l'entrée de menu. Si l'entrée est désactivée, rien ne se produit. Si une commande est associée à l'entrée, la valeur de retour de cette commande sert de valeur de retour à l'opération **invoke**. Sinon, le résultat est une chaîne vide. On notera que l'invocation d'une entrée de menu ne referme pas automatiquement le menu. Ce sont les liaisons par défaut qui s'en chargent normalement avant l'exécution de **invoke**.

cheminMenu **post** *x y*
Cette commande fait en sorte d'afficher le menu à l'écran au point de la fenêtre de base de coordonnées x et y. Ces coordonnées sont ajustées si nécessaire pour garantir que le menu soit entièrement visible à l'écran. Cette commande renvoie normalement une chaîne vide. Si l'option **postCommand** a été spécifiée, sa valeur est exécutée comme un script Tcl avant d'afficher le menu et le résultat de ce script sert de valeur de retour pour l'opération **post**. Si une erreur se produit, elle est renvoyée et le menu n'est pas affiché.

cheminMenu **postcascade** *index*
Affiche le sous-menu associé à l'entrée de type *cascade* désignée par l'indice *index* et referme tout sous-menu précédemment affiché. Si l'indice *index* ne correspond pas à une entrée de type *cascade* ou si le menu *cheminMenu* n'est pas lui-même affiché, cette commande aura pour seul effet de refermer un sous-menu ouvert.

cheminMenu **type** *index*
Renvoie le type de l'entrée de menu spécifiée par l'argument *index*. C'est l'argument *type* passé lors de l'opération **add** qui a créé cette entrée. Par exemple, *command*, *separator* ou *tearoff*.

cheminMenu **unpost**
Cette opération met fin à la fenêtre qui n'est donc plus affichée, et renvoie une chaîne vide. Si un sous-menu était ouvert, il sera lui aussi effacé. L'opération n'a de sens que sous Unix : sous Windows et MacOS, le système gère lui-même

l'affichage des menus et sous-menus.

cheminMenu **yposition** *index*
Renvoie une valeur décimale en pixels indiquant la coordonnée verticale où se place, dans la fenêtre du menu, le point supérieur de l'entrée spécifiée par l'argument *index*.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les menus, avec les caractéristiques suivantes :

1. lorsque la souris passe sur un menu, l'entrée qui se trouve sous le pointeur est activée ; les entrées réagissent en fonction des déplacements de la souris ;
2. lorsque la souris sort d'un menu, toutes les entrées sont rendues inactives sauf si le pointeur entre alors dans l'un des sous-menus ;
3. lorsqu'un bouton de la souris est relevé au-dessus d'un menu, l'entrée active (s'il y en a une) est invoquée. Le menu s'efface s'il n'est pas détaché ;
4. les touches d'espacement et de retour-chariot invoquent l'entrée active et effacent le menu ;
5. si certaines des lettres d'une des entrées du menu sont soulignées grâce à l'option **-underline**, le fait de presser l'une de ces lettres (indifféremment en minuscules ou en majuscules) invoque l'entrée de menu et referme le menu ;
6. la touche d'échappement (*escape*) interrompt une sélection de menu en cours en n'invoquant aucune entrée. Elle referme le menu à moins qu'il ne s'agisse d'un menu détaché ;
7. les touches flèches vers le haut et vers le bas permettent de se déplacer parmi les entrées d'un menu. Lorsqu'une extrêmité du menu est atteinte, on passe à l'autre extrêmité ;
8. la touche flèche vers la gauche permet de se déplacer au menu situé à gauche du menu actuel. Si le menu courant est un sous-menu, celui-ci est refermé et l'entrée courante est celle de type cascade qui correspond à ce sous-menu. Si le menu courant résulte d'un bouton de menu, ce menu de bouton est refermé et c'est le prochain bouton de menu vers la gauche qui est affiché. L'ordre de gauche à droite des boutons de menus est, par convention, l'ordre de leur empilement. Tk considère que le premier bouton créé est celui de gauche ;
9. la touche flèche vers la droite permet de se déplacer au menu situé à droite du menu actuel. Si l'entrée courante est de type cascade, le sous-menu correspondant est ouvert et c'est la première entrée de ce sous-menu qui devient active. Si le menu courant résulte d'un bouton de menu, ce menu de bouton est refermé et c'est le prochain bouton de menu vers la droite qui est affiché.

Les comportements décrits sont les comportements par défaut et peuvent être modifiés en définissant de nouvelles liaisons pour des composants particuliers ou en redéfinissant les liaisons de classe.

-activebackground	activeBackground	Foreground
-activeforeground	activeForeground	Background
-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-font	font	Font
-foreground ou -fg	foreground	Foreground
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus

TAB. 9 – *Options standard reconnues par les composants de type* menu.

Options standard

Les options standard supportées par les composants de type *menu* sont rassemblées dans le tableau 9. Ces options sont décrites en détail à la page 142.

Options spécifiques

-activeborderwidth **activeBorderWidth** **BorderWidth**

Spécifie une valeur non négative indiquant la largeur de la bordure 3-D dessinée autour des éléments actifs du menu. La valeur est exprimée en unités d'écran.

-postcommand **postCommand** **Command**

Si cette option est spécifiée, elle fournit une commande Tcl à exécuter chaque fois que le menu est affiché. La commande est invoquée par la commande **post** *avant* d'afficher le menu.

-selectcolor **selectColor** **Background**

Pour des articles de menu qui sont des cases à cocher ou des boutons-radios, cette option spécifie la couleur à afficher dans l'indicateur lorsque l'article correspondant est sélectionné.

-tearoff **tearOff** **TearOff**

Cette option doit prendre une valeur booléenne valide qui spécifie si le menu doit contenir à son sommet une ligne pointillée pour le rendre détachable. Si c'est le cas, elle sera considérée comme l'article d'indice 0 dans le menu et les autres articles seront numérotés à partir de 1. Les liaisons par défaut font en sorte que le menu soit effectivement détaché lorsque cette ligne pointillée est invoquée.

-tearoffcommand **tearOffCommand** **TearOffCommand**

Si cette option a une valeur non vide, elle spécifie une commande Tcl à invoquer lorsque le menu est détaché. La véritable commande sera constituée en fait de la valeur de cette option, suivie par une espace, suivie par le nom de la fenêtre du menu, suivi par une autre espace puis par le nom de fenêtre du menu détaché. Par exemple, si la valeur de l'option est *abc* et si le menu *.x.y* est détaché et crée un nouveau menu appelé *.x.detache1*, la commande *abc .x.y .x.detache1* sera invoquée.

-title **title** **Title**

La chaîne définie par cette option servira de titre à la fenêtre créée lorsqu'un menu sera détaché. Si le titre est NULL, le fenêtre aura pour titre le titre du bouton de menu ou de l'article de sous-menu à partir desquels ce menu aura été invoqué.

-type **type** **Type**

Cette option peut prendre une des valeurs **menubar**, **tearoff** ou *normal* et est fixée lorsque le menu est créé. Cette option n'est utilisée en principe que par la bibliothèque standard de Tk.

menubutton

Permet de créer et de manipuler des boutons menus.

Syntaxe

menubutton *cheminBouton ?options?*

Introduction

La commande **menubutton** crée au premier niveau une nouvelle fenêtre (donnée par l'argument *cheminBouton*) et en fait un composant de type *bouton menu*. Des options complémentaires peuvent être spécifiées aussi bien dans la ligne de commandes que dans la base de données de ressources pour configurer certains aspects du bouton menu tels que les couleurs, la police, le texte et le relief initial utilisés. La commande **menubutton** renvoie son argument *cheminBouton*. Au moment de l'invocation de cette commande, il ne doit pas exister de fenêtre nommée *cheminBouton*, mais le composant parent de *cheminBouton* doit exister effectivement.

Un bouton menu est un composant graphique qui affiche un texte, une icone bitmap ou une image et qui est associé à un menu. Si un texte est affiché, il est entièrement composé dans une même police ; le texte peut occuper plusieurs lignes à l'écran, soit par utilisation de sauts de lignes, soit parce qu'un paramètre **-wrapLength** a été fixé pour forcer les coupures de ligne. Il est enfin possible de souligner certains caractères du texte avec l'option **-underline**, le plus souvent pour en faire par la suite des raccourcis clavier.

Dans le fonctionnement normal, lorsqu'on presse le premier bouton de la souris sur un bouton menu, le menu associé est affiché immédiatement en dessous du bouton graphique. Si la souris est, déplacée sur le contenu du menu et si le bouton est relâché au-dessus d'une entrée, celle-ci est invoquée. Le menu est effacé dès que l'on relâche le bouton de la souris.

Les boutons menus sont habituellement organisés en groupes qui constituent des barres de menus : si le bouton de la souris est pressé sur un bouton menu (ce qui affiche le menu associé) puis le pointeur déplacé sur un autre bouton menu de la même barre, le premier menu est effacé et le menu associé à ce deuxième bouton est affiché.

Opérations sur les boutons menus

Avec l'instruction :

menubutton *cheminBoutonMenu ?options?*

la commande **menubutton** crée une nouvelle commande Tcl dont le nom est précisément *cheminBoutonMenu*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le menu. La forme générale est :

cheminBoutonMenu opération ?arg arg ...?

Les opérations reconnues par les composantes de type *menubutton* sont les suivantes :

cheminBoutonMenu **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **menubutton**.

cheminBoutonMenu **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration du bouton de menu. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminBoutonMenu*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **menubutton**.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les boutons menus avec les caractéristiques suivantes :

1. un bouton menu est activé dès que le pointeur de la souris passe au-dessus de lui et désactivé dès qu'il le quitte ;
2. lorsque le premier bouton de la souris est pressé au-dessus d'un bouton menu, le relief change et le menu associé est affiché. Si le pointeur est ensuite déplacé sur une des entrées du menu et le bouton relâché, cette entrée est invoquée ;
3. lorsque le premier bouton de la souris est pressé au-dessus d'un bouton menu et relâché au-dessus de ce bouton, le menu associé reste affiché. On peut alors aller cliquer sur une des entrées pour l'invoquer. Le menu se referme alors ;
4. si le premier bouton de la souris est pressé sur un bouton menu et le pointeur déplacé sur un autre bouton menu, le premier menu est effacé et le menu associé au deuxième bouton menu est affiché.
5. si le premier bouton de la souris est pressé sur un bouton menu puis relâché en dehors de tout autre bouton menu, le menu associé se referme et aucune action n'est déclenchée ;
6. si l'option **-underline** a été spécifiée pour un bouton menu, le menu peut être affiché en tapant au clavier la combinaison Alt+x, où x désigne le caractère qui est souligné dans le texte du bouton menu ;
7. la touche de clavier F10 permet d'afficher le premier bouton menu de la fenêtre ;
8. si un bouton menu est focalisé par l'application, la barre d'espacement et le retour-chariot permettent de l'activer.

-activebackground	activeBackground	Foreground
-activeforeground	activeForeground	Background
-anchor	anchor	Anchor
-background ou -bg	background	Background
-bitmap	bitmap	Bitmap
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-image	image	Image
-justify	justify	Justify
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus
-text	text	Text
-textvariable	textVariable	Variable
-underline	underline	Underline
-wraplength	wrapLength	WrapLength

TAB. 10 – *Options standard reconnues par les composants de type* menubutton.

Si l'état d'un bouton menu est *disabled*, aucune des actions mentionnées ne se produit. Les comportements décrits sont les comportements par défaut et peuvent toujours être modifiés en définissant de nouvelles liaisons pour des composants particuliers ou en redéfinissant les liaisons de classe.

Options standard

Les options standard supportées par les composants de type *menubutton* sont rassemblées dans le tableau 10. Ces options sont décrites en détail à la page 142.

Options spécifiques

-direction **direction** **Direction**

Spécifie l'endroit où le menu doit apparaître. La valeur d'option *above* essaie de faire surgir le menu au-dessus du bouton menu et la valeur *below* en dessous. De même, si l'option prend les valeurs *left* ou *right*, le menu essaiera d'apparaître respectivement à gauche ou à droite du bouton menu. Enfin la valeur *flush* fait apparaître le menu directement au-dessus du bouton.

-height **height** **Height**

Spécifie une hauteur pour le bouton menu. Si une image ou un bitmap est affiché dans le bouton menu, la valeur est mesurée en unités d'écran ; pour du texte elle est mesurée en lignes de texte. Si cette option n'est pas spécifiée, la hauteur pour le bouton menu est calculée à partir de la taille de l'image, du bitmap ou du texte qu'il contient.

-indicatoron **indicatorOn** **IndicatorOn**

Cette option doit prendre une valeur booléenne valide. Si cette valeur est *true*, un petit indicateur rectangulaire sera affiché à la droite du bouton de menu et les liaisons par défaut le traiteront comme un bouton de menu d'options. Si la valeur est *false*, aucun indicateur ne sera affiché.

-menu **menu** **MenuName**

Spécifie le nom de chemin du menu associé à ce bouton menu. Le menu doit être un enfant du bouton menu.

-state **state** **State**

Spécifie l'un des trois états *normal*, *active* ou *disabled* pour le bouton menu. Dans l'état normal, le bouton menu est affiché en utilisant les options **foreground** et **background**. Dans l'état activé (typiquement utilisé lorsque le curseur de la souris se trouve au-dessus du bouton menu), le bouton menu est affiché en utilisant les options **activeForeground** et **activeBackground**. Dans l'état désactivé, les options **disabledForeground** et **background** déterminent la façon dont le bouton menu est affiché.

-width **width** **Width**

Spécifie une largeur pour le bouton menu. Si une image ou un bitmap est affiché dans le bouton menu, la valeur est mesurée en unités d'écran ; pour du texte, elle est mesurée en nombre de caractères. Si cette option n'est pas spécifiée, la largeur du bouton menu est calculée à partir de la taille de l'image, du bitmap ou du texte qu'il contient.

message

Permet de créer et de manipuler des messages graphiques.

Syntaxe

message *cheminMessage ?options?*

Description

La commande **message** crée une nouvelle fenêtre (indiquée par l'argument *cheminMessage*) et en fait un composant graphique de type *message*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques du message telles que la couleur, la police, le texte, le relief initial etc. La commande **message** renvoie la valeur de son argument *cheminMessage*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminMessage*, mais en revanche l'objet parent de *cheminMessage* doit exister effectivement.

Un message est un composant graphique qui affiche une petite chaîne de texte. Il possède trois caractéristiques principales :

- il coupe les lignes de façon à proportionner le texte dans la fenêtre. Les coupures sont faites autant que possible entre les mots. On peut forcer une coupure de ligne en utilisant le caractère de saut de ligne ;
- le texte peut être justifié à gauche, centré ou justifié à droite ;
- les fenêtres de messages gèrent les caractères de contrôle et les caractères non imprimables de façon particulière. Les caractères de tabulation et de saut de ligne ont l'effet habituel. Les autres caractères dont le code ASCII est inférieur à 0x20 (32 en décimal) sont affichés sous la forme `\x`*hh* où *hh* est le nombre hexadécimal correspondant au caractère. Il en va de même des caractères absents de la fonte utilisée.

On notera que les tabulations sont à éviter dans un texte centré ou justifié à droite.

Opérations sur les messages

Avec l'instruction :

message *cheminMessage ?options?*

la commande **message** crée une nouvelle commande Tcl dont le nom est précisément *cheminMessage*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le message. La forme générale est :

cheminMessage opération ?arg arg ...?

Les opérations reconnues par les composantes de type *message* sont les suivantes :

cheminMessage **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **message**.

cheminMessage **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration du message. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminMessage*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **message**.

Liaisons par défaut

Lorsqu'une nouvelle fenêtre de message est créée, elle ne possède aucune liaison par défaut. Les messages n'ont pas pour fonction d'être interactifs.

Options standard

Les options standard supportées par les composants de type *message* sont rassemblées dans le tableau 12 à la page 169. Ces options sont décrites en détail à la page 142.

Options spécifiques

-aspect **aspect** **Aspect**

Spécifie une valeur entière non négative indiquant le rapport d'aspect souhaité pour le texte. Le rapport d'aspect est défini comme 100*largeur/hauteur. Une valeur de 100 signifie que le texte doit être aussi large que haut, 200 qu'il doit être deux fois plus large que haut, 50 qu'il doit être deux fois plus haut que large, et ainsi de suite. Cette option sert à choisir la longueur de ligne pour le texte si l'option **width** n'est pas spécifiée. La valeur par défaut est 150.

-justify **justify** **Justify**

Spécifie comment justifier les lignes de texte. L'option peut prendre une des valeurs **left**, **center** ou **right**. La valeur par défaut est **left**. Cette option fonctionne de concert avec les options **anchor**, **aspect**, **padX**, **padY** et **width** procurant ainsi une grande variétés d'arrangements du texte à l'intérieur de la fenêtre. Les options **aspect** et **width** déterminent la quantité d'espace nécessaire à l'écran pour afficher le texte. Les options **anchor**, **padX** et **padY** déterminent l'endroit où cette région

-anchor	anchor	Anchor
-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus
-text	text	Text
-textvariable	textVariable	Variable
-width	width	Width

TAB. 11 – *Options standard reconnues par les message*

rectangulaire est affichée à l'intérieur de la fenêtre du message, et l'option **justify** détermine comment chaque ligne est affichée dans la région.

Par exemple, si l'option *anchor* vaut *e* et *justify* prend la valeur *left* et si la fenêtre du message est plus large que nécessaire pour le texte, alors le texte sera affiché de telle sorte que les extrêmités gauches de toutes les lignes soient alignées et l'extrêmité droite de la plus longue ligne soit à la distance **padX** du bord droit de la fenêtre ; le bloc entier de texte sera centré dans l'étendue verticale de la fenêtre.

-width **width** **Width**

Spécifie la longueur des lignes dans la fenêtre. Si cette option a une valeur positive, l'option **aspect** est ignorée et l'option **width** détermine la longueur de ligne. Si cette option a une valeur négative ou nulle, c'est l'option **aspect** qui détermine la longueur de ligne.

option

Permet de manipuler les options de fenêtre dans la base de données d'options.

Syntaxe

option add *motif valeur ?priorité?*
option clear
option get *fenêtre nom classe*
option readfile *nomFichier ?priorité?*

Description

La commande **option** permet d'ajouter des entrées à la base de données d'options de Tk ou de récupérer certaines options de cette base de données. L'argument *motif* représente l'option spécifiée et consiste en une série de noms et de classes séparés par des astérisques ou des points dans le format X usuel. L'argument *valeur* contient une chaîne de texte à associer à *motif*; c'est la valeur qui sera renvoyée par l'invocation de la commande **option** dans un script Tcl ou, dans du code C, par des appels à la fonction **Tk_GetOption**. Si l'argument *priorité* est spécifié, il indique le niveau de priorité pour cette option (voir ci-dessous les valeurs autorisées); la valeur par défaut est *interactive*. Cette commande renvoie toujours une chaîne vide.

La commande **option clear** efface la base de données d'options. Les options par défaut seront rechargées automatiquement la prochaine fois qu'une option est ajoutée à la base ou en est retirée. Cette commande renvoie toujours une chaîne vide.

La commande **option get** renvoie la valeur de l'option de *fenêtre* spécifiée par les arguments *nom* et *classe*. Si plusieurs entrées dans la base de données correspondent aux arguments *fenêtre*, *nom* et *classe*, la commande renvoie celle qui a été créée avec le plus haut niveau de priorité. Si plusieurs entrées correspondent avec le même niveau de priorité, la commande renvoie celle qui a été déclarée le plus récemment dans la base de données, même si elle est moins spécifique qu'une autre plus ancienne. S'il n'y a aucune correspondance, une chaîne vide est renvoyée.

La sous-commande **readfile** de la commande **option** lit le fichier désigné par l'argument *nomFichier*, qui devrait adopter le format standard pour un fichier texte de base de données sous système X (comme par exemple dans les fichiers .Xdefaults) et ajoute à la base de données toutes les options spécifiées dans ce fichier. Si l'argument *priorité* est précisé, il indique le niveau de priorité pour ces options; la valeur par défaut est *interactive*.

Les arguments *priorité* de la commande **option** peuvent prendre une des valeurs symboliques suivantes:

widgetDefault
: Niveau 20. Utilisé pour les valeurs par défaut définies dans les composants graphiques.

startupFile
: Niveau 40. Utilisé pour les options spécifiées dans des fichiers de démarrage spécifiques à une application.

userDefault
: Niveau 60. Utilisé pour les options spécifiées dans des fichiers de valeurs par défaut appartenant à l'utilisateur comme par exemple .Xdefaults chargé par le serveur X, ou des fichiers de démarrage spécifiques à l'utilisateur.

interactive
: Niveau 80. Utilisé pour les options spécifiées interactivement après le démarrage de l'application. Si *priorité* n'est pas spécifié, ce niveau est la valeur par défaut.

Chacun de ces mots-clés peut être abrégé. En outre, les priorités peuvent aussi être spécifiées numériquement en utilisant des nombres entiers entre 0 et 100.

Options standard

Les options standard supportées par les composants graphiques.

Description

Cette section décrit les options standard supportées par les composants graphiques par opposition aux options spécifiques à certains d'entre eux. Chaque composant graphique ne supporte pas nécessairement chacune des options standard mentionnées car elles peuvent ne pas avoir de signification pour lui, mais s'il la supporte, elle aura exactement l'effet décrit ici. Les sections consacrées à chaque composant graphique se contentent de mentionner les options standard supportées par le composant : il faudra se reporter à la présente section pour en trouver la signification. Au contraire, les options spécifiques sont décrites dans le cadre de chaque composant.

Dans les descriptions qui suivent, sont indiquées systématiquement les trois formes que prennent les options en fonction du contexte dans lequel on les utilise. Chaque option possède en effet trois dénominations selon la façon dont on s'y réfère : il existe un nom de commande, un nom de ressource et un nom de classe. Une convention de dénomination permet de s'y retrouver facilement. Le nom de commande commence par un tiret suivi du nom en lettres minuscules ; le nom de ressource commence par une lettre minuscule et peut éventuellement contenir ensuite des majuscules ; le nom de classe commence obligatoirement par une majuscule. Par exemple, l'option qui gère la largeur de la bordure d'un composant graphique a pour nom de commande **-borderwidth**, pour nom de ressource **borderWidth** et pour nom de classe **BorderWidth**. Ces trois aspects correspondent aux situations suivantes :

- le *nom de commande* est la forme que prend l'option lorsqu'on l'utilise sur une ligne d'instructions, autrement dit comme option d'une commande particulière et en particulier avec une sous-commande telle que **configure**. Par exemple, la commande suivante attribue une couleur de premier plan noire à l'objet graphique nommé .a.b.c :

  ```
  .a.b.c  configure  -foreground black
  ```

  Utilisées ainsi, les options portent parfois le nom de commutateurs (*switches*).
- le *nom de ressource* est le nom utilisé lorsqu'on se réfère à cette option dans la base de données de ressources (cf. **??** p. **??**).
- le *nom de classe* est le nom utilisé lorsqu'on se réfère à cette option en tant que classe générale d'options dans la base de données de ressources (cf. **??** p. **??**).

Les options suivantes sont considérées comme standard :

-activebackground **activeBackground** **Foreground**

Spécifie la couleur de fond à utiliser pour dessiner les éléments actifs. Un élément (un composant ou une portion de composant) est actif si le pointeur de la souris est positionné sur cet élément et si un clic sur l'un des boutons de la souris provoque une certaine action. Si la variable **tk_strictMotif** a été fixée, cette option sera normalement ignorée; la couleur de fond normale sera utilisée à la place. Pour certains éléments sur les systèmes Windows et Macintosh, la couleur active est utilisée seulement lorsque le premier bouton de la souris est enfoncé au-dessus de cet élément.

-activeforeground **activeForeground** **Background**

Spécifie la couleur de premier plan à utiliser pour dessiner les éléments actifs.

-anchor **anchor** **Anchor**

Spécifie la façon dont le contenu d'un composant (par exemple texte ou bitmap) doit être affiché dans le composant. Cette option doit prendre une des valeurs suivantes: **n**, **ne**, **e**, **se**, **s**, **sw**, **w**, **nw** ou **center**. Par exemple, **nw** signifie que le coin supérieur gauche de ce qui doit être affiché doit coïncider avec le coin supérieur gauche du composant.

-background ou -bg **background** **Background**

Spécifie la couleur de fond normale à utiliser pour afficher le composant.

-bitmap **bitmap** **Bitmap**

Spécifie une image bichrome (*bitmap* dans la terminologie Tk) à afficher dans le composant, sous une quelconque des formes acceptées pour la désignation des bitmaps. La manière exacte dont le bitmap est affiché peut être affectée par d'autres options telles que **anchor** ou **justify**. Typiquement, si cette option est spécifiée, elle supplante toute autre option spécifiant une valeur textuelle à afficher dans le composant; l'option **bitmap** devra recevoir une valeur de chaîne vide si l'on veut réactiver l'affichage de texte. Dans les composants qui supportent à la fois les options **bitmap** et **image**, **image** l'emporte habituellement sur **bitmap**.

-borderwidth ou -bd **borderWidth** **BorderWidth**

Spécifie une valeur non négative indiquant la largeur de la bordure 3-D à dessiner à l'extérieur du composant (si une telle bordure a été requise au moyen d'une option **relief**). La valeur peut aussi être utilisée pour produire des effets 3-D à l'intérieur du composant. La valeur est exprimée en unités d'écran.

-cursor **cursor** **Cursor**

Spécifie le pointeur de souris à utiliser pour le composant. La valeur doit prendre une quelconque des formes acceptées pour la désignation des curseurs.

-disabledforeground disabledForeground DisabledForeground

Spécifie la couleur de premier plan à utiliser pour dessiner un élément désactivé. Si cette option est spécifiée comme une chaîne vide (ce qui est typiquement le cas sur des écrans monochromes), les éléments désactivés sont dessinés avec la couleur normale de premier plan mais ils sont grisés au moyen d'un motif pointillé.

-exportselection exportSelection ExportSelection

Spécifie si une sélection dans le composant doit être également une sélection conforme du système X. La valeur doit prendre une quelconque des formes acceptées pour les valeurs booléennes, comme **true**, **false**, **0**, **1**, **yes** ou **no**. Si la sélection est exportée, une sélection opérée dans le composant désélectionne la sélection X courante, une sélection opérée hors du composant désélectionne toute sélection du composant, et le composant répondra aux requêtes pour récupérer la sélection lorsqu'il en possède une. Le comportement par défaut pour les composants est habituellement d'exporter les sélections.

-font font Font

Spécifie la police à utiliser pour dessiner le texte à l'intérieur du composant. La valeur doit prendre une quelconque des formes acceptées pour la désignation d'une police.

-foreground ou -fg foreground Foreground

Spécifie la couleur de premier plan normale à utiliser pour afficher le composant.

-highlightbackground highlightBackground HighlightBackground

Spécifie la couleur à afficher pour une région sélectionnée lorsque le composant ne détient pas la focalisation.

-highlightcolor highlightColor HighlightColor

Spécifie la couleur à utiliser pour le rectangle de sélection qui est dessiné autour du composant lorsque celui-ci est focalisé.

-highlightthickness highlightThickness HighlightThickness

Spécifie une valeur non négative indiquant la largeur du rectangle de sélection qui est dessiné autour du composant à l'extérieur lorsque celui-ci est focalisé. La valeur est exprimée en unités d'écran. Si la valeur est zéro, aucune marque de sélection n'est dessinée autour du composant focalisé.

-image image Image

Spécifie une image à afficher dans le composant : cette image aura été créée antérieurement au moyen d'une commande **image create**. Typiquement, si cette

option est spécifiée, elle supplante toute autre option spécifiant un bitmap ou une valeur textuelle à afficher dans le composant ; l'option **image** devra recevoir une valeur de chaîne vide si l'on veut réactiver l'affichage de bitmaps ou de texte. Dans les composants qui supportent à la fois les options **bitmap** et **image**, **image** l'emporte habituellement sur **bitmap**.

-insertbackground **insertBackground** **Foreground**

Spécifie la couleur à utiliser comme arrière-plan dans la zone couverte par le curseur d'insertion. Cette couleur remplace normalement la couleur en vigueur à l'arrière-plan du composant.

-insertborderwidth **insertBorderWidth** **BorderWidth**

Spécifie une valeur entière non négative indiquant la largeur de la bordure 3-D à dessiner autour du curseur d'insertion. La valeur est exprimée en unités d'écran.

-insertofftime **insertOffTime** **OffTime**

Spécifie une valeur entière non négative indiquant le nombre de millisecondes pendant lesquelles le curseur d'insertion reste invisible à chaque cycle de son clignotement. Si cette option est 0, le curseur ne clignote pas : il est visible tout le temps.

-insertontime **insertOnTime** **OnTime**

Spécifie une valeur entière non négative indiquant le nombre de millisecondes pendant lesquelles le curseur d'insertion reste visible à chaque cycle de son clignotement.

-insertwidth **insertWidth** **InsertWidth**

Spécifie une valeur indiquant la largeur totale du curseur d'insertion. La valeur est exprimée en unités d'écran. Si une bordure a été spécifiée pour le curseur d'insertion au moyen de l'option **insertBorderWidth**, la bordure sera dessinée à l'intérieur de la largeur spécifiée par **insertWidth**.

-justify **justify** **Justify**

Quand plusieurs lignes de texte sont affichées dans un composant, cette option détermine comment les lignes s'alignent entre elles. L'option doit prendre une des valeurs symboliques suivantes : *left*, *center* ou *right* qui correspondent respectivement à une justification, un centrage ou une justification à droite des lignes de texte.

-padx **padX** **Pad**

Spécifie une valeur non négative indiquant la quantité d'espace supplémentaire que l'on souhaite insérer dans le composant dans la direction horizontale. La valeur

est exprimée en unités d'écran. Lorsqu'il calcule la taille qui lui est nécessaire, un composant ajoute cette quantité à la largeur qu'il aurait normalement compte-tenu des objets qu'il contient ; si le gestionnaire géométrique peut satisfaire cette requête, le composant possèdera un espace interne supplémentaire à gauche et/ou à droite de son contenu. La plupart des composants utilisent cette option pour le texte et l'ignorent lorsqu'il s'agit d'images.

-pady **padY** **Pad**

Spécifie une valeur non négative indiquant la quantité d'espace supplémentaire que l'on souhaite insérer dans le composant dans la direction des ordonnées. Les mêmes remarques que pour l'option **-padx** s'appliquent ici.

-relief **relief** **Relief**

Spécifie l'effet 3-D (tridimensionnel) souhaité pour le composant. Les valeurs autorisées sont : *raised*, *sunken*, *flat*, *ridge*, *solid* et *groove*. La valeur indique comment l'intérieur du composant devrait apparaître par rapport à l'extérieur. Par exemple, *raised* donnera l'impression d'un intérieur en relief par rapport à l'extérieur du composant.

-repeatdelay **repeatDelay** **RepeatDelay**

Spécifie le nombre de millisecondes pendant lesquelles un bouton de la souris ou une touche du clavier doivent être enfoncés avant de déclencher la répétition de l'action qu'ils représentent. Cela concerne par exemple les flèches aux extrêmités des barres de défilement.

-repeatinterval **repeatInterval** **RepeatInterval**

Utilisée en conjonction avec **repeatDelay** : dès que l'auto-répétition d'une action commence, cette option détermine le nombre de millisecondes séparant chaque répétition.

-selectbackground **selectBackground** **Foreground**

Spécifie la couleur de fond à utiliser pour afficher les éléments sélectionnés.

-selectborderwidth **selectBorderWidth** **BorderWidth**

Spécifie une valeur non négative indiquant la largeur de la bordure 3-D à dessiner autour des éléments sélectionnés. La valeur est exprimée en unités d'écran.

-selectforeground **selectForeground** **Background**

Spécifie la couleur de premier plan à utiliser pour afficher les éléments sélectionnés.

-takefocus **takeFocus** **TakeFocus**

Détermine si la fenêtre accepte la focalisation lorsque des composants sont « traversés », c'est-à-dire parcourus à tour de rôle au moyen des commandes Tab et Majuscule-Tab. Avant d'attribuer la focalisation à une fenêtre, les scripts examinent la valeur de l'option **takeFocus**. Une valeur 0 signifie que la fenêtre doit être ignorée pendant une traversée. Une valeur 1 signifie au contraire que la fenêtre sera focalisée aussi longtemps qu'elle restera visible. Une valeur vide signifie que l'on s'en remet aux scripts pour déterminer s'il faut ou non focaliser la fenêtre : le comportement par défaut consiste à négliger la fenêtre si elle est désactivée, si elle n'a pas de liaisons ou si elle n'est pas visible.

Enfin, pour toute autre valeur, les scripts prennent cette valeur, y ajoutent le nom de la fenêtre séparé par une espace et évaluent la chaîne ainsi formée comme une instruction Tcl. Le script doit renvoyer 0, 1 ou une chaîne vide avec les significations qui viennent d'être dites. Il s'agit d'une option qui ne concerne pas l'implémentation du composant et dont l'interprétation relève entièrement des scripts qui implémentent une traversée des composants.

-text **text** **Text**

Spécifie une chaîne à afficher à l'intérieur du composant. La façon dont la chaîne est affichée dépend du composant et peut être déterminée par d'autres options, telles que **-anchor** ou **-justify**.

-textvariable **textVariable** **Variable**

Spécifie le nom d'une variable. La *valeur* de la variable est une chaîne de texte à afficher à l'intérieur du composant ; si la valeur de la variable change, le composant se met automatiquement à jour par rapport à la nouvelle valeur. La façon dont la chaîne est affichée dépend du composant et peut être déterminée par d'autres options, telles que **-anchor** ou **-justify**.

-underline **underline** **Underline**

Spécifie l'indice entier d'un caractère à souligner dans le composant. Cette option est utilisée dans les liaisons par défaut pour implémenter des parcours à travers les menus ou les boutons menus au moyen de touches du clavier. La valeur 0 correspond au premier caractère du texte affiché dans le composant, 1 au caractère suivant et ainsi de suite.

-wraplength **wrapLength** **WrapLength**

Pour les composants qui peuvent pratiquer des coupures de lignes de texte, cette option spécifie la longueur maximale d'une ligne. Les lignes qui dépassent cette longueur subissent une coupure et se poursuivent sur la ligne suivante. La valeur peut être spécifiée sous une quelconque des formes admises pour les distances d'écran. Si la valeur est inférieure ou égale à zéro, aucune coupure n'est pratiquée :

les lignes ne seront coupées qu'aux endroits où se place explicitement un symbole de saut de ligne.

-xscrollcommand **xScrollCommand** **ScrollCommand**

Spécifie le préfixe pour une commande utilisée pour communiquer avec les barres de défilement horizontales. Lorsque la vue contrôlée par une barre de défilement change, le composant génère une commande Tcl obtenue en concaténant la commande de défilement et deux valeurs numériques: ces valeurs sont des fractions comprises entre 0 et 1 qui indiquent une position dans le document. La valeur 0 indique le début du document et une valeur 0,333 indiquerait la position au tiers du document etc. La première valeur indique la première information du document qui est visible dans la fenêtre et la seconde indique l'information qui se trouve immédiatement après la dernière portion visible. La commande obtenue est alors passée à l'interpréteur Tcl pour être exécutée. Typiquement, l'option **-xScrollCommand** consiste en un nom de chemin d'une barre de défilement suivi de la commande **set** comme par exemple :

```
.x.scrollbar set
```

qui permettra de mettre à jour la barre de défilement lorsque la vue dans la fenêtre changera. Si cette option n'est pas spécifiée, aucune commande n'est exécutée.

-yscrollcommand **yScrollCommand** **ScrollCommand**

Spécifie le préfixe pour une commande utilisée pour communiquer avec les barres de défilement verticales. Cette option est traitée de la même manière que l'option **-xScrollCommand**, si ce n'est qu'elle concerne la direction verticale plutôt qu'horizontale.

pack

Gestionnaire de placement utilisant un modèle de cavités.

Syntaxe

pack *sous-commande arg?arg ...?*

Description

La commande **pack** est utilisée pour interagir avec un gestionnaire de placement particulier dit *packer* qui arrange les descendants d'un widget parent en les agglomérant le long des bords de ce parent. Cette hiérachie des objets utilise la terminologie d'objets maîtres et d'objets assujettis, dits esclaves. La commande **pack** peut prendre plusieurs formes selon l'argument *sous-commande* qui lui est adjoint :

pack *esclave?esclave ...??options?*
Si le premier argument de **pack** est un nom de fenêtre (n'importe quelle valeur commençant par un point), la commande est exécutée de la même manière que **pack configure** : les commandes **pack** et **pack configure** sont synonymes.

pack configure *esclave?esclave ...??options?*
Les arguments sont les noms d'une ou de plusieurs fenêtres esclaves suivis par des paires d'arguments qui spécifient comment gérer les esclaves. On se reportera au paragraphe consacré à l'algorithme de placement pour comprendre comment les options sont utilisées par le gestionnaire de placement. Les options suivantes sont supportées :

-after *autre*
L'argument *autre* doit être le nom d'une autre fenêtre. Le maître de cette autre fenêtre sera utilisé comme maître pour les esclaves, et les esclaves sont insérés immédiatement après *autre* dans l'ordre d'empaquetage.

-anchor *ancre*
L'argument *ancre* doit être une position d'ancre valide comme par exemple **n** ou **sw** ; il spécifie où positionner chaque esclave dans sa parcelle. La valeur par défaut est **center**.

-before *autre*
L'argument *autre* doit être le nom d'une autre fenêtre. Le maître de cette autre fenêtre sera utilisé comme maître pour les esclaves, et les esclaves sont insérés immédiatement avant *autre* dans l'ordre d'empaquetage.

-expand *bool*
Spécifie si les esclaves doivent être étirés pour consommer l'espace excédentaire dans leur maître. L'argument *bool* vaut 0 par défaut.

-fill *style*
Si la parcelle pour un esclave est plus grande que les dimensions requises,

cette option peut être utilisée pour étirer l'objet esclave. L'argument *style* doit prendre une des valeurs suivantes :

none
Donne à l'esclave les dimensions requises plus une marge interne éventuellement spécifiées par des options **-ipadx** ou **-ipady**. C'est la valeur par défaut.

x
Étire l'objet esclave horizontalement pour remplir la largeur complète de sa parcelle (tout en préservant une éventuelle valeur supplémentaire spécifiée par une option **-padx**).

y
Étire l'objet esclave verticalement pour remplir la hauteur complète de sa parcelle (tout en préservant une éventuelle valeur supplémentaire spécifiée par une option **-pady**).

both
Étire l'objet esclave à la fois horizontalement et verticalement

-in *autre*
Insère le ou les esclaves à la fin de l'ordre d'empaquetage de la fenêtre maître désignée par l'argument *autre*.

-ipadx *quantité*
L'argument *quantité* spécifie la quantité de remplissage interne à insérer horizontalement de chaque côté du ou des esclaves. L'argument *quantité* doit être une distance d'écran valide (cf. p. 32). La valeur par défaut est 0.

-ipady *quantité*
L'argument *quantité* spécifie la quantité de remplissage interne à insérer verticalement au-dessus et en dessous du ou des esclaves. La valeur par défaut est 0.

-padx *quantité*
L'argument *quantité* spécifie la quantité de remplissage externe à insérer horizontalement de chaque côté du ou des esclaves. La valeur par défaut est 0.

-pady *quantité*
L'argument *quantité* spécifie la quantité de remplissage externe à insérer verticalement au-dessus et en dessous du ou des esclaves. La valeur par défaut est 0.

-side *côté*
Spécifie sur quel côté du maître le ou les esclaves seront agglomérés. Les valeurs possibles sont : *left*, *right*, *top* (valeur par défaut) ou *bottom*.

Si aucune option **-in**, **-after** ou **-before** n'est spécifiée, chaque esclave sera inséré à la fin de la liste d'empaquetage de son parent à moins qu'il ne soit déjà géré par le gestionnaire de placement. Si une de ces options est spécifiée,

tous les esclaves seront insérés au point spécifié. Si l'un des esclaves est déjà géré par le gestionnaire de placement toute option non spécifiée conservera sa valeur précédente au lieu de prendre les valeurs par défaut.

pack forget *esclave ?esclave ...?*
Retire chaque objet *esclave* de l'ordre d'empaquetage de son maître et supprime sa fenêtre. Ces esclaves ne seront plus contrôlés par le gestionnaire.

pack info *esclave*
Renvoie une liste dont les éléments représentent l'état de la configuration actuelle de l'esclave désigné par l'argument *esclave* sous la forme de paires *option-valeur* identiques à celles de la commande **pack configure**. Les deux premiers éléments de la liste sont « **-in** *maître* » où *maître* désigne le maître de cet esclave.

pack propagate *maître ?bool?*
Si l'argument *bool* a une valeur vraie (comme 1 ou *on*), la propagation est activée pour le maître spécifié par l'argument *maître* qui doit être un nom de fenêtre (voir la section *Propagation géométrique* ci-dessous). C'est la situation par défaut. Si l'argument *bool* a une valeur correspondant à *faux* (comme 0 ou *off*), la propagation est désactivée. Dans les deux cas, une chaîne vide est renvoyée. Si *bool* est omis, la commande renvoie **0** ou **1** selon que la propagation est actuellement activée ou non pour le maître *maître*.

pack slaves *maître*
Renvoie une liste de tous les esclaves dans l'ordre d'empaquetage pour le maître spécifié par l'argument *maître*. Si *maître* n'a pas d'esclaves, une chaîne vide est renvoyée.

L'algorithme de paquetage

Pour chaque maître, le gestionnaire maintient une liste ordonnée de ses esclaves, c'est-à-dire des objets qui lui sont assujettis. Cette liste est appelée *packing list* (liste de paquetage). Les options de configuration **-in**, **-after** et **-before** permettent de spécifier le maître de chaque esclave et la position de l'esclave dans la liste de paquetage. Si aucune de ces options n'est donnée pour un esclave particulier, l'esclave est ajouté à la fin de la liste de paquetage de son parent.

Le gestionnaire arrange les esclaves d'un maître en parcourant la liste de paquetage dans l'ordre. Au moment de traiter chaque esclave, une zone rectangulaire est laissée à l'intérieur du maître. Cette zone est appelée *cavité*; pour le premier esclave, elle correspond à l'aire entière du maître.

Pour chaque esclave, le gestionnaire accomplit les étapes suivantes :

1. Le gestionnaire alloue une *parcelle* rectangulaire pour l'esclave le long d'un bord de la cavité, spécifié par l'option **-side** de l'esclave. S'il s'agit du bord supérieur ou inférieur, la largeur de la parcelle est la largeur de la cavité et la hauteur est la hauteur requise pour l'esclave augmentée des valeurs des options **-ipady** et **-pady**. Pour les bords gauche ou droit, la hauteur de la parcelle est la hauteur de la cavité et la largeur est la largeur requise pour

l'esclave augmentée des valeurs des options **-ipadx** et **-padx**. La parcelle peut être élargie au-delà avec l'option **-expand**.

2. Le gestionnaire choisit les dimensions de l'esclave. La largeur sera normalement la largeur requise pour l'esclave augmentée de deux fois la valeur de l'option **-ipadx** et la hauteur sera la hauteur requise pour l'esclave augmentée de deux fois la valeur de l'option **-ipady** Toutefois, si l'option **-fill** est *x* ou *both*, la largeur de l'esclave est étendue pour remplir la largeur de la parcelle, moins deux fois l'option **-padx**. Si l'option **-fill** est *y* ou *both*, la hauteur de l'esclave est étendue pour remplir la hauteur de la parcelle, moins deux fois l'option **-pady**.
3. Le gestionnaire positionne l'esclave sur sa parcelle. Si l'esclave est plus petit que la parcelle, l'option **-anchor** détermine l'endroit dans la parcelle où l'esclave sera placé. Si **-padx** ou **-pady** sont non nuls, la quantité d'espace additionnel externe fixée sera toujours laissée entre l'esclave et les côtés de la parcelle.

Une fois qu'un esclave donné a été placé, la zone correspondant à sa parcelle est soustraite de la cavité, laissant une cavité rectangulaire plus petite pour l'esclave suivant. Si un esclave n'utilise pas toute sa parcelle, l'espace inutilisé dans la parcelle ne sera pas utilisé par les esclaves suivants. Si la cavité devait devenir trop petite pour staisfaire les besoins d'un esclave, l'esclave recevrait l'espace qui est laissé dans la cavité. Si la cavité rétrécit jusqu'à une taille nulle, tous les esclaves restants sur la liste de paquetage seront retirés de la fenêtre d'écran jusqu'à ce que la fenêtre maître devienne assez large pour les contenir.

Expansion

Si une fenêtre maître est tellement large qu'il reste de l'espace après que tous les esclaves ont été placés, cet espace supplémentaire est distribué uniformément entre tous les esclaves pour lesquels l'option **-expand** a été fixée à la valeur *true*. L'espace excédentaire horizontal est distribué entre les esclaves extensibles dont l'option **-side** prend les valeurs *left* ou *right*, et l'espace excédentaire vertical est distribué entre les esclaves extensibles dont l'option **-side** prend les valeurs *top* ou *bottom*.

Propagation géométrique

Le gestionnaire calcule normalement les dimensions que doit avoir un maître pour satisfaire exactement les besoins de ses esclaves et il fixe la largeur et la hauteur du maître à ces valeurs. Cela conduit l'information géométrique à se propager vers le haut dans la hiérarchie des fenêtres jusqu'à la fenêtre de premier niveau, ce qui fait que le sous-arbre entier se dimensionne en fonction des dimensions des widgets les plus profonds. On peut néanmoins utiliser la commande **pack propagate** pour désactiver la propagation pour un ou plusieurs maîtres. Cela s'avère utile lorsque, par exemple, on souhaite qu'une fenêtre maître ait une taille fixée d'avance.

Restrictions concernant les fenêtres maître

Pour chaque esclave, le maître doit être soit le parent de l'esclave (situation par défaut), soit un widget collatéral (descendant d'un parent de l'esclave). Cette restriction garantit que l'esclave peut être placé sur n'importe quelle partie visible de son maître sans risquer que l'objet esclave ne soit tronqué par son parent.

Ordre d'assemblage

Si le maître d'un esclave n'est pas son parent, il faut s'assurer que l'esclave est bien au-dessus du maître dans l'ordre d'assemblage. Sinon, le maître masquera l'esclave et on aura l'impression que l'esclave n'aura pas été placé correctement. La meilleure façon de s'en assurer est de créer la fenêtre maître en premier : la fenêtre la plus récente est toujours la plus haute dans l'ordre d'empilement. Une autre solution consiste à utiliser les commandes **lower** et **raise** pour modifier l'ordre d'empilement du maître ou de l'esclave.

photo

Permet de créer et de manipuler des images couleur 32-bits.

Syntaxe

image create photo *?nom? ?options?*

Description

Une photo est une image dont les pixels peuvent afficher n'importe quelle couleur ou être transparents. Une photo est stockée en utilisant 32 bits par pixel et affichée en utilisant le procédé de conversion par matrice de pixellisation (*dithering*) si nécessaire. Les données d'image d'une photo peuvent être obtenues soit à partir d'un fichier, soit à partir d'une chaîne. Pour le moment, seul les formats GIF et PPM/PGM sont supportés, mais il existe une interface pour permettre facilement l'addition de formats d'image additionnels. Une photo est transparente dans les régions où aucune donnée d'image n'est fournie.

Création de photos

Comme n'importe quelle image, une photo est créée au moyen de la commande **image create** (cf. p. 103). Les photos supportent les options suivantes :

-data *chaîne*
: Spécifie le contenu de l'image sous forme de chaîne. La chaîne peut contenir des données encodées en base64 ou des données binaires. Le format de la chaîne peut être n'importe quel format pour lequel il existe un gestionnaire (*handler*) de format de fichier d'image acceptant ces données. Si à la fois les options **-data** et **-file** sont spécifiées, c'est l'option **-file** qui l'emporte.

-format *nomFormat*
: Spécifie le nom du format de fichier pour les données spécifiées avec les options **-data** ou **-file**.

-file *nom*
: L'argument *nom* indique le nom du fichier à lire pour récupérer les données de la photo. Le format du fichier peut être n'importe quel format pour lequel il existe un gestionnaire (*handler*) de format de fichier d'image capable de lire ces données.

-gamma *valeur*
: Spécifie un facteur de correction gamma pour les couleurs allouées pour l'affichage de cette image pour un moniteur non linéaire. L'intensité produite par la plupart des moniteurs CRT est une puissance de la valeur entrée et le coefficient gamma en est l'exposant. Sa valeur est typiquement voisine de 2. La valeur spécifiée doit être supérieure à zéro. La valeur par défaut est 1 (pas de correction). En général, des valeurs supérieures à 1 rendront l'image plus

claire et des valeurs inférieures la rendront plus sombre.

-height *nombre*
Spécifie la hauteur de l'image, en pixels. Cette option est utile principalement dans les situations où l'utilisateur souhaite construire le contenu d'une image pièce par pièce. Une valeur 0 (valeur par défaut) permet à l'image de s'étirer ou de se rétracter verticalement pour correspondre aux données qu'elle contient.

-palette *paletteSpec*
Spécifie la résolution du cube de couleurs à allouer pour afficher cette image, déterminant ainsi le nombre de couleurs prises dans la carte de couleurs de la fenêtre où elle est affichée (cf. la section *Allocation de couleurs* p. 160). La valeur de l'argument *paletteSpec* peut être soit un unique nombre décimal spécifiant le nombre de niveaux de gris à utiliser, soit une série de trois nombres séparés par une barre oblique / définissant le cube de couleurs, c'est-à-dire les niveaux de rouge, de vert et de bleu respectivement. Si la première forme est utilisée (un nombre unique), l'image sera monochrome, c'est-à-dire en niveaux de gris.

-width *nombre*
Spécifie la largeur de l'image, en pixels. Cette option est utile principalement dans les situations où l'utilisateur souhaite construire le contenu d'une image pièce par pièce. Une valeur de 0 (valeur par défaut) permet à l'image de s'étirer ou de se rétracter horizontalement pour correspondre aux données qu'elle contient.

Commandes spécifiques des photos

Lorsqu'une photo est créée, Tk crée une nouvelle commande ayant le nom *nomImage* qui aura été donné à la nouvelle image. Cette commande peut être utilisée pour invoquer diverses opérations concernant l'image. La forme générale est :

nomImage opération ?arg arg ...?

Les options qui écrivent des données dans l'image modifient, si nécessaire, les dimensions de l'image pour correspondre aux données écrites à moins qu'une valeur non nulle ait été explicitement spécifiée pour les options **-width** et/ou **-height**.

Les opérations reconnues par les composantes de type *photo* sont les suivantes :

nomImage **blank**
Efface l'image, c'est-à-dire supprime toutes les données de l'image entière, la rendant transparente. Tout ce qui pourrait se trouver en arrière-plan de cette image deviendra visible.

nomImage **cget** *option*
Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **image create photo**.

nomImage **configure** *?option? ?valeur option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration de la photo. Si

l'argument *option* n'est pas spécifié, la commande renvoie la liste de toutes les options disponibles pour *nomImage*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **image create photo**.

nomImage **copy** *sourceImage ?option valeurs ...?*
Copie une région de l'image *sourceImage* (qui doit être de type photo) dans l'image *nomImage* en exécutant éventuellement une opération de zoom et/ou de sous-échantillonnage. Si aucune option n'est spécifiée, cette commande copie l'image *sourceImage* entière dans l'image *nomImage*, en partant de l'origine (0,0) de *nomImage*. Les options suivantes peuvent être spécifiées :

-from *x1 y1 x2 y2*
Spécifie une sous-région rectangulaire de l'image source à copier. Les valeurs (*x1,y1*) et (*x2,y2*) spécifient des coins diagonalement opposés du rectangle. Si *x2* et *y2* ne sont pas spécifiés, la valeur par défaut sera le coin inférieur droit de l'image source. Les pixels copiés incluent les bords gauche et supérieur du rectangle spécifié mais pas les bords droit et inférieur. Si l'option **-from** est omise, la région sera l'image source entière.

-to *x1 y1 x2 y2*
Spécifie une sous-région rectangulaire de l'image de destination affectée par la copie. Les valeurs (*x1,y1*) et (*x2,y2*) spécifient les coins diagonalement opposés du rectangle. Si *x2* et *y2* ne sont pas spécifiés, leurs valeurs par défaut seront (*x1,y1*) augmentées de la taille de la région source (calculée après sous-échantillonnage et zoom éventuels). Si *x2* et *y2* sont spécifiés, la région source sera dupliquée si nécessaire pour remplir la région de destination à la manière d'un pavage.

-shrink
Spécifie que la taille de l'image de destination doit être réduite, si nécessaire, afin que la région copiée coïncide avec le coin inférieur droit de *nomImage*. Cette option n'affectera pas la largeur ou la hauteur de l'image si l'utilisateur a spécifié une valeur non nulle pour les options **-width** ou **-height** respectivement.

-zoom *x y*
Spécifie que la région source doit être magnifiée d'un facteur x dans la direction des abscisses et y dans la direction des ordonnées. Si y est omis, la valeur par défaut est celle de x. Avec cette option, chaque pixel de l'image source sera dilaté en un bloc de $x \times y$ pixels (tous de la même couleur) dans l'image de destination. x et y doivent être supérieurs à 0.

-subsample *x y*
Spécifie que l'image source doit être réduite en taille en utilisant seule-

ment tous les *x*-ièmes pixels dans la direction des abscisses et les *y*-ièmes pixels dans la direction des ordonnées. Des valeurs négatives auront pour effet d'inverser l'image par rapport aux axes des ordonnées ou des abscisses respectivement. Si *y* est omis, la valeur par défaut est celle de l'argument *x*.

nomImage **data** *?option valeurs ...?*

Renvoie les données de l'image sous la forme d'une chaîne. Les options suivantes peuvent être spécifiées:

-background *couleur*

Si la couleur est spécifiée, les données ne contiendront aucune information de transparence: dans tous les pixels transparents, la couleur sera remplacée par la couleur spécifiée.

-format *nomFormat*

Spécifie le nom du gestionnaire (*handler*) de format de fichier d'image à utiliser. Très exactement, la sous-commande **data** recherche le premier gestionnaire dont le nom correspond au début de la chaîne *nomFormat* et qui est capable de lire ces données d'image. Si cet argument est omis, la sous-commande utilise le premier gestionnaire capable de lire ces données d'image.

-from *x1 y1 x2 y2*

Spécifie une région rectangulaire de *nomImage* à renvoyer. Si seuls *x1* et *y1* sont spécifiés, la région s'étend depuis *(x1,y1)* jusqu'au coin inférieur droit de *nomImage*. Si les quatre coordonnées sont spécifiées, elles désignent des coins diagonalement opposés de la région rectangulaire.

Par défaut, si cette option est omise, c'est l'image entière qui est utilisée.

-grayscale

Si cette option est spécifiée, les données ne contiendront pas d'information de couleur. Tous les pixels seront transformés en niveaux de gris.

nomImage **get** *x y*

Renvoie la couleur du pixel de coordonnées (*x*,*y*) dans l'image sous la forme d'une liste de trois entiers compris entre 0 et 255 et représentant les composantes rouge, verte et bleue respectivement.

nomImage **put** *data ?option valeurs ...?*

Fixe les pixels de *imageName* à la valeur spécifiée dans l'argument *data*. Cette commande recherche dans la liste des gestionnaires de format de fichier d'image un gestionnaire capable d'interpréter les données de *data*, puis lit l'image obtenue pour la placer dans *nomImage* (l'image destination). Les options suivantes peuvent être spécifiées:

-format *nomFormat*

Spécifie le format des données de l'image contenues dans l'argument *data*. En particulier, seuls les gestionnaires de format de fichier d'image

dont le nom commence par *nomFormat* seront utilisés lors de la recherche d'un gestionnaire capable de lire ces données.

-from *x1 y1 x2 y2*
Spécifie une sous-région rectangulaire des données *data* à récupérer. Si seuls *x1* et *y1* sont spécifiés, la région s'étend depuis *(x1,y1)* jusqu'au coin inférieur droit de l'image dans le fichier d'image. Si les quatre coordonnées sont spécifiées, elles désignent des coins diagonalement opposés de la région rectangulaire. Par défaut, si cette option est omise, c'est l'image entière représentée par *data* qui est utilisée.

-shrink
Avec cette option, la taille de *nomImage* sera réduite, si nécessaire, afin que la région dans laquelle les données provenant de *data* seront insérées, coïncide avec le coin inférieur droit de *nomImage*. Cette option n'affectera pas la largeur ou la hauteur de l'image si l'utilisateur a spécifié une valeur non nulle pour les options **-width** ou **-height** respectivement.

-to *x y*
Spécifie les coordonnées du coin supérieur gauche de la région de *nomImage* dans laquelle les données de *data* seront placées. La valeur par défaut est (0,0).

nomImage **read** *nomFichier?option valeurs ...?*
Cette commande lit les données d'image contenues dans le fichier *nomFichier* pour les insérer dans l'image *nomImage*. Cette commande recherche dans la liste des gestionnaires de format de fichier d'image un gestionnaire capable d'interpréter les données contenues dans *nomFichier*, puis lit l'image trouvée pour la placer dans *nomImage* (qui est donc l'image de destination). Les options suivantes peuvent être spécifiées:

-format *nomFormat*
Spécifie le format des données de l'image contenues dans l'argument *nomFichier*. En particulier, seuls les gestionnaires de format de fichier d'image dont le nom commence par *nomFormat* seront utilisés lors de la recherche d'un gestionnaire capable de lire ces données.

-from *x1 y1 x2 y2*
Spécifie une sous-région rectangulaire des données du fichier *nomFichier* à récupérer pour les copier dans *nomImage*. Si seuls *x1* et *y1* sont spécifiés, la région s'étend depuis *(x1,y1)* jusqu'au coin inférieur droit de l'image dans le fichier d'image. Si les quatre coordonnées sont spécifiées, elles désignent des coins diagonalement opposés de la région rectangulaire. Par défaut, si cette option est omise, c'est l'image entière du fichier d'image qui est utilisée.

-shrink
Avec cette option, la taille de *nomImage* sera réduite, si nécessaire, afin que la région dans laquelle les données provenant de *nomFichier* seront insérées, coïncide avec le coin inférieur droit de *nomImage*. Cette option

n'affectera pas la largeur ou la hauteur de l'image si l'utilisateur a spécifié une valeur non nulle pour les options **-width** ou **-height** respectivement.

-to *x y*
Spécifie les coordonnées du coin supérieur gauche de la région de *nomImage* dans laquelle les données de *nomFichier* seront placées. La valeur par défaut est (0,0).

nomImage **redither**
L'algorithme de conversion par matrice de pixellisation (*dithering*) utilisé pour l'affichage des photos propage les modifications de quantification d'un pixel à ses voisins. Il en résulte que si les données d'image pour *nomImage* sont fournies par morceaux, l'image pixellisée peut ne pas être tout à fait correcte. Normalement, la différence n'est pas perceptible mais, en cas de problème, cette commande peut être utilisée pour recalculer l'image pixellisée dans chacune des fenêtres où elle est affichée.

nomImage **write** *nomFichier ?option valeurs ...?*
Cette commande écrit des données d'image provenant de *nomImage* dans un fichier désigné par l'argument *nomFichier*. Les options suivantes peuvent être spécifiées :
-format *nomFormat*
-from *x1 y1 x2 y2*
-shrink
-to *x y*
Elles ont exactement la même signification qu'avec la commande **data** vue ci-dessus.

Les formats d'images

Le code de la commande **photo** est structuré de telle sorte qu'il soit possible (moyennant programmation en langage C) de créer des gestionnaires de format de fichier additionnels. Le code maintient une liste de ces gestionnaires. Les nouveaux gestionnaires peuvent être ajoutés à cette liste au moyen de la fonction **Tk_CreatePhotoImageFormat**. La distribution standard de Tk comporte par défaut des gestionnaires pour les formats PPM/PGM et GIF.

Lorsqu'il doit lire ou exécuter une chaîne de données spécifiée au moyen de l'option **-data**, le code appelle tous les gestionnaires à tour de rôle jusqu'à ce qu'il en trouve un qui se déclare capable de traiter ces données. Normalement, le gestionnaire approprié sera trouvé mais, si ce n'est pas le cas, l'utilisateur peut indiquer un nom de format avec l'option **-format** afin de spécifier le gestionnaire à utiliser. En fait, la commande recherche (sans distinction de minuscules et de majuscules) les gestionnaires dont le nom commence par la chaîne spécifiée par l'option **-format** : par exemple, si l'utilisateur spécifie `-format gif`, un gestionnaire appelé GIF87 ou GIF89 pourra être invoqué, mais un gestionnaire appelé JPEG ne le sera pas (à supposer que de tels gestionnaires aient été enregistrés).

Au cours de l'écriture de données d'image dans un fichier, le processus est légèrement différent. La valeur de l'option **-format** doit commencer par le nom

complet du gestionnaire requis et peut être suivie d'informations additionnelles que le gestionnaire pourra utiliser pour spécifier quelle variante des formats qu'il supporte il faut utiliser.

Allocation de couleurs

Lorsqu'une photo est affichée dans une fenêtre, la commande **photo** alloue les couleurs à utiliser pour l'affichage et convertit l'image pixellisée (*dithering*), si nécessaire, afin d'afficher une approximation raisonnable de l'image originale utilisant les couleurs qui sont disponibles. Les couleurs sont allouées sous la forme d'un cube de couleurs, c'est-à-dire que le nombre de couleurs allouées est le produit des nombres de nuances de rouge, vert et bleu respectivement.

Normalement, ce nombre est choisi en fonction de la profondeur de la fenêtre. Par exemple, pour une fenêtre avec des couleurs codées sur 8 bits, le code essayera d'allouer 7 nuances de rouge, 7 de vert et 7 de bleu conduisant à un total de 198 couleurs. Avec un moniteur monochrome (1 bit), deux couleurs seront allouées: le noir et le blanc. En 24 bits (avec des fenêtres en mode visuel DirectColor ou TrueColor), il y a 256 nuances de rouge, de vert et de bleu mais heureusement, étant donné le mode de combinaison des valeurs de pixels avec les fenêtres en mode DirectColor et TrueColor, cela conduit à seulement 256 couleurs. S'il n'est pas possible d'allouer toutes les couleurs, la commande **photo** se charge de réduire le nombre de nuances dans chaque couleur primaire et recommence.

L'utilisateur dispose d'un certain contrôle sur le nombre de couleurs qu'une photo utilise, grâce à l'option **-palette**. Si cette option est utilisée, elle permet de spécifier le nombre maximal de nuances dans chaque couleur primaire qu'il faut essayer d'allouer. Cette option peut aussi servir à forcer l'affichage en niveaux de gris (même sur un moniteur couleur) en spécifiant une valeur unique dans cette option plutôt que trois valeurs séparées par une barre oblique.

place

Gestionnaire de placement pour réaliser des dispositions fixes ou élastiques de composants graphiques.

Syntaxe

place *fenêtre option valeur?option valeur ...?*

Description

Le gestionnaire de placement *place* permet de réaliser des dispositions fixes simples de fenêtres dans lesquelles on spécifie la taille exacte et l'emplacement d'une fenêtre, appelée *esclave*, à l'intérieur d'une autre fenêtre, appelée *maître*. Le gestionnaire de placement permet aussi d'effectuer des dispositions élastiques, dans lesquelles on spécifie la taille de l'esclave en fonction des dimensions du maître, de telle sorte que l'esclave change de taille et d'emplacement lorsque la taille du maître change. Enfin, le gestionnaire de placement permet de mélanger ces styles de placement de façon, par exemple, que l'esclave ait une largeur et une hauteur fixes mais soit centré à l'intérieur du maître.

La commande **place** peut prendre plusieurs formes selon l'argument *option* qui lui est adjoint :

place *fenêtre option valeur?option valeur ...?*
: Charge le gestionnaire de placement de gérer la disposition d'un esclave dont le chemin est *fenêtre*. Les arguments restants consistent en une ou plusieurs paires de type *option-valeur* qui précisent la façon d'exploiter la géométrie de la fenêtre. L'argument *option* peut prendre une quelconque des valeurs acceptées par la commande **place configure**.

place configure *fenêtre?option??valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de l'esclave *fenêtre*. Si l'argument *option* n'est pas spécifié, la commande renvoie la liste de toutes les options disponibles pour *cheminCadre*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide.
L'argument *option* peut être une des valeurs suivantes :

-anchor *emplacement*
: L'argument *emplacement* spécifie quel point de la fenêtre désignée par l'argument *fenêtre* doit être positionné aux coordonnées (x,y) fixées par les options **-x** , **-y** , **-relx** et **-rely**. Le point d'ancrage est fonction de l'aire extérieure de *fenêtre* avec la bordure incluse s'il y a lieu. Ainsi, si l'argument *emplacement* est **se**, le coin inférieur droit de la bordure

de l'esclave *fenêtre* apparaîtra au point (x,y) dans le maître. Le point d'ancrage par défaut est **nw** (coin supérieur gauche).

-bordermode *mode*
L'argument *mode* détermine la façon dont les bordures sont prises en compte dans les calculs pour déterminer le placement des composants. La valeur par défaut est *inside*. Dans ce cas, le gestionnaire de placement considère que l'aire du maître est la zone strictement intérieure de la fenêtre de ce maître : une option -x 0 correspond à une abscisse immédiatement à l'intérieur de la bordure et une option -relwidth 1.0 signifie que l'esclave *fenêtre* doit remplir la zone intérieure de la bordure du maître (voir ci-dessous)
Si l'argument *mode* est *outside*, le gestionnaire de placement considère que l'aire du maître inclut sa bordure ; ce mode est typiquement utilisé pour placer un esclave *fenêtre* en dehors de son maître, comme par exemple avec les options :

```
-x 0 -y 0 -anchor ne
```

Enfin, l'argument *mode* peut être fixé à la valeur *ignore*, auquel cas les bordures sont ignorées : l'aire du maître est considérée comme étant son aire officielle dans le système X de fenêtrage (sous Unix), qui inclut les bordures internes mais pas les bordures externes.

-height *taille*
L'argument *taille* spécifie la hauteur de l'esclave *fenêtre* en unités d'écran valides (cf. p. 32). La hauteur sera la dimension externe de l'esclave *fenêtre*, bordure incluse s'il y a lieu. Si l'argument *taille* est une chaîne vide, ou si aucune option **-height** ou **-relheight** n'est spécifiée, la hauteur requise en interne par la fenêtre sera utilisée.

-in *maître*
L'argument *maître* spécifie le nom de chemin de la fenêtre relativement à laquelle *fenêtre* doit être placée. L'argument *maître* doit être n'importe quel parent de *fenêtre* ou descendant d'un parent de *fenêtre*. En outre, l'argument *maître* et l'argument *fenêtre* doivent tous les deux être des descendants de la même fenêtre de premier niveau. Ces restrictions garantissent que l'esclave sera visible dès que son maître le sera. Si cette option n'est pas spécifiée, le maître sera par défaut le parent de l'esclave *fenêtre*.

-relheight *taille*
L'argument *taille* spécifie la hauteur de l'esclave *fenêtre*. La hauteur est spécifiée par un nombre en virgule flottante relativement à la hauteur du maître : 0.5 signifie que la hauteur de *fenêtre* sera la moitié de celle du maître, 1.0 signifie que *fenêtre* aura la même hauteur que le maître, et ainsi de suite. Si à la fois les options **-height** et **-relheight** sont spécifiées pour un esclave, leurs valeurs sont additionnées. Par exemple, l'instruction

```
-relheight 1.0 -height -2
```

fait l'esclave plus petit que le maître de 2 pixels.

-relwidth *taille*
L'argument *taille* spécifie la largeur de l'esclave *fenêtre*. La largeur est spécifiée par un nombre en virgule flottante relativement à la largeur du maître : 0.5 signifie que *fenêtre* sera moitié aussi haut que le maître, 1.0 signifie que *fenêtre* aura la même largeur que le maître, et ainsi de suite. Si à la fois les options **-width** et **-relheight** sont spécifiées pour un esclave, leurs valeurs sont additionnées. Par exemple, l'instruction

```
-relheight 1.0 -width 5
```

fait l'esclave plus large que le maître de 5 pixels.

-relx *emplacement*
L'argument *emplacement* spécifie l'abscisse dans la fenêtre maître du point d'ancrage de l'esclave *fenêtre*. Avec cette option, l'emplacement est spécifié de manière relative par un nombre en virgule flottante : 0.0 correspond au bord gauche du maître et 1.0 correspond au bord droit. L'argument *emplacement* doit donc être dans l'intervalle [0.0 , 1.0]. Si à la fois les options **-x** et **-relx** sont spécifiées pour un esclave, leurs valeurs sont additionnées. Par exemple, l'instruction

```
-relx 0.5 -x -2
```

positionne le bord gauche de l'esclave 2 pixels à gauche du centre de son maître.

-rely *emplacement*
L'argument *emplacement* spécifie l'ordonnée dans la fenêtre maître du point d'ancrage de l'esclave *fenêtre*. Avec cette option, l'emplacement est spécifié de manière relative par un nombre en virgule flottante : 0.0 correspond au bord supérieur du maître et 1.0 au bord inférieur. L'argument *emplacement* doit donc être dans l'intervalle [0.0 , 1.0]. Si à la fois les options **-y** et **-rely** sont spécifiées pour un esclave, leurs valeurs sont additionnées. Par exemple, l'instruction

```
-rely 0.5 -x 3
```

positionne le bord supérieur de l'esclave 3 pixels en dessous du centre de son maître.

-width *taille*
L'argument *taille* spécifie la largeur de l'esclave *fenêtre* en unités d'écran valides (cf. p. 32). La largeur sera la largeur externe de l'esclave *fenêtre*, bordure incluse s'il y a lieu. Si l'argument *taille* est une chaîne vide, ou si aucune option **-width** ou **-relwidth** n'est spécifiée, la largeur requise en interne par la fenêtre sera utilisée.

-x *emplacement*
L'argument *emplacement* spécifie l'abscisse dans la fenêtre maître du point d'ancrage pour l'esclave *fenêtre*. L'emplacement est spécifié en unités d'écran valides (cf. p. 32) et n'a pas besoin de se trouver dans les limites de la fenêtre maître.

-y *emplacement*
L'argument *emplacement* spécifie l'ordonnée dans la fenêtre maître du point d'ancrage pour l'esclave *fenêtre*. L'emplacement est spécifié en unités d'écran valides (cf. p. 32) et n'a pas besoin de se trouver dans les limites de la fenêtre maître.

Si une même valeur est spécifiée séparément avec deux options différentes, comme par exemple **-x** et **-relx**, c'est la plus récente qui est utilisée et l'autre est ignorée.

place forget *fenêtre*
Retire au gestionnaire de placement le contrôle de l'esclave désigné par l'argument *fenêtre* et supprime en conséquence la fenêtre à l'écran. Si l'argument *fenêtre* n'est pas actuellement contrôlé par le gestionnaire de placement, la commande est sans effet. Cette commande renvoie une chaîne vide.

place info *fenêtre*
Renvoie une liste donnant la configuration courante de l'esclave *fenêtre*. La liste consiste en une série de paires de type *option-valeur* exactement sous la même forme que celles spécifiées avec la commande **place configure**.

place slaves *fenêtre*
Renvoie une liste de toutes les fenêtres esclaves dont l'argument *fenêtre* est le maître. S'il n'y a pas d'esclaves pour *fenêtre*, une chaîne vide est renvoyée. Si la configuration de la fenêtre a été récupérée avec la commande **place info** , cette configuration pourra être rechargée par la suite en utilisant tout d'abord une commande **place forget** pour effacer toute information existante puis une commande **place configure** avec l'information sauvegardée.

Propriétés spécifiques

Il n'est pas nécessaire pour la fenêtre maître d'être le parent de la fenêtre esclave. Cette propriété est utile en particulier dans les deux situations suivantes :

- pour des dispositions de fenêtres complexes, cela permet de créer une hiérarchie de sous-fenêtres dans le seul but d'aider à la mise en forme du parent. Les véritables enfants du parent peuvent être placés en réalité dans d'autres sous-fenêtres de la hiérarchie. Leur nom de chemin ne reflètera pas leur place dans la hiérarchie et on pourra cependant spécifier des options sans connaître la structure de cette hiérarchie.
- cela permet aussi de lier deux widgets collatéraux entre eux. Par exemple, le gestionnaire de placement peut forcer une fenêtre à être constamment centrée juste sous une des fenêtres collatérales, de telle sorte que si cette dernière change de position l'autre se repositionne en conséquence, au moyen d'une instruction telle que :
  -in *collatéral* -relx 0.5 -rely 1.0 -anchor n -bordermode outside

Enfin, à la différence des autres gestionnaires de placement, la commande **place** ne cherche pas à manipuler les dimensions des fenêtres maîtres ou des fenêtres parents de fenêtres esclaves. Pour contrôler les dimensions de ces fenêtres, on devra

se servir de fenêtres de type cadre (*frame*) ou canevas qui possèdent des options pour ce genre de manipulation (cf. p. 31 et).

radiobutton

Permet de créer et de manipuler les boutons-radios.

Syntaxe

radiobutton *cheminBoutonRadio ?options?*

Description

La commande **radiobutton** crée une nouvelle fenêtre (indiquée par l'argument *cheminBoutonRadio*) et en fait un composant graphique de type *bouton radio*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques du bouton radio telles que la couleur, la police, le texte, le relief initial etc. La commande **radiobutton** renvoie la valeur de son argument *cheminBoutonRadio*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminBoutonRadio*, mais en revanche l'objet parent de *cheminBoutonRadio* doit déjà exister.

Un bouton radio est un composant graphique qui affiche un petit losange ou un petit cercle appelé indicateur, accompagné d'un texte, d'une icone bitmap ou d'une image que l'on appelle l'étiquette du bouton radio. Si l'étiquette est textuelle, elle est entièrement composée dans une même police ; le texte peut occuper plusieurs lignes à l'écran, soit par utilisation de sauts de lignes, soit parce qu'un paramètre **-wrapLength** a été fixé pour forcer les coupures de ligne. Il est enfin possible de souligner certains caractères du texte avec l'option **-underline**, le plus souvent pour en faire par la suite des raccourcis clavier.

Un bouton radio possède tous les comportements d'un simple bouton. Il peut s'afficher dans trois états différents selon la valeur de l'option **-state**. D'autre part, son apparence peut être bombée, creusée ou plate. On peut le faire clignoter et il peut invoquer une commande Tcl lorsque le premier bouton de la souris est cliqué sur lui.

En outre, les boutons-radios peuvent être sélectionnés :

- si le bouton radio est sélectionné, l'indicateur est dessiné avec une apparence particulière et une variable Tcl associée est fixée à une valeur particulière (en général 1). La manière dont l'indicateur est marqué dépend du système d'exploitation : sous Unix il prend une forme de relief en creux avec une couleur particulière. Sous Windows et MacOS, une marque en forme de point circulaire est insérée à l'intérieur.
- si le bouton radio n'est pas sélectionné, l'indicateur est dessiné avec une apparence différente et la variable Tcl associée est fixée à une autre valeur particulière (en général 0). Sous Unix il prend une forme de relief bombé sans couleur particulière. Sous Windows et MacOS, la petite marque circulaire est effacée.

Dans le fonctionnement normal, plusieurs boutons-radios partagent une même variable et la valeur de cette variable indique le bouton radio actuellement sélectionné. Quand un bouton radio est sélectionné, il se charge de fixer la valeur de la variable en conséquence. Par ailleurs, les boutons-radios sont à l'écoute de leur variable associée : si la valeur de celle-ci est modifiée, chaque bouton radio se marquera lui-même en conséquence.

Par défaut, la variable nommée **selectedButton** est utilisée : elle contient le nom du bouton sélectionné ou une chaîne vide si aucun bouton radio n'est sélectionné. Le nom de la variable, de même que les valeurs qu'elle stocke pour représenter les états *sélectionné* et *désélectionné*, peuvent toujours être modifiés au moyen des options appropriées.

Des options de configuration peuvent aussi être employées pour modifier l'aspect de l'indicateur. Par défaut, un bouton radio est configuré pour se sélectionner et se désélectionner alternativement à chaque clic de souris.

Opérations sur les boutons-radios

Avec l'instruction :

radiobutton *cheminCase ?options?*

la commande **radiobutton** crée une nouvelle commande Tcl dont le nom est précisément *cheminBoutonRadio*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant le bouton radio. La forme générale est :

cheminBoutonRadio opération ?arg arg ...?

Les opérations reconnues par les composantes de type *radiobutton* sont les suivantes :

cheminBoutonRadio **cget** *option*

Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **radiobutton**.

cheminBoutonRadio **configure** *?option? ?valeur option valeur ...?*

Permet d'obtenir ou de modifier les options de configuration du bouton radio. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminCase*.

Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **radiobutton**.

cheminBoutonRadio **deselect**

Désélectionne le bouton radio et donne à la variable associée la valeur d'une chaîne vide. Si le bouton radio n'était pas sélectionné, la commande est sans effet.

cheminBoutonRadio **flash**
: Fait clignoter le bouton radio. Cet effet est obtenu en réaffichant à plusieurs reprises le bouton radio en faisant alterner les couleurs de son état normal et de son état activé. À la fin du clignotement, le bouton radio est laissé dans le même état que lorsque la commande a été invoquée. Cette commande est ignorée si l'état du bouton radio est *désactivé* (*disabled*).

cheminBoutonRadio **invoke**
: Accomplit exactement ce qui se produirait si un utilisateur cliquait sur le bouton radio avec la souris : l'état du bouton radio est commuté et la commande Tcl associée, si elle existe, est invoquée. La valeur de retour est celle de la commande Tcl ou une chaîne vide si aucune commande n'est associée au bouton. Cette commande est ignorée si l'état du bouton radio est *désactivé* (*disabled*).

cheminBoutonRadio **select**
: Sélectionne le bouton radio et donne à la variable associée la valeur correspondant à ce bouton.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les boutons-radios avec les caractéristiques suivantes :

1. sur les systèmes Unix, un bouton radio est activé dès que le pointeur de la souris passe au-dessus et désactivé dès qu'il le quitte. Sur les systèmes Mac et Windows, si le premier bouton de la souris est enfoncé, le bouton radio est activé dès que le pointeur se trouve dedans et désactivé dès qu'il en ressort ;
2. lorsque le premier bouton de la souris est enfoncé au-dessus d'un bouton radio, l'état est commuté et la commande associée, si elle existe, est invoquée ;
3. lorsque le bouton radio est focalisé par l'application, la barre d'espacement peut être utilisée pour l'invoquer.

Si l'état du bouton radio est *disabled*, aucune des actions mentionnées ne se produit. Les comportements décrits sont les comportements par défaut et peuvent toujours être modifiés en définissant de nouvelles liaisons pour des composants particuliers ou en redéfinissant les liaisons de classe.

Options standard

Les options standard supportées par les composants de type *message* sont rassemblées dans le tableau 12 de la page suivante. Ces options sont décrites en détail à la page 142.

-activebackground	activeBackground	Foreground
-activeforeground	activeForeground	Background
-anchor	anchor	Anchor
-background ou -bg	background	Background
-bitmap	bitmap	Bitmap
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-disabledforeground	disabledForeground	DisabledForeground
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-image	image	Image
-justify	justify	Justify
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus
-text	text	Text
-textvariable	textVariable	Variable
-underline	underline	Underline
-wraplength	wrapLength	WrapLength

TAB. 12 – *Options standard reconnues par les composants de type* message.

Options spécifiques

-command **command** **Command**

Spécifie une commande Tcl à associer au bouton. Cette commande est typiquement invoquée lorsque le premier bouton de la souris est relâché au-dessus de la fenêtre du bouton radio. La variable globale du bouton spécifiée avec l'option **-variable** est mise à jour avant que la commande ne soit invoquée.

-height **height** **Height**

Spécifie une hauteur pour le bouton. Si une image ou un bitmap est affiché dans le bouton radio, la valeur est mesurée en unités d'écran; pour du texte elle est mesurée en lignes de texte. Si cette option n'est pas spécifiée, la hauteur pour le bouton radio est calculée à partir de la taille de l'image, du bitmap ou du texte qu'il contient.

-indicatoron **indicatorOn** **IndicatorOn**

Spécifie si l'indicateur doit être dessiné ou non. La valeur de cette option doit être une valeur booléenne valide. Si la valeur est *false*, l'option **relief** est ignorée

et le relief du composant est toujours enfoncé (*sunken*) lorsqu'il est sélectionné et relevé (*raised*) le cas échéant.

-overrelief **overRelief** **OverRelief**

Spécifie un relief alternatif pour le bouton radio, à utiliser lorsque le curseur de la souris se trouve au-dessus de lui. Cette option peut être utilisée pour fabriquer des boutons de barre d'outils en utilisant la configuration suivante :

```
-relief flat -overrelief raised
```

Si la valeur passée à cette option est la chaîne vide, aucun relief alternatif n'est utilisé lorsque le curseur de la souris passe au-dessus du bouton radio. La chaîne vide est la valeur par défaut.

-selectcolor **selectColor** **Background**

Spécifie une couleur de fond à utiliser lorsque le bouton radio est sélectionné. Si **indicatorOn** est *true*, la couleur s'applique à l'indicateur. Sous Windows, cette couleur est utilisée comme arrière-plan pour l'indicateur indépendamment de l'état de la sélection. Si **indicatorOn** est *false*, cette couleur est utilisée comme arrière-plan pour le composant entier, à la place de **-background** ou **-activebackground**, chaque fois qu'il est sélectionné. Si l'option est spécifiée comme une chaîne vide, aucune couleur spéciale n'est utilisée lorsque le bouton radio est sélectionné.

-selectimage **selectImage** **SelectImage**

Spécifie une image à afficher (à la place de l'option **image**) lorsque le bouton radio est sélectionné. Cette option est ignorée à moins que l'option **image** n'ait été spécifiée.

-state **state** **State**

Spécifie l'un des trois états *normal*, *active*, ou *disabled* pour le bouton radio. Dans l'état normal, le bouton radio est affiché en utilisant les options **foreground** et **background**. L'état activé est typiquement utilisé lorsque la souris est au-dessus du bouton radio. Dans l'état activé, le bouton radio est affiché en utilisant les options **activeForeground** et **activeBackground**. L'état désactivé (*disabled*) signifie que le bouton radio ne réagit plus : les liaisons par défaut refuseront d'activer le bouton et ignoreront les clics de souris. Dans cet état les options **disabledForeground** et **background** déterminent comment le bouton doit être affiché.

-value **value** **Value**

Spécifie la valeur à stocker dans la variable associée au bouton radio chaque fois que celui-ci est sélectionné.

-variable **variable** **Variable**

Spécifie le nom d'une variable globale à fixer pour indiquer que le bouton radio est sélectionné ou non. Les modifications de cette variable ont pour effet de

sélectionner ou déselectionner le bouton lui-même. Par défaut, cette variable s'appelle *selectedButton*.

-width **width** **Width**

Spécifie une largeur pour le bouton. Si une image ou un bitmap est affiché dans le bouton radio, la valeur est mesurée en unités d'écran ; pour du texte elle est mesurée en nombre de caractères. Si cette option n'est pas spécifiée, la largeur pour le bouton radio est calculée à partir de la taille de l'image, du bitmap ou du texte qu'il contient.

raise

Modifie l'ordre d'empilement des fenêtres.

Syntaxe

raise *fenêtreA ?fenêtreB?*

Description

Si l'argument *fenêtreB* est omis, la commande fait passer la fenêtre *fenêtreA* au-dessus de toutes les fenêtres collatérales dans l'ordre d'empilement.

Si l'argument *fenêtreB* est spécifié, il doit indiquer le nom de chemin d'une fenêtre qui est soit une fenêtre collatérale (descendante du même parent) de la fenêtre *fenêtreA*, soit un descendant d'une fenêtre collatérale. Dans ce cas, la commande **raise** insère la fenêtre *fenêtreA* immédiatement au-dessus de celle désignée par *fenêtreB* ou du parent de *fenêtreB* si celui-ci est un collatéral de *fenêtreA* (ce qui revient alors à faire descendre la fenêtre dans l'ordre d'empilement !).

scale

Permet de créer et de manipuler des échelles graduées

Syntaxe

scale *cheminÉchelle ?options?*

Description

La commande **scale** crée une nouvelle fenêtre (indiquée par l'argument *cheminÉchelle*) et en fait un composant graphique de type *scale*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques du bouton telles que la couleur, la police, le texte, le relief initial etc. La commande **scale** renvoie la valeur de son argument *cheminÉchelle*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminÉchelle*, mais en revanche l'objet parent de *cheminÉchelle* doit exister effectivement.

Une échelle est un composant graphique qui affiche une glissière rectangulaire et un petit curseur. La glissière correspond à un intervalle de nombres réels que l'on définit au moyen des options **-from**, **-to** et **-resolution**. La position du curseur sur l'échelle permet de sélectionner une valeur réelle dans cet intervalle. Chaque fois que la valeur de l'échelle est modifiée, une commande Tcl définie grâce à l'option **-command** est invoquée afin d'informer d'autres composants graphiques. Par ailleurs, la valeur de l'échelle peut être liée à une variable Tcl au moyen de l'option **-variable** pour que les modifications apportées à l'une se reflètent sur l'autre et réciproquement.

On peut préciser trois caractéristiques dans les échelles que l'on peut activer ou désactiver à volonté au moyen d'options appropriées :

- une étiquette qui apparaît en haut à droite des échelles verticales ou en haut à gauche des échelles horizontales ;
- un nombre disposé immédiatement à gauche du curseur si l'échelle est verticale ou immédiatement au-dessus si elle est horizontale ;
- une série de repères numériques à gauche de la glissière si celle-ci est verticale ou immédiatement en dessous si elle est horizontale.

Opérations sur les échelles

Avec l'instruction :

scale *cheminÉchelle ?options?*

la commande **scale** crée une nouvelle commande Tcl dont le nom est précisément *cheminÉchelle*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant l'échelle. La forme générale est :

cheminÉchelle opération ?arg arg ...?

Les opérations reconnues par les composantes de type *scale* sont les suivantes :

cheminÉchelle **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **scale**.

cheminÉchelle **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de l'échelle graduée. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminÉchelle*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **scale**.

cheminÉchelle **coords** *?valeur?*
: Renvoie une liste de deux éléments qui sont les coordonnées x et y du point situé sur la ligne médiane de la glissière et correspond à la valeur spécifiée par l'argument *valeur*. Si *valeur* est omis, la valeur courante de l'échelle est utilisée.

cheminÉchelle **get** *?x y?*
: Si x et y sont omis, cette commande renvoie la valeur courante de l'échelle. Si x et y sont spécifiés, ils indiquent les coordonnées en pixels d'un point dans l'échelle ; la commande renvoie alors la valeur de l'échelle correspondant à ce point. Seule une des coordonnées x et y est utilisée : x pour les échelles horizontales et y pour les échelles verticales.

cheminÉchelle **identify** x y
: Renvoie une chaîne qui indique quelle partie de l'échelle se trouve sous le point de coordonnées x et y. Une valeur de retour peut prendre une des valeurs symboliques suivantes :
  - *slider* désigne le curseur ;
  - *trough1* désigne la partie de la glissière qui se trouve avant le curseur ;
  - *trough2* désigne la partie de la glissière qui se trouve après le curseur ;

  Si le point de coordonnées x et y n'est pas au-dessus d'un de ces éléments, la commande renvoie une chaîne vide.

cheminÉchelle **set** *valeur*
: Cette commande est invoquée pour modifier la valeur courante de l'échelle, et par conséquent la position où le curseur est affiché. L'argument *valeur* donne la nouvelle valeur de l'échelle. Cette commande est sans effet si l'échelle est désactivée.

-activebackground	activeBackground	Foreground
-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-relief	relief	Relief
-repeatdelay	repeatDelay	RepeatDelay
-repeatinterval	repeatInterval	RepeatInterval
-takefocus	takeFocus	TakeFocus
-troughcolor	troughColor	Background

TAB. 13 – *Options standard reconnues par les composants de type* scale.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les échelles graduées avec les caractéristiques suivantes (le comportement peut différer pour les échelles horizontales et verticales) :

1. Si le premier bouton de la souris est enfoncé dans la glissière, la valeur de l'échelle sera incrémentée ou décrémentée de la valeur de l'option **-resolution** de telle sorte que le curseur se déplace dans la direction du pointeur. Si le bouton est maintenu, l'action se répète.
2. Si le premier bouton de la souris est enfoncé au-dessus du curseur, le curseur peut être déplacé en glissant la souris.
3. Si le premier bouton de la souris est pressé dans la glissière avec la touche Contrôle enfoncée, le curseur se déplace jusqu'à l'extrêmité de l'intervalle dans la direction du pointeur.
4. Si le second bouton de la souris est enfoncé, la valeur de l'échelle est fixée à la position de la souris. Si la souris est glissée avec le second bouton enfoncé, la valeur de l'échelle change en suivant le déplacement.
5. Les touches Haut et Gauche déplacent le curseur vers le haut (resp. la gauche) de la valeur de l'option **-resolution**.
6. Les touches Bas et Droite déplacent le curseur vers le bas (resp. la droite) de la valeur de l'option **-resolution**.
7. Les combinaisons de touches Contrôle-Haut et Contrôle-Gauche déplacent le curseur vers le haut (resp. la gauche) de la valeur de l'option **-bigIncrement**.
8. Les combinaisons de touches Contrôle-Bas et Contrôle-Droite déplacent le curseur vers le bas (resp. la droite) de la valeur de l'option **-bigIncrement**.
9. La touche Début (*Home*) déplace le curseur à l'extrêmité supérieure (ou gauche) de l'intervalle.

10. La touche Fin (*End*) déplace le curseur à l'extrêmité inférieure (ou droite) de l'intervalle.

Si l'échelle est désactivée au moyen de l'option **-state**, toutes les liaisons ci-dessus sont sans effet.

Les comportements décrits sont les comportements par défaut et peuvent toujours être modifiés en définissant de nouvelles liaisons pour des échelles particulières ou en redéfinissant les liaisons de classe.

Options standard

Les options standard supportées par les composants de type *scale* sont rassemblées dans le tableau 13 de la page précédente. Ces options sont décrites en détail à la page 142.

Options spécifiques

-bigincrement **bigIncrement** **BigIncrement**

Certaines interactions avec l'échelle graduée font que sa valeur change par de « grandes » différences; cette option spécifie ce que l'on considère comme une grande différence. Si la valeur spécifiée est 0, les incréments en question vaudront par défaut 1/10 de l'étendue complète de l'échelle graduée.

-command **command** **Command**

Spécifie le préfixe d'une commande Tcl à invoquer chaque fois que la valeur de l'échelle graduée est modifiée par une commande. La commande véritable qui sera exécutée est constituée de la valeur de cette option suivie par une espace puis un nombre réel indiquant la nouvelle valeur de l'échelle graduée.

-digits **digits** **Digits**

Il s'agit d'un nombre entier qui spécifie combien de chiffres significatifs doivent être retenus en convertissant la valeur de l'échelle graduée en une chaîne. Si le nombre est négatif ou nul l'échelle graduée choisit la plus petite valeur garantissant que chaque position possible de l'échelle puisse être marquée par une chaîne différente.

-from **from** **From**

Une valeur réelle correspondant, selon son orientation, à l'extrêmité gauche ou supérieure de l'échelle graduée.

-label **label** **Label**

Une chaîne à afficher comme étiquette pour l'échelle graduée. Pour les échelles verticales l'étiquette est affichée immédiatement à droite de l'extrêmité supérieure

de l'échelle graduée. Pour les échelles horizontales l'étiquette est affichée immédiatement au-dessus de l'extrêmité gauche de l'échelle graduée. Si l'option est spécifiée comme une chaîne vide, aucune étiquette n'est affichée.

-length **length** **Length**

Spécifie la valeur souhaitée pour la dimension longue de l'échelle graduée, en unités d'écran. Pour les échelles verticales c'est la hauteur de l'échelle graduée; pour les échelles horizontales, c'est sa largeur.

-orient **orient** **Orient**

Cette option spécifie l'orientation (horizontale ou verticale) à utiliser. Elle doit prendre une des valeurs symboliques suivantes : *horizontal* ou *vertical* ou une abréviation de ces termes.

-resolution **resolution** **Resolution**

Une valeur réelle spécifiant la « résolution » de l'échelle graduée: si cette valeur est positive, la valeur de l'échelle graduée sera toujours arrondie à un multiple entier de cette valeur, tout comme les graduations et les extrêmités de l'échelle.

-showvalue **showValue** **ShowValue**

Spécifie une valeur booléenne indiquant si la valeur courante de l'échelle graduée doit être affichée ou non.

-sliderlength **sliderLength** **SliderLength**

Spécifie la taille du curseur mesurée en unités d'écran le long de la dimension longue du curseur. La valeur est exprimée en unités d'écran.

-sliderrelief **sliderRelief** **SliderRelief**

Spécifie le relief à utiliser pour dessiner le curseur, comme par exemple *raised* ou *sunken*.

-state **state** **State**

Spécifie l'un des trois états *normal*, **active** ou *disabled* pour l'échelle graduée. Si l'échelle graduée est désactivée, la valeur ne peut être modifiée et l'échelle graduée ne réagira pas. Si l'échelle graduée est active, le curseur est affiché en utilisant la couleur spécifiée par l'option **activeBackground**.

-tickinterval **tickInterval** **TickInterval**

Cette option doit prendre une valeur réelle qui détermine l'espace entre les marques de graduation numériques affichées en dessous ou à gauche du curseur. Si la valeur est 0, aucune marque de graduation n'est affichée.

-to **to** **To**

Spécifie une valeur réelle correspondant à l'extrêmité droite ou inférieure de l'échelle graduée. Cette valeur doit être au moins égale à celle spécifiée par l'option **from**.

-troughcolor **troughColor** **Background**

Spécifie la couleur à utiliser pour la glissière.

-variable **variable** **Variable**

Spécifie le nom d'une variable globale à lier à l'échelle graduée. Chaque fois que la valeur de la variable change, l'échelle graduée est mise à jour pour refléter la nouvelle valeur. Chaque fois que l'échelle graduée est manipulée interactivement, la variable est modifiée elle aussi pour refléter la nouvelle valeur de l'échelle graduée.

-width **width** **Width**

Spécifie la largeur souhaitée pour la glissière, exprimée en unités d'écran.

scrollbar

Permet de créer et de manipuler les barres de défilement.

Syntaxe

scrollbar *cheminBarre ?options?*

Description

La commande **scrollbar** crée une nouvelle fenêtre (indiquée par l'argument *cheminBarre*) et en fait un composant graphique de type *barre de défilement*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques de la barre de défilement telles que la couleur, la police ou le relief etc. La commande **scrollbar** renvoie la valeur de son argument *cheminBarre*. Au moment de son invocation, il ne doit exister aucune autre fenêtre nommée *cheminBarre*, mais en revanche l'objet parent de *cheminBarre* doit déjà exister.

Une barre de défilement est un composant graphique qui affiche des flèches aux deux extrêmités d'une glissière et un curseur dans sa partie intermédiaire. Elle renseigne sur la partie visible de la fenêtre de document qui lui est associée selon la position et la taille du curseur. Inversement, on peut ajuster la vue d'un document en déplaçant le curseur.

Éléments

Une barre de défilement affiche cinq éléments auxquels les commandes associées se réfèrent de la manière suivante :

arrow1
: La flèche supérieure ou gauche de la barre de défilement.

trough1
: La région entre le curseur et l'élément **arrow1**.

slider
: Le rectangle dit « curseur » qui indique la proportion de ce qui est visible dans la fenêtre associée.

trough2
: La région entre le curseur et l'élément **arrow2**.

arrow2
: La flèche inférieure ou droite de la barre de défilement.

Opérations sur les barres de défilement

Avec l'instruction :
scrollbar *cheminBarre ?options?*

la commande **scrollbar** crée une nouvelle commande Tcl dont le nom est précisément *cheminBarre*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant la barre de défilement. La forme générale est :

cheminBarre opération ?arg arg ...?

Les opérations reconnues par les composantes de type *scrollbar* sont les suivantes :

cheminBarre **activate** *?élément?*
: Marque l'élément indiqué par l'argument *élément* comme actif, ce qui a pour effet de l'afficher avec les caractéristiques spécifiées au moyen des options **activeBackground** et **-activeRelief**. Les seuls éléments reconnus par cette commande sont **arrow1**, **slider** et **arrow2**. Si l'argument *élément* n'est pas spécifié, la commande renvoie le nom de l'élément actif courant, ou une chaîne vide lorsqu'aucun élément n'est actif.

cheminBarre **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **scrollbar**.

cheminBarre **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de la barre de défilement. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminCase*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **scrollbar**.

cheminBarre **delta** *deltaX deltaY*
: Cette commande renvoie un nombre réel indiquant en proportion le changement de valeur dans la barre de défilement qui correspond à un déplacement donné du curseur. Si, par exemple, la barre de défilement est horizontale, le résultat indique de combien il faut modifier le réglage de la barre de défilement pour que le curseur se déplace de *deltaX* pixels vers la droite (et dans ce cas *deltaY* est ignoré). Si la barre de défilement est verticale, le résultat indique de combien il faut modifier le réglage de la barre de défilement pour que le curseur se déplace de *deltaY* pixels vers le bas. Les arguments et le résultat peuvent être nuls ou même négatifs.

cheminBarre **fraction** x y
: Cette commande renvoie un nombre entre 0 et 1 qui indique où le point de coordonnées x et y se trouve dans la partie creuse dans laquelle se déplace le curseur. La valeur 0 correspond au sommet ou à la gauche de la partie creuse, la valeur 1 au bas ou à la droite, 0.5 au milieu etc. Les coordonnées x et y sont exprimées en pixels relativement à la barre de défilement. Si les valeurs

passées pour x et y correspondent à un point situé en dehors de la barre, c'est le point le plus proche qui est utilisé à la place.

cheminBarre **get**
Cette commande renvoie les réglages de la barre de défilement sous la forme d'une liste dont les éléments sont les arguments de la plus récente commande **set** invoquée.

cheminBarre **identify** *x y*
Cette commande renvoie le nom de l'élément de la barre de défilement situé sous le point de coordonnées x et y (comme par exemple **arrow1**) ou une chaîne vide si le point ne correspond à aucun élément de la barre de défilement. Les coordonnées x et y sont exprimées en pixels relativement à la barre de défilement.

cheminBarre **set** *premier dernier*
Cette commande est invoquée par l'objet associé à la barre de défilement pour informer celle-ci sur sa position courante. La commande prend deux arguments qui sont tous deux des nombres réels compris entre 0 et 1. Ces fractions indiquent la portion du document qui est visible dans l'objet associé. Par exemple si l'argument *premier* vaut 0.2 et l'argument *dernier* vaut 0.4, cela signifie que le début de la partie visible est à 20% du document total et que la fin est à 40%.

Commandes de défilement

Lorsque l'utilisateur actionne le curseur, la barre de défilement notifie l'objet associé à propos de la modification. Cette notification se fait par l'évaluation d'un script Tcl défini au moyen de l'option **-command** de la barre de défilement. Cette commande Tcl peut prendre une des formes suivantes où *préfixe* désigne le contenu de l'option **-command** qui prend habituellement la forme `.t xview` ou `.t yview`:

préfixe **moveto** *fraction*
L'argument *Fraction* est un nombre réel compris entre 0 et 1. Le composant doit ajuster la vue qu'il contient de telle sorte que le point correspondant à *fraction* apparaisse au début. Ainsi une valeur de 0 pour *fraction* correspond au début du document, une valeur de 1.0 à la fin du document, 0.333 à un point placé au tiers du document etc.

préfixe **scroll** *nb units*
Le composant doit ajuster la vue qu'il contient d'un nombre *nb* d'unités. Les unités peuvent être quoi que ce soit qui ait un sens pour le composant, comme des caractères ou des lignes dans un objet de texte. L'argument *nb* est soit 1 pour un défilement d'une unité vers le haut ou la gauche de la fenêtre, soit -1 pour un défilement d'une unité vers le bas ou la droite de la fenêtre.

préfixe **scroll** *nb pages*
Le composant doit ajuster la vue qu'il contient d'un nombre *nb* de pages. C'est au composant lui-même de définir ce qu'il faut entendre par *page*. En général cela correspond à un tout petit peu moins que le contenu de la fenêtre pour

qu'il y ait une petite partie commune entre la vue précédente et la suivante. L'argument *nb* est soit 1 pour passer à la page suivante, soit -1 pour passer à la page précédente.

Signalons que les versions de Tk antérieures à 4.0 utilisaient une syntaxe différente maintenant devenue obsolète pour les opérations **set** et **get**.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les barres de défilement avec les caractéristiques suivantes (le comportement peut différer pour les barres horizontales et verticales ; celui des barres horizontales est indiqué entre parenthèses) :

1. Presser le premier bouton de la souris sur l'élément **arrow1** déplace l'objet associé vers le haut (la gauche) d'une unité, donnant l'impression que le contenu du document s'est déplacé vers la bas (la droite). Si le bouton est maintenu pressé, l'action est répétée.
2. Presser le premier bouton de la souris sur l'élément **trough1** déplace l'objet associé vers le haut (la gauche) du contenu d'un écran entier, donnant l'impression que le contenu du document s'est déplacé vers la bas (la droite) d'un écran entier.
3. Presser le premier bouton de la souris sur le curseur et la déplacer provoque un glissement de l'objet associé. Si l'option **-jump** a la valeur *true*, l'objet ne suit pas le déplacement du curseur continûment : il saute à sa nouvelle position seulement lorsque le bouton de la souris est relâché.
4. Presser le premier bouton de la souris sur l'élément **trough2** déplace l'objet associé vers le bas (la droite) du contenu d'un écran entier, donnant l'impression que le contenu du document s'est déplacé vers la bas (la droite) d'un écran entier. Si le bouton est maintenu pressé, l'action est répétée.
5. Presser le premier bouton de la souris sur l'élément **arrow1** déplace l'objet associé vers le bas (la droite) d'une unité, donnant l'impression que le contenu du document s'est déplacé vers la bas (la droite). Si le bouton est maintenu pressé, l'action est répétée.
6. Si le second bouton de la souris est enfoncé au-dessus de la partie creuse ou du curseur, l'objet se place en fonction de la position de la souris ; en faisant glisser la souris avec le second bouton enfoncé le contenu de l'objet associé glisse simultanément. Si le second bouton de la souris est enfoncé sur une des flèches, le résultat est le même que lorsqu'on presse le premier bouton.
7. Si le premier bouton de la souris est enfoncé avec la touche Contrôle, alors si la souris est au-dessus des éléments **arrow1** ou **trough1** le contenu de l'objet associé se déplace au sommet (à la gauche) du document. Si la souris est au-dessus des éléments **arrow2** ou**trough2**, le contenu de l'objet associé se déplace à la fin (à la droite) du document. Ailleurs l'action est sans effet.
8. Dans les barres de défilement verticales les touches flèches vers le haut et vers le bas se comportent comme un clic sur les flèches **arrow1** et **arrow2**,

-activebackground	activeBackground	Foreground
-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-relief	relief	Relief
-repeatdelay	repeatDelay	RepeatDelay
-repeatinterval	repeatInterval	RepeatInterval
-takefocus	takeFocus	TakeFocus

TAB. 14 – *Options standard reconnues par les composants de type* scrollbar.

respectivement. Dans les barres de défilement horizontales ces touches sont sans effet.

9. Dans les barres de défilement verticales, les touches Contrôle-Haut et Contrôle-Bas se comportent comme un clic sur les éléments **trough1** et **trough2**, respectivement. Dans les barres de défilement horizontales ces touches sont sans effet.
10. Dans les barres de défilement horizontales les touches flèches vers le haut et vers le bas se comportent comme un clic sur les flèches **arrow1** et **arrow2**, respectivement. Dans les barres de défilement verticales ces touches sont sans effet.
11. Dans les barres de défilement horizontales, les touches Contrôle-Haut et Contrôle-Bas se comportent comme un clic sur les éléments **trough1** et **trough2**, respectivement. Dans les barres de défilement verticales ces touches sont sans effet.
12. Les touches Précédent et Suivant se comportent comme un clic sur les éléments **trough1** et **trough2**, respectivement.
13. La touche Début (*Home*) ajuste la vue au sommet (côté gauche) du document.
14. La touche Fin (*End*) ajuste la vue à la fin (côté droit) du document.

Options standard

Les options standard supportées par les composants de type *scrollbar* sont rassemblées dans le tableau 14. Ces options sont décrites en détail à la page 142.

Options spécifiques

-activerelief **activeRelief** **ActiveRelief**

Spécifie le relief à utiliser pour afficher l'élément lorsqu'il est actif. Les éléments autres que l'élément actif sont toujours affichés en relief.

-command **command** **Command**

Spécifie le préfixe d'une commande Tcl à invoquer pour changer la vue dans l'objet associé à la barre de défilement. Lorsqu'un utilisateur requiert une modification de la vue par manipulation de la barre de défilement, une commande Tcl est invoquée. La commande véritable qui est exécutée est constituée de la valeur de cette option suivie d'arguments additionnels. Cette option a presque toujours une valeur telle que *.t xview* ou *.t yview*, constituée du nom de l'objet contrôlé par la barre et de **xview** ou **yview** selon que la barre est destinée à un défilement horizontal ou vertical. Tous les objets servant au défilement possèdent des commandes **xview** et **yview** qui acceptent précisément les arguments qui seront ajoutés par la barre de défilement (voir la description complète au paragraphe *Commandes de défilement* ci-dessus).

-elementborderwidth **elementBorderWidth** **BorderWidth**

Spécifie la largeur de la bordure dessinée autour des éléments internes de la barre de défilement (les deux flèches et le curseur). La valeur est exprimée en unités d'écran. Si cette valeur est négative, la valeur de l'option **borderWidth** est utilisée à la place.

-jump **jump** **Jump**

Cette option détermine quand les notifications concernant les changements de valeur doivent être exécutées. La valeur de l'option doit être une valeur booléenne valide : si la valeur est *false*, les mises à jour se font continûment lorsque le curseur est déplacé ; si la valeur est *true*, les mises à jour sont retardées jusqu'au moment où le bouton de la souris est relâché. Dans ce dernier cas, la valeur fait un saut (*jump*) à sa nouvelle valeur.

-orient **orient** **Orient**

Cette option spécifie l'orientation (horizontale ou verticale) à utiliser. Elle doit prendre une des valeurs symboliques suivantes : *horizontal* ou *vertical* ou une abréviation de ces termes.

-troughcolor **troughColor** **Background**

Spécifie la couleur à utiliser pour la glissière. Cette option est ignorée pour les barres de défilement sous Windows.

-width **width** **Width**

Spécifie la largeur souhaitée pour le barre de défilement, sans compter la bordure 3-D s'il y en a. La valeur est exprimée en unités d'écran.

selection

Manipule les sélections sous système X.

Syntaxe

selection *sous-commande ?arg arg ...?*

Description

Cette commande fournit une interface Tcl pour le mécanisme de sélection du système graphique X sous Unix et implémente les fonctionnalités complètes décrites dans le manuel ICCCM (*Inter-Client Communication Conventions Manual*). La commande **selection** peut prendre plusieurs formes en fonction de la sous-commande qui lui est adjointe. Les formes supportées sont les suivantes :

selection clear ?-**displayof** *fenêtre*? ?-**selection** *selection*?
Si *selection* existe quelque part sur le moniteur de la fenêtre désignée par l'argument *fenêtre*, cette commande l'efface de telle sorte qu'aucune fenêtre ne détienne plus la sélection. L'argument *sélection* spécifie la sélection X qui doit être effacée : il doit prendre comme valeur un des noms-clés tels que PRIMARY ou CLIPBOARD (on se reportera au manuel ICCCM pour tous les détails). La valeur par défaut est PRIMARY pour *sélection* et « . » pour *fenêtre*. Renvoie une chaîne vide.

selection get ?-**displayof** *fenêtre*? ?-**selection** *selection*? ?-**type** *type*?
Récupère la valeur de *sélection* du moniteur de la fenêtre désignée par l'argument *fenêtre* et la renvoie comme résultat. La valeur par défaut est PRIMARY pour *sélection* et « . » pour *fenêtre*. L'argument *type* spécifie la forme sous laquelle la sélection doit être renvoyée (la cible de conversion, dans la terminologie ICCCM) : il doit prendre comme valeur un des noms-clés tels que STRING ou FILE_NAME (on se reportera au manuel ICCCM pour tous les détails). La valeur par défaut pour *type* est STRING. Le propriétaire de la sélection peut choisir de renvoyer la sélection dans divers formats de représentation différents, comme STRING, ATOM, INTEGER, etc. (ce format est différent du type de la sélection type ; voir le manuel ICCCM). Si la sélection est renvoyée sous un format autre qu'une chaîne, comme INTEGER ou ATOM, la commande **selection** la convertit au format chaîne comme une collection de champs séparés par des espaces : les atomes sont convertis en leur dénomination textuelle et le reste est converti en entiers hexadécimaux.

selection handle ?-**selection** *sélection*? ?-**type** *type*? ?-**format** *format*? *fenêtre commande*
Crée un gestionnaire pour les requêtes de sélection, de telle sorte que la commande désignée par l'argument *commande* sera exécutée dès que *sélection* est possédée par *fenêtre* et que quelqu'un essaie de la récupérer sous la forme

spécifiée par l'argument *type* (par exemple *type* est spécifié dans la commande **selection get**). Les valeurs par défaut sont PRIMARY pour *sélection*, STRING pour *type* et STRING pour *format*. Si *commande* est une chaîne vide tout gestionnaire déjà existant pour *fenêtre*, *type* et *sélection* est supprimé.

Lorsque *sélection* est requise, que *fenêtre* est propriétaire de la sélection, et que *type* est le type requis, *commande* sera exécutée comme une commande avec deux nombres supplémentaires ajoutés. Ces deux nombres supplémentaires sont *décalage* et *maxCars*: *décalage* spécifie une position de caractère de départ dans la sélection et *maxCars* indique le nombre maximum de caractères à récupérer. La commande devrait renvoyer une valeur consistant constituée de *maxCars* caractères de la sélection au plus, en partant de la position *décalage*. Pour de très grandes sélections (supérieures à *maxCars*) la sélection sera récupérée en utilisant plusieurs invocations de *commande* avec des valeurs de *décalage* croissantes. Si *commande* renvoie une chaîne dont la longueur est inférieure à *maxCars*, la valeur de retour est censée contenir tout le reste de la sélection; si la longueur du résultat de *commande* est égale à *maxCars* alors *commande* sera invoquée à nouveau, jusqu'à ce qu'elle renvoie un résultat inférieur à *maxCars*. La valeur de *maxCars* sera toujours relativement grande (des milliers de caractères).

Si *commande* renvoie une erreur, la récupération de la sélection sera rejetée exactement comme si la sélection n'existait pas du tout.

L'argument *format* spécifie la représentation à utiliser pour transmettre la sélection à l'émetteur de la requête et vaut par défaut STRING. Si *format* est STRING, la sélection est transmise en caractères ASCII 8-bits (c'est-à-dire simplement sous la forme renvoyée par *commande*). Si *format* est ATOM, la valeur de retour de *commande* est découpée en champs séparés par des espaces; chaque champ est converti à sa valeur atomique, et la valeur atomique 32-bits est transmise au lieu du nom de l'atome lui-même. Pour tout autre *format*, la valeur de retour de *commande* est découpée en champs séparés par des espaces; chaque champ est converti en entier 32-bits; un tableau d'entiers est transmis au demandeur de la sélection. L'argument *format* est rendu nécessaire seulement pour la compatibilité avec les requêtes de sélection qui ne sont pas faites dans Tk. Si Tk est utilisé pour récupérer la sélection, la valeur est reconvertie en une chaîne et l'argument *format* est sans objet.

selection own ?**-displayof** *fenêtre*? ?**-selection** *sélection*?

selection own ?**-command** *commande*? ?**-selection** *sélection*? *fenêtre*

La première forme de la commande **selection own** renvoie le nom de chemin de la fenêtre de l'application qui détient la *sélection* sur le moniteur contenant *fenêtre*, ou une chaîne vide si aucune fenêtre de cette application ne détient la sélection. La valeur par défaut est PRIMARY pour l'argument *sélection* et « . » pour *fenêtre*.

La seconde forme de la commande **selection own** rend la fenêtre désignée par l'argument *fenêtre* propriétaire de *sélection* sur le moniteur de *fenêtre*, renvoyant une chaîne vide comme résultat. Le propriétaire existant, s'il y en a un, est avertit qu'il a perdu la sélection. Si l'argument *commande* est

spécifié, il désigne un script Tcl à exécuter lorsqu'une autre fenêtre réclame la sélection à *fenêtre*. La valeur par défaut de *sélection* est PRIMARY.

send

Exécute une commande dans une autre application.

Syntaxe

send *?options? nomApplication nomCommande ?arg arg ...?*

Description

Cette commande fait exécuter la commande *nomCommande* (avec les arguments éventuels *arg*) par l'application nommée *nomApplication*. Elle renvoie le résultat ou une erreur résultant de l'exécution de cette commande. L'argument *nomApplication* peut désigner le nom de n'importe quelle application dont la fenêtre principale se trouve sur le moniteur contenant la fenêtre principale de l'émetteur de l'instruction **send**. Il n'est pas nécessaire que ce soit dans le même processus. Si aucun argument *arg* n'est présent, la commande à exécuter est contenue entièrement dans l'argument *nomCommande*. Si un ou plusieurs arguments *arg* sont présents, ils sont concaténés pour former la commande à exécuter, comme avec la commande Tcl **eval**.

Les options actuellement supportées sont les suivantes :

-async
: Établit une invocation asynchrone. Dans ce cas, la commande **send** retournera immédiatement sans attendre que *nomCommande* achève son exécution dans l'application cible ; aucun résultat ne sera donc disponible et les éventuelles erreurs dans la commande envoyée seront ignorées. Si l'application cible est dans le même processus que l'application émettrice, l'option **-async** est ignorée.

-displayof *cheminFenêtre*
: Spécifie que la fenêtre principale de l'application cible se trouve sur le moniteur de la fenêtre désignée par l'argument *cheminFenêtre*, au lieu du moniteur contenant la fenêtre principale de l'application émettrice de la commande.

--
: Indique la fin des options sur la ligne de commande.

Le nom d'une application Tk est fixé initialement à partir du nom du programme ou du script qui a créé l'application. On peut obtenir ou modifier le nom d'une application grâce à la commande **tk appname**.

Désactivation de la commande *send*

Si la commande **send** est retirée d'une application, par exemple au moyen d'une instruction telle que

```
rename send {}
```

alors l'application ne répondra plus aux requêtes émises par **send**, et ne pourra plus émettre elle-même de requêtes de ce type. Pour rétablir ce type de communication, il faudra invoquer la commande **tk appname**.

Sécurité

La commande **send** est potentiellement un danger pour la sécurité. Sous Unix, toute application qui peut se connecter au serveur X peut envoyer des scripts à vos applications. Ces scripts peuvent fort bien utiliser Tcl pour lire et écrire vos fichiers et invoquer des sous-processus en votre nom. Les contrôles d'accès tels que ceux fournis par xhost sont particulièrement peu sûrs, puisqu'ils autorisent quiconque possède un compte sur des hôtes particuliers à se connecter à votre serveur et, s'il est désactivé, autorise tout le monde à se connecter n'importe où sur le serveur.

Afin de procurer un minimum de sécurité, Tk vérifie quel est le contrôleur d'accès utilisé par le serveur et rejette les commandes **send** qui arrivent à moins que, d'une part, les contrôles d'accès du type xhost ne soient activés et que, d'autre part, la liste des hôtes activés soit vide. Cela signifie que les applications ne pourront se connecter à votre serveur qu'en ayant recours à une autre forme d'authentification telle que xauth par exemple.

Sous Windows, la commande **send** est totalement désactivée. La plupart de ses fonctionnalités sont fournies par la commande **dde** à la place.

spinbox

Permet de créer et de manipuler des champs de saisie « cycliques »

Syntaxe

spinbox *chemin ?options?*

Description

Un champ de saisie « cyclique », appelé en anglais *spinbox*, est un champ de saisie particulier permettant à l'utilisateur, en plus de la simple saisie de valeurs comme avec la commande *entry*, de se déplacer vers le haut ou vers le bas à travers une suite prédéfinie de valeurs. Ils sont munis dans la partie droite de la fenêtre de saisie de deux petites flèches superposées, l'une orientée vers le haut, l'autre vers le bas afin de faire défiler dans un sens ou dans l'autre les éléments prédéfinis. Ces deux flèches fonctionnent comme des boutons et on peut leur associer une commande à exécuter lorsqu'elles sont invoquées. Le terme cyclique n'est pas tout à fait approprié car il suggère que, lorsqu'on arrive en fin de liste, on passe ensuite à l'élément en début de liste, ce qui n'est pas le cas.

Les spinbox sont une extension des champs de saisie et héritent donc de toutes les propriétés de ces derniers en particulier pour ce qui est de l'insertion de caractères, la sélection de parties du texte, la focalisation, la validation des saisies, les substitutions avec le symbole % dans les options **-validateCommand** et **-invalidCommand**. On se reportera à la référence de la commande **entry** à la page 67 pour plus de détails. Lorsqu'une *spinbox* est créée pour le première fois, le champ de saisie est vide.

Opérations sur les spinbox

Toutes les opérations applicables à un composant de type *entry* sont également applicables à une *spinbox*. Pour mémoire, elles sont rassemblées dans le tableau 15 de la page suivante. En outre, les spinbox acceptent quatre opérations supplémentaires qui leur sont spécifiques :

chemin **identify** *x y*
: Renvoie le nom de l'élément de fenêtre correspondant aux coordonnées *x* et *y* dans la *spinbox*. La valeur renvoyée peut être une des constantes symboliques suivantes : *none*, *spindown*, *spinup* ou *entry*.

chemin **invoke** *élément*
: Cette commande invoque l'élément spécifié, qui peut être *spindown* ou *spinup*, et déclenche l'action qui lui est associée.

chemin **set** *?chaîne?*
: Si l'argument *chaîne* est spécifié, la *spinbox* essayera d'attribuer cette valeur au champ de saisie. Si l'argument *chaîne* n'est pas spécifié, la commande

bbox	**scan dragto**	**selection to**
cget	**scan mark**	**selection**
configure	**scan**	**validate**
delete	**selection adjust**	**xview moveto**
get	**selection clear**	**xview scroll**
icursor	**selection from**	**xview**
index	**selection present**	
insert	**selection range**	

TAB. 15 – *Les opérations communes aux composants de type* entry *et* spinbox.

renvoie simplement la valeur du champ. Si une commande de validation a été déclarée, elle se déclenchera lorsque la chaîne est fixée.

chemin **validate**
Cette commande est utilisée pour forcer une évaluation du test spécifié par l'option **-validateCommand** indépendamment de la valeur qui a été attribuée à l'option **-validate** (l'option **-validate** est fixée temporairement à la valeur *all*). La valeur de retour doit être 0 ou 1.

Options standard

Les options standard supportées par les composants de type *spinbox* sont rassemblées dans le tableau 16 de la page suivante. Ces options sont décrites en détail à la page 142.

Options spécifiques

-buttonbackground **buttonBackground** **Background**
La couleur de fond à utiliser pour les boutons de rotation.

-buttoncursor **buttonCursor** **Cursor**
La forme de curseur à utiliser lorsque la souris passe sur les boutons de rotation. Si c'est la chaîne vide (valeur par défaut), un curseur par défaut est utilisé.

-buttondownrelief **buttonDownRelief** **Relief**
Le relief à utiliser pour le bouton supérieur.

-buttonuprelief **buttonUpRelief** **Relief**
Le relief à utiliser pour le bouton inférieur.

-command **command** **Command**
Spécifie une commande Tcl à invoquer lorsqu'un bouton est pressé. La commande reconnaît plusieurs types de séquences de substitution introduites par le

Options standard supportées par les *spinbox*		
-activebackground	activeBackground	Foreground
-background ou bg	background	Background
-borderwidth ou bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-exportselection	exportSelection	ExportSelection
-font	font	Font
-foreground ou fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-insertbackground	insertBackground	Foreground
-insertborderwidth	insertBorderWidth	BorderWidth
-insertofftime	insertOffTime	OffTime
-insertontime	insertOnTime	OnTime
-insertwidth	insertWidth	InsertWidth
-justify	justify	Justify
-relief	relief	Relief
-repeatdelay	repeatDelay	RepeatDelay
-repeatinterval	repeatInterval	RepeatInterval
-selectbackground	selectBackground	Foreground
-selectborderwidth	selectBorderWidth	BorderWidth
-selectforeground	selectForeground	Background
-takefocus	takeFocus	TakeFocus
-textvariable	textVariable	Variable
-xscrollcommand	xScrollCommand	ScrollCommand
Options spécifiques communes aux *entry* et aux *spinbox*		
-disabledbackground	disabledBackground	DisabledBackground
-disabledforeground	disabledForeground	DisabledForeground
-invalidcommand ou invcmd	invalidCommand	InvalidCommand
-readonlybackground	readonlyBackground	ReadonlyBackground
-state	state	State
-validate	validate	Validate
-validatecommand ou vcmd	validateCommand	ValidateCommand
-width	width	Width
Options spécifiques aux *spinbox*		
-buttonbackground	buttonBackground	Background
-buttoncursor	buttonCursor	Cursor
-buttondownrelief	buttonDownRelief	Relief
-buttonuprelief	buttonUpRelief	Relief
-command	command	Command
-format	format	Format
-from	from	From
-increment	increment	Increment
-to	to	To
-values	values	Values
-wrap	wrap	Wrap

TAB. 16 – *Options reconnues par les composants de type* spinbox.

symbole % : *%W* pour le chemin du composant, *%s* pour la valeur courante du composant et *%d* pour la direction du bouton pressé (*up* ou *down*). On se reportera à la référence de la commande *entry* p. 67 pour plus de détails sur ces séquences de substitution.

-format **format** **Format**

Spécifie un format alternatif à utiliser lorsque la valeur de la chaîne est fixée et qu'un intervalle est spécifié avec les options **-from** et **-to**. Cela doit être un spécificateur de format de la forme **%¡m¿.¡n¿f**, avec *m* et *n* entiers, car il doit formater un nombre en virgule flottante (par exemple %4.3f). En l'absence de format, les nombres sont affichés en valeurs entières.

-from **from** **From**

Une valeur en virgule flottante correspondant à la plus petite valeur de la *spinbox*. Cette option est à utiliser en conjonction avec les options **-to** et **-increment**. Si ces trois options sont spécifiées correctement, la *spinbox* les utilisera pour contrôler son contenu. La valeur de **-from** doit être inférieure à celle de **-to**. Si l'option **-values** est spécifiée, elle supplante celle-ci.

-increment **increment** **Increment**

Une valeur en virgule flottante spécifiant l'incrément. Utilisée conjointement avec les options **-from** et **-to**, la valeur du champ de saisie sera augmentée ou diminuée de la valeur de cet incrément lorsqu'un des boutons est utilisé (le bouton *up* pour augmenter et le bouton *down* pour diminuer).

-to **to** **To**

Une valeur en virgule flottante correspondant à la plus grande valeur de la *spinbox*. Cette option est à utiliser en conjonction avec les options **-from** et **-increment**. Si ces trois options sont spécifiées correctement, la *spinbox* les utilisera pour contrôler son contenu. La valeur de **-to** doit être supérieure à celle de **-from**. Si l'option **-values** est spécifiée, elle supplante celle-ci.

-values **values** **Values**

Cette option doit être une liste Tcl valide. Si elle est spécifiée, la *spinbox* utilisera les éléments de la liste pour contrôler le contenu du champ de saisie en partant du premier élément. Cette option supplante d'éventuelles options **-from** et **-to**.

-wrap **wrap** **Wrap**

Cette option doit être une valeur booléenne. Si elle est fixée à *true*, la spinbox opérera des coupures de lignes si nécessaire.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les champs de saisie de type *spinbox* avec des caractéristiques en tous points identiques à celles des champs de saisie de type *entry* décrites page 73.

text

Permet de créer et de manipuler des objets de texte.

Syntaxe

text *cheminTexte ?options?*

Description

La commande **text** crée une nouvelle fenêtre (indiquée par l'argument *cheminTexte*) et en fait un composant graphique de type *texte*. On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques telles que la couleur de fond ou le relief etc. La commande **text** renvoie la valeur de son argument *cheminTexte*.

Un objet texte affiche une ou plusieurs lignes de texte et permet d'éditer ce texte. Les objets textes supportent quatre types différents d'annotations sur le texte, appelées respectivement balises, marques, fenêtres insérées et images insérées :

- les balises permettent d'afficher différentes parties du texte avec des attributs variés. En outre, il est possible d'associer des commandes ou des procédures Tcl à des balises de telle sorte que certaines actions se déclenchent lorsqu'un événement (clic de souris, entrée de clavier) se produit dans une certaine portion du texte. Les balises sont décrites en détail à la page 197.
- les marques sont des marqueurs flottants que l'on place dans le texte afin de garder la trace de l'emplacement de certains éléments au fur et à mesure que le texte est modifié. Les marques sont décrites en détail à la page 201.
- les fenêtres insérées dans le texte sont décrites en détail à la page 202.
- les images insérées dans le texte sont décrites en détail à la page 203.

Indices

De nombreuses commandes pour les objets textes utilisent des indices comme arguments. Un indice est une chaîne servant à désigner une position particulière à l'intérieur d'un texte. Les indices ont la syntaxe suivante :

base modificateur modificateur modificateur ...

où *base* désigne le point de départ et les arguments *modificateur* ajustent l'indice à partir de ce point de départ (par exemple désignent un déplacement vers l'avant ou vers l'arrière dans le texte). L'argument *base* est obligatoire tandis que les modificateurs sont optionnels. La *base* pour un indice peut prendre une des formes suivantes :

ligne.*n*
: indique le *n*-ième caractère sur la ligne *ligne*. Les lignes sont numérotées à

partir de 1. Dans une ligne, les caractères sont numérotés à partir de 0. Si l'argument *caractère* est **end**, il fait référence au caractère de saut de ligne placé à la fin de cette ligne.

@*x*,*y*
: indique le caractère qui couvre le pixel de coordonnées x et y dans la fenêtre du texte.

end
: indique la fin du texte (le caractère immédiatement après le dernier saut de ligne).

mark
: indique le caractère immédiatement après la marque dont le nom est *marque*.

*balise***.first**
: indique le premier caractère dans le texte qui porte la balise *balise*. Cette forme génère une erreur si aucun caractère n'a actuellement la balise *balise*.

*balise***.last**
: indique le caractère immédiatement après le dernier dans le texte qui porte la balise *balise*. Cette forme génère une erreur si aucun caractère n'a actuellement la balise *balise*.

cheminTexte
: indique la position de la fenêtre imbriquée dont le nom est *cheminTexte*. Cette forme génère une erreur s'il n'existe aucune fenêtre imbriquée de ce nom.

nomImage
: indique la position de l'image imbriquée dont le nom est *nomImage*. Cette forme génère un erreur s'il n'existe aucune image imbriquée de ce nom.

Si la *base* correspond à plusieurs des formes décrites ci-dessus, comme par exemple si une marque porte le même nom qu'une image, la première forme rencontrée dans la liste est retenue. Si des modificateurs suivent l'indice de base, chacun d'eux doit avoir une des formes indiquées dans la liste qui suit. Des mots-clés tels que **chars** et **wordend** peuvent être abrégés tant que cela ne conduit pas à des ambiguïtés. Dans ce qui suit, un « mot » est toute série contiguë de lettres, chiffres ou caractères de soulignement (_) ou n'importe quel caractère isolé.

\+ *nb chars*
: ajuste l'indice vers l'avant de *nb* caractères, en allant sur les lignes suivantes si nécessaire. S'il reste moins de *nb* caractères dans le texte, l'indice désigne alors le dernier caractère. Les espaces autour de la valeur *nb* sont optionnels.

\- *nb chars*
: Ajuste l'indice vers l'arrière de *nb* caractères, en remontant sur les lignes précédentes si nécessaire. S'il y a moins de *nb* caractères dans le texte qui précède la base, l'indice désigne alors le premier caractère. Les espaces autour de la valeur *nb* sont optionnels.

\+ *nb lines*
: Ajuste l'indice vers l'avant de *nb* lignes tout en gardant la même position à

l'intérieur de la ligne. S'il reste moins de *nb* lignes après la ligne contenant l'indice courant, l'indice se réfère au caractère de même position sur la dernière ligne du texte. Si une ligne n'est pas assez longue pour contenir un caractère à la position souhaitée, la position est ajustée à celle du dernier caractère de la nouvelle ligne. Les espaces autour de la valeur *nb* sont optionnels.

- *nb lines*
Ajuste l'indice vers l'arrière de *nb* lignes tout en gardant la même position à l'intérieur de la ligne. S'il reste moins de *nb* lignes avant la ligne contenant l'indice courant, l'indice se réfère au caractère de même position sur la première ligne du texte. Si une ligne n'est pas assez longue pour contenir un caractère à la position souhaitée, la position est ajustée à celle du dernier caractère de la nouvelle ligne. Les espaces autour de la valeur *nb* sont optionnels.

linestart
Ajuste l'indice sur le premier caractère de la ligne.

lineend
Ajuste l'indice sur le dernier caractère de la ligne.

wordstart
Ajuste l'indice sur le premier caractère du mot contenant l'indice courant.

wordend
Ajuste l'indice sur le dernier caractère du mot contenant l'indice courant.

Si plus d'un modificateur est présent, ils sont appliqués dans l'ordre de gauche à droite. Par exemple, la séquence :

end - 1 char

se réfère à l'avant-dernier caractère du texte tandis qu'une instruction

```
insert wordstart - 1 c
```

se réfère au caractère qui précède le premier caractère du mot contenant la position courante.

Balises

Une balise est une chaîne textuelle que l'on associe à certains caractères du texte. Elles peuvent avoir n'importe quel nom mais il est préférable d'éviter d'utiliser des espaces ou des signes + ou − pour éviter des ambiguïtés avec les indices. Il peut y avoir un nombre quelconque de balises associées à des caractères dans le texte. Une balise peut se référer à un caractère unique, ou à une ou plusieurs séquences de caractères successifs. Un caractère individuel peut posséder un nombre quelconque de balises.

Un ordre de priorité est défini parmi les balises et cet ordre est utilisé par certaines opérations. Lorsqu'une balise est définie, elle reçoit *a priori* un ordre de priorité supérieur à celui des balises qui existent déjà. On peut par la suite modifier cet ordre au moyen des opérations **tag raise** ou **tag lower**.

Les balises ont trois fonctions :

– elles contrôlent la façon dont l'information est affichée à l'écran. Par défaut

les caractères sont affichés selon les valeurs passées aux options **-background**, **-font** et **-foreground**.
Des options peuvent cependant être associées à des balises individuelles au moyen d'une commande :
cheminTexte **tag configure**
Si un caractère possède une balise, les options associées supplantent les réglages de style par défaut.
Si un caractère a plusieurs balises associées, et si leurs options d'affichage ne sont pas contradictoires, ce sont celles de la balise ayant le plus haut degré de priorité qui sont utilisées. Si une option particulière n'a pas été spécifiée pour une certaine balise, ou si elle est spécifiée comme une chaîne vide, cette option ne sera jamais utilisée. C'est celle de la balise suivante dans l'ordre de priorité qui sera utilisée à la place. Si aucune balise n'est spécifiée pour une certaine option d'affichage, les réglages par défaut du composant graphique sont utilisés.

– elles servent aux liaisons d'événements. On peut associer des liaisons à des balises de la même façon qu'avec des composants graphiques. Dès qu'un événement particulier se produit sur un caractère possédant cette balise une commande Tcl qui aura été déclarée avec la liaison est exécutée. Les liaisons sur des balises peuvent servir à créer des comportements particuliers sur certaines séquences de caractères comme, par exemple, en faire des hyperliens à la manière des navigateurs Internet.
– elles servent enfin à manipuler les sélections de texte. Cet usage est expliqué en détail à la section *Sélection*, page 204.

Attributs des balises

Les options suivantes sont actuellement supportées pour fixer les attributs des balises :

-background *couleur*
couleur spécifie la couleur de fond à utiliser pour les caractères associés à une balise. Il peut prendre une quelconque des formes acceptées pour la désignation des couleurs.

-bgstipple *bitmap*
L'argument *bitmap* spécifie une image bichrome (*bitmap* dans la terminologie de Tk) qui est utilisée comme motif grisé pour l'arrière-plan. Il peut prendre une quelconque des formes acceptées pour la désignation des bitmaps. Si *bitmap* n'a pas été spécifié, ou s'il est spécifié comme une chaîne vide, l'arrière-plan est rempli en couleur unie.

-borderwidth *pixels*
L'argument *Pixels* spécifie la largeur d'une bordure simulant un effet de trois dimensions à dessiner autour de l'arrière-plan. Elle est exprimée en unités d'écran. Cette option est utilisée en conjonction avec l'option **-relief** pour créer une apparence de trois dimensions à l'arrière-plan des caractères. L'option est ignorée à moins que l'option **-background** n'ait été établie pour cette balise.

-elide *bool*
L'option **-elide** spécifie si les données doivent être « élidées ». Une donnée élidée n'est pas affichée et n'occupe aucun espace sur l'écran mais, pour le reste, se comporte comme des données ordinaires.

-fgstipple *bitmap*
L'argument *bitmap* spécifie une image bichrome (*bitmap* dans la terminologie de Tk) qui est utilisée comme motif pour dessiner le texte et tous les éléments de premier plan comme les soulignements. Il peut prendre une quelconque des formes acceptées pour la désignation des bitmaps. Si *bitmap* n'a pas été spécifié, ou s'il est spécifié comme une chaîne vide, les éléments concernés sont remplis en couleur unie.

-font *nomPolice*
L'argument *nomPolice* est le nom d'une police à utiliser pour dessiner les caractères. Il peut prendre une quelconque des formes acceptées pour la désignation d'une police.

-foreground *couleur*
L'argument *couleur* spécifie la couleur à utiliser pour dessiner le texte et d'autres éléments qui viennent au premier plan comme les soulignements. Il peut prendre une quelconque des formes acceptées pour la désignation des couleurs.

-justify *justif*
Si le premier caractère d'une ligne affichée possède une balise pour laquelle cette option a été spécifiée, *justif* détermine quelle justification appliquer à cette ligne. L'argument peut prendre une des valeurs **left**, **right** ou **center**. Si une ligne doit être coupée, la justification pour chaque ligne sur l'écran est déterminée par le premier caractère de cette ligne.

-lmargin1 *pixels*
Si le premier caractère d'une ligne de texte possède une balise pour laquelle cette option a été spécifiée, l'argument *pixels* spécifie de combien la ligne devrait être indentée à partir du bord gauche de la fenêtre. L'argument *pixels* peut prendre n'importe quelle forme standard représentant les distances d'écran. Si une ligne de texte doit être coupée, cette option s'applique seulement à la première ligne ; l'option **-lmargin2** contrôle l'indentation des lignes suivantes.

-lmargin2 *pixels*
Si le premier caractère d'une ligne de texte possède une balise pour laquelle cette option a été spécifiée et si cette ligne n'est pas la première (ce qui signifie qu'il y a eu des coupures d'une ligne logique en plusieurs lignes physiques) alors l'argument *pixels* spécifie de combien la ligne devrait être indentée à partir du bord gauche de la fenêtre. L'argument *pixels* peut prendre n'importe quelle forme standard représentant les distances d'écran. Cette option n'est utilisée que si les coupures de lignes ont été autorisées et s'applique seulement à partir de la seconde ligne de texte et pour toutes les suivantes.

-offset *pixels*
L'argument *pixels* spécifie la quantité de pixels dont la ligne de base du texte devrait être décalée verticalement par rapport à la ligne de base globale de la ligne. Par exemple, un décalage positif créera des exposants et un déclagae négatif des indices. L'argument *pixels* peut prendre n'importe quelle forme standard représentant les distances d'écran.

-overstrike *bool*
Spécifie le tracé ou non d'une ligne horizontale traversant les caractères en leur milieu. L'argument *bool* peut prendre une quelconque des formes acceptées pour les valeurs booléennes.

-relief *relief*
L'argument *relief* spécifie un relief en trois dimensions à utiliser pour dessiner l'arrière-plan. Cette options peut prendre une des valeurs *flat*, *raised*, *sunken*, *groove*, *solid* ou *ridge*. Cette option est utilisée en conjonction avec l'option **-borderwidth** afin de donner une apparence tridimensionnelle à l'arrière-plan des caractères. Elle est ignorée si l'option **-background** n'a pas elle-même été établie pour la balise.

-rmargin *pixels*
Si le premier caractère d'une ligne de texte possède une balise pour laquelle cette option a été spécifiée, l'argument *pixels* spécifie la largeur d'une marge à laisser entre la fin de la ligne et le bord droit de la fenêtre. L'argument *pixels* peut prendre n'importe quelle forme standard représentant les distances d'écran. Si une ligne subit une coupure, la marge de droite de chaque ligne est déterminée par le premier caractère de cette ligne.

-spacing1 *pixels*
L'argument *pixels* spécifie quelle quantité d'espace additionnel laisser au-dessus de chaque ligne, en utilisant une quelconque des formes stantards représentant les distances d'écran. Si une ligne subit une coupure, cette option s'applique uniquement à la première ligne affichée.

-spacing2 *pixels*
Pour les lignes subissant une coupure, cette option spécifie quelle quantité d'espace additionnel laisser entre chaque ligne. L'argument *pixels* peut prendre n'importe quelle forme standard représentant les distances d'écran.

-spacing3 *pixels*
L'argument *pixels* spécifie quelle quantité d'espace additionnel laisser en dessous de chaque ligne, en utilisant une quelconque des formes stantards représentant les distances d'écran. Si une ligne subit une coupure, cette option s'applique uniquement à la dernière.

-tabs *liste*
L'argument *liste* spécifie un ensemble de tabulations sous la même forme que pour l'option **-tabs** des objets textes. Cette option s'applique à une ligne affichée seulement si elle s'applique au premier caractère de cette ligne. Si cette option est spécifiée comme une chaîne vide, elle a un effet d'annulation, laissant les tabulations non spécifiées (situation par défaut). Si l'option

est spécifiée comme une chaîne non-vide qui est une liste vide, comme dans l'instruction

```
-tags {}
```

alors elle installe des tabulations par défaut de huit caractères d'espacement.

-underline *bool*
L'argument *bool* spécifie le tracé ou non d'une ligne pour souligner les caractères. Il peut prendre une quelconque des formes acceptées pour les valeurs booléennes.

-wrap *mode*
L'argument *mode* spécifie la manière de traiter les lignes qui sont plus larges que la largeur de la fenêtre. Elle peut prendre une des valeurs symboliques suivantes comme pour l'option **-wrap** des objets textes eux-mêmes : *none*, *char* ou *word*. Si cette option de balise est spécifiée, elle supplante l'option d'objet texte de même nom.

Marques

Les marques sont la seconde forme d'annotation dans les objets textes. Les marques sont utilisées pour mémoriser des emplacements particuliers dans le texte.

Elles sont analogues aux balises, en ce sens qu'elles ont un nom et se réfèrent à des emplacements dans le fichier mais elles ne sont pas associées à des caractères particuliers. Une marque est en fait associée à l'interstice entre deux caractères. Une seule position peut être associée à une marque à un instant donné.

Si les caractères autour de la marque sont détruits, celle-ci persiste néanmoins. Ce sont seulement les caractères qui l'entourent qui auront changé. C'est une autre différence avec les balises : si des caractères possédant une balise sont détruits, la balise ne sera plus associée à des caractères dans le fichier.

La manipulation des marques se fait au moyen d'une instruction :
cheminTexte **mark**
et les emplacements qu'elles représentent peuvent être utilisés en se servant du *nom* de la marque dans les commandes appropriées.

Chaque marque possède aussi une *gravité*, qui est soit *left* soit *right*. La *gravité* d'une marque indique ce qui lui arrive lorsque du texte est inséré au point qu'elle désigne : avec une gravité à gauche, la marque est traitée comme si elle était attachée au caractère qui se trouve à sa gauche ce qui fait que la marque reste à gauche du nouveau texte inséré. Avec une gravité à droite (ce qui est la valeur par défaut), le texte inséré est placé à gauche de la marque et la marque se retrouve alors à la droite du dernier caractère inséré.

Deux marques prédéfinies par Tk ne peuvent pas être détruites et possèdent une signification particulière :

- la marque *insert* est associée au curseur d'insertion qui se trouve décrit à la section *Curseur d'insertion* page 204 ;
- la marque *current* est associée au caractère le plus proche de la souris et ajustée automatiquement pour suivre la position de la souris (si le bouton n'est pas enfoncé) et tout changement apporté au texte.

Fenêtres insérées

Les fenêtres insérées sont la troisième forme d'annotation dans les objets textes. Chaque annotation de type fenêtre imbriquée provoque l'affichage d'une fenêtre en un point particulier du texte. Il peut y avoir un nombre quelconque de fenêtres insérées et n'importe quel composant graphique peut servir de fenêtre imbriquée du moment que la fenêtre de texte soit un parent de la fenêtre imbriquée ou un descendant de sa fenêtre mère. La position de la fenêtre imbriquée sur l'écran est mise à jour au fur et à mesure que le texte est modifié ou qu'on le fait défiler.

Du point de vue des indices, une fenêtre imbriquée est considérée comme un caractère ordinaire : on peut donc s'y référer aussi bien par son nom que par l'indice de la position qu'elle occupe dans le texte. Si une portion de texte contenant une fenêtre imbriquée est détruite, la fenêtre est également détruite.

Lorsqu'une fenêtre imbriquée est ajoutée à un objet texte avec la commande **window create**, plusieurs options de configuration peuvent lui être attribuées. Ces options peuvent être modifiées par la suite avec la commande **window configure**. Les options suivantes sont actuellement supportées :

-align *positionnement*
: Si la fenêtre est moins haute que la ligne dans laquelle elle est imbriquée, cette option détermine l'emplacement de la fenêtre dans la ligne. L'argument *positionnement* doit prendre une des valeurs symboliques suivantes : *top* pour un alignement du sommet de la fenêtre avec le sommet de la ligne, *center* pour centrer la fenêtre, *bottom* pour un alignement du bas de la fenêtre avec le bas de la ligne ou *baseline* pour un alignement du bas de la fenêtre avec la ligne de base.

-create *script*
: Spécifie un script Tcl qui peut être évalué pour créer la fenêtre pour l'annotation. Si aucune option **-window** n'a été spécifiée pour l'annotation, ce script sera évalué lorsque l'annotation sera sur le point d'être affichée à l'écran. L'argument *script* doit créer une fenêtre pour l'annotation et renvoyer le nom de cette fenêtre comme valeur de retour. Si la fenêtre d'annotation doit être détruite, le *script* sera évalué à nouveau la prochaine fois que l'annotation sera affichée.

-padx *pixels*
: L'argument *pixels* spécifie la quantité d'espace supplémentaire à laisser de chaque côté de la fenêtre imbriquée. Il peut prendre une quelconque des formes standard représentant les distances d'écran.

-pady *pixels*
: L'argument *pixels* spécifie la quantité d'espace supplémentaire à laisser au-dessus et en dessous de la fenêtre imbriquée. Il peut prendre une quelconque des formes standard représentant les distances d'écran.

-stretch *bool*
: Si la hauteur requise pour une fenêtre imbriquée est inférieure à la hauteur de la ligne qui la contient, cette option permet de spécifier si la fenêtre doit être étirée verticalement pour s'ajuster sur la hauteur de la ligne. Si l'option **-pady**

a été également spécifiée, l'espace additionnel requis sera respecté même si la fenêtre est étirée.

-window *cheminTexte*
Spécifie le nom d'une fenêtre à afficher dans l'annotation.

Images insérées

Les images insérées sont la quatrième forme d'annotation dans les objets textes. Chaque annotation de type image imbriquée provoque l'affichage d'une image en un point particulier du texte. Il peut y avoir un nombre quelconque d'images insérées et une image particulière peut être imbriquée en plusieurs endroits d'un même objet texte. La position de l'image imbriquée sur l'écran est mise à jour au fur et à mesure que le texte est modifié ou qu'on le fait défiler.

Du point de vue des indices, une image imbriquée est considérée comme un caractère ordinaire : on peut donc s'y référer aussi bien par le nom qui lui a été attribué à sa création avec la commande **image create** que par l'indice de la position qu'elle occupe dans le texte. Si une portion de texte contenant une fenêtre imbriquée est détruite, la fenêtre est également détruite.

Lorsqu'une image imbriquée est ajoutée à un objet texte avec la commande **image create**, un nom unique pour cette instance de l'image est renvoyé. Ce nom peut servir par la suite pour se référer à cette instance de l'image. Ce nom est pris comme valeur de l'option **-name**. Si l'option **-name** n'est pas fournie, le nom de l'option **-image** est utilisé à la place. Si un nom *nomImage* est déjà utilisé dans un objet texte, alors une numérotation sous la forme $\#n$ est ajoutée en suffixe à *nomImage*, avec n entier arbitraire. Cela garantit que le nom est unique. Une fois le nom attribué à cette instance de l'image, il ne change plus, et cela bien que les valeurs des options **-image** ou **-name** puissent être changées avec **image configure**.

Au moment de la création d'une image imbriquée, plusieurs options de configuration peuvent lui être attribuées. Ces options peuvent être modifiées par la suite avec la commande **window configure**. Les options suivantes sont actuellement supportées :

-align *positionnement*
Si l'image est moins haute que la ligne dans laquelle elle est imbriquée, cette option détermine le positionnement de l'image dans la ligne. L'argument *positionnement* doit prendre une des valeurs symboliques suivantes : *top* pour un alignement du sommet de l'image avec le sommet de la ligne, *center* pour centrer l'image, *bottom* pour un alignement du bas de l'image avec le bas de la ligne ou *baseline* pour un alignement du bas de l'image avec la ligne de base de la ligne.

-image *image*
Spécifie le nom de l'image Tk à afficher dans l'annotation. Si *image* ne désigne pas une image Tk valide, une erreur est renvoyée.

-name *nomImage*
Spécifie le nom par lequel cette instance de l'image doit être référencée dans

l'objet texte. Si l'argument *nomImage* n'est pas fourni, le nom de l'image Tk est utilisé à la place. Si *nomImage* est déjà utilisé, une numérotation lui est ajoutée en suffixe comme expliqué précedemment.

-padx *pixels*
L'argument *pixels* spécifie la quantité d'espace supplémentaire à insérer de chaque côté de l'image imbriquée. Il peut prendre une quelconque des formes standard représentant les distances d'écran.

-pady *pixels*
L'argument *pixels* spécifie la quantité d'espace supplémentaire à insérer au-dessus et en dessous de l'image imbriquée. Il peut prendre une quelconque des formes standard représentant les distances d'écran.

Sélections de texte

Le mécanisme de sélection de texte est implémenté au moyen de balises. Si l'option **exportSelection** pour l'objet texte est vraie (*true*), la balise nommée **sel** sera associée à la sélection compte-tenu des règles suivantes :

1. dès que des caractères sont marqués avec la balise **sel** l'objet texte devient propriétaire de la sélection.
2. toute tentative pour récupérer la sélection sera accomplie par l'objet texte qui renverra tous les caractères ayant la balise **sel**.
3. Si la sélection est réclamée par une autre application ou par une autre fenêtre de l'application, la balise **sel** est retirée de tous les caractères du texte.

La balise **sel** est automatiquement définie lorsqu'un objet texte est créé et ne peut être détruite par la commande **tag delete**. D'autre part, les options **selectBackground**, **selectBorderWidth** et **selectForeground** pour l'objet texte sont liées aux options homologues **background**, **borderwidth** et **foreground** concernant la balise **sel** : les changements des unes sont immédiatement reflétées par les autres.

Curseur d'insertion

La marque nommée *insert* a une signification spéciale dans les objet textes. Elle est définie automatiquement lorsqu'un objet texte est créé et elle ne peut pas être détruite par la commande **mark unset**. Cette marque *insert* représente la position du point d'insertion et le curseur d'insertion sera automatiquement affiché à cet emplacement chaque fois que l'objet texte sera focalisé par l'application.

Commandes des objets textes

Avec l'instruction :
text *cheminTexte ?options?*
la commande **text** crée une nouvelle commande Tcl dont le nom est précisément

cheminTexte. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant l'objet texte. La forme générale est :

cheminTexte opération ?arg arg ...?

Les opérations reconnues par les objets textes sont les suivantes :

cheminTexte **bbox** *index*
: Renvoie une liste de quatre nombres décrivant la zone d'écran occupée par le caractère d'indice *index*. Les deux premiers éléments de cette liste sont l'abscisse et l'ordonnée (mesurées en pixels relativement à la boîte elle-même) du coin supérieur gauche de la zone d'écran occupée par le caractère et les deux derniers sont la largeur et la hauteur du caractère. Si le caractère est seulement partiellement visible à l'écran, la valeur renvoyée ne concerne que la partie visible. Si le caractère n'est pas visible à l'écran, la valeur renvoyée est une liste vide.

cheminTexte **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **text**.

cheminTexte **compare** *index1 op index2*
: Compare les indices spécifiés par *index1* et *index2* selon l'opérateur relationnel *op* et renvoie 1 si la relation est satisfaite ou sinon 0. L'argument *op* désigne l'un des opérateurs $<$, $\leq$, $==$, $\geq$, $>$, ou $!=$. Si l'opérateur *op* est $==$, la valeur 1 est renvoyée si les deux indices se réfèrent au même caractère. Si l'opérateur *op* est $<$, la valeur 1 est renvoyée si *index1* se réfère à un caractère du texte placé avant l'indice *index2* et ainsi de suite.

cheminTexte **configure** *?option? ?valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration du texte. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminTexte*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **text**.

cheminTexte **debug** *?bool?*
: Si une valeur *bool* est spécifiée, elle doit prendre une des formes acceptées pour les valeurs booléennes. Si la valeur correspond à *true*, le programme mettra en place des procédures de contrôle interne pour vérifier la cohérence de l'arbre binaire du code associé à l'objet texte. Dans le cas contraire, ces vérifications seront désactivées. Dans tous les cas, la commande renvoie une chaîne vide.
Si l'argument *bool* n'est pas spécifié, la commande renvoie *on* ou *off* indiquant si oui ou non le débogage a été installé. Pour des objets contenant de grandes

quantités de texte, les vérifications de cohérence peuvent sensiblement ralentir l'application.

cheminTexte **delete** *index1 ?index2?*
Détruit une série de caractères du texte. Si à la fois *index1* et *index2* sont spécifiés, la commande détruit tous les caractères depuis celui qui se trouve à l'indice *index1* jusqu'à la position de l'indice *index2*, autrement dit le caractère d'indice *index2* n'est pas détruit. Si *index2* ne désigne pas une position ci-dessous dans le texte que celle d'indice *index1*, aucun caractère n'est détruit. Si *index2* n'est pas spécifié seul le caractère d'indice *index1* est détruit. Il n'est pas permis de procéder à des suppressions de caractères qui laisseraient la dernière ligne sans un caractère de saut de ligne. La commande renvoie une chaîne vide.

cheminTexte **dlineinfo** *index*
Renvoie une liste de cinq éléments décrivant la zone occupée par la ligne affichée contenant l'indice *index*. Les deux premiers éléments de la liste donnent les coordonnées x et y du coin supérieur gauche de la zone occupée par la ligne, les troisième et quatrième correspondent à la largeur et la hauteur de la zone et le cinquième indique la position de la ligne de base de cette ligne de texte, mesurée vers le bas à partir du sommet de la zone. Toutes ces valeurs sont mesurées en pixels.
Si le mode de coupures de lignes est *none* et que la ligne s'étend au-delà des limites de la fenêtre, la zone renvoyée reflète la zone entière, incluant les parties non visibles. Si la ligne est plus courte que la largeur de la fenêtre, la zone renvoyée est juste la portion de la ligne occupée par les caractères (et d'éventuelles fenêtres insérées dans le texte).
Si la ligne contenant l'indice *index* n'est pas visible à l'écran, la valeur de retour est une liste vide.

cheminTexte **dump** *?options? index1 ?index2??*
Renvoie le contenu de l'objet texte de l'indice *index1* jusqu'à l'indice *index2* exclu, avec à la fois le texte lui-même et les informations concernant les balises, les marques et les fenêtres insérées. Si *index2* n'est pas spécifié, il désigne par défaut le caractère après celui d'indice *index1*. L'information est renvoyée sous la forme suivante :

clé1 valeur1 index1 clé2 valeur2 index2 ...

Les valeurs de *clé* possibles sont *text*, *mark*, **tagon**, *tagoff* et *window*. L'argument *valeur* correspondant est le texte, un nom de marque, un nom de balise ou un nom de fenêtre. L'argument *index* est l'indice de départ du texte, de la marque, de la balise ou de la fenêtre.
Une ou plusieurs options supplémentaires peuvent être spécifiées pour contrôler la sortie produite par la commande **dump** :

-all
Renvoie l'information concernant *tous* les éléments : texte, marques, balises, images et fenêtres. C'est la situation par défaut.

-command *cmd*
Au lieu de renvoyer l'information comme résultat de l'opération de sortie, invoque la commande *cmd* sur chaque élément de l'objet texte dans la séquence concernée. La commande a trois arguments qui lui sont ajoutés avant d'être évaluée : *clé*, *valeur* et *index*.

-image
Inclut l'information sur les images dans les résultats.

-mark
Inclut l'information sur les marques dans les résultats.

-tag
Inclut l'information sur les transitions de balises dans les résultats. Cette information est renvoyée comme des éléments *tagon* et *tagoff* qui indiquent le début et la fin des intervalles couverts par chaque balise respectivement.

-text
Inclut l'information sur le texte dans les résultats. La valeur est le texte lui-même jusqu'à l'élément suivant ou la fin de la séquence indiquée par l'indice *index2*. Un élément texte ne couvre pas les fins de lignes. Un bloc de texte de plusieurs lignes qui ne contient ni marques, ni balises, sera quand même renvoyé sous la forme de segments de textes terminés chacun par un caractère de fin de ligne. Le symbole de fin de ligne fait partie de la valeur.

-window
Inclut, dans les résultats, l'information concernant les fenêtres insérées. La valeur d'une fenêtre est son chemin dans Tk, à moins que la fenêtre n'ait pas encore été créée. Dans ce cas, une chaîne vide est renvoyée, et il faut interroger la fenêtre par son indice de position pour obtenir davantage d'information.

cheminTexte **get** *index1 ?index2?*
Renvoie une séquence de caractères du texte. La valeur de retour est faite de tous les caractères du texte partant de celui d'indice *index1* et allant jusqu'avant celui d'indice *index2* (le caractère placé à l'indice *index2* n'est donc pas renvoyé). Si *index2* est omis, seul le caractère d'indice *index1* est renvoyé. S'il n'y a pas de caractères dans l'intervalle spécifié (par exemple si *index1* est au-delà de la fin du fichier ou si *index2* est inférieur ou égal à *index1*), une chaîne vide est renvoyée. Si l'intervalle spécifié contient des fenêtres insérées, aucune information les concernant n'est incluse dans le résultat.

cheminTexte **image** *opération ?arg arg ...?*
Cette commande est utilisée pour manipuler les images insérées. Le comportement de la commande dépend de l'argument *opération* qui suit l'argument **image**. Les opérations suivantes sont actuellement supportées :

cheminTexte **image cget** *index option*
Renvoie la valeur courante de l'option de configuration désignée par

l'argument *option*. Cet argument peut prendre une quelconque des valeurs admises par les images insérées (voir à la section *Images insérées*, page 203).

cheminTexte **image configure** *index ?option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration pour une image imbriquée. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminMenu*. Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. Cet argument peut prendre uniquement les valeurs admises par les images insérées : voir à la section *Images insérées*, page 203.

cheminTexte **image create** *index ?option valeur ...?*
Cette commande crée une nouvelle annotation de type image, qui apparaîtra dans le texte à la position spécifiée par l'indice *index*. On peut préciser un nombre quelconque de paires *option-valeur* afin de configurer l'annotation. Renvoie un identificateur unique qui peut être utilisé comme référence à cette image. On se reportera à la section *Images insérées*, page 203, pour connaître les options supportées.

cheminTexte **image names**
Renvoie une liste dont les éléments sont les noms de toutes les instances d'images actuellement insérées dans la *fenêtre*.

cheminTexte **index** *index*
Renvoie la position correspondant à l'indice *index* sous la forme *ligne.car* où *ligne* est le numéro de la ligne et *car* est l'indice du caractère dans la ligne. L'indice peut prendre une quelconque des formes décrites à la section *Indices*, page 195.

cheminTexte **insert** *index caractères ?listeBalise chars listeBalise ...?*
Insère tous les arguments *caractères* immédiatement après le caractère placé à l'indice *index*. Si *index* se réfère à la fin du texte (le caractère après le dernier symbole de fin de ligne), le texte est inséré en fait immédiatement avant le dernier symbole de fin de ligne. Si l'argument *caractères* est unique et qu'il n'y a aucun argument *listeBalise*, le texte se verra affecter toute balise qui serait présente à la fois sur le caractère qui précède et sur celui qui suit le point d'insertion. Si une balise est présente sur seulement un de ces caractères, elle ne sera pas appliquée au texte.

Si un argument *listeBalise* est spécifié, il consiste en une liste de noms de balises. Les nouveaux caractères insérés recevront toutes les balises de cette liste à l'exclusion de toute autre qui pourrait être présente autour du point d'insertion.

Si plusieurs paires d'arguments *caractères-listeBalise* sont présentes, elles produisent le même effet que si autant d'instructions **insert** étaient invoquées pour chacune de ces paires dans l'ordre. Le dernier argument *listeBalise* peut

être omis.

cheminTexte **mark** *option ?arg arg ...?*

Cette commande est utilisée pour manipuler les marques. Le comportement exact de la commande dépend de l'argument *option* qui suit le mot **mark**. Les formes suivantes de la commande **mark** sont actuellement supportées :

cheminTexte **mark gravity** *nomMarque ?direction?*

Si l'argument *direction* n'est pas spécifié, la commande renvoie *left* ou *right* pour indiquer à quel caractère adjacent *nomMarque* est attaché. Si *direction* est spécifié, il doit prendre une des valeurs symboliques *left* ou *right*, ce qui fixe la *gravité* à cette valeur (cf. p. 201).

cheminTexte **mark names**

Renvoie une liste dont les éléments sont les noms de toutes les marques qui sont actuellement installées.

cheminTexte **mark next** *index*

Renvoie le nom de la prochaine marque placée à l'indice *index* ou après. Si *index* est spécifié sous forme numérique, la recherche de la prochaine marque commence à cet indice. Si *index* est le nom d'une marque, la recherche commence immédiatement après cette marque. Cette opération peut renvoyer une marque à la même position étant donné que plusieurs marques peuvent figurer à un même indice : on peut donc utiliser la commande **mark next** pour parcourir toutes les marques d'un objet texte dans le même ordre que celui qui est renvoyé par la commande **dump**. Si une marque a été installée à l'indice *end*, elle apparaît après *end* du point de vue de l'opération **mark next**. Une chaîne vide est renvoyée s'il ne se trouve aucune marque après l'indice *index*.

cheminTexte **mark previous** *index*

Renvoie le nom de la marque placée à l'indice *index* ou avant. Si l'indice *index* est spécifié sous forme numérique, la recherche de la prochaine marque commence avec le caractère immédiatement avant cet indice. Si *index* est le nom d'une marque, la recherche commence immédiatement avant cette marque. Cette opération peut renvoyer une marque à la même position étant donné que plusieurs marques peuvent figurer à un même indice : on peut donc utiliser la commande **mark previous** pour parcourir toutes les marques d'un objet texte dans l'ordre inverse de celui qui est renvoyé par la commande **dump**. Une chaîne vide est renvoyée s'il ne se trouve aucune marque après l'indice *index*.

cheminTexte **mark set** *nomMarque index*

Établit une marque nommée *nomMarque* à la position immédiatement avant le caractère d'indice *index*. Si *nomMarque* existe déjà, elle est déplacée depuis son ancienne position, sinon une nouvelle marque est créée. Cette commande renvoie une chaîne vide.

cheminTexte **mark unset** *nomMarque ?nomMarque nomMarque ...?*

Supprime les marques correspondant à chacun des arguments *nomMarque*. Cette commande renvoie une chaîne vide.

cheminTexte **scan** *option args*
Cette commande est utilisée pour implémenter des fonctions permettant de parcourir rapidement un objet texte. Elle peut prendre deux formes selon la valeur de l'argument *option* :

cheminTexte **scan mark** *x y*
Enregistre les valeurs *x* et *y* et la vue courante de la fenêtre de texte pour une utilisation ultérieure avec une commande **scan dragto**. Cette commande est typiquement associée à une pression du bouton de la souris dans la liste. Elle renvoie une chaîne vide.

cheminTexte **scan dragto***x y*
Cette commande calcule la différence entre les valeurs des arguments *x* et *y* et celles de ces mêmes arguments lors de la dernière commande **scan mark** exécutée pour cet objet texte. Elle renvoie une chaîne vide. La vue du contenu de la fenêtre est alors ajustée de 10 fois la valeur de cette différence. Cette commande est typiquement utilisée pour simuler l'effet de déplacement rapide du texte à travers la fenêtre avec la souris. On associe fréquemment les deux commandes **scan mark** et **scan dragto** au second bouton de la souris comme ceci :

```
bind $c <2> "$c scan mark %x %y"
bind $c <B2-Motion> "$c scan dragto %x %y"
```

cheminTexte **search** *?options? motif index ?indexArrêt?*
Effectue une recherche dans le texte *cheminTexte* en partant de l'indice *index* : la commande recherche une séquence de caractères correspondant au motif *motif*. Si une correspondance est trouvée, l'indice du premier caractère de celle-ci est renvoyé comme résultat ; autrement une chaîne vide est renvoyée. On peut, en outre, préciser une ou plusieurs des options suivantes afin de contrôler le mode de recherche :

-forwards
La recherche se fait vers l'avant dans le texte et pourra trouver une correspondance à partir de la position spécifiée par *index* ou après. C'est la valeur par défaut.

-backwards
La recherche se fait vers l'arrière dans le texte et pourra trouver la correspondance la plus proche de la position spécifiée par *index* et dont le premier caractère est avant cet indice.

-exact
Les caractères d'une séquence en correspondance avec le motif doivent être rigoureusement identiques à ceux du motif lui-même. C'est la valeur par défaut.

-regexp
Traite l'argument *motif* comme une expression régulière et le compare

au texte en utilisant les règles des expressions régulières (voir au chapitre *Les expressions régulières*).

-nocase
Ignore les différences de casse (majuscule ou minuscule) entre le motif et le texte.

-count *nomVar*
L'argument qui suit la commande **-count** indique le nom d'une variable ; si une correspondance est trouvée, le nombre de positions d'indice entre le début et la fin de la correspondance sera stocké dans cette variable. S'il n'y a pas de fenêtres ou d'images insérées dans la chaîne trouvée ce nombre est équivalent au nombre des caractères trouvés. Dans tous les cas, l'intervalle allant de l'indice trouvé à cet indice augmenté de la valeur de **-count** correspondra intégralement au texte trouvé.

-elide
Permet de trouver même du texte qui a été caché au moyen de l'option de balises **-elide**.

--
Le double tiret indique comme toujours la fin des options sur une ligne de commandes.

La séquence en correspondance doit être entièrement dans une seule ligne de texte. Dans le cas des expressions régulières cependant les symboles de fin de ligne ne sont par pris en compte pendant la recherche : on utilisera le méta-caractère $ des expressions régulières si l'on veut désigner la fin d'une ligne. Dans une recherche exacte au contraire, les fins de lignes sont conservées.
Si l'argument *indexArrêt* est spécifié, la recherche s'arrêtera à cet indice : pour des recherches en avant, aucune correspondance à cet indice ou au-delà ne sera considérée ; pour des recherches en arrière, aucune correspondance avant cet indice ne sera considérée. Si l'argument *indexArrêt* est omis, la recherche s'opère dans le texte entier : lorsqu'une extrêmité du texte est atteinte, la recherche reprend à l'autre extrêmité et se poursuit jusqu'à l'indice de départ.

cheminTexte **see** *index*
Ajuste la vue dans la fenêtre de telle sorte que le caractère d'indice *index* soit complètement visible. Si l'argument *index* est déjà visible, rien ne se produit. S'il est à courte distance, l'ajustement sera juste ce qu'il faut pour le rendre visible. S'il est loin, il se retrouvera centré dans la fenêtre.

cheminTexte **tag** *option ?arg arg ...?*
Cette commande est utilisée pour manipuler les balises. Le comportement exact de la commande dépend de l'argument *option* qui suit le mot **tag**. Les formes suivantes de la commande **tag** sont actuellement supportées :

cheminTexte **tag add** *nomBalise index1 ?index2 index1 index2 ...?*
Associe la balise *nomBalise* à tous les caractères depuis l'indice *index1* jusqu'au caractère immédiatement avant l'indice *index2* (le caractère d'indice *index2* n'est pas touché). Une commande peut contenir un

nombre quelconque de paires d'indices *index1-index2*. Si le dernier indice *index2* est omis, seul le caractère placé à l'indice *index1* reçoit une balise. S'il n'y a pas de caractères dans l'intervalle spécifié (par exemple si *index1* se trouve après *index2* ou après la fin du fichier), la commande est sans effet.

cheminTexte **tag bind** *nomBalise ?sequence? ?script?*

Cette commande associe le script *script* à la balise désignée par *nomBalise* de telle sorte que dès que la suite d'événements représentée par l'argument *séquence* se produit, la commande soit invoquée. Cette opération est analogue à la commande **bind** à la différence qu'elle agit sur les caractères d'un texte plutôt que sur un composant graphique entier. Les règles régissant les arguments *séquence* et *commande* sont les mêmes que pour **bind** (cf. p. 13).

Si tous les arguments sont spécifiés, une nouvelle liaison est créée remplaçant toute liaison déjà existante pour cette suite d'événements et cet élément *nomBalise* : toutefois, si le premier caractère de l'argument *commande* est un signe +, l'argument s'ajoute à la définition de la liaison au lieu de la remplacer. Dans ce cas, la valeur de retour est une chaîne vide.

Si l'argument *script* est omis, l'opération **bind** renvoie la définition du script *script* associé à *nomBalise* et à la *séquence*. Si à la fois *commande* et *séquence* sont omis, l'opération renvoie la liste de toutes les séquences pour lesquelles des liaisons ont été définies concernant l'élément *nomBalise*.

Les seuls événements acceptant des liaisons sont ceux relatifs à la souris et au clavier comme *Enter*, *Leave*, *ButtonPress*, *Motion* et *KeyPress* ou bien les événements virtuels. Un événement *Enter* se déclenche pour une balise lorsque celle-ci devient présente sur le caractère courant. Les événements **Enter** et **Leave** peuvent se produire ou parce que la marque courante a bougé ou parce que le caractère à cette position a changé. Ces événements sont différents des événements de même nom pour les fenêtres. D'autre part, les événements *ButtonPress*, *Motion* et *KeyPress* sont dirigés vers le caractère courant.

Si un événement virtuel est utilisé dans une liaison, cette liaison se déclenche seulement si l'événement virtuel est défini par un événement de souris ou de clavier sous-jacent. Il est possible que le caractère courant ait plusieurs balises et que chacune ait une liaison correspondant à un événement particulier. Dans ce cas, une liaison est invoquée pour chaque balise par ordre de la priorité la plus faible à la plus forte.

Enfin si des liaisons ont été créées pour le texte entier au moyen de la commande Tk **bind**, elles s'ajoutent à celles créées pour un élément particulier : les liaisons de la balise sont invoquées avant celles de la fenêtre de texte elle-même.

cheminTexte **tag cget** *nomBalise option*

Cette commande renvoie la valeur courante de l'option *option* associée à la balise désignée par *nomBalise*. L'argument *Option* peut prendre une quelconque des valeurs acceptées par la commande **tag configure**.

cheminTexte **tag configure** *nomBalise ?option? ?valeur? ?option valeur ...?*
Cette commande est analogue à la commande de même nom concernant le texte entier à la différence qu'elle permet de modifier les options associées à la balise *nomBalise* plutôt qu'à l'objet texte globalement.
Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminCanevas*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par les balises (voir à la section *Balises*, page 197.

cheminTexte **tag delete** *nomBalise ?nomBalise ...?*
Détruit toutes les informations concernant les balises désignées par les arguments *nomBalise*. La commande retire les balises de tous les caractères du fichier et détruit simultanément toute information associée aux balises telles que liaisons ou conditions d'affichage. La commande renvoie une chaîne vide.

cheminTexte **tag lower** *nomBalise ?autreBalise?*
Modifie la priorité de la balise *nomBalise* de telle sorte qu'elle soit immédiatement en dessous de la balise dont le nom est *autreBalise*. Si l'argument *autreBalise* est omis, la priorité de *nomBalise* devient la plus basse de toutes les balises.

cheminTexte **tag names** *?index?*
Renvoie une liste dont les éléments sont les noms de toutes les balises qui sont actives à la position désignée par l'indice *index*. Si l'argument *index* est omis, la valeur de retour décrit toutes les balises existant dans le texte : la liste est triée par ordre de priorité croissante.

cheminTexte **tag nextrange** *nomBalise index1 ?index2?*
Cette commande recherche dans le texte une séquence de caractères marquée par la balise *nomBalise* où le premier caractère de la séquence n'est pas avant le caractère d'indice *index1* et pas après le caractère immédiatement avant *index2*. Si plusieurs séquences correspondent, la première rencontrée est choisie. La valeur de retour de la commande est une liste de deux éléments qui sont l'indice du premier caractère de la séquence et l'indice du caractère immédiatement après le dernier. Si aucune séquence n'est trouvée, la valeur de retour est une chaîne vide. Si *index2* n'est pas précisé, il correspond par défaut à la fin du texte.

cheminTexte **tag prevrange** *nomBalise index1 ?index2?*
Cette commande recherche dans le texte une séquence de caractères marquée par la balise *nomBalise* où le premier caractère de la séquence

est avant le caractère d'indice *index1* mais pas avant le caractère d'indice *index2*. Si plusieurs séquences correspondent, celle qui est la plus proche de l'indice *index2* est choisie. La valeur de retour de la commande est une liste de deux éléments qui sont l'indice du premier caractère de la séquence et l'indice du caractère immédiatement après le dernier. Si aucune séquence n'est trouvée, la valeur de retour est une chaîne vide. Si *index2* n'est pas précisé, il correspond par défaut au début du texte.

cheminTexte **tag raise** *nomBalise ?autreBalise?*
Modifie la priorité de la balise *nomBalise* de telle sorte qu'elle soit immédiatement au-dessus de la balise dont le nom est *autreBalise*. Si l'argument *autreBalise* est omis, la priorité de *nomBalise* devient la plus haute parmi toutes les balises.

cheminTexte **tag ranges** *nomBalise*
Renvoie une liste décrivant toutes les séquences de texte marquées par la balise *nomBalise*. Les deux premiers éléments de la liste décrivent la première séquence dotée de la balise, les deux suivants la seconde séquence et ainsi de suite. Dans chaque paire, le premier élément représente l'indice du premier caractère de la séquence et le second l'indice du caractère immédiatement après le dernier de la séquence. S'il n'y a pas de caractères ayant la balise *nomBalise*, une chaîne vide est renvoyée.

cheminTexte **tag remove** *nomBalise index1 ?index2 index1 index2 ...?*
Supprime la balise *nomBalise* de tous les caractères depuis l'indice *index1* jusqu'au caractère immédiatement avant l'indice *index2*. Une commande peut contenir un nombre quelconque de paires d'indices *index1-index2*. Si le dernier indice *index2* est omis, seul le caractère placé à l'indice *index1* reçoit une balise. S'il n'y a pas de caractères dans l'intervalle spécifié (par exemple si *index1* se trouve après *index2* ou après la fin du fichier), la commande est sans effet. Cette commande renvoie une chaîne vide.

cheminTexte **window** *opération ?arg arg ...?*
Cette commande est utilisée pour manipuler les fenêtres insérées. Le comportement exact de la commande dépend de l'argument *opération* qui suit le mot **window**. Les formes suivantes de la commande **window** sont actuellement supportées :

cheminTexte **window cget** *index option*
Renvoie la valeur courante de l'option de configuration désignée par l'argument *option*. L'argument *index* identifie la fenêtre et l'argument *option* peut prendre une quelconque des valeurs admises par les fenêtres insérées (voir à la section *Fenêtres insérées*, page 202).

cheminTexte **window configure***index ?option valeur ...?*
Permet d'obtenir ou de modifier les options de configuration pour une fenêtre imbriquée. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *cheminMenu*. Si l'argument *option* est spécifié sans une valeur correspondante, la com-

mande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-valeur* sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. Cet argument peut prendre uniquement les valeurs admises par les fenêtres insérées : voir à la section *Fenêtres insérées*, page 202.

cheminTexte **window create** *index ?option valeur ...?*
Cette commande crée une nouvelle annotation de type fenêtre, qui apparaîtra dans le texte à la position spécifiée par l'indice *index*. On peut préciser un nombre quelconque de paires *option-valeur* afin de configurer l'annotation. Cette opération renvoie une chaîne vide. On se reportera à la section *Fenêtres insérées*, page 202 pour connaître les options supportées.

cheminTexte **window names**
Renvoie une liste dont les éléments sont les noms de toutes les fenêtres actuellement insérées dans le texte.

cheminTexte **xview** *option args*
Cette commande est utilisée pour obtenir ou modifier la position horizontale du texte dans la fenêtre. Elle peut prendre une des formes suivantes :

cheminTexte **xview**
Renvoie une liste de deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : ces valeurs indiquent horizontalement les proportions de l'élément qui sont visibles ou non dans la fenêtre. Par exemple, si les valeurs renvoyées sont 0,2 et 0,6 cela signifie que 20% de la zone du texte sont situés à gauche de la partie visible, que 40% sont visibles au milieu et que les 40% restants sont non visibles à droite de la fenêtre. Les fractions concernent seulement les lignes qui sont visibles dans la fenêtre : si toutes les lignes visibles sont plus courtes que la largeur de la fenêtre et sont donc entièrement visibles, les fractions renvoyées seront 0 et 1 même si d'autres lignes ailleurs sont plus larges que la fenêtre. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de l'option **-xscrollcommand** .

cheminTexte **xview moveto** *fraction*
Ajuste la vue dans la fenêtre de texte de telle sorte que la portion désignée en pourcentage de la largeur totale par la fraction *fraction* soit masquée par le bord gauche de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1].

cheminTexte **xview scroll** *nb unité*
Cette commande décale la vue dans la fenêtre de texte vers la gauche ou vers la droite selon la valeur donnée aux arguments *nb* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage est de *nb* largeurs moyennes de caractères à l'écran. Avec *pages*, l'unité de décalage correspond à la valeur d'un écran entier. Si *nb* est négatif, le décalage se fait vers la droite ; s'il est positif, le décalage se fait vers la gauche.

cheminTexte **yview** *?args?*

Cette commande est utilisée pour obtenir ou modifier la position verticale du texte dans la fenêtre. Elle peut prendre une des formes suivantes :

cheminTexte **yview**

Renvoie une liste de deux éléments. Chaque élément est une fraction dans l'intervalle [0,1] : ces valeurs indiquent verticalement les proportions de l'élément qui sont visibles ou non dans la fenêtre. Le premier élément donne la position du premier caractère de la ligne placée en haut de la fenêtre, par rapport au texte entier (0,5 désigne le milieu du texte en hauteur par exemple). Le second élément donne la position du caractère immédiatement après le dernier de la ligne qui se trouve au bas de la fenêtre, relativement au texte entier. Ce sont les mêmes valeurs qui sont passées aux barres de défilement au moyen de l'option **-yscrollcommand** .

cheminTexte **yview moveto** *fraction*

Ajuste la vue dans la fenêtre de texte de telle sorte que la portion désignée en pourcentage de la hauteur totale par la fraction *fraction* soit masquée par le bord supérieur de cette fenêtre. L'argument *fraction* est une valeur dans l'intervalle [0,1] : 0 indique le premier caractère dans le texte, 0,33 indique le caractère au tiers du texte etc.

cheminTexte **yview scroll** *nb unité*

Cette commande décale la vue dans la fenêtre de texte vers le haut ou vers le bas selon la valeur donnée aux arguments *nombre* et *unité*. L'argument *nb* est un nombre entier. L'argument *unité* est l'un des mots *units* ou *pages* (ou une abréviation). Avec *units*, le décalage se fait en nombre de lignes. Avec *pages*, l'unité de décalage correspond à la hauteur de la fenêtre. Si *nb* est négatif, le décalage se fait vers le bas ; s'il est positif, le décalage se fait vers le haut.

cheminTexte **yview**?**-pickplace**? *index*

Modifie la vue dans la fenêtre de texte pour rendre l'indice *index* visible. Si l'option **-pickplace** n'est pas spécifiée, l'indice *index* apparaîtra au haut de la fenêtre. Si **-pickplace** est spécifié, l'objet choisit où l'indice apparaîtra dans la fenêtre : si *index* est déjà visible quelque part, rien ne se produit. S'il n'est qu'à quelques lignes au-dessus de la fenêtre, il sera positionné au sommet. S'il n'est qu'à quelques lignes en dessous de la fenêtre, il sera positionné en bas. Dans les autres cas, il sera positionné au centre.

L'option **-pickplace** a été rendue obsolète par la commande **see** qui gère les déplacements horizontaux et verticaux pour rendre une position visible alors que **-pickplace** ne gère que les ajustements verticaux.

cheminTexte **yview** *nb*

Cette commande rend le premier caractère de la ligne après celui désigné par l'argument *nb* visible au sommet de la fenêtre. L'argument *nb* doit

être un entier. Cette commande servait anciennement pour les fonctions de défilement, elle est maintenant obsolète.

Liaisons par défaut

Tk crée automatiquement des liaisons de classe pour les objets textes avec les caractéristiques indiquées ci-dessous. La notion de « mot » est ici définie par le contenu de la variable **tcl_wordchars** :

1. Un clic du premier bouton de la souris positionne le curseur d'insertion immédiatement avant le caractère sous le pointeur de la souris, focalise l'objet et efface toute sélection déjà existante. Glisser avec le premier bouton de la souris dessine une sélection entre le curseur d'insertion et le caractère sous la souris.
2. Un double clic avec le premier bouton de la souris sélectionne le mot sous la souris et positionne le curseur d'insertion au début du mot. Glisser après un double clic dessine une sélection faite de mots entiers.
3. Un triple clic avec le premier bouton de la souris sélectionne la ligne sous la souris et positionne le curseur d'insertion au début de la ligne. Glisser après un triple clic dessine une sélection faite de lignes entières.
4. Les extrêmités de la sélection peuvent être ajustées en glissant avec le premier bouton de la souris en maintenant la touche Majuscule enfoncée ; cela ajuste l'extrêmité de la sélection la plus proche du curseur de la souris au moment où le bouton a été enfoncé. Avec un double ou un triple clic, la sélection se fait en mots ou en lignes entiers respectivement.
5. Un clic du premier bouton de la souris avec la touche Contrôle enfoncée repositionne le curseur d'insertion sans affecter la sélection.
6. Si des caractères normaux sont saisis au clavier, ils sont insérés à la position du curseur d'insertion.
7. La vue peut être ajustée en glissant avec le second bouton de la souris enfoncé. Si le second bouton de la souris est cliqué sans la déplacer, la sélection est copiée dans le texte à la position du pointeur de la souris. La touche Insert insère elle aussi la sélection mais à la position du curseur d'insertion.
8. Si la souris est glissée hors de la fenêtre de texte avec le premier bouton enfoncé, le texte défilera automatiquement pour faire apparaître les portions qui ne sont pas visibles.
9. Les touches Gauche et Droite déplacent le curseur d'insertion d'un caractère vers la gauche ou vers la droite ; elles effacent toute sélection dans le texte. Si en même temps la touche Majuscule est enfoncée, le curseur d'insertion se déplace et la sélection est étendue pour inclure le nouveau caractère. Les touches Contrôle-Gauche et Contrôle-Droite déplacent le curseur d'insertion par mots entiers, et Contrôle-Majuscule-Gauche et Contrôle-Majuscule-Droite déplacent le curseur d'insertion par mots entiers tout en étendant la sélection. Contrôle-b et Contrôle-f sont synonymes de Gauche et Droite,

respectivement. Méta-b et Méta-f sont synonymes de Contrôle-Gauche et Contrôle-Droite, respectivement.

10. Les touches Haut et Bas déplacent le curseur d'insertion d'une ligne vers le haut ou vers le bas et effacent toute sélection dans le texte. Si en même temps la touche Majuscule est enfoncée, le curseur d'insertion se déplace et la sélection est étendue pour inclure le nouveau caractère. Les touches Contrôle-Haut et Contrôle-Bas déplacent le curseur d'insertion par paragraphes entiers, et Contrôle-Majuscule-Haut et Contrôle-Majuscule-Bas déplacent le curseur d'insertion par paragraphes entiers tout en étendant la sélection. Contrôle-p et Contrôle-n sont synonymes de Haut et Bas, respectivement.
11. Les touches Précédent et Suivant déplacent le curseur d'insertion vers l'avant ou l'arrière d'un écran entier et effacent toute sélection dans le texte. Si en même temps la touche Majuscule est enfoncée, la sélection est étendue pour inclure le nouveau caractère. Contrôle-v déplace la vue vers le bas d'un écran sans bouger le curseur d'insertion ni ajuster la sélection.
12. Les touches Contrôle-Précédent et Contrôle-Suivant font défiler la vue vers la droite ou vers la gauche d'une page sans bouger le curseur d'insertion ni affecter la sélection.
13. Les touches Début et Contrôle-a déplacent le curseur d'insertion au début de la ligne et effacent toute sélection dans le texte. Majuscule-Début déplace le curseur d'insertion au début de la ligne et étend la sélection jusqu'à ce point.
14. Les touches Fin et Contrôle-e déplacent le curseur d'insertion à la fin de la ligne et effacent toute sélection dans le texte. Majuscule-Fin déplace le curseur à la fin de la ligne et étend la sélection jusqu'à ce point.
15. Les touches Contrôle-Début et Méta-< déplacent le curseur d'insertion au début du texte et effacent toute sélection dans le texte. Contrôle-Majuscule-Début déplace le curseur d'insertion au début du texte et étend la sélection jusqu'à ce point.
16. Les touches Contrôle-Fin et Méta-> déplacent le curseur d'insertion à la fin du texte et effacent toute sélection dans le texte. Contrôle-Majuscule-Fin déplace le curseur d'insertion à la fin du texte et étend la sélection jusqu'à ce point.
17. Les touches Sélection et Contrôle-Espace fixent le point d'ancrage de la sélection à la position du curseur d'insertion. Elles n'affectent pas la sélection courante. Majuscule-Sélection et Contrôle-Majuscule-Espace ajustent la sélection à la position courante du curseur d'insertion, sélectionnant depuis le point d'ancrage jusqu'au curseur d'insertion s'il n'y avait pas de sélection auparavant.
18. Contrôle-/ sélectionne le contenu entier de l'objet texte.
19. Contrôle-\ efface toute sélection dans l'objet.
20. Les touches F16 ou Méta-w copient la sélection du texte, s'il y a une sélection.
21. Les touches F20 ou Contrôle-w copient la sélection du texte et détruisent la sélection. S'il n'y a pas de sélection, ces touches sont sans effet.

22. Les touches F18 ou Contrôle-y insèrent le contenu d'une sélection copiée à la position du curseur d'insertion.
23. La touche Suppression supprime la sélection s'il y en a une. S'il n'y a pas de sélection, elle supprime le caractère placé à droite du curseur d'insertion.
24. Les touches Retour-Arrière et Contrôle-h suppriment la sélection s'il y en a une. S'il n'y a pas de sélection, elles suppriment le caractère placé à gauche du curseur d'insertion.
25. Contrôle-d supprime le caractère placé à droite du curseur d'insertion.
26. Méta-d supprime le mot placé à droite du curseur d'insertion.
27. Contrôle-k supprime tout depuis le curseur d'insertion jusqu'à la fin de la ligne ; si le curseur d'insertion est déjà à la fin d'une ligne, Contrôle-k supprime le caractère de fin de ligne.
28. Contrôle-o commence une nouvelle ligne en insérant un caractère de nouvelle ligne devant le curseur d'insertion sans déplacer le curseur d'insertion.
29. Méta-Arrière et Méta-Suppression suppriment le mot placé à gauche du curseur d'insertion.
30. Contrôle-x supprime tout ce qui est sélectionné dans l'objet texte.
31. Contrôle-t inverse l'ordre des deux caractères à droite du curseur d'insertion.
32. Contrôle-z (et Contrôle-soulignement sous UNIX lorsque la variable globale **tk_strictMotif** est vraie) défait la dernière action d'édition si l'option **-undo** est vraie.
33. Contrôle-Z (ou Contrôle-y sur Windows) réapplique la dernière action d'édition défaite si l'option **-undo** est vraie.

Si l'objet texte a été désactivé au moyen de l'option **-state**, sa vue peut quand même être ajustée et le contenu est néanmoins sélectionnable mais le curseur d'insertion n'est pas affiché et aucune modification du texte n'est possible.

Options standard

Les options standard supportées par les composants de type *texte* sont rassemblées dans le tableau 17 de la page suivante. Ces options sont décrites en détail à la page 142.

Options spécifiques

-autoseparators **autoSeparators** **AutoSeparators**
Cette option a été introduite avec la version 8.4 de Tk. Elle spécifie, lorsque l'option **-undo** est vraie, si des séparateurs sont automatiquement insérés entre les actions d'édition susceptibles d'être défaites.

-background ou -bg	background	Background
-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-exportselection	exportSelection	ExportSelection
-font	font	Font
-foreground ou -fg	foreground	Foreground
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-insertbackground	insertBackground	Foreground
-insertborderwidth	insertBorderWidth	BorderWidth
-insertofftime	insertOffTime	OffTime
-insertontime	insertOnTime	OnTime
-insertwidth	insertWidth	InsertWidth
-padx	padX	Pad
-pady	padY	Pad
-relief	relief	Relief
-selectbackground	selectBackground	Foreground
-selectborderwidth	selectBorderWidth	BorderWidth
-selectforeground	selectForeground	Background
-takefocus	takeFocus	TakeFocus
-xscrollcommand	xScrollCommand	ScrollCommand
-yscrollcommand	yScrollCommand	ScrollCommand

TAB. 17 – *Options standard reconnues par les composants de type* text.

-height **height** **Height**

Spécifie la hauteur désirée pour la fenêtre, mesurée en nombre de caractères dans la fonte spécifiée par l'option **-font** . La valeur doit être au moins 1.

-maxundo **maxUndo** **MaxUndo**

Cette option a été introduite avec la version 8.4 de Tk. Elle désigne, lorsque l'option **-undo** est vraie, le nombre maximal d'actions d'édition qui peuvent être défaites. La valeur 0 correspond à un nombre illimité.

-setgrid **setGrid** **SetGrid**

Spécifie une valeur booléenne qui détermine si le texte contrôle la grille de redimensionnement pour la fenêtre qui le contient. Les composants textes possèdent une unité « naturelle » (la taille d'un caractère) et il semble logique que les dimensions de la fenêtre soient un multiple entier de cette unité. Ces tailles naturelles des fenêtres forment une grille. Si l'option **setGrid** est fixée à la valeur vraie (*true*) alors le composant communiquera avec le serveur de fenêtres afin que, lorsque l'utilisateur redimensionne interactivement la fenêtre du composant, les dimensions de celle-ci s'affichent en unités de grille et que la taille soit astreinte à rester un multiple entier de ces unités (cf. la discussion de la commande **wm** p. 251 et suivantes).

-spacing1 **spacing1** **Spacing1**

Permet de spécifier un espace supplémentaire au-dessus de chaque ligne logique de texte. La valeur de l'option peut être exprimée au moyen des unités d'écran habituelles. Si une ligne doit être coupée, cette option s'applique uniquement à la première ligne affichée à l'écran. Cette option peut être supplantée par des options **spacing1** dans les balises (*tags*).

-spacing2 **spacing2** **Spacing2**

Pour les lignes qui doivent être coupées (ce qui signifie qu'elles couvrent plus d'une ligne lorsqu'elles sont affichées à l'écran), cette option spécifie un espacement additionnel à insérer entre les lignes d'écran qui correspondent à une même ligne de texte. La valeur de l'option peut être exprimée au moyen des unités d'écran habituelles. Cette option peut être supplantée par des options **spacing2** dans les balises (*tags*).

-spacing3 **spacing3** **Spacing3**

Permet de spécifier un espace supplémentaire en dessous de chaque ligne logique de texte. La valeur de l'option peut être exprimée au moyen des unités d'écran habituelles. Si une ligne doit être coupée, cette option s'applique uniquement à la dernière ligne affichée à l'écran. Cette option peut être supplantée par des options **spacing3** dans les balises (*tags*).

-state **state** **State**

Spécifie un des deux états *normal* ou *disabled* pour le texte. Si l'objet texte est désactivé, aucun caractère ne peut être inséré ou détruit et aucun curseur d'insertion n'est affiché, même si l'objet est focalisé : le texte est alors en lecture seule.

-tabs **tabs** **Tabs**

Spécifie un ensemble de taquets de tabulation pour la fenêtre de texte. La valeur de cette option est constituée d'une liste de distances d'écran indiquant les positions des taquets de tabulation. Chaque position peut être éventuellement complétée par l'un des mots-clés *left*, *right*, *center* ou *numeric*, qui spécifient la justification du texte par rapport au taquet de tabulation :

- La valeur par défaut est *left*. Elle a pour effet d'aligner l'extrêmité gauche du texte suivant le caractère de tabulation sur le taquet ;
- la valeur *right* signifie que l'extrêmité droite du texte suivant le caractère de tabulation est alignée sur le taquet ;
- la valeur *center* signifie que le texte est centré sur le taquet ;
- la valeur *numeric* signifie qu'un point décimal dans le texte sera positionné au taquet. S'il n'y a pas de point décimal, le chiffre le moins significatif du nombre est positionné immédiatement à la gauche du taquet. S'il n'y a pas de nombre dans le texte, celui-ci est justifié à droite à l'emplacement du taquet.

Par exemple, l'instruction :

```
-tabs {2c left 4c 6c center}
```

crée trois taquets de tabulation à des intervalles de deux centimètres ; les deux premiers utilisent une justification à gauche et le troisième une justification centrée.

Si la liste des taquets de tabulation n'a pas assez d'éléments pour couvrir toutes les tabulations d'une ligne de texte, Tk extrapole de nouveaux taquets de tabulation en utilisant les valeurs du dernier taquet de la liste (espacement et justification). La valeur de l'option **tabs** peut être supplantée par des options **-tabs** dans les balises (*tags*). Si aucune option **-tabs** n'est spécifiée, ou si elle est spécifiée comme une liste vide, Tk utilise des tabulations par défaut tous les huit caractères (en taille moyenne de caractères).

-undo **undo** **Undo**

Cette option a été introduite avec la version 8.4 de Tk. Elle permet de spécifier si des actions d'édition peuvent être défaites.

-width **width** **Width**

Spécifie une largeur pour la fenêtre de texte, mesurée en nombre de caractères dans la fonte spécifiée par l'option **-font**. Si la fonte n'a pas une largeur fixe, la largeur du caractère 0 est utilisée pour transformer les unités de caractères en unités d'écran.

-wrap **wrap** **Wrap**

Spécifie comment traiter les lignes du texte qui sont trop longues pour être affichée sur une ligne unique à l'écran. La valeur de cette option doit être *none*, *char* ou *word* :

- la valeur **none** signifie qu'aucune ligne ne sera coupée. Chaque ligne de texte apparaît exactement comme une ligne à l'écran et les caractères excédentaires ne sont pas affichés ;
- la valeur **char** signifie que les lignes trop longues peuvent être coupées après n'importe quel caractère ;
- la valeur **word** signifie que les lignes trop longues ne seront coupées qu'entre les mots.

tk

Permet de paramétrer l'état interne de Tk.

Syntaxe

tk *sous-commande ?arg arg ...?*

Description

La commande **tk** permet d'accéder à divers éléments concernant l'état interne de Tk. Cette commande permet de manipuler des informations à propos de l'application en général ou d'un écran ou un moniteur, plutôt que des fenêtres particulières. La commande peut prendre des formes différentes en fonction de la sous-commande qui lui est adjointe. Les sous-commandes autorisées sont les suivantes :

tk appname *?nouveauNom?*

Si *nouveauNom* n'est pas spécifié, cette commande renvoie le nom de l'application : c'est le nom qui serait utilisé par une commande **send** pour communiquer avec l'application. Si *nouveauNom* est spécifié, le nom de l'application est changé en *nouveauNom*. Si le nom indiqué est déjà utilisé, un suffixe numérique est adjoint afin d'en faire un nom unique. Le résultat de la commande est le nom effectivement choisi.

L'argument *nouveauNom* ne devrait pas commencer par une lettre majuscule. Cela risquerait d'interférer avec le traitement des options car les noms commençant par une majuscule sont censés désigner des classes ; de ce fait, Tk pourrait ne pas trouver certaines options pour l'application.

Si la commande **send** a été désactivée (par exemple pour des raisons de sécurité), on peut la recréer et la réactiver avec **tk appname**.

tk scaling ?**-displayof** *fenêtre*? *?nombre?*

Permet d'obtenir et de fixer le facteur de proportionnalité courant utilisé par Tk pour convertir des unités physiques (telles que points, pouces ou millimètres) en pixels et réciproquement. L'argument *nombre* est un nombre en virgule flottante qui spécifie le nombre de pixels par point sur le moniteur de la fenêtre désignée par l'argument *fenêtre*. Si l'argument *fenêtre* est omis, la fenêtre principale est utilisée par défaut. Si l'argument *nombre* est omis, la valeur courante du facteur de proportionnalité est renvoyée.

Un point est une unité de mesure égale à 1/72 pouce (1 pouce vaut 2,54 mm). Un facteur de proportionnalité de 1.0 correspond à 1 pixel par point, ce qui correspond à la résolution standard des moniteurs de 72 dpi (points par pouce). Un facteur de proportionnalité de `1.25` signifie 1,25 pixels par point, ce qui est le réglage pour un moniteur à 90 dpi ; fixer ce facteur de proportionnalité à `1.25` sur un moniteur à 72 dpi aurait pour effet que tous les objets de l'application seraient affichés 1,25 fois plus grands que la normale. La valeur initiale du facteur de proportionnalité est fixée au démarrage de

l'application, en fonction des propriétés du moniteur installé, mais on peut la modifier à tout moment. Le nouveau facteur est alors utilisé mais il n'est pas garanti que les objets déjà existants se redimensionneront dynamiquement pour tenir compte de la nouvelle valeur.

tk useinputmethods ?**-displayof** *fenêtre*? *?bool?*

Permet d'obtenir et de fixer l'état de Tk concernant les méthodes d'entrée XIM (*X Input Methods*) pour les événements de filtrage. L'état résultant est renvoyé. XIM est utilisé dans certaines locales (par exemple *Japanese*, *Korean*), pour prendre en charge des périphériques de saisie particuliers. Cela n'a de sens que sous le système X. Si les méthodes XIM ne sont pas supportées, la commande renvoie toujours 0. Si l'argument *fenêtre* est omis, la fenêtre principale est utilisée par défaut. Si l'argument *bool* est omis, la valeur courante de l'état est renvoyée. La valeur par défaut est 1 pour le moniteur principal.

tk_chooseColor

Syntaxe

Tk_chooseColor *?option valeur ...?*

Description

La procédure **Tk_chooseColor** présente une boîte de dialogue pour permettre à l'utilisateur de sélectionner une couleur. Les options suivantes sont disponibles :

-initialcolor *couleur*
: Spécifie la couleur à afficher dans le dialogue lorsqu'il est appelé. L'argument *couleur* doit prendre une des valeurs acceptées pour la désignation des couleurs.

-parent *fenêtre*
: Fait de *fenêtre* le parent logique du sélectionneur de couleurs. Le sélectionneur de couleurs est affiché au-dessus de la fenêtre parent.

-title *titre*
: Spécifie une chaîne à afficher comme titre pour la boîte de dialogue. Si cette option n'est pas spécifiée, le titre par défaut sera affiché.

Si l'utilisateur sélectionne une couleur, **Tk_chooseColor** renvoie le nom de la couleur sous une des formes acceptées pour la désignation des couleurs. Si l'utilisateur annule l'opération, la commande renvoie une chaîne vide.

Exemple

```
button .b -fg [tk_chooseColor -initialcolor gray -title "Choisir une couleur"]
```

tk_chooseDirectory

Syntaxe

Tk_chooseDirectory *?option valeur ...?*

Description

La procédure **Tk_chooseDirectory** présente une boîte de dialogue permettant à l'utilisateur de sélectionner un répertoire. Les options suivantes sont disponibles :

-initialdir *répertoire*
: Spécifie que les répertoires indiqués dans l'argument *répertoire* doivent être affichés lorsque le dialogue est présenté. Si ce paramètre n'est pas spécifié, les sous-répertoires du répertoire de travail courant sont affichés. Si le paramètre spécifie un chemin relatif, la valeur renvoyée convertira le chemin relatif en chemin absolu. Cette option ne fonctionne pas toujours sur les Macintosh. Ce n'est pas un bogue : sur cette plate-forme, le tableau de bord *Général* permet de spécifier le comportement souhaité au niveau du système (Système 9 ou antérieur).

-parent *fenêtre*
: Fait de la fenêtre *fenêtre* le parent logique du dialogue. Le dialogue est affiché au-dessus de la fenêtre parent.

-title *titre*
: Spécifie une chaîne à afficher comme titre de la boîte de dialogue. Si cette option n'est pas spécifiée, un titre par défaut sera affiché.

-mustexist *bool*
: Indique si l'utilisateur a le droit de spécifier des répertoires non-existants. Si ce paramètre a la valeur *true*, l'utilisateur peut uniquement sélectionner des répertoires qui existent déjà. La valeur par défaut est *false*.

tk_dialog

Syntaxe

Tk_dialog *fenêtre titre texte bitmap défaut nomBouton nomBouton ...*

Description

Cette procédure fait partie de la bibliothèque de scripts standard de Tk. Ses arguments permettent de décrire une boîte de dialogue :

fenêtre est le nom de la fenêtre de premier niveau à utiliser pour ce dialogue. Toute autre fenêtre qui aurait déjà ce nom là sera détruite.

titre est le texte qui figurera dans la barre de titre de la fenêtre.

texte est le texte qui apparaîtra dans la partie supérieure du dialogue.

bitmap permet de spécifier une image bitmap à afficher dans la partie supérieure de la boîte de dialogue, à gauche du texte. Si une chaîne vide est spécifiée pour cet argument, aucune image n'est affichée.

défaut permet de préciser le bouton par défaut. Si c'est un nombre supérieur ou égal à zéro, il désigne l'indice du bouton qui sera choisi comme bouton par défaut (l'indice 0 correspond au bouton le plus à gauche). Si c'est une valeur négative ou une chaîne vide, il n'y aura pas de bouton par défaut.

nomBouton désigne le texte affiché par chaque bouton. Il y aura autant de boutons que d'arguments *nomBouton* spécifiés. Ceux-ci seront placé de gauche à droite dans le même ordre que les arguments correspondants.

Une fois créée la boîte de dialogue, la procédure **Tk_dialog** attend que l'utilisateur sélectionne l'un des boutons au moyen de la souris ou en tapant la touche Retour-Chariot qui invoque le bouton par défaut. Elle renvoie l'indice du bouton qui a été invoqué. Si cette fenêtre est détruite avant que l'utilisateur n'ait sélectionné un bouton, la valeur -1 est renvoyée. En attendant que l'utilisateur agisse, la procédure **Tk_dialog** exécute une commande **grab** locale afin d'empêcher toute interaction avec le reste du programme.

tk_getOpenFile

Syntaxe

Tk_getOpenFile *?option valeur ...?*
Tk_getSaveFile *?option valeur ...?*

Description

Les procédures **Tk_getOpenFile** et **Tk_getSaveFile** présentent une boîte de dialogue permettant à l'utilisateur de sélectionner un fichier à ouvrir ou à sauvegarder. La commande **Tk_getOpenFile** est habituellement associée à l'article *Ouvrir* d'un menu *Fichier*. Son but est pour l'utilisateur de sélectionner uniquement un fichier déjà existant. Si l'utilisateur entre le nom d'un fichier qui n'existe pas, la boîte de dialogue présente à l'utilisateur un message d'erreur demandant d'opérer une autre sélection. La commande **Tk_getSaveFile** est habituellement associée à l'article *Enregistrer sous...* d'un menu *Fichier*. Si l'utilisateur entre le nom d'un fichier qui existe existe, la boîte de dialogue présente à l'utilisateur un message demandant confirmation pour écraser le fichier déjà existant.

Les options suivantes sont disponibles pour ces deux commandes :

-defaultextension *extension*
: Spécifie une chaîne à ajouter au nom du fichier si l'utilisateur entre un nom de fichier sans extension. La valeur par défaut est la chaîne vide. Cette option est ignorée sur les plates-formes Macintosh car le système ne requiert pas d'extensions aux noms de fichiers.

-filetypes *motifFichier*
: Si les dialogues sont capables d'afficher une liste de types de fichiers particuliers, cette option permet de spécifier les types à afficher dans cette liste. Lorsque l'utilisateur sélectionne un type de fichier dans la liste, seuls les fichiers de ce type sont listés. Si cette option n'est pas spécifiée, ou si elle désigne une liste vide, ou encore si les listes de types de fichiers ne sont pas supportées sur une plate-forme particulière, tous les fichiers seront listés indépendamment de leurs types. La section *Spécification des motifs de fichiers* ci-dessous précise quelle est la syntaxe autorisée pour l'argument *motifFichier*.

-initialdir *répertoire*
: Spécifie que les fichiers indiqués dans l'argument *répertoire* doivent être affichés lorsque le dialogue est présenté. Si ce paramètre n'est pas spécifié, les fichiers du répertoire de travail courant sont affichés. Si le paramètre spécifie un chemin relatif, la valeur renvoyée convertira le chemin relatif en chemin absolu. Cette option ne fonctionne pas toujours sur les Macintosh. Ce n'est pas un bogue : sur cette plate-forme, le tableau de bord *Général* permet de spécifier le comportement souhaité au niveau du système (Système 9 ou antérieur).

-initialfile *nomFichier*
Spécifie un nom de fichier à afficher dans le dialogue lorsqu'il est présenté. Cette option est ignorée sur les plates-formes Macintosh.

-multiple
Permet à l'utilisateur de choisir plusieurs fichiers à la fois dans les dialogues d'ouverture de fichiers. Sous MacOS cette option n'est disponible que si les Services de Navigation (*Navigation Services*) sont disponibles (ceux-ci sont intégrés au système depuis la version 8.5).

-message
Spécifie un message à faire figurer dans le dialogue. Cette option est disponible uniquement sur Macintosh et si les Services de Navigation (*Navigation Services*) sont installés (version 8.5 ou ultérieure).

-parent *fenêtre*
Fait de la fenêtre *fenêtre* le parent logique du dialogue. Le dialogue de fichier est affiché au-dessus de la fenêtre parent.

-title *titre*
Spécifie une chaîne à afficher comme titre de la boîte de dialogue. Si cette option n'est pas spécifiée, un titre par défaut est affiché.

Si l'utilisateur sélectionne un fichier, les deux commandes **Tk_getOpenFile** et **Tk_getSaveFile** renvoient le chemin complet de ce fichier. Si l'utilisateur annule l'opération, les deux commandes renvoient une chaîne nulle.

Spécification des motifs de fichiers

La valeur *motifFichier* figurant dans l'option **-filetypes** est une liste de motifs de fichiers. Chaque motif de fichier doit être conforme au modèle suivant :
nomType *extension ?extension ...? ?typeMac ...?*
où *nomType* est le nom du type de fichier décrit par ce motif et qui apparaîtra dans la liste de types affichée dans la boîte de dialogue. Les arguments *extension* sont des extensions de fichiers pour ce motif. Sur MacOS, on peut spécifier des arguments *typeMac ...* qui sont des types de fichiers (codes de quatre caractères maintenus par le système et caractérisant le type d'un fichier particulier, comme TEXT, APPL, PICT etc.).

Plusieurs motifs de fichiers peuvent avoir le même argument *nomType,* auquel cas ils se réfèrent au même type de fichier et celui-ci n'apparaîtra bien entendu qu'une fois dans la liste des types. Lorsque l'utilisateur sélectionne une entrée dans la liste des types, tous les fichiers correspondant à ce type sont listés. Habituellement chaque motif correspond à un type distinct. L'utilisation de plusieurs motifs associés à un même type de fichiers est utile uniquement pour les plates-formes Macintosh.

On est en fait amené à distinguer plusieurs cas de figure :

– Sur les plates-formes Macintosh, un fichier correspond à un motif si son nom comporte une des extensions figurant dans les arguments *extension* des déclarations de motif et que son type correspond à l'un des types spécifiés par des arguments *typeMac*.

– Sur les plates-formes Unix et Windows, un fichier correspond au motif si son nom comporte une des extensions figurant dans les arguments *extension* des déclarations de motif.

Considérons le code suivant :

```
set types {
    {{Fichiers Texte}     {.txt}       }
    {{Scripts Tcl}        {.tcl}       }
    {{Fichiers Source C}  {.c}     TEXT}
    {{Fichiers GIF}       {.gif}       }
    {{Fichiers GIF}       {}       GIFF}
    {{Tous Fichiers}       *           }
}
set filename [tk_getOpenFile -filetypes $types]

if {$filename != ""} {
#     Ouverture du fichier  ...
}
```

Dans cet exemple, les motifs *Fichiers Source C* correspondront, sur Macintosh, à des fichiers ayant une extension .c et possédant en même temps le type interne TEXT. Si l'on voulait que ce soit l'une *ou* l'autre de ces propriétés, il faudrait utiliser deux motifs de fichiers : c'est ce qui a été fait dans cet exemple pour les motifs de *Fichiers GIF*.

Spécification des extensions

Sur les plates-formes Unix et Macintosh, les extensions peuvent être décrites au moyen de motifs globalisants (comme ceux que l'on utilise avec la commande Tcl **glob** – cf. p. **??**). Sur les plates-formes Windows, les extensions sont identifiées par le système d'exploitation. Les types d'extensions acceptés sont :

– le caractère spécial $*$ qui décrit n'importe quel fichier ;
– l'extension spéciale `""` qui désigne n'importe quel fichier dépourvu d'extension ;
– toute chaîne de caractères ne comportant pas l'un des métacaractères `?` ou `*`.

Les règles de correspondance étant différentes selon les plates-formes, les métacaractères ne sont pas acceptés dans les extensions (à l'exception de l'extension particulière $*$) afin d'assurer la portabilité du code.

tk_messageBox

Syntaxe

Tk_messageBox *?option valeur ...?*

Description

Cette procédure crée et affiche une fenêtre de message comportant un texte, une icone et une série de boutons. Chaque bouton est identifié par un nom symbolique (voir l'option **-type** ci-dessous). Une fois que la fenêtre de message est affichée, la fonction **Tk_messageBox** attend que l'utilisateur sélectionne l'un des boutons. La procédure renvoie le nom symbolique du bouton sélectionné.

Les options suivantes sont disponibles :

-default *nom*
: L'argument *nom* donne le nom symbolique du bouton par défaut de cette fenêtre de message (*ok, cancel* etc). Voir la liste des noms symboliques avec l'option **-type** ci-dessous. Si cette option n'est pas spécifiée, le premier bouton du dialogue sera choisi comme bouton par défaut.

-icon *icone*
: Spécifie une icone à afficher. L'argument *icone* peut prendre une des valeurs suivantes : *error*, *info*, *question* ou *warning*. Si cette option n'est pas spécifiée, c'est l'icone *info* qui est affichée par défaut.

-message *chaîne*
: Spécifie le message à afficher.

-parent *fenêtre*
: Fait de la fenêtre *fenêtre* le parent logique du message. La fenêtre message est affichée au-dessus de la fenêtre parent.

-title *titre*
: Spécifie une chaîne à afficher comme titre pour la fenêtre de message. La valeur par défaut est la chaîne vide.

-type *typePrédef*
: Fait en sorte qu'un ensemble prédéfini de boutons soit affiché. Les valeurs suivantes sont possibles pour l'argument *typePrédef*:

  abortretryignore
  : Affiche trois boutons dont les noms symboliques sont *abort*, *retry* et *ignore*.

  ok
  : Affiche un seul bouton dont le nom symbolique est *ok*.

  okcancel
  : Affiche deux boutons dont les noms symboliques sont *ok* et *cancel*.

retrycancel
: Affiche deux boutons dont les noms symboliques sont *retry* et *cancel*.

yesno
: Affiche deux boutons dont les noms symboliques sont *yes* et *no*.

yesnocancel
: Affiche trois boutons dont les noms symboliques sont *yes*, *no* et *cancel*.

Exemple

```
set answer [tk_messageBox -message "Voulez-vous vraiment quitter ?" \
    -type yesno -icon question]
switch -- $answer {
    yes exit
    no {tk_messageBox -message "Opération annulée" -type ok}
}
```

tk_optionMenu

Syntaxe

Tk_optionMenu *nomMenu nomVar valeur ?valeur valeur ...?*

Description

Cette procédure crée un bouton de menu optionnel dont le nom est *nomMenu*, ainsi que le menu associé. Ils permettent à l'utilisateur de sélectionner une des valeurs spécifiées dans les arguments *valeur*. La valeur courante sera stockée dans la variable globale dont le nom est spécifié par l'argument *nomVar* et elle sera également affichée comme étiquette du bouton de menu.

L'utilisateur peut cliquer sur le bouton de menu pour afficher un menu contenant toutes les valeurs ci-dessus et en sélectionner une. Une fois sélectionnée, cette valeur est stockée dans la variable *nomVar* et apparaît dans le bouton de menu. La valeur courante peut aussi être modifiée en fixant une autre valeur pour la variable.

La procédure **Tk_optionMenu** renvoie le nom du menu associé à *nomMenu* afin que la procédure appelante puisse le manipuler ou modifier ses options de configuration.

tk_setPalette

Syntaxe

Tk_setPalette *fond*
Tk_setPalette *nom valeur ?nom valeur ...?*
Tk_bisque

Description

La procédure **Tk_setPalette** modifie le schéma des couleurs de Tk. Elle le fait en modifiant les couleurs des composants existants et en changeant la base de données d'options afin que les futurs composants graphiques utilisent le nouveau schéma. Si **Tk_setPalette** est invoquée avec un unique argument, cet argument représentera le nom d'une couleur à utiliser comme couleur de fond normale : la fonction calculera une palette de couleurs complète à partir de cette couleur de fond. Sinon, les arguments de **Tk_setPalette** peuvent être un nombre arbitraire de paires de type *nom-valeur*, où *nom* désigne le nom d'une option dans la base de ressources de Tk et *valeur* est la nouvelle valeur à utiliser pour cette option. Les termes suivants de la base de ressource sont actuellement supportés :

activeBackground	foreground	selectColor
activeForeground	highlightBackground	selectBackground
background	highlightColor	selectForeground
disabledForeground	insertBackground	troughColor

La fonction **Tk_setPalette** s'efforce de calculer des valeurs par défaut raisonnables pour les options qui n'ont pas été spécifiées. On peut d'autre part spécifier des options autres que celles qui viennent d'être mentionnées et Tk répercutera ces changements sur les composants. Cela s'avèrera utile si l'on a défini de nouveaux composants ayant des options propres.

Une fois qu'elle a calculé de nouvelles valeurs pour chacune des options de couleur, la fonction **Tk_setPalette** parcourt la hiérarchie des composants graphiques afin d'en modifier les options en conséquence. Pour chaque composant, elle vérifie si certaines des options ci-dessus sont destinées à ce composant. Si c'est le cas et si la valeur courante de l'option est sa valeur par défaut, alors elle est modifiée. Si l'option a une valeur autre que la valeur par défaut, **Tk_setPalette** ne modifie rien. La valeur par défaut pour un composant est la valeur renvoyée par l'instruction

```
lindex [$w configure $option] 3
```

à moins que la commande **Tk_setPalette** ait déjà été invoquée, auquel cas c'est la valeur fixée au cours de cette invocation.

Après avoir modifié tous les composants de l'application, **Tk_setPalette** ajoute les options à la base de ressource pour que soient modifiées les valeurs par défaut pour les composants qui seront créés par la suite. Les nouvelles options sont ajoutées avec la priorité **composantDefault**, ce qui fait qu'elles seront supplantées par

toute option spécifiée dans le fichier .Xdefaults ou par des options spécifiées sur la ligne d'instruction qui crée le composant.

La procédure **Tk_bisque** est fournie uniquement pour des raisons de compatibilité : elle rétablit le schéma de couleurs brun clair (bisque) qui était en vigueur dans les versions 3.6 et antérieures de Tk.

tk_popup

Syntaxe

Tk_popup *menu x y ?indice?*

Description

Cette procédure affiche un menu à une position donnée sur l'écran et configure Tk de telle sorte que ce menu et tous ses sous-menus puissent être parcourus avec la souris ou le clavier. L'argument *menu* est le nom du menu et x et y sont les coordonnées de base où ce menu doit être affiché. Si l'argument *indice* est omis ou est une chaîne vide, c'est le coin supérieur gauche du menu qui est positionné en x et y. Sinon *indice* désigne l'indice d'un article du menu et le menu sera positionné de telle sorte que cet article se place au point donné.

tkerror

Commande invoquée pour traiter les erreurs d'arrière-plan.

Syntaxe

tkerror *message*

Description

Depuis la version 4.1 de Tk, la commande **tkerror** a été rebaptisée **bgerror** car la boucle d'événements qui l'invoque habituellement fait maintenant partie de Tcl. Pour des raisons de compatibilité, la commande **bgerror** fournie par la version courante de Tk essaie toujours d'appeler une commande **tkerror** s'il y en a une afin que des scripts anciens qui définissent ce gestionnaire d'erreurs continuent de fonctionner mais il est recommandé de les modifier si possible pour utiliser maintenant **bgerror** : la commande **tkerror** est appelée à disparaître.

Si l'appel à **tkerror** échoue, **bgerror** affiche un dialogue montrant l'erreur et proposant de montrer la trace de la pile à l'utilisateur.

Pour implémenter son propre mécanisme de gestion d'erreur il faut redéfinir directement **bgerror** plutôt que **tkerror**.

Variables globales

Variables internes maintenues par Tk.

Description

Les variables globales Tcl suivantes sont fixées par Tk et utilisées à différents stades de son exécution :

tk_library
: Cette variable contient le nom d'un répertoire possédant une bibliothèque de scripts Tcl relatifs à Tk. Ces scripts incluent un fichier d'initialisation qui est normalement exécuté chaque fois qu'une application Tk démarre ainsi que plusieurs fichiers comportant les procédures qui implémentent les comportements par défaut des composants graphiques.
La valeur initiale de **tk_library** est fixée lorsque Tk est ajouté à un interpréteur ; cela se fait en recherchant parmi un certain nombre de répertoires, un script de démarrage approprié pour Tk. Si la *variable d'environnement* TK_LIBRARY existe, le répertoire qu'elle désigne est examiné en premier. Si TK_LIBRARY n'est pas fixée ou ne se réfère pas à un répertoire approprié, Tk examine d'autres répertoires en fonction d'un emplacement qui a été défini à la compilation, de l'emplacement du répertoire de la bibliothèque de Tcl, de l'emplacement du répertoire contenant les exécutables et enfin du répertoire courant. Cette variable peut être modifiée par une application pour permettre par exemple de basculer à une autre bibliothèque.

tk_patchLevel
: Cette variable contient un entier décimal indiquant le niveau de modification (*patchlevel*) de Tk. Ce numéro est augmenté à chaque nouvelle version ou à chaque nouveau lot de modifications (*patch*) : il identifie de manière unique une version officielle particulière de Tk.

tk_textRedraw
tk_textRelayout
: Ces variables sont fixées par les widgets de texte lorsque le débogage est installé. Les valeurs écrites dans ces variables peuvent être utilisées pour tester les opérations sur les widgets de texte. Elles sont utilisées principalement par la suite de tests de Tk.

tk_strictMotif
: Cette variable vaut 0 par défaut. Si une application lui donne la valeur 1, Tk tente de se rapprocher le plus possible de l'apparence des réglages standard de l'interface Motif. Par exemple, les éléments actifs tels que boutons et barres de défilement ne changeront pas de couleur quand le curseur de la souris passera dessus.

tk_version
: Tk fixe cette variable dans l'interpréteur pour chaque application. La variable

désigne le numéro de version courant de la bibliothèque Tk sous la forme *majeur.mineur*. Les arguments *majeur* et *mineur* sont des nombres entiers. Le numéro majeur augmente pour une nouvelle version de Tk qui comportera des changements incompatibles avec les précédentes versions. Le numéro mineur augmente à toute nouvelle version de Tk mais est remis à zéro lorsque le numéro majeur change.

Mentionnons aussi la variable **tk::Priv** qui est un tableau comportant un certain nombre d'informations privées concernant Tk. Les éléments de **tk::Priv** sont utilisés par des procédures de la bibliothèque et par les liaisons par défaut. Ils ne devraient jamais être modifiés en dehors de Tk.

tkwait

Attend qu'une variable soit modifiée ou qu'une fenêtre soit supprimée.

Syntaxe

tkwait variable *nom*
tkwait visibility *nom*
tkwait window *nom*

Description

La commande **tkwait** attend que certains événements se produisent et retourne alors sans entreprendre aucune action. La valeur de retour est toujours une chaîne vide :

- Si le premier argument est **variable** (ou une abréviation), le second argument sera le nom d'une variable globale et la commande **tkwait** forcera à attendre que cette variable soit modifiée.
- Si le premier argument est **window** (ou une abréviation), le second argument sera le nom d'une fenêtre et la commande **tkwait** forcera à attendre que cette fenêtre soit détruite. Cette forme est typiquement utilisée pour attendre qu'un utilisateur ait fini d'interagir avec une certaine fenêtre de dialogue avant d'utiliser le résultat de cette interaction.
- Si le premier argument est **visibility** (ou une abréviation), le second argument sera le nom d'une fenêtre et la commande **tkwait** forcera à attendre que l'état de visibilité de la fenêtre change[3]. Cette forme est typiquement utilisée pour attendre qu'une nouvelle fenêtre apparaisse effectivement à l'écran à la suite d'une certaine action.

Pendant que la commande **tkwait** attend, les événements sont traités de façon normale et l'application n'est pas figée : elle peut continuer de réagir aux interactions en provenance de l'utilisateur. Si une procédure de rappel invoque la commande **tkwait** à nouveau, l'appel imbriqué à cette commande doit s'achever avant que le premier appel puisse lui même terminer.

3. Techniquement, la commande attend de recevoir la notification d'un événement *VisibilityNotify*.

toplevel

Permet de créer et de manipuler des fenêtres de premier niveau.

Syntaxe

toplevel *chemin ?options?*

Description

La commande **toplevel** crée une nouvelle fenêtre (spécifiée par l'argument *chemin*). On peut spécifier des options supplémentaires soit par une ligne de commandes, soit directement dans la base de ressources pour configurer certaines caractéristiques du composant telles que la couleur de fond ou le relief. La commande **toplevel** renvoie la valeur de son argument *chemin*.

Un composant de type *toplevel* est analogue à un composant de type *frame* sauf qu'il est créé comme une fenêtre de premier niveau : son parent dans le système de fenêtrage X est la fenêtre racine d'un écran et non le parent logique déduit de son nom de chemin. La finalité des composants de type *toplevel* est de servir de conteneur pour des boîtes de dialogue et des collections d'autres objets. Les seules caractéristiques visibles d'un composant de premier niveau sont sa couleur de fond et sa bordure tridimensionnelle optionnelle pour le faire apparaître en creux ou en relief.

Opérations sur un composant de premier niveau

Avec l'instruction :
toplevel *chemin ?options?*
la commande **toplevel** crée une nouvelle commande Tcl dont le nom est précisément *chemin*. Cette commande peut être complétée par des sous-commandes permettant d'effectuer diverses opérations concernant la fenêtre. La forme générale est :
chemin opération ?arg arg ...?

Les opérations reconnues par les composantes de type *toplevel* sont les suivantes :

chemin **cget** *option*
: Renvoie la valeur courante de l'option de configuration spécifiée par l'argument *option*. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **toplevel**.

chemin **configure** *?option??valeur option valeur ...?*
: Permet d'obtenir ou de modifier les options de configuration de la fenêtre. Si l'argument *option* n'est pas spécifié, la commande renvoie une liste de toutes les options disponibles pour *chemin*.
Si l'argument *option* est spécifié sans une valeur correspondante, la commande renvoie une liste décrivant l'option nommée. Si un ou plusieurs couples *option-*

-borderwidth ou -bd	borderWidth	BorderWidth
-cursor	cursor	Cursor
-highlightbackground	highlightBackground	HighlightBackground
-highlightcolor	highlightColor	HighlightColor
-highlightthickness	highlightThickness	HighlightThickness
-relief	relief	Relief
-takefocus	takeFocus	TakeFocus

TAB. 18 – *Options standard reconnues par les composants de type* toplevel.

valeur sont spécifiés, les options en question se voient attribuer la valeur correspondante : la commande renvoie alors une chaîne vide. L'argument *option* peut être une quelconque des valeurs acceptées par la commande **toplevel**.

Liaisons

Lorsqu'un nouveau composant de premier niveau est créé, il ne possède aucune liaison par défaut. Ils n'ont, en effet, pas pour fonction d'être interactifs.

Options standard

Les options standard supportées par les composants de type *toplevel* sont rassemblées dans le tableau 18. Ces options sont décrites en détail à la page 142.

Options spécifiques

-background **background** **Background**

Cette option est identique à l'option standard **-background** si ce n'est que sa valeur peut aussi être spécifiée comme une chaîne vide. Dans ce cas-là, l'objet n'affichera aucun fond et aucune bordure, et aucune couleur de sa carte de couleurs ne sera utilisée pour le fond et la bordure.

-class **class** **Class**

Spécifie une classe pour la fenêtre. Cette classe sera utilisée pour rechercher dans la base de données de ressources d'autres options pour la fenêtre ou par la suite pour établir des liaisons (*bindings*). L'option **class** ne peut pas être modifiée au moyen de la commande **configure**.

-colormap **colormap** **Colormap**

Spécifie une carte de couleurs à utiliser pour la fenêtre. La valeur peut être ou bien *new*, auquel cas une nouvelle carte de couleurs est créée pour la fenêtre et ses enfants, ou le nom d'une autre fenêtre (qui doit être sur le même écran et avoir les mêmes caractéristiques visuelles que *chemin*), auquel cas la nouvelle fenêtre

utilisera la carte de couleurs de la fenêtre spécifiée. Si l'option **colormap** n'est pas spécifiée, la nouvelle fenêtre utilise la carte de couleurs par défaut de son écran. Cette option ne peut pas être modifiée au moyen de la commande **configure**.

-container **container** **Container**

Cette option doit prendre une valeur booléenne. Si la valeur est *true*, cela signifie que cette fenêtre sera utilisée pour contenir une autre application (par exemple, un objet Tk de type *toplevel* peut être inséré au moyen de l'option **-use**). La fenêtre supportera les protocoles appropriés du gestionnaire de fenêtres pour les requêtes concernant les caractéristiques géométriques. La fenêtre ne doit pas avoir d'enfants dans cette application. Cette option ne peut pas être modifiée au moyen de la commande **configure**.

-height **height** **Height**

Spécifie la hauteur désirée pour la fenêtre en unités d'écran. Si cette option est négative ou nulle, la fenêtre ne requerra aucune dimension.

-menu **menu** **Menu**

Spécifie un objet de type menu à utiliser comme une barre de menus. Sur un Macintosh, la barre de menus sera affichée au sommet du moniteur principal tandis que sur les plates-formes Unix ou sous Windows, le menu apparaît au sommet de la fenêtre de premier niveau.

-screen **screen** **Screen**

Spécifie l'écran sur lequel placer une nouvelle fenêtre. N'importe quel écran valide peut être utilisé, même s'il est associé à un autre moniteur. Par défaut le même écran que le parent est utilisé. Cette option est particulière en ce sens qu'elle ne peut pas être spécifiée dans la base de données de ressources, ni être modifiée au moyen de la commande **configure**.

-use **use** **Use**

Cette option est utilisée pour les imbrications. Si la valeur attribuée à l'option n'est pas une chaîne vide, elle doit être l'identificateur d'une fenêtre contenante, exprimée sous la forme d'une chaîne hexadécimale analogue à celles que renvoie la commande **winfo id**. L'objet de type *toplevel* doit être créé comme enfant du conteneur donné au lieu de la fenêtre racine de l'écran. Si la fenêtre conteneur est dans une application Tk, il doit s'agir d'un objet de type *toplevel* ou *frame* pour lequel l'option **-container** a été spécifiée. Cette option ne peut pas être modifiée au moyen de la commande **configure**.

-visual **visual** **Visual**

Spécifie la classe visuelle de la nouvelle fenêtre. Sa valeur peut être une des constantes symboliques suivantes : *directcolor*, *grayscale*, *greyscale*, *pseudocolor*, *staticcolor*, *staticgray*, *staticgrey*, *truecolor*, *best*, *default*, ou le nom d'une autre fenêtre

(cf. p. **??**). Si cette option n'est pas spécifiée, la nouvelle fenêtre utilisera les caractéristiques visuelles de son écran. L'option **visual** ne peut pas être modifiée au moyen de la commande **configure**.

-width **width** **Width**

Spécifie la largeur désirée pour la fenêtre en unités d'écran. Si cette option est négative ou nulle, la fenêtre ne requerra aucune dimension.

winfo

Renvoie l'information relative aux fenêtres.

Syntaxe

winfo *sous-commande ?arg arg ...?*

Description

La commande **winfo** est utilisée pour recueillir l'information concernant les fenêtres gérées par Tk. Elle peut prendre plusieurs formes différentes en fonction de la sous-commande qui lui est adjointe. Les formes reconnues sont les suivantes :

winfo atom ?**-displayof** *fenêtre*? *nomAtome*
: Renvoie une chaîne décimale indiquant l'identificateur complet de l'atome dont le nom est *nomAtome*. Si aucun atome de ce nom n'existe, un nouveau est créé. Si l'option **-displayof** est spécifiée, l'atome est recherché sur l'écran de la fenêtre désignée par l'argument *fenêtre* ; autrement il est recherché sur l'écran de la fenêtre principale de l'application.

winfo atomname ?**-displayof** *fenêtre*? *identificateur*
: Renvoie le nom textuel de l'atome dont l'identificateur entier est *identificateur*. Si l'option **-displayof** est spécifiée, l'identificateur est recherché sur l'écran de la fenêtre désignée par l'argument *fenêtre* ; autrement il est recherché sur l'écran de la fenêtre principale de l'application. Cette commande est l'inverse de la commande **winfo atom**. Elle génère une erreur si aucun atome n'existe.

winfo cells *fenêtre*
: Renvoie une chaîne décimale indiquant le nombre d'entrées dans la carte de couleurs de *fenêtre*.

winfo children *fenêtre*
: Renvoie une liste contenant les noms de chemin de tous les descendants de *fenêtre*. La liste est dans l'ordre d'empilement, avec en premier la fenêtre placée le plus bas. Les fenêtres de premier niveau (créées par la commande **toplevel**) sont renvoyées comme descendants de leurs parents logiques.

winfo class *fenêtre*
: Renvoie le nom de classe de *fenêtre*.

winfo colormapfull *fenêtre*
: Renvoie 1 si la carte de couleurs de *fenêtre* est pleine, sinon 0. La carte de couleurs d'une fenêtre est réputée pleine si la dernière tentative d'allouer une nouvelle couleur à la fenêtre a échoué et si l'application n'a pas libéré de couleurs de cette table depuis l'échec de l'allocation.

winfo containing ?**-displayof** *fenêtre*? *racineX racineY*
: Renvoie le nom de chemin de la fenêtre contenant le point de coordonnées

racineX et *racineY*. Celles-ci sont spécifiées en unités d'écran dans le système de coordonnées de la fenêtre racine (y compris pour les fenêtres virtuelles). Si l'option **-displayof** est spécifiée, les coordonnées se réfèrent à l'écran contenant *fenêtre*; autrement elles se réfèrent à l'écran de la fenêtre principale de l'application. Si aucune fenêtre de cette application ne contient le point, une chaîne vide est renvoyée. En sélectionnant la fenêtre contenante, les enfants reçoivent une priorité supérieure à celle de leurs parents et parmi les enfants de même niveau les plus haut placés dans l'ordre d'empilement sont choisis.

winfo depth *fenêtre*
Renvoie une chaîne décimale indiquant la profondeur de *fenêtre* (nombre de bits par pixel).

winfo exists *fenêtre*
Renvoie 1 s'il existe une fenêtre nommée *fenêtre*, sinon 0.

winfo fpixels *fenêtre nombre*
Renvoie une valeur en virgule flottante indiquant le nombre de pixels dans *fenêtre* correspondant à la distance spécifiée par l'argument *nombre*. Ce dernier peut être spécifié en unités d'écran. La valeur de retour peut être décimale ; pour obtenir une valeur entière, on utilisera plutôt **winfo pixels**.

winfo geometry *fenêtre*
Renvoie le dimensions géométriques de *fenêtre*, sous la forme :
largeur **x** *hauteur* $+x$ $+y$
Toutes les dimensions sont en pixels.

winfo height *fenêtre*
Renvoie une chaîne décimale indiquant la hauteur de *fenêtre* en pixels. Lorsqu'une fenêtre est créée pour la première fois, sa hauteur est de 1 pixel ; la hauteur est ensuite adaptée par le gestionnaire de placement pour satisfaire les exigences de la fenêtre. Si l'on a besoin d'une hauteur réelle immédiatement après la création du composant, il faut invoquer la commande **update** pour forcer le gestionnaire de placement à recalculer, ou encore utiliser la commande **winfo reqheight** pour attribuer à la fenêtre une hauteur requise plutôt que sa hauteur naturelle.

winfo id *fenêtre*
Renvoie une chaîne hexadécimale indiquant un identificateur de bas niveau dépendant de la plate-forme pour l'argument *fenêtre*. Sur les plates-formes Unix, il s'agit de l'identificateur de fenêtre X. Sous Windows, il s'agit du HWND (*Window Handle*). Sous MacOS, la valeur n'a pas de signification particulière en dehors de Tk.

winfo interps ?**-displayof** *fenêtre*?
Renvoie une liste dont les membres sont les noms de tous les interpréteurs Tcl (autrement dit toutes les applications Tk) actuellement enregistrées pour un moniteur particulier. Si l'option **-displayof** est spécifiée, la valeur renvoyée se réfère à l'écran contenant *fenêtre*; autrement elle se réfère à l'écran de la fenêtre principale de l'application.

winfo ismapped *fenêtre*
Renvoie 1 si *fenêtre* est actuellement affichée, sinon 0.

winfo manager *fenêtre*
Renvoie le nom du gestionnaire de placement actuellement responsable de *fenêtre*, ou une chaîne vide si *fenêtre* n'est gérée par aucun gestionnaire de placement. Le nom est habituellement celui de la commande Tcl pour le gestionnaire de placement, comme par exemple **pack** ou **place**. Si le gestionnaire de placement est lui-même un composant, comme un canevas ou un objet texte, le nom est la commande de classe de ce composant, comme par exemple *canvas*.

winfo name *fenêtre*
Renvoie le nom de *fenêtre* (autrement dit son nom à l'intérieur de son parent, par opposition au nom de chemin complet). Le cas de la racine « . » est particulier. L'instruction

```
winfo name .
```

renvoie le nom de l'application.

winfo parent *fenêtre*
Renvoie le nom de chemin du parent de *fenêtre*, ou une chaîne vide si *fenêtre* est la fenêtre principale de l'application.

winfo pathname ?**-displayof** *fenêtre*? *identificateur*
Renvoie le nom de chemin de la fenêtre dont l'identificateur X est *identificateur*. L'argument *identificateur* doit être un entier décimal, hexadécimal ou octal et doit correspondre à la fenêtre de l'application invoquante. Si l'option **-displayof** est spécifiée, l'identificateur est recherché sur l'écran de la fenêtre désignée par l'argument *fenêtre*; autrement il est recherché sur l'écran de la fenêtre principale de l'application.

winfo pixels *fenêtre nombre*
Renvoie le nombre de pixels dans *fenêtre* correspondant à la distance spécifiée par l'argument *nombre*. Ce dernier est exprimé en unités d'écran. Le résultat est arrondi à l'entier le plus proche; pour obtenir une valeur fractionnaire, utiliser **winfo fpixels**.

winfo pointerx *fenêtre*
winfo pointery *fenêtre*
winfo pointerxy *fenêtre*
Si le pointeur de la souris est sur le même écran que *fenêtre*, cette commande renvoie respectivement l'abscisse ou l'ordonnée ou les deux coordonnées du pointeur, mesurées en pixels dans la fenêtre racine de l'écran. Si une fenêtre racine virtuelle est utilisée sur l'écran, la position est mesurée dans la racine virtuelle. Si le pointeur de la souris n'est pas sur le même écran que *fenêtre*, une valeur -1 est renvoyée.

winfo reqheight *fenêtre*
winfo reqwidth *fenêtre*
Renvoie une chaîne décimale indiquant respectivement la hauteur ou la largeur requises pour *fenêtre* en pixels. C'est la valeur utilisée par le gestionnaire

de géométrie de la fenêtre pour calculer ses dimensions.

winfo rgb *fenêtre couleur*
Renvoie une liste contenant trois valeurs décimales, qui sont les intensités de rouge, de vert et de bleu correspondant à la couleur de la fenêtre désignée par l'argument *fenêtre*. L'argument *couleur* peut être spécifié dans n'importe laquelle des formes acceptées pour les options de couleur.

winfo rootx *fenêtre*
winfo rooty *fenêtre*
Renvoie une chaîne décimale indiquant respectivement l'abscisse ou l'ordonnée, dans la fenêtre racine de l'écran, du coin supérieur gauche de la bordure de *fenêtre* ou de *fenêtre* elle-même si elle n'a pas de bordure.

winfo screen *fenêtre*
Renvoie le nom de l'écran associé à *fenêtre*, sous la forme *nomEcran.indiceEcran*.

winfo screencells *fenêtre*
Renvoie une chaîne décimale indiquant le nombre d'entrées dans la carte de couleurs par défaut de l'écran de *fenêtre*.

winfo screendepth *fenêtre*
Renvoie une chaîne décimale indiquant la profondeur de la fenêtre racine de l'écran de *fenêtre* (nombre de bits par pixel).

winfo screenheight *fenêtre*
Renvoie une chaîne décimale indiquant la hauteur de l'écran de *fenêtre*, en pixels.

winfo screenwidth *fenêtre*
Renvoie une chaîne décimale indiquant la largeur de l'écran de *fenêtre*, en pixels.

winfo screenmmheight *fenêtre*
Renvoie une chaîne décimale indiquant la hauteur de l'écran de *fenêtre*, en millimètres.

winfo screenmmwidth *fenêtre*
Renvoie une chaîne décimale indiquant la largeur de l'écran de *fenêtre*, en millimètres.

winfo screenvisual *fenêtre*
Renvoie une des valeurs symboliques suivantes pour indiquer la classe visuelle par défaut de l'écran de *fenêtre*: *directcolor*, *grayscale*, *pseudocolor*, *staticcolor*, *staticgray* ou *truecolor*.

winfo server *fenêtre*
Renvoie une chaîne contenant l'information à propos du serveur du moniteur de *fenêtre*. Le format exact de cette chaîne varie d'une plate-forme à l'autre. Pour les serveurs X, la chaîne a la forme
X *maj***R** *min vendeur vsVendeur*
où *maj* et *min* désignent respectivement les numéros de version et de révision fournis par le serveur (par exemple X11R5), *vendeur* est le nom du vendeur du serveur, et *vsVendeur* est un numéro de version fourni par le vendeur.

winfo toplevel *fenêtre*
Renvoie le nom de chemin de la fenêtre de premier niveau contenant *fenêtre*.

winfo viewable *fenêtre*
Renvoie 1 si *fenêtre* et tous ses ancêtres jusqu'à la fenêtre la plus proche du niveau supérieur sont affichés. Renvoie 0 sinon.

winfo visual *fenêtre*
Renvoie une des valeurs symboliques suivantes pour indiquer la classe visuelle par défaut de la fenêtre désignée par l'argument *fenêtre* : *directcolor*, *grayscale*, *pseudocolor*, *staticcolor*, *staticgray* ou *truecolor*.

winfo visualid *fenêtre*
Renvoie l'identificateur X pour la classe visuelle de *fenêtre*.

winfo visualsavailable *fenêtre* ?**inclureIds**?
Renvoie une liste dont les éléments décrivent les classes visuelles disponibles pour l'écran de *fenêtre*. Chaque élément consiste en une classe visuelle suivie d'une profondeur en nombre entier. La classe a la même forme que la valeur renvoyée par la commande **winfo visual**. La profondeur indique le nombre de bits par pixel dans la classe. En outre, si l'argument **inclureIds** est fourni, la profondeur est suivie par l'identificateur X de la classe visuelle.

winfo vrootheight *fenêtre*
Renvoie la hauteur de la fenêtre racine virtuelle associée à *fenêtre* s'il en existe une ; autrement, renvoie la hauteur de l'écran de *fenêtre*.

winfo vrootwidth *fenêtre*
Renvoie la largeur de la fenêtre racine virtuelle associée à *fenêtre* s'il en existe une ; autrement, renvoie la largeur de l'écran de *fenêtre*.

winfo vrootx *fenêtre*
winfo vrooty *fenêtre*
Renvoie le décalage en abscisse ou en ordonnée respectivement de la fenêtre racine virtuelle associée à *fenêtre*, relativement à la fenêtre racine de son écran. Cette valeur est normalement négative ou nulle. Renvoie 0 s'il n'y a pas de fenêtre racine virtuelle pour *fenêtre*.

winfo width *fenêtre*
Renvoie une chaîne décimale indiquant la largeur de *fenêtre* en pixels. Lorsqu'une fenêtre est créée pour la première fois, sa largeur est de 1 pixel ; la largeur est ensuite adaptée par le gestionnaire de placement pour satisfaire les exigences de la fenêtre. Si l'on a besoin d'une largeur réelle immédiatement après la création du composant, il faut invoquer la commande **update** pour forcer le gestionnaire de placement à recalculer immédiatement, ou encore utiliser la commande **winfo reqwidth** pour attribuer à la fenêtre une largeur requise plutôt que la largeur réelle.

winfo x *fenêtre*
winfo y *fenêtre*
Renvoie une chaîne décimale indiquant l'abscisse ou l'ordonnée respectivement, dans le parent de *fenêtre* du coin supérieur gauche de la bordure de *fenêtre* ou de *fenêtre* elle-même si elle n'a pas de bordure.

wm

Permet de communiquer avec le serveur de fenêtres.

Syntaxe

wm *sous-commande fenêtre ?args?*

Description

La commande **wm** (abréviation de *window manager*) est utilisée pour interagir avec les gestionnaires de fenêtres afin de contrôler des éléments tels que le titre d'une fenêtre, ses dimensions, ou la manière de la redimensionner.

Propriétés géométriques

Par défaut, une fenêtre de premier niveau apparaît à l'écran dans sa taille naturelle qui est celle qui est déterminée de façon interne en fonction de ses composants et du type de gestionnaire de placement utilisé (*pack*, *grid* ou *place*). Si la taille naturelle d'une fenêtre de premier niveau change, la taille de la fenêtre change en conséquence. Une fenêtre de premier niveau peut se voir attribuer une taille autre que sa taille naturelle de deux façons différentes :

- l'utilisateur peut redimensionner la fenêtre manuellement ;
- l'application peut exiger une taille particulière au moyen d'une commande **wm geometry**.

Ces deux cas sont traités de manière identique par Tk : la taille requise l'emporte sur la taille naturelle. Pour qu'une fenêtre retrouve par la suite sa taille naturelle, il suffit d'invoquer une commande **wm geometry** avec une chaîne vide.

Normalement une fenêtre de premier niveau peut avoir une taille allant de 1 pixel dans chaque dimension à la taille complète de l'écran. Cependant, les commandes **wm minsize** et **wm maxsize** permettent d'imposer des limites inférieures et supérieures différentes. L'intervalle fixé par ces commandes s'applique à toutes les formes de redimensionnement. On peut aussi utiliser la commande **wm resizable** pour activer ou désactiver le redimensionnement interactif dans une dimension particulière ou dans les deux à la fois.

Sous-commandes

La commande **wm** possède de nombreuses sous-commandes. Toutes les sous-commandes attendent au moins un argument supplémentaire, *fenêtre*, qui doit être le nom de chemin d'une fenêtre de premier niveau. Les sous-commandes définies sont les suivantes :

wm aspect *fenêtre ?minNumer minDenom maxNumer maxDenom?*
Si *minNumer*, *minDenom*, *maxNumer* et *maxDenom* sont tous spécifiés, ils

seront transmis au serveur de fenêtres qui les utilise pour imposer un intervalle de ratios d'aspect acceptables pour *fenêtre*. Le ratio d'aspect de *fenêtre* (rapport entre la largeur et la hauteur) se situera entre les rapports *minNumer*/*minDenom* et *maxNumer*/*maxDenom*. Si *minNumer* etc. sont tous spécifiés comme des chaînes vides, toute restriction existante concernant les ratios d'aspect sera annulée. Si *minNumer* etc. sont spécifiés, la commande renvoie une chaîne vide. Sinon, elle renvoie une liste Tcl contenant quatre éléments qui sont les valeurs courantes de *minNumer*, *minDenom*, *maxNumer* et *maxDenom* (si aucune restriction d'aspect n'est en vigueur, une chaîne vide est renvoyée).

wm client *fenêtre ?nom?*

Si l'argument *nom* est spécifié, cette commande stocke *nom* (qui désigne le nom d'un hôte sur lequel l'application est exécutée) dans la propriété WM_CLIENT_MACHINE de *fenêtre* pour une utilisation par le serveur de fenêtres ou le gestionnaire de session. La commande renvoie une chaîne vide dans ce cas. Si *nom* n'est pas spécifié, la commande renvoie le dernier nom déclaré par une instruction **wm client** pour la fenêtre. Si *nom* est spécifié comme une chaîne vide, la commande détruit la propriété WM_CLIENT_MACHINE de *fenêtre*.

wm colormapwindows *fenêtre ?listeFenêtres?*

Cette commande sert à manipuler la propriété WM_COLORMAP_WINDOWS, qui fournit des informations au serveur de fenêtres à propos des fenêtres possédant des cartes de couleurs privées. Si *listeFenêtres* n'est pas spécifié, la commande renvoie une liste dont les éléments sont les noms des fenêtres dans la propriété WM_COLORMAP_WINDOWS.

Si *listeFenêtres* est spécifié, il consiste en une liste de noms de chemin de fenêtres ; la commande réécrit la propriété WM_COLORMAP_WINDOWS avec les fenêtres données et renvoie une chaîne vide.

La propriété WM_COLORMAP_WINDOWS devrait normalement contenir une liste des fenêtres intérieures à *fenêtre* et dont la carte de couleurs diffère de celle de leurs parents. L'ordre des fenêtres dans la propriété indique un ordre de priorité : le serveur de fenêtres essaiera d'installer autant de cartes de couleurs que possible à partir de la tête de cette liste lorsque *fenêtre* est focalisée. Si *fenêtre* ne fait pas partie de *listeFenêtres*, Tk l'ajoute implicitement à la fin de la propriété WM_COLORMAP_WINDOWS, de telle sorte que sa carte de couleurs ait la priorité la plus basse.

Si la commande **wm colormapwindows** n'est pas invoquée, Tk fixera automatiquement la propriété pour chaque fenêtre de premier niveau à toutes les fenêtres intérieures dont les cartes de couleurs diffèrent de celles de leurs parents, suivies par celle de premier niveau elle-même ; l'ordre des fenêtres intérieures n'est pas défini. On se reportera à la documentation ICCCM (*Inter-Client Communication Conventions Manual*) pour plus d'information concernant la propriété WM_COLORMAP_WINDOWS.

wm command *fenêtre ?valeur?*
Si l'argument *valeur* est spécifié, cette commande stocke *valeur* dans la propriété WM_COMMAND de la fenêtre en vue d'une utilisation par le serveur de fenêtres ou le gestionnaire de session et renvoie une chaîne vide. L'argument *valeur* doit avoir une structure de liste valide ; les éléments doivent contenir les termes de la commande utilisée pour invoquer l'application. Si *valeur* n'est pas spécifiée, la commande renvoie la dernière valeur fixée par une instruction **wm command** pour *fenêtre*. Si *valeur* est spécifiée comme une chaîne vide, la commande détruit la propriété WM_COMMAND de *fenêtre*.

wm deiconify *fenêtre*
Fait en sorte que *fenêtre* soit affichée sous forme normale (non iconifiée). Si la fenêtre n'a jamais été installée, cette commande ne l'installera pas, mais lorsqu'elle sera installée pour la première fois, elle le sera sous forme non iconifiée. Sous Windows, une fenêtre mise sous forme normale est remontée au premier plan et focalisée. Cette commande renvoie une chaîne vide.

wm focusmodel *fenêtre* ?**active**—**passive**?
Si **active** ou **passive** sont spécifiés comme argument optionnel à la commande, ils indiquent le modèle de focalisation pour *fenêtre*. Dans ce cas, la commande renvoie une chaîne vide. Si aucun argument supplémentaire n'est spécifié, la commande renvoie le modèle de focalisation courant pour *fenêtre*. La valeur **active** signifie que *fenêtre* réclamera la focalisation pour elle-même et pour ses descendants, même lorsque la focalisation est actuellement placée sur une autre application. La valeur **passive** signifie que *fenêtre* ne réclamera jamais la focalisation pour elle-même : c'est le gestionnaire de fenêtres qui assigne la focalisation à *fenêtre* au moment voulu. Cependant, une fois que la focalisation a été donnée à *fenêtre* ou à un de ses descendants, l'application peut réassigner la focalisation parmi les descendants de *fenêtre*. La valeur par défaut est **passive** et la commande **focus** de Tk suppose un modèle passif de focalisation.

wm frame *fenêtre*
Si le gestionnaire de fenêtres a dessiné un cadre de fenêtre autour de *fenêtre*, la commande renvoie l'identificateur de fenêtre (spécifique à la plate-forme) pour le cadre le plus extérieur contenant *fenêtre* (la fenêtre dont le parent est la racine ou la racine virtuelle). Si *fenêtre* n'a pas été « reparentée » par le gestionnaire de fenêtres, c'est-à-dire entourée d'un cadre, la commande renvoie l'identificateur de fenêtre spécifique à la plate-forme pour *fenêtre*.

wm geometry *fenêtre ?nouvellesDim?*
Si l'argument *nouvellesDim* est spécifié, la géométrie de *fenêtre* est modifiée et une chaîne vide est renvoyée. Sinon, la géométrie courante pour *fenêtre* est renvoyée. L'argument *nouvellesDim* a la forme :

$$=largeur \times hauteur \pm x \pm y$$

où l'un des termes =, $largeur \times hauteur$ ou $\pm x \pm y$ peut être omis. Les arguments *largeur* et *hauteur* sont des entiers positifs spécifiant les dimensions souhaitées de *fenêtre*. Si *fenêtre* est une grille, les dimensions sont spécifiées

en unités de grille ; sinon elles sont spécifiées en pixels. Les arguments x et y spécifient l'emplacement souhaité pour *fenêtre* sur l'écran, en pixels.

Si x est précédé d'un signe +, il spécifie le nombre de pixels entre le bord gauche de l'écran et le bord gauche de la bordure de *fenêtre* ; s'il est précédé d'un signe −, x spécifie le nombre de pixels entre le bord droit de l'écran et le bord droit de la bordure de *fenêtre*.

Si y est précédé de +, il spécifie le nombre de pixels entre le bord supérieur de l'écran et le bord supérieur de la bordure de *fenêtre* ; s'il est précédé de −, y spécifie le nombre de pixels entre le bord inférieur de l'écran et le bord inférieur de la bordure de *fenêtre*.

Enfin, si l'argument *nouvellesDim* est spécifié comme une chaîne vide, toutes dimensions spécifiées antérieurement par l'utilisateur pour *fenêtre* seront annulées et la fenêtre reprendra la taille requise intérieurement par ses composants.

wm grid *fenêtre ?largeurBase hauteurBase pasLargeur pasHauteur?*

Cette commande indique que *fenêtre* doit être considérée comme une fenêtre de grille. Elle spécifie également la relation entre les unités de grille et les pixels. Les arguments *largeurBase* et *hauteurBase* spécifient le nombre d'unités de grille correspondant aux dimensions en pixels requises intérieurement par *fenêtre*.

Les arguments *pasLargeur* et *pasHauteur* spécifient le pas, c'est-à-dire le nombre de pixels dans chaque unité de grille horizontale et verticale. Ces quatre valeurs déterminent un intervalle de tailles acceptables pour *fenêtre*, correspondant à des hauteurs et largeurs en termes de grille qui sont des entiers non négatifs.

Tk transmet cette information au gestionnaire de fenêtres ; au cours d'un redimensionnement manuel, le gestionnaire de fenêtres limitera la taille de la fenêtre à ces valeurs. De plus, le gestionnaire de fenêtres affichera la taille courante de la fenêtre en termes d'unités de grille plutôt qu'en pixels. Si *largeurBase* etc. sont tous spécifiés comme des chaînes vides, alors *fenêtre* ne sera plus considérée comme une fenêtre de grille. Si *largeurBase* etc. sont spécifiés, la valeur de retour de la commande est une chaîne vide. Sinon, la valeur de retour est une liste Tcl contenant quatre éléments correspondant aux valeurs courantes de *largeurBase*, *hauteurBase*, *pasLargeur* et *pasHauteur* ; si *fenêtre* n'est pas actuellement une grille, une chaîne vide est renvoyée.

On notera que l'option standard **-setGrid** des composants (ou la fonction **Tk_SetGrid** dans du code C) permettent un accès plus aisé à ces fonctionnalités.

wm group *fenêtre ?chemin?*

Si l'argument *chemin* est spécifié, il indique le nom de chemin de la tête d'un groupe de fenêtres associées. Le gestionnaire de fenêtres peut se servir de cette information, par exemple, pour désinstaller toutes les fenêtres d'un groupe lorsque la première fenêtre du groupe est iconifiée. L'argument *chemin* peut être spécifié comme une chaîne vide pour retirer *fenêtre* d'un groupe. Si

chemin est spécifié, la commande renvoie une chaîne vide ; sinon, elle renvoie le nom de chemin de la première fenêtre du groupe courant auquel appartient *fenêtre* ou une chaîne vide si *fenêtre* ne fait partie d'aucun groupe.

wm iconbitmap *fenêtre ?bitmap?*
Si l'argument *bitmap* est spécifié, il désigne une image bichrome (dite *bitmap* dans la terminologie de Tk) sous les formes standard acceptées pour ce type d'objet (cf. p. 24 et l'exemple p. **??**). Cette image bitmap est passée au gestionnaire de fenêtres pour être affichée comme icone de *fenêtre* et la commande renvoie une chaîne vide. Si une chaîne vide est spécifiée pour *bitmap*, toute icone bitmap courante est supprimée pour *fenêtre*. Si *bitmap* est spécifié, la commande renvoie une chaîne vide. Sinon, elle renvoie le nom de l'icone bitmap courante associée à *fenêtre* ou une chaîne vide si *fenêtre* n'a pas d'icone bitmap.

wm iconify *fenêtre*
Iconifie la fenêtre désignée par l'argument *fenêtre*. Si *fenêtre* n'a pas encore été installée, cette commande fera en sorte qu'elle apparaisse dans l'état iconifié lorsqu'elle sera installée.

wm iconmask *fenêtre ?bitmap?*
Si l'argument *bitmap* est spécifié, il désigne une image bichrome (dite *bitmap* dans la terminologie de Tk) sous les formes standard acceptées pour ce type d'objet (cf. p. 24). Cette image bitmap est passée au gestionnaire de fenêtres qui l'utilise comme masque en conjonction avec l'option **iconbitmap** : là où le masque contient des zéros, aucune icone n'est affichée ; là où il contient des 1, les bits correspondants de l'icone bitmap sont affichés. Si une chaîne vide est spécifiée pour *bitmap*, tout masque d'icone courant est supprimé pour *fenêtre*.
Si *bitmap* est spécifié, la commande renvoie une chaîne vide. Sinon, elle renvoie le nom du masque d'icone courant associé à *fenêtre* ou une chaîne vide si aucun masque n'est en vigueur.

wm iconname *fenêtre ?nouveauNom?*
Si l'argument *nouveauNom* est spécifié, il est passé au gestionnaire de fenêtres qui affiche alors *nouveauNom* dans l'icone associée à *fenêtre*. Dans ce cas, une chaîne vide est renvoyée comme résultat. Si *nouveauNom* n'est pas spécifié, la commande renvoie le nom d'icone courante pour *fenêtre*, ou une chaîne vide si aucun nom d'icone n'a été spécifié (dans ce cas, le gestionnaire de fenêtres affiche normalement le titre de la fenêtre tel qu'il a été défini avec la commande **wm title**).

wm iconposition *fenêtre ?x y?*
Si x et y sont spécifiés, ils sont passés au gestionnaire de fenêtres comme indication concernant la position de l'icone pour *fenêtre*. Dans ce cas, une chaîne vide est renvoyée. Si x et y sont spécifiés comme des chaînes vides, toute indication de position d'icone est annulée. Si ni x ni y ne sont spécifiés, la commande renvoie une liste Tcl contenant deux valeurs, qui sont les indications de position d'icone courante (si aucune indication n'est en vigueur,

une chaîne vide est renvoyée).

wm iconwindow *fenêtre ?chemin?*

Si l'argument *chemin* est spécifié, il désigne le nom de chemin pour une fenêtre à utiliser comme icone pour *fenêtre*: lorsque *fenêtre* est iconifiée, *chemin* sera affiché pour servir d'icone, et lorsque *fenêtre* est déiconifiée, *chemin* sera désinstallé. Si *chemin* est spécifié comme une chaîne vide, toute association de fenêtre d'icone pour *fenêtre* sera annulée.

Si l'argument *chemin* est spécifié, la valeur de retour est une chaîne vide. Sinon, la commande renvoie le nom de chemin de l'icone courante pour *fenêtre*, ou une chaîne vide s'il n'y a aucune icone actuellement spécifiée pour *fenêtre*. Les événements de souris sont désactivés pour *fenêtre* tant qu'elle a la forme d'une icone[4].

wm maxsize *fenêtre ?largeur hauteur?*

wm minsize *fenêtre ?largeur hauteur?*

Si *largeur* et *hauteur* sont spécifiés, ils indiquent les dimensions maximales (resp. minimales) autorisées pour *fenêtre*. Pour des fenêtres de grille, les dimensions sont spécifiées en unités de grille ; sinon, elles sont spécifiées en pixels. Le gestionnaire de fenêtres restreint les dimensions de la fenêtre pour qu'elles restent inférieures ou égales à *largeur* et *hauteur*. Si *largeur* et *hauteur* sont spécifiés, la commande renvoie une chaîne vide. Sinon, elle renvoie une liste Tcl avec deux éléments, qui sont la largeur et la hauteur maximales (resp. minimales) actuellement en vigueur. Par défaut, la taille maximale (resp. minimale) est la taille de l'écran. Si le redimensionnement a été désactivé par une commande **wm resizable**, cette commande est sans effet.

wm overrideredirect *fenêtre ?bool?*

Permet de fixer la valeur de l'option *override-redirect* pour *fenêtre*. Si *bool* n'est pas spécifié, une valeur 0 ou 1 est renvoyée pour indiquer si l'option *override-redirect* est actuellement en vigueur pour *fenêtre*. Si l'option *override-redirect* est fixée pour une fenêtre, celle fenêtre sera ignorée par le serveur de fenêtres pour un certain nombre d'opérations. La principale conséquence est de supprimer le cadre que le serveur dessine autour de la fenêtre : la fenêtre n'est plus qu'une zone rectangulaire comme les images d'accueil qu'affichent de nombreuses applications pendant leur chargement.

wm positionfrom *fenêtre ?qui?*

Si l'argument *qui* est spécifié, il doit prendre une des valeurs *program* ou *user* ou une abréviation de ces valeurs. Il indique si la position courante de *fenêtre* a été requise par le programme ou par l'utilisateur respectivement. Plusieurs gestionnaires de fenêtres ignorent les requêtes de programme pour les positions initiales et demandent à l'utilisateur de positionner manuellement la fenêtre ; si *user* est spécifié, le gestionnaire de fenêtres devrait positionner la fenêtre à l'emplacement donné sans demander l'aide de l'utilisateur.

Si *qui* est spécifié comme une chaîne vide, la source de la position courante est annulée. Si *qui* est spécifié, la commande renvoie une chaîne vide. Sinon,

4. Les gestionnaires de fenêtres ne supportent pas tous la notion d'iconification.

elle renvoie les valeurs *program* ou *user* pour indiquer la source de la position courante de la fenêtre ou une chaîne vide si aucune source n'a encore été spécifiée. La plupart des gestionnaires de fenêtres interprètent l'absence de source comme équivalente à la valeur *program*. Tk fixe automatiquement la source de positionnement à la valeur *user* lorsqu'une commande **wm geometry** est invoquée à moins que la source n'ait été explicitement déclarée avec la valeur *program*.

wm protocol *fenêtre ?nom? ?commande?*

Cette commande est utilisée pour contrôler les protocoles du gestionnaire de fenêtres comme WM_DELETE_WINDOW. L'argument *nom* est le nom d'un atome correspondant à un protocole de gestionnaire de fenêtres, comme par exemple WM_DELETE_WINDOW ou WM_SAVE_YOURSELF ou WM_TAKE_FOCUS. Si à la fois *nom* et *commande* sont spécifiés, *commande* est associé au protocole spécifié par *nom*. L'argument *nom* sera ajouté à la propriété WM_PROTOCOLS de *fenêtre* afin d'indiquer au gestionnaire de fenêtres que l'application a un gestionnaire de protocole *nom* : la commande *commande* sera invoquée par la suite chaque fois que le gestionnaire de fenêtres enverra un message au client pour ce protocole. Dans ce cas, la commande renvoie une chaîne vide.

Si *nom* est spécifié mais que *commande* ne l'est pas, la commande courante pour *nom* est renvoyée, ou une chaîne vide s'il n'y a pas de gestionnaire défini pour *nom*. Si l'argument *commande* est spécifié comme une chaîne vide, le gestionnaire courant pour *nom* est détruit et il est retiré de la propriété WM_PROTOCOLS sur la fenêtre *fenêtre* ; une chaîne vide est renvoyée.

Enfin, si ni *nom*, ni *commande* ne sont spécifiés, la commande renvoie une liste de tous les protocoles pour lesquels des gestionnaires sont actuellement définis pour *fenêtre*.

Tk définit toujours un gestionnaire de protocole pour WM_DELETE_WINDOW, même si on ne l'a pas explicitement demandé avec une instruction **wm protocol**. Si un message WM_DELETE_WINDOW arrive alors qu'un gestionnaire n'a pas été défini, Tk traite le message en détruisant la fenêtre pour laquelle il a été reçu.

wm resizable *fenêtre ?largeur hauteur?*

Cette commande contrôle si l'utilisateur peut redimensionner interactivement une fenêtre de premier niveau. Si *largeur* et *hauteur* sont spécifiés, ce sont des valeurs booléennes qui déterminent si la largeur et la hauteur de *fenêtre* peuvent être modifiées par l'utilisateur. Dans ce cas, la commande renvoie une chaîne vide. Si les arguments *largeur* et *hauteur* sont omis, la commande renvoie une liste avec deux éléments prenant les valeurs 0 ou 1 qui indiquent si la largeur et la hauteur de *fenêtre* sont actuellement redimensionnables. Par défaut, les fenêtres peuvent être redimensionnées dans les deux dimensions. Si la redimension est désactivée, la taille de la fenêtre sera la taille du plus récent changement de dimension dû à l'utilisateur ou à la commande **wm geometry**. S'il n'y a pas eu d'opération de ce type, la taille naturelle de la fenêtre est utilisée.

wm sizefrom *fenêtre ?qui?*

Si l'argument *qui* est spécifié, il doit prendre une des valeurs *program* ou *user* ou une abréviation de ces valeurs. Il indique si la taille courante de *fenêtre* a été requise par le programme ou par l'utilisateur respectivement. Les mêmes remarques que pour la sous-commande **positionfrom** s'appliquent à **sizefrom** concernant les requêtes au serveur.

wm state *fenêtre ?étatNouveau?*

Si l'argument *étatNouveau* est spécifié, la fenêtre sera mise dans ce nouvel état, autrement la commande renvoie l'état courant de *fenêtre* qui peut être une des valeurs suivantes : *normal*, *iconic*, *withdrawn*, *icon* ou *zoomed* (pour Windows uniquement). La différence entre *iconic* et *icon* est que *iconic* se réfère à une fenêtre qui a été iconifiée (avec la commande **wm iconify**) tandis que *icon* se réfère à une fenêtre dont la seule fonction est de servir d'icone pour une autre fenêtre (par la commande **wm iconwindow**). L'état *icon* ne peut pas être fixé par cette commande.

wm title *fenêtre ?chaîne?*

Si l'argument *chaîne* est spécifié, il sera passé au gestionnaire de fenêtres qui l'utilise comme titre pour *fenêtre* (le gestionnaire de fenêtres affiche cette chaîne dans la barre de titre de *fenêtre*). Dans ce cas, la commande renvoie une chaîne vide. Si l'argument *chaîne* n'est pas spécifié, la commande renvoie le titre courant de *fenêtre*. Le titre par défaut d'une fenêtre est son nom.

wm transient *fenêtre ?maître?*

Si l'argument *maître* est spécifié, le gestionnaire de fenêtres est informé que *fenêtre* est une fenêtre temporaire (comme par exemple un menu déroulant ou un menu popup) placée sous le contrôle de l'argument *maître* (qui désigne le nom de chemin d'une fenêtre de premier niveau). Certains gestionnaires de fenêtres utiliseront cette information pour gérer la fenêtre de manière particulière : une fenêtre temporaire ne reçoit ni cadre, ni barre de titre.

Si *maître* est spécifié comme une chaîne vide, la fenêtre est marquée comme n'étant plus une fenêtre temporaire, ce qui est une manière de supprimer le statut de fenêtre temporaire.

Si *maître* est spécifié, la valeur de retour de la commande est une chaîne vide. Sinon, la commande renvoie le nom de chemin du maître courant de *fenêtre*, ou une chaîne vide si *fenêtre* n'est pas actuellement une fenêtre temporaire.

wm withdraw *fenêtre*

Fait en sorte que la fenêtre désignée par l'argument *fenêtre* soit effacée de l'écran. La fenêtre est alors désinstallée et n'est plus connue du gestionnaire de fenêtres. Si la fenêtre n'avait jamais été installée, cette commande a pour effet de l'installer avec un état de fenêtre effacée. Les gestionnaires de fenêtres ne savent pas tous traiter ce genre de situation. Il est parfois nécessaire d'effacer une fenêtre puis de la réafficher afin que le gestionnaire de fenêtres remarque les changements intervenus sur les attributs de la fenêtre.

Index

SYMBOLES

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z