Tcl - machines à états - mbl@lri.fr

Le fichier sm.tcl contient une extension de Tcl/Tk qui permet de définir des machines à états. Le ficl smUtils.tcl permet de tester ces machines à états. Elles peuvent également être utilisées directement d des applications Tcl. La déclaration d'une machine à états a la forme suivante :

```
state_machine CreateLine {                        déclaration de la machine de nom CreateLine
        local P1 P2 line                          déclaration de trois variables locales rémanentes
        state Start {                             déclaration de l'état Start
                when Down at P1 do {                      déclaration de la transition sur Down
                        set line [CreateLine $P1 $P1]     action à exécuter
                } -> Drag                                         état final
        }
        state Drag {                              déclaration de l'état Drag
                when Move at P2 do {                      déclaration de la transition sur Move
                        ChangeCoords $line $P1 $P2        action à exécuter
                }                                                 pas d'état final : on boucle sur Drag

                when Up at P2 do {                                déclaration de la transition sur Up
                        DeleteItem $line                          action à exécuter
                        AppCreateLine $P1 $P2
                } -> Start                                        état final
        }
}
```

Une machine à état est constituées d'*états*. Dans chaque état, des *transitions* sont associées à occurences d'*événements* et comportent des *clauses.*

Etats -

Une machine à états est définie par un ensemble de variables locales (commande local) et un ensem d'états (commande state). Le premier état dans la liste est l'état de départ. A chaque état peuvent associés un script Tcl exécuté lorsque l'on arrive dans cet état (commande enter) et un script exéc lorsque l'on quitte cet état (commande leave) :

```
        state S {
                enter { puts "on arrive dans S" }
                leave { puts "on quitte S" }
                . . .
        }
```

Evénements -

Chaque transition est définie par un événement suivi de zéro, une ou plusieurs clauses qui définissen condition, l'action et l'état d'arrivée de la transition. Les événements sont :

Down	Appui sur le bouton de gauche de la souris
Move	Déplacement de la souris avec le bouton de gauche de la souris enfoncé
Motion	Déplacement de la souris, quel que soit l'état des boutons
Up	Relâchement du bouton de gauche de la souris
Enter	Entrée du curseur dans un objet graphique
Leave	Sortie du curseur d'un objet graphique
Key	Appui sur une touche du clavier
Key-xx	Appui sur la touche xx du clavier
TimeOut	Expiration d'un temporisateur

Pour l'événement Key-xx, xx peut être:

- la valeur ASCII d'une touche du clavier (Key-a, Key-8, etc.)
- le nom d'un touche (Key-BackSpace, Key-Escape, etc.)

Chaque machine à état a un temporisateur (et un seul), qui est activé par la commande arm. La comma disarm peut être utilisé pour annuler l'armement du temporisateur.

arm 100	déclenche un TimeOut dans 100 millisecondes
disarm	désarme le temporisateur d'identificateur id

Clauses -

Chaque clause d'une transition est introduite par un mot-clé. Toutes les clauses sont optionnelles. E doivent apparaître dans l'ordre ci-dessous, et seule la clause "and" peut être répétée plusieurs fois. Seu les clauses "and", "do" et "->" sont valides avec un événement de type TimeOut. Toutes les clauses s valides pour les autres événements.

with mod	définit le modificateur clavier qui doit être enfoncé au moment de la transition. mod peut valoir Shift, Control, ou Alt.
at pt	affecte à la variable pt la position de la souris.
onItem i	définit la condition que l'événement concerne un objet graphique. L'identificateur de l'objet est mis dans la variable i.
withTag t	définit la condition que l'événement concerne un objet graphique marqué par le "tag" t (voir ci-dessous).
and {cond}	définit une condition pour que la transition soit activée. cond est un script Tcl dont la valeur de retour est interprétée comme un booléen.
do {script}	définit l'action à exécuter lorsque l'on active la transition.
-> S	définit l'état d'arrivée de la transition. Si cette clause est absente, l'état d'arrivée est le même que l'état de départ (boucle). Dans ce cas, les actions d'entree et de sortie de l'état (enter et leave) ne sont pas exécutée.

Conditions et Actions -

Le fichier smUtils.tcl définit quelques commandes qui simplifient l'écriture de machines à états simp Certaines de ces fonctions prennent des points en paramètre. Un point est une liste de 2 éléments {x y}.

Géométrie :

PointsClose P1 P2	retourne vrai si P1 et P2 sont proches (moins de 3 pixels).
Delta P1 P2	retourne une liste {dx dy} correspondant au vecteur P1P2.

Objets graphiques :

Ces commandes permettent de gérer des objets graphiques. Les commandes de création retournent identificateur d'objet utilisé par les commandes de manipulation. L'argument "opts" de certai commandes est une liste d'options de la forme "-option valeur -option valeur...". Les options disponit sont par exemple : "-fill couleur" et "-outline couleur" pour la couleur de fond et de bord de l'objet width nn" pour l'épaisseur de trait, "-tag t" pour associer un "tag" t à l'objet, qui peut être testé pa clause "withTag" des transitions de la machine à états.

CreateLine P1 P2 opts	crée une ligne d'extrémités P1 P2.
CreateRect P1 P2 opts	crée un rectangle de points diagonaux P1 P2.
CreatePolygon points opts	crée un polygone dont les sommets sont la liste points.
ChangeCoords obj P1 P2	change les coordonnées d'un segment ou d'un rectangle.
ChangeItem obj opts	change les attributs de l'objet graphique obj.
DeleteItem obj	détruit l'objet graphique obj.

Icones :

CreateIcon P	crée un icone à la position P.
ShadowIcons icons	crée les "ombres" des icones de la liste icons et retourne la liste de leurs identificateurs.
SelectIcons icons	affiche les icones de la liste icons dans l'état sélectionné.
DeselectIcons icons	affiche les icones de la liste icons dans l'état normal.
HiliteIcons icons	affiche les icones de la liste icons dans l'état activé.
UnhiliteIcons icons	affiche les icones de la liste icons dans l'état désactivé.
MoveIcons icons delta	déplace les icones de la liste icons de delta = {dx dy}.
DeleteIcons icons	détruit les icones de la liste icons.
IconAtPoint P	retourne l'icone à la position P, ou la chaine vide s'il n'y a pas d'icone à cet endroit.
IconsInRect P1 P2	retourne la liste (éventuellement vide) des icones qui sont en partie intérieurs au rectangle de diagonale P1 P2.

Autres :

Command cmd	crée un bouton en bas de la fenêtre permettant d'activer la commande cmd (appelée sans arguments).