Actions et Translations sous Xspect

Gauthier LEBRUN

10 juin 1997

Résumé

Pour traiter des actions liées aux widgets, la librairie Xt utilise le mécanisme des *callbacks*. Ainsi, lorsque l'utilisateur clique sur un bouton, il suffit de lier l'événement à la fonction correspondante voulue. Or, il se peut que l'utilisateur veuille implémenter une action qui n'est pas prévu par le widget. A ce moment là, on utilise le mécanisme des actions/translations.

1 Généralités

Lorsqu'aucune classe de widget n'a les *callbacks* nécessaires ou les caractéristiques voulues, on utilise la technique des Actions / Translations. Cette méthode est en effet plus simple que d'écrire un widget et effectue la même chose.

Ainsi, on utilise les actions pour ajouter des comportements à un *Primitive* ou *Core* widget. (On peut également les utiliser dans le code d'un widget pour implémenter une caractéristique du widget).

1.1 Actions

Les actions et les *callbacks* diffèrent par la méthode d'implémentation des fonctions:

- Pour les **callbacks**, le déclencheur est défini dans le widget (qui peut ou non être lié à un événement). Quand le déclencheur est activé, les routines sont appelées par `XtCallCallbacks()` ou `XtCallCallbackList()`.

- En revanche, les **Actions** sont appelées directement par le mécanisme de translation de Xt, comme le résultat d'une combinaison d'événements.

Pour déclarer une action, il faut d'abord déclarer une **Table des Actions** qui organise les chaînes de noms d'actions avec des pointeurs sur fonctions. La table est déclarée par `XtAppAddActions()`. On crée ensuite une table de translations stockée généralement dans le fichier de resources de l'application (on peut également la coder en "dur").

Le schéma suivant montre une table de translation et une table d'action, et indique comment chacune est utilisée lorsqu'un événement est reçu pour activer la fonction correspondante.

```
Key Event (Return key pressed)
```

Translation Table

```
static char default Translations[]=
        ''<Key>Return:Activate()'';
```

Action Table

```
static XtActionsRec actions[]={
{''Activate'',Activate}
};
```

Action Function

```
Activate(w,event,params,num_params)
widget w;
XEvent *event;
String *params;
Cardinal *num_params
{
/*action code */
}
```

Translation : Traduit un événement en un nom d'action. Les translations se trouvent donc dans le **fichier de resources.**

Action Table : Traduit le nom d'action en un pointeur sur fonction. Elle est donc spécifiée dans le **code**.

Schéma type de la création de la table des actions :

```
main(...)
{
        /* autres declarations */
        ...
        static XtActionsRec actions[]={
               {''RedrawSmallPicture'',RedrawSmallPicture},
        }
        ...
```

```
        /* XtAppInitialize */
        ...
        XtAppAddActions(app_context,actions,XtNumber(actions));
        ...
        /* XtMainLoop */
}
```

Les actions ajoutées par `XtAppAddActions` sont globales au contexte de l'application, et donc utilisables par n'importe quelle instance de widget de l'application.
Typiquement, une action est juste un pointeur sur une fonction à 4 arguments :

1. un widget.
2. un événement.
3. une chaîne contenant tous les arguments spécifiés pour l'action dans la table de translation.
4. le nombre d'arguments de la chaîne.

1.2 Translations

La table des translations permet de déterminer à quels widgets les actions sont destinés et lesquels vont les déclencher.
Bien sur, la librairie Xt comporte les fonctions pour charger cette table : ce sont `XtParseTranslationTable`, `XtAugmentTranslation`, et `XtOverrideTranslation`.
Fragment de code montrant la déclaration des tranlations :

```
static char defaultTranslations[]=
       ``<Btn1Down> : set()\n\
         <Btn2Down>, <Btn2Up> : quit()'';

XtTranslations myTranslations;
...

myTranslation = XtParseTranslationTable(defaultTranslation);
XtOverrideTranslations(farewell,myTranslations);
```

On remarquera que la table des translations (`defaultTranslations`) est définie en tant que chaîne de caractères et peut donc se trouver dans un fichier de resources. L'utilisation du fichier de resources est avantageux au départ car il permet de modifier cette table sans devoir recompiler tout le programme.
En pratique, certaines parties de la table des translations seront codées en “dur” dans le programme pour définir certains comportements qui ne

seront pas modifiables par l'utilisateur. En revanche, d'autres parties de la table des translations seront insérées dans un fichier de resources pour permettre à l'utilisateur de personaliser les comportements du logiciel à sa manière.

Voici un exemple montrant une partie du fichier de resources d'une application où l'on définit une table de translations :

```
*XmPrimitive.baseTranslations:\
    <Expose>:  RedrawSmallPicture()
```

On notera que puisque les fichiers de resources sont des chaînes de caractères, nous n'avons pas besoin de guillemets. Néanmoins, pour les tables de translations comportants plusieurs lignes, nous devons terminer chaque ligne par `\n\`.
La resource affectée ici est `baseTranslations` (nouveau dans R5). Cette resource a le même nom pour tous les widgets. Comme le fichier de resources est constitué exclusivement de chaînes de caractères, Xt affecte donc la table des translations avec les données du fichier de resources en passant par un convertisseur de resuurce. Ce convertisseur appelle la fonction `XtParseTranslationTable()` pour nous quand nous spécifions les translations dans un fichier de resources.

2 Utilisation dans Xspect

Xspect utilise fortement les actions / translations.

Tables de translations définies dans `xspect.h` :

translationsWindow	translationsMoveButton	translationsDrawVisSel
translationsPopupPage	translationsDrawNoView	translationsDrawCallUdi
translationsPopupZoom	translationsMixMode	translationsSynchro
translationsIdw	translationsDrawSelMode	translationsTranslationOutput
translationsIdwChgActView	translationsDrawSigMode	

On trouve la table des actions dans `resources.h` :

```
static XtActionsRec actions[] =
{
  {"Motion",            XsAcMotion},
  {"Up",                XsAcUp},
  {"ActivateFrame",     XsAcActivateFrame},
  {"ActivateWindow",    XsAcActivateWindow},
  {"DeActivateWindow",  XsAcDeActivateWindow},
  {"ActivateSelection", XsAcActivateSelection},
  {"CreateSelectionX",  XsAcCreateSelectionX},
  {"CreateSelectionXY", XsAcCreateSelectionXY},
  {"CreateSelectionY",  XsAcCreateSelectionY},
```

```
  {"IdwChgActView",     XsAcIdwChgActView},
  {"PagePopupMenu",     XsAcPagePopupMenu},
  {"ZoomPopupMenu",     XsAcZoomPopupMenu},
  {"IdwPopupMenu",      XsAcIdwPopupMenu},
  {"ModifySelection",   XsAcModifySelection},
  {"ModifySelectionXY", XsAcModifySelectionXY},
  {"MoveButtonWidget",  XsAcMoveButtonWidget},
  {"OkCallUdi",         XsAcOkCallUdi},
  {"OkVisitSelection",  XsAcOkVisitSelection},
  {"OkSynchro",         XsAcOkSynchro},
  {"OkOutput",          XsAcOkOutput},
  {"SelectAll",         XsAcSelectAll},
  {"ShiftSelection",    XsAcShiftSelection},
  {"ShiftSelectionXY",  XsAcShiftSelectionXY},
  {"ShiftSignalX",      XsAcShiftSignalX},
  {"ShiftSignalXY",     XsAcShiftSignalXY},
  {"ShiftSignalY",      XsAcShiftSignalY},
};
```

Les fonctions indiquées ci-dessus se trouvent se trouvent dans `Xa.c`.