

UTILISATION DES API INSTALL-CALLBACK, CALL_PREV CALLBACK ET REMOVE_CALLBACK

Explication générale

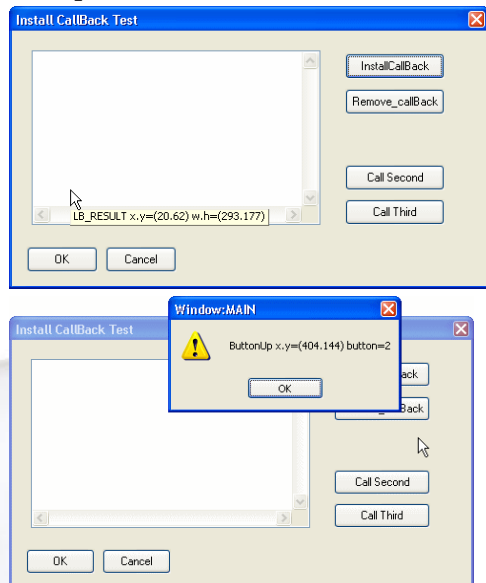
La librairie NSMISC.NCL exporte ces trois API très puissantes. Elles permettent d'appliquer des traitements sur **TOUTES** les fenêtres ou bien sur **TOUS** les contrôles d'une application, et ceci sans modifier ou ouvrir dans l'éditeur le moindre fichier SCR.

Le principe est le suivant : une fonction unique reçoit **tous** les événements envoyés aux contrôles et les fenêtres. Elle effectue ses traitements propres et peut appeler le traitement par défaut si nécessaire, ensuite elle peut effectuer d'autres traitements puis rendre la main.

Nous verrons dans un exemple d'utilisation comment on peut (entre autres) :

- Appliquer un nouveau traitement avant et après l'INIT de toutes les fenêtres
- Centrer toutes les fenêtres d'une application, indépendamment de la taille de l'écran, de la taille de la fenêtre et du paramétrage initial
- Afficher automatiquement une bulle d'aide sur tous les contrôles et fenêtres d'une application, cette bulle d'aide contenant le nom du contrôle, sa position et sa taille
- Afficher un message spécifique quand on clique avec la souris dans une fenêtre

Exemple d'utilisation :



Rappel : le callBack

Un callback est une fonction écrite par l'utilisateur et qui est appelée par le système. Il faudra donc que l'utilisateur informe de système de l'adresse de sa fonction pour qu'il puisse l'appeler.

Utilisation des APIs INSTALL_CALLBACK

Déclarer dans une librairie NCL une fonction CallBack avec le prototype suivant :

```
global pointer hCallback

Function MY_PREV CALLBACK (POINTER mySelf, INTEGER id, INTEGER msg, \
    POINTER parm1, POINTER parm2) return INTEGER
```

Ensuite, installer la fonction de callBack par l'appel suivant :

```
hCallBack = INSTALL_CALLBACK(@MY_PREV CALLBACK)
```

Le système étant maintenant informé de l'adresse de votre fonction, il pourra l'appeler.

Exemple

Voici un exemple complet de traitement de la fonction CallCack.

```
1 ; Library LIBCBACK.NCL
2 ; Created 05/02/08
3
4 Const NS_MSG_INIT 1
5 Const NS_MSG_MOUSEMOVE 5
6 Const NS_MSG_BUTTONUP 7
7
8 global pointer hCallback
9
10 Function MY_PREV CALLBACK (POINTER mySelf, INTEGER id, INTEGER msg, \
11     POINTER parm1, POINTER parm2) return INTEGER
12
13     local res%, control ctrl
14
15     if id = 0
16 ; traitement des événements des fenêtres
17         evaluate msg
18         where NS_MSG_INIT
19             settex ("PRE-INIT " & gettext$ (mySelf) ) to mySelf
20         endwhile
21         where NS_MSG_MOUSEMOVE
22             MAIN (mySelf) .client.tooltip=windowName$ (mySelf) && "x.y=(" & \
23                 getxpos$ (mySelf) & "." & getyPos$ (mySelf) & ") w.h=(" & \
24                 getWidth$ (mySelf) & "." & getheight$ (mySelf) & ")"
25         endwhile
26         where NS_MSG_BUTTONUP
27             message "Window:" & windowName$ (mySelf), "ButtonUp x.y=(" & \
28                 LOW (parm1) & "." & HIW (parm1) & ") button=" & parm2
29         endwhile
30     endevaluate
31     else
32 ; traitement des contrôles
33         evaluate msg
34         where NS_MSG_MOUSEMOVE
35             makeControl (ctrl, mySelf, id)
36 ; création automatique de la bulle d'aide avec les infos pertinentes.
37             ctrl.tooltip=controlname$ (ctrl) && "x.y=(" & \
38                 getxpos$ (ctrl) & "." & getyPos$ (ctrl) & ") w.h=(" & \
39                 getWidth$ (ctrl) & "." & getheight$ (ctrl) & ")"
40         endwhile
41     endevaluate
42     endif
```

Explications du code

Les lignes 4,5,6 contiennent les N° des événements sous forme de constantes. Il est facile de récupérer toutes les constantes d'événements depuis le fichier **nslib.h**. Dans ce fichier, les n° d'événements sont déclarés sous la forme suivante :

```
#define NS_MSG_INIT 1
#define NS_MSG_TERMINATE 2
#define NS_MSG_HELP 3
#define NS_MSG_CHARACTER 4
#define NS_MSG_MOUSEMOVE 5
#define NS_MSG_BUTTONDOWN 6
#define NS_MSG_BUTTONUP 7
#define NS_MSG_BUTTONDBLCLK 8
#define NS_MSG_TIMER 9
#define NS_MSG_PAINT 10
#define NS_MSG_CHANGED 11
```

Pour les utiliser en NCL, il suffit de copier les lignes pertinentes dans votre source NCL, et de changer «#define» en «Const».

Les paramètres passés

mySelf% : Indique l'identifiant de la fenêtre (SELF%).

Id% : s'il vaut 0, le message est passé à la fenêtre, sinon il contient l'identifiant du contrôle.

Msg% : le numéro de l'événement passé.

Parm1 et Parm2 : les paramètres associés (équivalent de param12% et param34%).

Ligne 15 :

On teste si l'ID est 0 pour savoir si l'on traite un événement de fenêtre ou bien un événement de contrôle.

Ligne 18 :

Traitement de l'événement INIT de la fenêtre : Dans ce cas, on modifie le titre de la fenêtre **AVANT** que l'événement INIT soit exécuté. (ajout de «PRE-INIT» au début)

Ligne 21 :

Traitement de l'événement MOUSEMOVE de la fenêtre.

On construit dynamiquement la tooltip (bulle d'aide) à afficher quand on bouge la souris sur la fenêtre. Elle contient le nom de la fenêtre, sa position et sa taille.

2 choses sont à noter dans ce traitement :

- Le nom de la fenêtre «MAIN» n'est pas utilisé réellement et cette instruction fonctionnera quel que soit le nom de fenêtre que l'on mettra dans cette ligne, mais il est obligatoire d'en préciser une pour coller à la syntaxe du NCL.

- On accède aux qualificatifs dynamiques des fenêtres via leur zone client.

Ligne 26 :

Traitement de l'événement BUTTONUP de la fenêtre. Sur clic de souris, on affiche une boîte de message contenant le nom de la fenêtre, la position du clic de souris et le N° du bouton enfoncé.

- A noter : l'utilisation de LOW et HIW pour «couper en 2» le paramètre parm1. Ceci est l'équivalent en NCL de la transformation automatique de PARAM12% à PARAM1% et PARAM2%.

On pourra voir la documentation de l'événement BUTTONUP.

Ligne 31 :

else du test «if id = 0». On va maintenant traiter les événements des contrôles.

Ligne 34 :

Traitement de l'événement MOUSEMOVE du contrôle.

Ligne 35 :

- On reconstruit dynamiquement une variable de type CONTROL à partir de l'identifiant de la fenêtre et de l'ID du contrôle.

- On utilise ensuite le qualificatif dynamique «TOOLTIP» pour afficher une bulle d'aide. Elle contient le nom du contrôle, sa position et sa taille.

Ligne 42 :

Fin du traitement des messages de contrôles.

Suite du code source

```

43 res% = CALL_PREV CALLBACK (hCallback,mySelf, id, msg, parm1, parm2)
44
45
46 ; traitement à la fin de l'INIT des fenêtres.
47 if (id = 0) and (msg = NS_MSG_INIT)
48   settext (gettext$(mySelf) & " POST-INIT") to mySelf
49   ; centrage automatique de la fenêtre à l'ouverture Si elle n'est pas fille.
50   if ParentWindow$(mySelf) = 0
51     setpos (getDeskTopWidth% - getWidth$(mySelf) / 2, \
52            (getDeskTopHeight% - getHeight$(mySelf) / 2 to mySelf
53   endif
54 endif
55
56 return res%
57
58 EndFunction ; MY_PREV CALLBACK
59

```

Ligne 44 :

On appelle le traitement par défaut de la fenêtre ou du contrôle.

Ligne 47 :

Exemple de traitement effectué **après** le traitement par défaut.

Si l'id vaut 0 (fenêtre) et que le message vaut NS_MSG_INIT, alors :

- On modifie le titre de la fenêtre (ajout de « POST-INIT » à la fin)
- On regarde si la fenêtre est fille du DESKTOP, dans ce cas elle est centrée sur l'écran

Ligne 56 :

La fonction renvoie le code retour fourni par l'appel de la fonction d'origine.

Désinstallation du Callback

Il faut, à la fin, désinstaller la fonction de callback par l'appel suivant :

```
remove_callback (hCallback)
```

Récupération des paramètres

Rappel : Parm1 et parm2 sont équivalents à PARAM12% et PARAM34%

- PARAM1% se récupère par LOW (parm1)
- PARAM2% se récupère par HIW (parm1)
- PARAM3% se récupère par LOW (parm2)

- PARAM4% se récupère par HIW (parm2)

Les fonctions ControlName\$ et WindowsName\$ ne fonctionnent que si l'option « Symbol info » a été cochée pour la génération.

Domaine d'utilisation de ces fonctions et conclusion

On l'aura compris, ces fonctions sont extrêmement puissantes. Elles permettent de modifier les comportements sur tous les types de contrôles et de fenêtres, sans les modifier. Leur champ d'action est quasi-illimité.