WindowKey.doc

COLLABORATORS			
	TITLE : WindowKey.doc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		August 19, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

WindowKey.doc

1.1 Documentation de WindowKey - Français

Bienvenue à WindowKey 1.01 copyright 1993-94 Frédéric Delacroix.

Ceci est le document qui doit toujours être distribué avec les autres fichiers. Il est fait pour être lu par AmigaGuide (copyright Commodore), mais peut aussi être vu par des yeux humains, le confort en moins.

TABLE DES MATIERES

1.2 1.1 Distribution et décharge

WindowKey est Copyright 1994 par Frédéric Delacroix. La permission de le copier et de le distribuer est accordée à quiconque respecte ces conditions (généralement connues sous le nom de SHAREWARE):

- Tous les fichiers (ou toute l'archive) soient distribués ensemble (cela concerne les exécutables, les fichiers de documentation, les fichiers catalogues, le fichier de transcription de catalogue et tous les icônes). Il y a une exception toutefois: si vous prévoyez une distribution pour une communauté "mono-linguistique" (disons, par exemple, la France uniquement), vous êtes autorisé à ne distribuer que les fichiers qui sont relatifs à votre langue, c'est-à-dire: le fichier de documentation et le fichier catalogue. Tous les autres fichiers doivent être présents.
- Tous les fichiers distribués ne soient MODIFIES EN AUCUNE FACON (pas de message idiot du genre "distribué par..."). Si vous avez des commentaires à ajouter, faites-le dans un fichier séparé! L'archivage est toutefois autorisé, mais le compactage de l'exécutable n'est pas recommandé car le programme se détache lui-même du CLI (il a besoin de couper sa SegList).
- Vous ne pouvez pas demander d'argent pour ce programme. Une petite somme est autorisée pour la copie et l'envoi, mais vous ne pouvez PAS demander plus que Fred Fish pour un AmigaLib disk unique.

- Ceci est pour tous les utilisateurs de WindowKey: comme le programme est distribué en SHAREWARE, vous devez m'envoyer une petite contribution de $10 (50FF ou équivalent) si vous continuez à l'utiliser après une courte période d'évaluation. Si vous voulez le source du programme (écrit pour Devpac 3), ajoutez $10 de plus et je vous l'enverrai.

Mon adresse se trouve à la fin de ce document.

WindowKey a été beaucoup testé, mais je ne peux pas garantir qu'il marchera toujours comme prévu. Je ne peux pas être tenu pour responsable de dommages/ perte de données directs ou indirects qui pourraient resulter de l'utilisation de ce programme. Souvenez-vous que vous l'utilisez A VOS RISQUES ET PERILS.

Note particulière: les catalogues et les fichiers de documentation ne sont disponibles qu'en anglais et en français (mon espagnol scolaire est si mauvais :-). Si vous pouvez faire une traduction de ces fichiers dans votre langue, j'aprécierais grandement que vous me les envoyiez pour que je les inclue dans la prochaine version. Pour le fichier de documentation, c'est facile: éditez simplement une copie de celui-ci (merci de ne pas changer le nom des noeuds, ils ne sont jamais affichés de toute façon). Pour les catalogues, remplissez les fichiers .ct avec vos propres textes et envoyez-les moi. Je vous retournerai les catalogues correspondants. Notez que certaines chaînes (comme le message de description pour commodities et les étiquettes de gadgets) sont limités en longueur. Pour certaines, il y a un raccourci-clavier (en majuscules SVP) ou un équivalent à un menu juste avant l'étiquette.

Les icônes que j'ai utilisées pour les tiroirs et d'autres fichiers sont tirées d'IconPack, qui est une collection de jolies icônes dessinées et copyrightées par Tom Ekström.

1.3 1.2 Crédits et remerciements

WindowKey a été écrit avec le merveilleux Devpac 3 d'Hisoft sur un vieil A500 avec l'OS 2.1. Il utilise la reqtools.library qui est Copyright Nico François et errormsg.library, qui est copyright par moi-même. ARexx est copyright ©1987 par William S. Hawes.

Remerciements à: Nico François pour reqtools.library
Khalid Aldoseri pour sa kd_freq.library
William S. Hawes pour ARexx
Hisoft pour Devpac 3
Commodore pour l'amiga et son OS
J.P. Deloffre et C.Peugnet de l'association ProMedia
Yves Quiquempois pour les tests sous Enforcer et idées d'amiéliorations
AmigaNews.

1.4 1.3 Configuration minimale

WindowKey devrait marcher sur n'importe quel amiga qui remplit ces conditions:

- Il vous faut le Kickstart 2.04 ou plus (V37+), ou WindowKey refusera de fonctionner. La seule chose à faire si vous avez encore le 1.3 est de vous mettre à jour ! Croyez-moi, ça en vaut le coup.

- Il vous faut la reqtools.library V38+ (release 2.1) installée dans votre tiroir LIBS:. Elle n'est pas fournie dans cette archive mais vous pouvez la trouver presque partout dans les collections de domaine public.

- Il vous faut l'errormsg.library V1.0+ installée dans votre tiroir LIBS:. Elle n'est pas fournie pour réduire la taille de l'archive, vous pouvez la trouver dans les collections de domaine public.

- Il vous faut l'iffparse.library V37+ installée dans votre tiroir LIBS:. Elle est fournie avec le Workbench 2.0+. La V39 est encore mieux car exempte de bugs.

- WindowKey utilise la locale.library V38+ si elle est disponible pour utiliser plusieurs langues. Cette bibliothèque est normalement fournie avec l'OS 2.1+.

- Le programme de préférences utilise l'amigaguide.library pour afficher ce fichier dès que la touche Help est pressée, mais elle n'est pas obligatoire.

- ARexx est nécessaire pour les options ARexx de WindowKey (!). Vous pouvez trouver ARexx avec votre distribution originale du Workbench 2.0 et supérieur.

1.5 2.1 Introduction

WindowKey est encore un autre programme 'input helper'. Le rôle de ces programmes est bien connu: ils activent, amènent à l'avant, renvoient à l'arrière, zooment, etc... n'importe quelle fenêtre ou écran (au sens Intuition du terme) à la demande de l'utilisateur.

Il existe beaucoup de programmes qui font cela, y compris le FKey de Commodore, fourni sur la disquette workbench, mais aucun n'avait une interface utilisateur aussi puissante et étaient aussi flexible que WindowKey. L'interface de WindowKey a été directement reproduite à partir de celle de mon autre programme, Injector. La raison pour laquelle j'ai choisi d'écrire deux programmes différents au lieu d'inclure les possibilités de WindowKey dans Injector est que vous pouvez utiliser un programme sans l'autre, ce qui paraît normal puisque leurs rôles sont totalement différents (c'est un reproche aux autres input helpers qui voudraient tout faire et finissent par ne rien faire).

Une autre raison à l'existence de WindowKey est la possibilité de contrôler Intuition uniquement au clavier, au lieu d'avoir à retrouver la souris sous une pile de papiers quand on veut changer de fenêtre active.

Pour que cela marche, WindowKey a son propre langage, fait de mot-clés, avec ou sans arguments entre parenthèses. Vous devrez lire les sections de référence pour plus d'informations.

1.6 2.2 Lancer Injector

WindowKey peut être démarré à partir du CLI ou du Workbench.

S'il est utilisé à partir du CLI, WindowKey se détache automatiquement, autorisant la fênêtre du CLI à se fermer ou la startup-sequence de continuer, pas besoin d'utiliser la commande Run. Aucune option n'est reconnue sur la ligne de commande du CLI. C'est le rôle du fichier de configuration. Quelques options supplémentaires sont décrites dans la section

tooltypes

.

Si vous utilisez WindowKey à partir du Workbench, le meilleur endroit est le tiroir WBStartup. De cette façon, WindowKey sera lancé chaque fois que le système démarre (n'oubliez pas de vous enregistrer). Vous pouvez bien sûr double-cliquer son icône aussi.

Même quand il est lancé du Workbench, WindowKey coupe sa SegList pour que le Workbench puisse se fermer quand WindowKey est actif. De plus, il n'y a pas besoin du tooltype DONOTWAIT.

1.7 2.3 Les tooltypes

Cette section décrit les tooltypes qui sont reconnus par WindowKey.
Les tooltypes sont la SEULE façon de passer ces arguments, même à partir du CLI. Ce sont:
PUBSCREEN
,
CX_PRIORITY
,
PREFSPATH
, et
AREXXCONSOLE
.

1.8 Le tooltype PUBSCREEN

Ce tooltype dit à WindowKey quel écran public doit être utilisé pour les requêtes qu'il pourrait produire. Par défaut, c'est '*', qui signifie que l'écran le plus en avant sera utilisé pourvu qu'il soit public (tout comme pour les fenêtre CON:). Si l'écran n'est pas disponible, l'écran public par défaut est utilisé.

1.9 Le tooltype CX_PRIORITY

Il est utilisé pour donner un nombre qui sera utilisé comme la priorité du Broker de WindowKey dans la liste de Commodities. C'est utile si vous voulez que les hotkeys d'WindowKey surpassent celles d'une autre commodité (ou le contraire). Plus la priorité est haute, plus WindowKey reçoit les InputEvents tôt (relativement aux autres brokers). Par défaut, c'est 0.

1.10 Le tooltype PREFSPATH

Cette option dit à WindowKey où trouver le programme de préférences quand
il est invoqué par la commande
EXEC_PREFS
. Par défaut, c'est
'WindowKeyPrefs', qui doit bien sûr être dans votre chemin d'accès.
Personnellement, j'utilise PREFSPATH=SYS:Prefs/WindowKey .

1.11 Le tooltype AREXXCONSOLE

Ce tooltype décrit la fenêtre de console que WindowKey doit ouvrir quand il lance un programme ARexx. Ce DOIT être une console interactive, par exemple une fenêtre CON: (peut-être un canal AUX: ? Je n'ai pas essayé).

Par défaut, c'est CON:////WindowKey and ARexx/SCREEN*/AUTO/WAIT/CLOSE, qui vous fournira une fenêtre de console raisonable sur l'écran le plus en avant s'il est public, ou sur l'écran public par défaut.

1.12 Le tooltype BREAKKEY

Ce tooltype es nouveau pour la version 1.01. Il s'agit d'une chaîne de description pour commodities utilisée pour dire à WindowKey quelle touche doit arrêter toute action de WindowKey.
Cette possibilité de "touche d'arrêt" a été ajoutée principalement pour des raisons d'homogénéité avec Injector, car les actions produites par WindowKey sont généralement courtes et ont rarement besoin d'être arrêtées par l'utilisateur. Cependant, la possibilité existe.

Contrairement aux hotkeys "normales", l'évènement n'est PAS retiré de la chaîne. De cette façon, vous pouvez quitter WindowKey en même temps, ou arrêter un autre programme comme Injector en même temps. Cela pourrait avoir des effets désastreux si l'application qui possède la fenêtre active prend ce message en compte.

La valeur par défaut est lcommand rcommand lshift rshift q

1.13 Le tooltype TOOLPRI

Ce tooltype existe depuis la version 1.01. C'est un tooltype standard, tel qu'il est compris par le Workbench, pour régler la priorité de la tâche de WindowKey. Cependant, comme WindowKey lance son propre processus, la priorité était toujours -1 dans les versions précédentes. Maintenant vous pouvez la régler avec ce tooltype, qui est aussi reconnu depuis le CLI.

Je vous conseille une priorité au moins égale à 1 (jamais plus de 5).

1.14 2.4 Terminer WindowKey

WindowKey peut être terminé de plusieurs façons. Vous pouvez ↩
bien sûr
utiliser le programme Exchange et sélectionner Supprimer WindowKey. Vous pouvez aussi lancer WindowKey (le programme principal, pas le programme de préférences) à nouveau. La dernière façon de quitter WindowKey est de le faire exécuter la commande

CMD_QUIT

.

Dans tous les cas, WindowKey ne peut pas se terminer si un programme ARexx est encore en exécution, car WindowKey doit attendre le message de réponse d'ARexx (vous ne voulez pas réveiller le gourou n'est-ce pas ? :-). De prochaines versions pourraient implémenter une option "delayed quit" (est-ce vraiment utile?).

1.15 3.1 Lancer le programme de préférences

Pour modifier sa configuration, Window utilise un programme séparé, appelé le programme de préférences. De cette façon, de la mémoire n'est pas utilisée pour stocker des données inutilisées pendant que WindowKey est actif: le programme de préférences n'est chargé que quand c'est nécessaire.

Le programme de préférences peut être lancé à partir du CLI, du Workbench, ou WindowKey. Il reconnaît TOUJOURS les tooltypes suivants (même à partir du CLI):

PUBSCREEN=<nom d'écran public>. Ceci définit l'écran sur lequel doit s'ouvrir la fenêtre. Par défaut, c'est *, signifiant que l'écran le plus en avant sera utilisé s'il est public.

CREATEICONS=<YES ou NO>. Ceci instruit le programme de préférences de l'état initial du menu "Créer icônes ?".

ACTION=<USE,SAVE ou EDIT>. Ceci est valide seulement pour les icônes projets dont l'outil par défaut est le programme de préférences, ou donnés par la multi-séléction. De tels projets sont des fichiers de configuration chargés immédiatement par le programme de préférences. Ce tooltype ACTION dit au programme de préférences ce qu'il doit en faire.

USE sauvera le fichier comme la configuration "courante", c'est-à-dire dans le fichier "ENV:WindowKey.Prefs".

SAVE sauvera le fichier comme la configuration "permanente", c'est-à-dire dans les deux fichiers "ENV:WindowKey.Prefs" et "ENVARC:WindowKey.Prefs".

EDIT est la valeur par défaut, il ne sauvegardera rien (encore), mais vous laissera éditer le fichier comme si vous aviez lancé le programme de préférences et séléctionné le menu Ouvrir... .

De plus, il y a quelques options qui peuvent être utilisées sur la ligne de commande, qui SURPASSENT les tooltypes mentionnés ci-dessus. Le format est le suivant:

```
 FILE,PUBSCREEN/K,USE/S,SAVE/S,EDIT/S,NCI=NOCREATEICONS/S,CI=CREATEICONS/S
```

FILE est bien sûr le nom du fichier à charger, PUBSCREEN a la même signification que le tooltype du même nom, USE,SAVE et EDIT sont des interrupteurs équivalents aux valeurs correspondantes du tooltype ACTION, et NOCREATEICONS est équivalent au tooltype CREATEICONS=NO. CREATEICONS est équivalent au tooltype CREATEICONS=YES. Les deux dernières options peuvent être utilisées pour surpasser les valeurs par défaut, je suppose qu'elles ne le seront pas beaucoup.

1.16 3.2 L'Interface Utilisateur Graphique

Quand le programme de préférences est lancé en mode EDIT, il ↩
ouvre une
fenêtre avec quelques gadgets et menus. Si vous avez l'habitude des programmes de préférences normaux, vous ne devriez pas avoir de problèmes

pour l'utiliser. Voici tout de même quelques explications.

L'affichage principal est constitué d'une liste. Elle contient toutes les hotkeys définies. Vous pouvez sélectionner un nom dans cette liste en cliquant dessus, ou en utilisant le raccourci clavier (t pour la version française) avec ou sans SHIFT pour traverser la liste sans utiliser la souris. Vous pouvez aussi utiliser les flèches vers le haut et vers le bas pour vous déplacer d'une entrée dans la direction correspondante, d'une page avec shift, ou pour aller au début ou à la fin de la liste avec control.

Sur la droite, il y a 5 gadgets qui contrôlent l'apparence de la liste. Ils ne changent pas la façon dont Injector traite la liste (excepté peut-être que si deux hotkeys sont en conflit, la première est utilisée). Premier enverra l'élément sélectionné tout en haut de la liste, Dernier l'enverra tout en bas, Haut le déplacera d'une ligne vers le haut, Bas d'une ligne vers le bas, et Trier triera la liste par ordre alphabétique.

Plus bas, il y a quelques gadgets de contrôle. Il y a trois gadgets de chaîne utilisés pour éditer les hotkeys. Pour WindowKey, une hotkey est une association de trois choses: un nom, une description de touche, et une ligne de commande. Les trois gadgets de chaîne vous permettent d'éditer ces trois champs.

Il y a une checkbox sur la droite du gadget descripteur de touche. Si elle est sélectionnée, alors la hotkey est autorisée. Si elle n'est pas sélectionnée, alors la hotkey n'est pas disponible. Elle l'est toutefois toujours grâce à la commande

EXEC_NAMED
.

Le champ Nom est utilisé pour afficher la liste et par la commande

EXEC_NAMED
.

Le gadget de chaîne nommé Touche vous permet d'éditer le champ qui dira à la commodities.library quelle touche doit déclencher l'action de WindowKey. Ce doit être une chaîne de description valide pour commodities. Voyez

Définir des touches de commande
pour plus d'informations. Après avoir tapé
dans ce gadget de chaîne, le programme de préférences demande à commodities si la description est bonne, et vous alerte dans le cas contraire. Si la description est correcte, alors la checkbox sera automatiquement sélectionnée, et desélectionnée sinon.

Le gadget de chaîne nommé Commande est là pour contenir la chaîne de commandes que WindowKey doit exécuter quand il reçoit l'évènement correspondant à cette touche. WindowKey utilise son propre "langage" pour définir de telles actions. Voyez la section

Référence du langage
pour plus
de détails.

Sur la droite des gadgets de chaîne, il y a trois gadgets booléens nommés Créer, Copier et Effacer. Ils sont utilisés respectivement pour créer une

nouvelle hotkey (qui apparaît à la fin de la liste), copier une hotkey existante (qui apparaît juste après celle qui est sélectionnée) et effacer une hotkey existante.

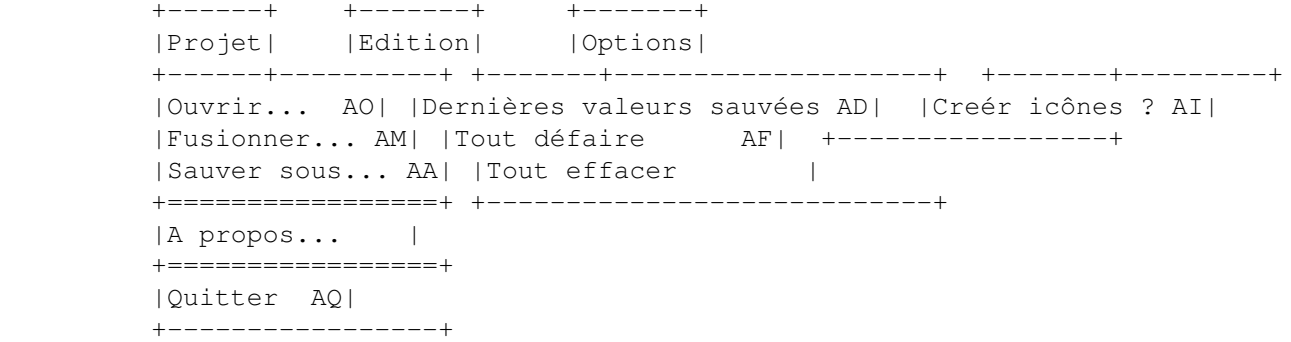
Plus bas encore il y a 5 gadgets. Ils contrôlent le comportement du programme.
Sauver sauvegardera le fichier édité comme configuration permanente. Utiliser le sauvegardera en tant que configuration courante. Tous deux termineront le programme de préférences (à moins qu'une erreur survienne pendant la sauvegarde).

Test changera la configuration courante mais ne terminera pas le programme de préférences, pour que vous puissiez tester si la configuration nouvellement créée vous convient vraiment, et la changer si ce n'est pas le cas.

Aide affichera ce fichier. Il vous faut amigaguide.library V34+ pour cela. A noter que vous devrez changer le nom de ce document si vous voulez que ce soit lui qui soit affiché et pas la version anglaise.

Annuler défera tout ce que vous avez fait. Si vous avez changé la configuration courante (par Tester), alors elle sera remise comme avant que le programme ne soit appelé. Ensuite le programme se termine.

Voilà pour les gadgets. Jetons un coup d'oeil aux menus:



Ouvrir chargera un nouveau fichier de configuration. Celui qui est en cours d'édition sera perdu. Fusionner chargera un nouveau fichier et le fusionnera avec la configuration que vous êtes en train d'éditer. Sauver sous écrira le fichier en cours d'édition dans un nouveau fichier. Pour ces trois actions, vous aurez un file requester d'ASL.

A propos vous informera de la version de WindowKey et de mon adresse. Quitter terminera le programme (Attention: aucun "Annuler" ou "Défaire" n'est effectué, une configuration en cours de test restera active).

Dernières valeurs sauvées récupèrera la dernière version sauvegardée du fichier de configuration permanent (la configuration sera chargée à partir de ENVARC:WindowKey.Prefs). Tout défaire annulera tous les changements que vous avez faits. Tout effacer effacera bien sûr tout le fichier que vous êtes en train d'éditer. Utilisez avec précautions !

Enfin, Créer icônes est un menu booléen qui dit au programme de préférences s'il doit ou non créer une icône pour un fichier qui sera sauvegardé sur disque par la fonction "Sauver sous...". L'icône sera un projet, dont l'image sera celle donnée par ENV:sys/def_prefs.info (ou l'icône projet par

défaut si inexistant), l'outil par défaut sera réglé sur le programme de préférences, et ACTION=USE sera en tooltype. De cette façon, double-cliquer sur cette icône changera la configuration courante de WindowKey sans entrer réellement dans le programme de préférences (ce qui ne veut pas dire qu'il n'est pas chargé).

A partir de la version 1.01, lancer le programme de préférences une nouvelle fois active, amène en avant et "dé-zoome" la fenêtre si nécessaire. Lui envoyer un CTRL-F a le même effet. L'envoi d'un CTRL-C le fera quitter.

1.17 3.3 Référence du langage

Cette section va vous apprendre comment écrire des commandes pour WindowKey. Elles doivent être entrées dans le gadget de chaîne Commande de la fenêtre du programme de préférences quand une hotkey est sélectionnée, ou envoyées par l'intermédiaire d'ARexx au port nommé WindowKey.

Une commande est constituée d'un mot-clé et d'un argument optionnel. Les commandes qui prennent un argument doivent avoir leur mot-clé suivi immédiatement par l'argument entre parenthèses. Ainsi, il peut être nécessaire pour les programmes ARexx d'entourer les parenthèses par des guillemets, car les premières sont traitées à part par ARexx. Mais les parenthèses doivent rester. Voyez les programmes d'exemple fournis. De multiples commandes sont séparées par des espaces.

Avant de lister toutes les commandes disponibles, je dois vous parler de la façon dont WindowKey nomme les fenêtres et les écrans, car de tels noms sont pris comme argument par la plupart de ces commandes. Tout est décrit en détail dans les sections nommés

Noms des fenêtres
et
Noms des écrans
.

Maintenant voici la liste des commandes que WindowKey comprend (celles qui prennent un argument ont () derrière elles):

WIN_ACTIVATE()

EXEC_NAMED()

WIN_TOFRONT()

EXEC_REXX()

WIN_TOBACK()

EXEC_PREFS

WIN_ZIP()

WIN_SETREMEMBER()

CMD_QUIT

WIN_MAKEVISIBLE()

CMD_UPDATE

WIN_SIZE()

CMD_NEWPREFS()

WIN_MOVE()

SCR_TOFRONT()

SCR_TOBACK()

SCR_SETREMEMBER()

SCR_MOVE()

SCR_STACK()

Avertissement: certaines de ces fonctions sont potentiellement dangereuse. Tous les input helpers ne vous le disent pas, mais il est impossible d'avoir une sécurité absolue. En effet, ce programme utilise des fenêtres et écrans d'autres programmes, qui peuvent se fermer à tout moment. Bien sûr, WindowKey examine toutes les structures d'Intuition en mode LockIBase(), mais il ne peut pas, par exemple, amener une fenêtre à l'avant dans ce mode. C'est pourquoi - bien que la transition soit faite en mode Forbid() - il se peut qu'il y ait des problèmes si vous invoquez les hotkeys de WindowKey lors d'activités de fenêtres ou écrans intenses. De tels problèmes sont très improbables, mais le risque existe (même le FKey de commodore agit comme cela). Vous êtes prévenus.

1.18 Conventions pour nommer les fenêtres

Pour trouver les fenêtres choisies par l'utilisateur dans les commandes WIN_, WindowKey utilise des noms, qui sont compris de différentes manières. Ce nom de fenêtre est soit le titre actuel de la fenêtre, ex:

Shell 8 in DH0{42MB}:Progs/WindowKey

soit l'un des noms spéciaux suivants (majuscules !):

- *A fenêtre Active
- *N fenêtre suivaNte (après l'active)
- *P fenêtre Précédente (avant l'active)
- *F fenêtre Frontale (sur l'écran frontal)
- *B fenêtre arrière (sur l'écran frontal)
- *M fenêtre sous la souris (tous les écrans sont examinés)
- *R fenêtre Remember

Les caractères escapes suivants sont disponibles (ce sont les mêmes que

ceux reconnus par Injector), bien qu'ils soient rarement utilisés dans les titres des fenêtres:

```
  \a  Bell (ASCII 7)
  \b  Backspace (ASCII 8)
  \c  Control sequence introducer (CSI, ASCII 155)
  \e  Escape (ASCII 27)
  \f  Form feed (ASCII 12)
  \n  New Line (ASCII 10)
  \r  Return (ASCII 13)
  \t  Tabulation horizontale (ASCII 9)
  \v  Tabulation verticale (ASCII 11)
  \xNN  Caractère dont le code ASCII est NN en hexadécimal.
  \NNN  Caractère dont le code ASCII est NNN en octal.
  \ Backslash
  \)  Parenthèse de fermeture (pas la fin des arguments).
```

L'option Remember marche de la façon suivante: chaque fois que vous agissez sur une fenêtre par l'intermédiaire de l'une des commandes WIN_, WindowKey stocke l'adresse de cette fenêtre pour qu'elle puisse être à nouveau utilisée avec le nom *R. Bien sur, des vérifications sont faites pour savoir si cette fenêtre est toujours ouverte avant d'être réutilisée. J'ai introduit cette possibilité pour résoudre ce petit problème: je voulais faire une hotkey qui amène la fenêtre arrière à l'avant et l'active. Malheureusement,

```
  WIN_TOFRONT(*B) WIN_ACTIVATE(*F)
```

ne marchait pas car Intuition lègue WindowToFront() et ActivateWindow() à son input handler, ce qui signifie que la fenêtre n'était pas encore en avant quand WIN_ACTIVATE() était exécutée. Par contre, ceci marche parfaitement:

```
  WIN_TOFRONT(*B) WIN_ACTIVATE(*R)
```

La valeur de remember peut aussi être altérée par la commande

```
WIN_SETREMEMBER
.
```

1.19 Conventions pour nommer les écrans

L'appellation des écrans diffère légèrement de celle des fenêtres, car il y a déjà une famille d'écrans possédant un nom: les écrans publics. WindowKey vous permet de spécifier un écran par son nom (ex: DEVPAC.1 ou Workbench, ...) ou son titre (ce qui marche aussi pour les écrans non publics). Si vous choisissez de référencer un écran par son titre, vous devez faire précéder celui-ci d'un signe @.

Vous pouvez aussi spécifier un écran par un nom de fenêtre: le premier écran contenant une fenêtre avec le titre spécifié sera retournée. Pour utiliser cette possibilité, utilisez simplement le titre de la fenêtre précédé d'un point d'exclamation (!).

Dernière note: le titre d'un écran signifie le titre par défaut de l'écran, et pas le titre courant de l'écran. La fenêtre active peut donner un autre titre à l'écran, mais ce n'est pas celui-là qui est vérifié (bien que c'est celui qui est affiché). ex, pour l'écran du Workbench:

"Workbench Screen" est le titre par défaut,
"Amiga Workbench 334 688 RAM graphique 892 184 autre RAM"
est le titre courant quand la fenêtre du workbench est active.

SCR_TOFRONT(@Workbench Screen) marchera, mais pas
SCR_TOFRONT(Amiga Work...).

Ces noms spéciaux (majuscules!) sont aussi reconnus:

*A écran Actif
*N écran suivaNt (après l'actif)
*P écran Précédent (avant l'actif)
*F écran Frontal
*B écran arrière
*M écran sous le pointeur de souris
*R écran Remember.

Vous pouvez bien sûr utiliser tous les caractères escapes (\a,\b,...) listés dans la section décrivant le
nom des fenêtres
.

L'option Remember est similaire à celle des fenêtres, sauf qu'elle concerne les écrans (!). Bien sûr, la validité du pointeur d'écran Remember est toujours vérifiée avant que celui-ci ne soit ré-utilisé.

1.20 La commande WIN_ACTIVATE()

Cette commande active la fenêtre sélectionnée. Inutile de préciser que WIN_ACTIVATE(*A) est sans effet...

1.21 La commande WIN_TOFRONT()

Cette commande amène la fenêtre sélectionnée en avant. L'écran sur lequel se ↩
trouve cette fenêtre
n'est _PAS_ amené en avant.

Petite particularité: les fenêtres BACKDROP ne peuvent pas être amenées en avant (ni en arrière); celles-ci restent toujours au fond. C'est pourquoi dans le cas de WIN_TOFRONT() et WIN_TOBACK(), les fenêtres BACKDROP sont ignorées, même si l'argument est *B. Ce comportement ne concerne pas les autres commandes WIN_.

1.22 La commande WIN_TOBACK

Cette commande renvoie la fenêtre sélectionnées au fond. L'écran sur lequel cette fenêtre se trouve n'est _PAS_ déplacé.

Petite particularité: les fenêtres BACKDROP ne peuvent pas être envoyées en arrière (ni à l'avant); celles-ci restent toujours au fond. C'est pourquoi dans le cas de WIN_TOFRONT() et WIN_TOBACK(), les fenêtres BACKDROP sont ignorées, même si l'argument est *B. Ce comportement ne concerne pas les autres commandes WIN_.

1.23 La commande WIN_ZIP

Cette commande "zoome" la fenêtre. Elle agit seulement si la fenêtre sélectionnée a un gadget de zoom. Par "zoomer", j'entends alterner la dimension et la position de la fenêtre avec les autres valeurs, telles qu'elles ont été définies par le possesseur de la fenêtre.

1.24 La commande WIN_SETREMEMBER()

Cette commande ajuste l'option
window remember
à la fenêtre spécifiée.
Toutes les autres commandes WIN_ ajustent le window remember automatiquement, cette commande est ainsi assez inutile (mais ne coûte rien).

1.25 La commande SCR_TOFRONT()

Cette commande amène l'écran sélectionné en avant. Aucune autre opération n'est effectuée.

Cette hotkey est vraiment utile pour cycler les écrans:

```
SCR_TOFRONT(*B) WIN_ACTIVATE(*F)
```

1.26 La commande SCR_TOBACK()

Cette commande envoie l'écran sélectionné en arrière.

1.27 La commande SCR_SETREMEMBER()

Cette commande ajuste l'option
screen remember
, dont le rôle est identique
au window remember, mais pour les écrans. Toutes les commandes SCR_ ajustent le tampon du screen remember.

1.28 La commande EXEC_NAMED()

Cette commande execute une hotkey comme si la touche correspondante avait été pressée par l'utilisateur. De cette façon vous pouvez faire des interprétations à la GOSUB des hotkeys. L'argument est bien évidemment le nom de la hotkey à exécuter. Il y a une sécurité dans WindowKey: quand une hotkey est exécutée, un bit spécial est mis à 1 dans sa structure, ce qui empêche les appels récursifs (vous aurez juste un message).

1.29 La commande EXEC_REXX()

Cette commande demande à WindowKey de lancer le programme ARexx dont le nom est en argument. Le programme aura son adresse hôte par défaut égale à "WindowKey". L'extension par défaut pour de tels programmes est .wndk .

Grâce à cette commande (et à ARexx !), vous pouvez générer des macros complexes, avec des tests, etc etc etc...

1.30 La commande EXEC_PREFS

Cette commande demande à WindowKey de lancer le programme de ↩
préférences.
Ceci est réalisé par la fonction SystemTagList() de la dos.library, donc le programme de préférences doit se trouver dans votre chemin. Le nom de fichier par défaut est "WindowKeyPrefs", mais il peut être modifié par le tooltype

PREFSPATH
.

Ceci peut aussi être fait en sélectionnant Montrer dans le programme commodities Exchange. A partir de la version 1.01, sélectionner Cacher terminera le programme de préférences (en lui envoyant un CTRL-C).

1.31 La commande CMD_QUIT

L'utilisation de cette commande est évidente: quand il la reçoit, et pourvu que ce soit possible, WindowKey se suicide. Ce n'est pas possible par exemple quand des programmes ARexx tournent encore.

1.32 La commande CMD_UPDATE

Cette commande n'est pas utile dans la plupart des cas. Quand il la reçoit, WindowKey recharge sa configuration. C'est inutile dans la plupart des cas car, comme le programme IPrefs, WindowKey utilise les possibilités de notification d'AmigaDOS pour détecter automatiquement les altérations du fichier de préférences (par le programme de préférences par exemple). Généralement, ENV: est assigné au Ram disk, donc ce n'est pas un problème.

Mais tous les gestionnaires ne gèrent pas la notification, alors, si quelqu'un utilise un assign sur un système de fichier en réseau par exemple, il doit utiliser cette commande pour qu'Injector mette à jour sa configuration.

Quand cette commande est reçue, WindowKey ne continue pas l'exécution de la ligne courante, même s'il restait des commandes.

1.33 La commande CMD_NEWPREFS

Cette commande changera de fichier de configuration. L'argument doit être un chemin valide jusqu'au fichier en question. Vous pouvez créer un tel fichier en sélectionnant le menu Sauver sous... dans le programme de configuration.

WindowKey fait le changement en invoquant le programme de préférences avec l'argument et le mot-clé USE, donc le programme de préférences doit être accessible. Vous pouvez utiliser l'option PREFSPATH pour cela.

1.34 La commande WIN_MAKEVISIBLE()

Cette commande ne marche que sur Kickstart 3.0 et plus pour l'instant. Si vous avez un écran plus grand que ne peut afficher votre moniteur (sans doute en mode autoscroll), cette commande déplacera l'écran de façon minimale pour rendre la fenêtre donnée en paramètre visible.

J'espère pouvoir la faire marcher sous Kickstart 2.0 dans le futur.

1.35 La commande SCR_MOVE()

Cette commande déplace un écran. L'argument doit être formatté de ↩
la façon
suivante:

SCR_MOVE(X,Y,Nom)

Nom est
nom d'écran
comme décrit plus haut, et X et Y sont des
valeurs décimales signées, indiquant de combien (de pixels) l'écran bougera sur les axes X et Y respectivement. Bien sûr, il y aura des restrictions si vous essayez de bouger l'écran trop loin.

Cette commande marche vraiment bien quand elle est utilisée en conjonction avec le qualifieur 'repeat' (voyez l'exemple de fichier de configuration).

1.36 La commande WIN_SIZE()

Cette commande change la taille d'une fenêtre. L'argument doit être formatté comme suit:

```
WIN_SIZE(X,Y,Nom)
```

X et Y sont des valeurs décimales (signées) indiquant les changements respectifs sur la largeur et la hauteur de la fenêtre. Nom est le nom de la fenêtre à affecter. Cette fenêtre sera réellement redimensionnée si et seulement si elle a un gadget de taille. Cette taille restera dans les limites définies par le possesseur de la fenêtre.

Cette commande n'est pas aussi rapide que, par exemple, SCR_MOVE() parce que chaque fois que la fenêtre est redimensionnée, le propriétaire peut avoir besoin de la rafraîchir, ainsi que les fenêtres éventuelles nouvellement découvertes. C'est pourquoi c'est une mauvaise idée que de l'utiliser avec le qualifier repeat.

1.37 La commande WIN_MOVE()

Cette commande change la position d'une fenêtre (cette position est bien sûr relative à l'écran de la fenêtre, et limitée par lui). L'argument doit être formatté de la façon suivante:

```
WIN_MOVE(X,Y,Nom)
```

X et Y sont les valeurs décimales (signées) qui renseignent WindowKey sur le déplacement de la fenêtre sur chaque axe. Nom est le nom de la fenêtre. La fenêtre sera effectivement bougée si et seulement si elle a une barre de déplacement.

Cette commande n'est pas très rapide (dépend de la vitesse de votre machine) pour les mêmes raisons que celles données pour WIN_SIZE() .

1.38 La commande SCR_STACK()

Cette commande est l'une des plus utiles dans WindowKey. Elle ré-organise toutes les fenêtres ouvertes sur un écran. L'argument pris par cette fonction doit être formatté de la façon suivante:

SCR_STACK(mode,nom)

Où mode est un paramètre numérique et nom un nom d'écran comme déjà défini. Les valeurs suivantes sont définies actuellement pour le mode. Les autres valeurs sont réservées pour une extension future et sont ignorées pour l'instant.

1 : Empile toutes les fenêtres, l'une sous la barre de titre de l'autre.
2 : Organise les fenêtres en diagonale, pour qu'au moins le gadget de profondeur de chaque fenêtre soit visible.
3 : Organise les fenêtres horizontalement. Toutes les fenêtres auront la même hauteur et seront placées l'une sous l'autre.
4 : Organise les fenêtres verticalement. Toutes les fenêtres auront la même largeur et seront placées l'une à coté de l'autre.
5 : Taille maximale. Toutes les fenêtres occuperont l'écran en entier.
6 : Grille. WindowKey essaiera de placer les fenêtres en grille pour qu'il y ait autant de lignes que de colonnes (dans la mesure du possible). L'écran sera occupé en entier, agrandissant les fenêtres de la dernière ligne si nécessaire.

Comme vous pouvez le remarquer, les cinq premières organisations sont les mêmes que celles disponibles dans l'éditeur de Devpac 3. Si vous avez des idées sur les nouvelles méthodes d'organisation que je pourrais ajouter, écrivez !

Bien sûr, aucune fenêtre ne sera rendue plus grande que sa taille maximale ou plus petite que sa taille minimale. La décision est laissée à Intuition alors ne m'en voulez pas si le résultat avec certaines fenêtres est un peu étrange !

Le processus de déplacement et de redimensionnement peut prendre un certain temps, surtout sur les machines lentes, si vous avez beaucoup de fenêtres. Plus que jamais, gardez l'avertissement (section Référence du langage) à l'esprit, ne fermez pas des fenêtres qui n'ont pas encore ↩ été bougées, ou vous aurez le gourou. Un homme averti en vaut deux.

Dernière chose mais pas la moindre, seules les fenêtres qui ont un gadget de redimensionnement ET une barre de mouvement seront effectivement bougées (ex: fenêtres Workbench, Shells, fenêtres DME...). Les autres (ex: docks toolmanager) restent intouchées.

Idée par Christophe Peugnet/ProMedia.

1.39 La commande WIN_CLOSE

Cette commande ne prend pas d'arguments. Son rôle est de fermer la fenêtre active. Elle marche sur toutes les fenêtres, celles qui traquent closewindow par le protocole IDCMP (la plupart) ainsi que celles qui utilisent le console.device (Shells).
Cependant, WindowKey ne ferme rien par lui même, car le système se planterait immédiatement. Au lieu de celà, il emet sur la chaîne de l'input.device un évènement IECLASS_CLOSEWINDOW. C'est pourquoi, d'ailleurs, vous ne pouvez pas sélectionner la fenêtre à fermer avec les noms de fenêtres habituels. Si vous voulez vraiment fermer une fenêtre qui n'est pas active, je suggère que vous utilisiez quelque chose comme celà:

```
WIN_ACTIVATE(Super nom de fenêtre) WIN_CLOSE WIN_ACTIVATE(*R)
```

Ceci marche car WIN_CLOSE n'affecte pas le tampon Window remember (vous vouliez garder la trace d'une fenêtre qui vient d'être fermée ?) et parce que WIN_CLOSE est mis en attente sur la queue d'entrées comme WIN_ACTIVATE (et donc parviennent à Intuition dans le même ordre).

1.40 4.1 Définir des touches de commande

Le texte qui suit ne dépend que du comportement de la commodities.library et pas de celui de WindowKey. C'est comme cela que commodities comprendra la combinaison de touches à laquelle vous voulez associer une action. Si vous avez des problèmes à vous faire comprendre de commodities, ou si vous êtes paresseux, Injector 2.12+ peut vous aider...

Une chaîne de description est constituée comme suit:

```
     [<classe>] {[-][<qualificateurs>]} [-][upstroke] [<code>]
```

Tous les mots-clés sont insensibles aux majuscules.

classe: doit être une classe d'InputEvent. Les classes supportées sont rawkey pour les évènements clavier, rawmouse pour les évènements souris, diskinserted et diskremoved. Par défaut c'est rawkey, qui doit généralement être utilisée.

qualificateurs: c'est une série de mots-clés représentant l'état des qualificateurs du clavier (Shift,Alt etc...). Voici une liste de mots-clés connus. Ceux qui sont marqués par un * sont nouveaux pour la version 38 de commodities.library.

```
lshift,left_shift *:        Touche shift gauche.
rshift,right_shift *:       Touche shift droite.
shift:            Une touche shift.
capslock,caps_lock *:       Touche Caps Lock.
caps:           Touche Caps Lock ou shift.
control,ctrl *:         Touche Control.
lalt,left_alt *:        Touche Alt gauche.
ralt,right_alt *:       Touche Alt droite.
alt:            Une touche Alt.
lcommand,lamiga *,left_amiga *,left_command *:  Touche amiga gauche.
rcommand,ramiga *,right_amiga *,right_command *:Touche amiga droite.
numericpad,numpad *,num_pad *,numeric_pad *:  Pour les touches du clavier
            numérique.
leftbutton,lbutton *,left_button *:   Bouton souris gauche.(1)
midbutton,mbutton *,middlebutton *,middle_button *:
```

```
            Bouton souris milieu.(1)
rbutton:,rightbutton *,right_button *:    Bouton souris droit.(1)
repeat:           Répétition active.(2)

Notes:  (1) la commodities.library V37 contenait une erreur qui l'empêchait
      d'utiliser     leftbutton,midbutton     et     rbutton     comme
      qualificateirs. Ce fut réparé pour la version 38.
  (2) pour la classe rawkey uniquement.
  (3) si  une  touche  de commande doit être insensible à l'état d'un
      qualificateur, placez un - avant son nom.

upstroke:   Normalement  un  évènement est généré seulement quand la touche
est  pressée.   Vous  pouvez  changer ce comportement en ajoutant upstroke.
Cela  générera  un  évènement uniquement quand la touche sera relachée.  Si
les faits de presser et de relacher doivent tous deux générer un évènement,
utilisez -upstroke.

code:   Ceux-ci  sont  significatifs  uniquement pour les classes rawkey et
rawmouse.   Pour rawkey,voici les codes de touches, * sont nouveaux pour la
version 38 de commodities.
a to z, 0 to 9:       Caractères ASCII normaux.
f1 to f10,f11 and f12:      Touches de fonction.
up,cursor_up *:       Flèche haut.
down,cursor_down *:     Flèche bas.
left,cursor_left *:     Flèche gauche.
right,cursor_right *:     Flèche droite.
esc,escape *:       Touche Esc.
backspace       Espace arrière.
del         Touche Del.
help          Touche Help.
tab         Touche Tab.
comma         Touche virgule (,).
return          Touche retour chariot.
space,spacebar *      Barre d'espace.
enter         Touche Enter.(4)
insert *        Touche pad 0.(4)
delete *        Touche pad 1.(4)
page_up *       Touche pad 9.(4)
page_down *       Touche pad 3.(4)
home *          Touche pad 7.(4)
end *         Touche pad 1.(4)

Notes:  (4) à utiliser avec le qualificateur numericpad.

Pour rawmouse, les codes valides sont:  (uniquement pour Commodities V38):
mouse_leftpress:      Bouton gauche pressé.(5)
mouse_middlepress:      Bouton du milieu pressé.(5)
mouse_rightpress:     Bouton droit pressé.(5)

Notes:  (5) vous devez utiliser aussi le qualificateur correspondant.
```

1.41 4.2 Historique

```
Revision V1.01
--------------
```

```
    Reproduit les changements d'Injector 2.30: tooltypes TOOLPRI
    et BREAKKEY, support de la commande Cacher, signaux breaks,
    tab pour activation du gadget de chaîne...

Revision V1.00
--------------
    --- Initial release ---
```

1.42 4.3 Contacter l'auteur

Je vous rappelle que WindowKey est SHAREWARE. Je sais que la plupart d'entre vous ne me paieront rien après leur période d'évaluation, mais sachez que j'ai passé beaucoup de temps à écrire et débugger ce programme, et que je voudrais juste une petite compensation qui me permettra d'acheter une bonne config hardware. Payer $10 vous fournira un support pour les prochaines versions. Je garde le source au chaud pour ceux d'entre vous qui sont intéressés par la programmation du merveilleux système d'exploitation de l'amiga, pour $10 de plus.

Pour tout ce qui concerne l'enregistrement, les commentaires, rapport de bugs, demandes d'améliorations, cartes postales, vous pouvez m'écrire à:

Frédéric DELACROIX
5 rue d'Artres
59269 QUERENAING
FRANCE.