

Récupérer les touches du clavier et de la souris

Fiche pratique de script LSL niveau intermédiaire

Ahuri Serenity
Professeur de script à l'Ecole SL
EM@iL

17 août 2010

Résumé

Dans cette fiche nous allons voir comment récupérer les touches du clavier et de la souris afin de donner plus de liberté à vos scripts.

Ce document n'est pas une substitution aux nombreux tutoriels disponibles sur Internet et ne vaut pas un cours, il est cependant indispensable et impératif que vous ayez bien intégré tous les points évoqués ici.

Table des matières

1 Introduction	1
2 Les éléments utilisés	1
3 Différentes façons d'appuyer	2
4 Implémentation	4
5 Récapitulatif	5

1 Introduction

Il existe différents types d'interface entre l'utilisateur et le script. On peut citer la souris avec le fait de toucher un objet, les commandes dans le chat ou encore les sites WEB. Une des plus utilisées est l'interface clavier. Il est en effet possible de récupérer la pression sur certaines touches du clavier et de la souris pendant l'exécution d'un script. Comme nous allons le voir les possibilités ne sont pas non plus énormes mais largement suffisantes dans la plupart des cas.

Après avoir vu les différents éléments utilisés, nous verrons de quelle manière sont récupérées les touches du clavier et de la souris puis comment mettre en place cette interface.

2 Les éléments utilisés

La capture des touches se fait grâce à la fonction [llTakeControls](#) dont voici le prototype :

```
1 llTakeControls( integer controles_a_interceptor, integer accept, integer transf );
```

Où *controles_a_interceptor* est l'ensemble des touches à récupérer (déparées deux à deux par le caractère "|"), *accept* permet d'activer ou pas la récupération et *transf* permet de priver ou non l'avatar de l'action standard de la touche (ex : si FALSE alors la touche flèche en haut pour avancer ne permet plus d'avancer).

Toutes les touches ne sont pas récupérables, voici la liste des touches pouvant être interceptées :

Type de touche	Constante associée
Déplacement vers l'avant	CONTROL_FWD
Déplacement vers l'arrière	CONTROL_BACK
Déplacement vers la gauche	CONTROL_LEFT
Déplacement vers la droite	CONTROL_RIGHT
Rotation vers la gauche	CONTROL_ROT_LEFT
Rotation vers la droite	CONTROL_ROT_RIGHT
Déplacement vers le haut	CONTROL_UP
Déplacement vers le bas	CONTROL_DOWN
Bouton gauche de la souris	CONTROL_LBUTTON
Bouton gauche de la souris en vue subjective	CONTROL_ML_LBUTTON

Cette fonction nécessite une demande de permission via la fonction [llRequestPermissions](#) :

```
1 llRequestPermissions( key agent, integer perm );
```

Le paramètre *agent* désigne la clé de l'avatar duquel la permission doit émaner et *perm* est le paramètre désignant quelle(s) permission(s) lui est(seront) demandée(s). Nous utiliserons ici la permission [PERMISSION_TAKE_CONTROLS](#).

Lorsque la permission est donnée ou refusée, on peut le savoir grâce à l'évènement [run_time_permissions](#) ce qui permet de réagir en conséquence.

```
1 run_time_permissions( integer perm ){ ; }
```

Ici, *perm* est un masque qui contient l'ensemble des permissions accordées pour le script.

Une fois que l'autorisation est accordée et que [llTakeControls](#) à été appelée et activée, chaque fois que l'utilisateur appuiera sur au moins une des touches interceptées, l'évènement [control](#) sera appelé.

Prototype de control :

```
1 control( key id, integer niveau, integer changement ){ ; }
```

Avec *id* la clé de l'utilisateur, *niveau* et *changement* deux données sur la façon d'appuyer sur les touches (voir Chap. 3).

Remarque : il existe un bug non résolu à cette date sur l'utilisation du paramètre *id*, quoiqu'il arrive celui-ci contiendra la clé du propriétaire du script ne permettant aucune distinction des utilisateurs.

Enfin, lorsque la capture des touches n'est plus nécessaire, on peut arrêter tout grâce à la fonction [llReleaseControls](#) qui libère aussi la permission.

3 Différentes façons d'appuyer

Les deux données *niveau* et *changement* de l'évènement control sont des masques. Il n'est pas question ici de donner un cours sur la notion de drapeau (flag) et de masque (mask), je vous conseille donc de vous

renseigner sur le sujet, si cela vous paraît étranger ou que ça vous intéresse. Ces données sont ici utilisées afin de rendre plus légers, pratiques et efficaces les tests à effectuer pour déterminer la façon de presser sur la touche.

En effet, il est possible de connaître différentes informations sur l'état de pression de la touche en inspectant les deux données *niveau* et *changement* :

- début de pression
- pendant la pression
- fin de pression
- pas de pression
- quelle(s) touche(s) est(sont) pressée(s)

Encore une fois, il n'est pas question ici de donner un cours sur les flags, voilà pourquoi je vais terminer ce chapitre sur des exemples :

Détecter une pression maintenue sur la touche 'avancer' :

```
1 if( niveau & CONTROL_FWD )
2 {
3     // ...
4 }
```

Détecter aucune pression sur la touche 'avancer' :

```
1 if( ~niveau & CONTROL_FWD )
2 {
3     // ...
4 }
```

Détecter le début de pression sur la touche 'avancer' :

```
1 if( niveau & changement & CONTROL_FWD )
2 {
3     // ...
4 }
```

Détecter la fin de pression sur la touche 'avancer' :

```
1 if( ~niveau & changement & CONTROL_FWD )
2 {
3     // ...
4 }
```

Tout se fait sur ce schéma. Par exemple si je veux faire la même chose mais avec la touche de rotation sur la gauche, je remplace `CONTROL_FWD` par `CONTROL_ROT_LEFT`. Si maintenant je veux détecter le début de pression simultanée sur les deux touches :

```
1 if( niveau & changement & (CONTROL_FWD|CONTROL_ROT_LEFT) )
2 {
3     // ...
4 }
```

Si je veux que la touche avancer soit pressée pendant que je détecte le début de pression sur la touche reculer :

```

1 if( (niveau & CONTROL_FWD) && (niveau & changement & CONTROL_ROT_LEFT) )
2 {
3     // ...
4 }

```

et si je veux détecter en plus de cela le début de pression simultanée sur la touche monter :

```

1 if( (niveau & CONTROL_FWD) && (niveau & changement & (CONTROL_ROT_LEFT|CONTROL_UP) ) )
2 {
3     // ...
4 }

```

Enfin pour garder un contrôle strict sur les touches pressées je vous conseille de procéder de la manière suivante :

```

1 if( niveau & changement == (CONTROL_FWD|CONTROL_ROT_LEFT) )
2 {
3     // ...
4 }

```

Avec cette dernière notation on s'assure que la combinaison est strictement identique à celle attendue, pratique pour les combos.

Pour plus d'informations ,je vous conseille [ce lien](#) (en anglais).

4 Implémentation

Voici un script permettant de faire monter / descendre un objet grâce aux touches PageUp/PageDown de votre clavier :

```

1 default
2 {
3     state_entry()
4     {
5         // On demande la permission au owner pour l'interception des touches
6         llRequestPermissions(llGetOwner(), PERMISSION_TAKE_CONTROLS);
7     }
8
9     run_time_permissions(integer perm)
10    {
11        // Lorsque l'utilisateur repond a la requete de permission, on trouve la reponse ici.
12
13        // Si il accepte on prend les controles sinon on l'avertit.
14        if (perm & PERMISSION_TAKE_CONTROLS)
15        {
16            // On intercepte les touches monter / descendre
17            // La fonction est activee et l'action standard des touches est substituee.
18            llTakeControls(CONTROL_UP | CONTROL_DOWN, TRUE, FALSE);
19        }
20        else
21        {
22            llOwnerSay("Il faut accepter la permission pour continuer !");
23            llResetScript();
24        }
25    }

```

```

26
27 control(key id, integer niveau, integer changement)
28 {
29     // L'utilisateur vient d'appuyer sur une des touches interceptees
30
31     // Si il presse la touche monter :
32     if (niveau & CONTROL_UP)
33     {
34         // On fait monter l'objet
35         llSetPos(llGetPos() + <0, 0, 0.18>);
36     }
37     // Sinon si il debute une pression sur la touche descendre :
38     else if (changement & niveau & CONTROL_DOWN)
39     {
40         // On fait descendre l'objet
41         llSetPos(llGetPos() + <0, 0, -0.18>);
42     }
43 }
44 }

```

Observez bien le comportement du déplacement suivant que vous maintenez ou pas la pression sur chacune des touches.

5 Récapitulatif

- Demande de permission : llRequestPermissions et PERMISSION_TAKE_CONTROLS.
- Réponse à la demande : run_time_permissions.
- Prise de contrôles : llTakeControls.
- Interception des touches : control.
- Tests multiples pour agir en fonction de l'état de pression des touches capturées : début/fin de pression et pression en cours ou pas.
- Libération des contrôles : llReleaseControls.