

Lecture de notecard

Fiche pratique de script LSL niveau intermédiaire

Ahuri Serenity

Professeur de script à l'Ecole SL

EM@iL

20 juillet 2010

Résumé

Dans cette fiche nous allons voir comment lire une notecard grâce à un script, à quoi cela peut servir et quels avantages cela apporte.

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 | 2 |
| 3 | L'implémentation | 2 |
| 4 | Aller plus loin ... paramétrer son script ! | 3 |
| 4.1 | Nettoyer une ligne | 3 |
| 4.2 | Prise en compte des commentaires et des lignes vides | 4 |
| 4.3 | Découper et reconnaître le paramètre courant | 4 |
| 4.4 | Tout en un | 5 |
| 5 | Aller plus loin ... pourcentage de progression ! | 7 |
| 6 | Récapitulatif | 8 |

1 Introduction

Un script LSL a la possibilité de lire les notecards, c'est à dire, des fichiers de texte bruts. Cette fonction est bien pratique lorsqu'il s'agit d'offrir au futur utilisateur la possibilité de paramétrer le système. Nous pouvons citer comme exemples significatifs les AOs (notecard avec les correspondances animation/action), les rentalboxes (notecard avec les tarifs, les contacts, les limites de location, les messages d'alerte) ou encore les portes (liste d'accès). Les notecards permettent aussi d'empêcher l'utilisateur d'avoir à ouvrir le script et donc d'avoir une configuration souple tout en ayant un script non modifiable. On s'aperçoit alors de l'avantage, si ce n'est parfois la nécessité, d'utiliser les notecards pour configurer nos scripts.

2 Les éléments utilisés

Dans un script, la lecture de notecard se fait ligne par ligne. Il est impossible d'obtenir en même temps le contenu de plusieurs lignes grâce à un seul script. Pour se faire, on utilise la fonction `llGetNotecardLine` qui appelle l'évènement `dataserver` et on utilise la constante `EOF`.

Voici le prototype de la fonction `llGetNotecardLine` :

```
1 key llGetNotecardLine( string nom, integer ligne );
```

Le paramètre *nom* est le nom de la notecard dans l'inventaire de l'objet et le paramètre *ligne* est le numéro de la ligne à lire dans cette notecard. À noter tout de même que les lignes sont numérotées à partir de 0 donc la 3^{ème} ligne porte le numéro 2. La fonction renvoie une clé qui est l'identifiant de la requête qui va servir lors des traitements et lors de son exécution l'évènement `dataserver` est appelé.

Et voici le prototype de l'évènement `dataserver` :

```
1 dataserver( key idrequete, string donnee ){ ; }
```

Le paramètre *idrequete* est la clé identifiant de la requête en train d'être traitée (si je viens de lancer `llGetNotecardLine` celle ci m'a renvoyé une clé qui est identique à *idrequete*) et le paramètre *donnee* est une chaîne de caractères contenant la ligne de la notecard. Lorsqu'on est arrivé à la fin de la notecard, *donnee* vaut `EOF`. Tester l'identifiant de la requête permet de faire la distinction avec d'autres types de requêtes faisant appel à `dataserver`.

Ajoutons maintenant qu'il existe une autre fonction : `llGetNumberOfNotecardLines`. Cette dernière fonction permet de récupérer le nombre de lignes de la notecard, bien pratique si l'on souhaite calculer au fur et à mesure le pourcentage de lignes lues afin d'informer l'utilisateur de la progression (voir Chap. 5). Voici son prototype :

```
1 key llGetNumberOfNotecardLines( string nom );
```

Elle aussi prend en paramètre le nom de la notecard, renvoie une clé identifiant et appelle `dataserver`. Sauf que pour cette requête le paramètre *donnee* de `dataserver` contient le nombre de lignes (sous forme de chaîne de caractères).

3 L'implémentation

Voyons ensemble un code inspiré du wiki :

```
1 key    kQuery;
2 integer iLine = 0;
3 string sNCName = "Note";
4
5 default
6 {
7
8     state_entry()
9     {
10         llOwnerSay( "Lecture de la note...");
11
12         // Lit la note "Note" (si possible)
13         // Lecture a partir de la ligne 0
14         kQuery = llGetNotecardLine( sNCName, iLine);
```

```

15     }
16
17     dataserver(key query_id, string data)
18     {
19         // Verification sur l'identifiant de la requete
20         if (query_id == kQuery)
21         {
22             // Si on est ici c'est qu'on est bien train de lire la notecard
23
24             // Si on est a la fin du document ...
25             if (data == EOF)
26             {
27                 // On peut stopper la lecture
28                 llOwnerSay( "Plus de lignes dans la note : " + (string)iLine + " ligne(s) lue(s).");
29             }
30             // Sinon ...
31             else
32             {
33                 // On continue a lire !
34
35                 // On affiche ici le numero de la ligne courante ainsi que son contenu
36                 llOwnerSay( "Ligne " + (string)iLine + ": " + data);
37
38                 // Puis on incremente le numero de la ligne de maniere a aller lire la ligne suivante
39                 ++iLine;
40
41                 // Lecture de la ligne ce qui va refaire appel a dataserver
42                 kQuery = llGetNotecardLine( sNCName, iLine);
43             }
44         }
45     }
46 }

```

Il ne faut pas oublier d'incrémenter la variable contenant le numéro de la ligne courante avant de relancer la lecture sinon on ne progresse pas comme il faut voir on boucle si elle n'est incrémentée nulle part !

4 Aller plus loin ... paramétrer son script !

Nous ne traiterons ici que d'une seule façon de paramétrer son script. Cette méthode est très souple et confortable pour l'utilisateur puisqu'il peut ajouter des commentaires, sauter des lignes, visualiser où il doit modifier les valeurs et comment il doit s'y prendre. On considère que la ligne courante est contenue dans la variable *data*.

4.1 Nettoyer une ligne

Avant tout traitement et toute interprétation d'une ligne de configuration, il est bon de la nettoyer, c'est à dire d'enlever les espaces avant et après le texte. De cette manière, la recherche de ligne de commentaire se fera simplement et les lignes vides apparaîtront d'elles mêmes :)

Afin de nettoyer *data*, on applique simplement la fonction `llStringTrim`. Voici directement l'implémentation :

```

1 data = llStringTrim( data, STRING_TRIM);

```

4.2 Prise en compte des commentaires et des lignes vides

Choisissons un caractère pour la déclaration des commentaires : "#". Ce caractère symbolisera donc une ligne de commentaire dans le texte de la notecard. Il devra être placé en tête de ligne (les espaces et tabulations à gauche sont autorisés). Voici un exemple de notecard avec des commentaires :

```
1 # Notecard de configuration de script
2 # Tout ce texte est commente et ne sert pas pour la configuration
3
4 # L utilisateur peut rajouter ses propres commentaires
5     # il peut meme decaller son texte !
6
7 # Premier parametre : vitesse du bolide (en m/s)
8     VITESSE = 30
9 # La ligne precedente n est pas commentee et n est pas vide, elle sert donc pour la configuration.
10
11 # Second parametre : nombre max de passagers (< 11)
12     PASSAGERS = 5
```

On suppose ici avoir effectué le nettoyage de *data* (Chap. 4.1). Voyons maintenant comment tester la validité de la ligne courante *data* :

```
1 // Si la ligne n'est pas vide et si le premier caractere n'est pas "#" :
2 if( data != "" && llGetSubString( data, 0, 0) != "#" )
3 {
4     // Ici on execute les decoupages et autres traitements
5 }
```

`llGetSubString` permet d'obtenir un segment de chaîne de caractères, ici on ne récupère que le premier caractère afin de le comparer à #.

4.3 Découper et reconnaître le paramètre courant

Nous voici maintenant dans l'ultime étape : le découpage de la ligne courante. Nous supposons que la forme des données est la suivante :

```
1 NOM_DONNEE = valeur
```

Ainsi, le découpage devra se faire suivant le signe "=" (on utilise `llParseString2List` qui permet de découper les chaînes de caractères selon des séparateurs déterminés). On récupère ainsi une liste de 2 chaînes de caractères qu'on appelle *duo* :

```
1 // Creation de la liste des 2 termes pour la ligne courante
2 list duo = llParseString2List( data, ["="], []);
```

Ensuite on récupère les deux morceaux qu'on s'empresse de nettoyer à la manière du chapitre 4.1 afin d'éliminer les espaces en trop autour du signe égal :

```
1 // nom de la donnee a configurer
2 string label = llStringTrim( llList2String( duo, 0), STRING_TRIM);
3 // valeur de configuration pour cette donnee
4 string valeur = llStringTrim( llList2String( duo, 1), STRING_TRIM);
```

Il nous reste ensuite à tester *label* pour reconnaître ce que nous devons faire et à traiter *valeur* :

```
1 if( label == "VITESSE" )
2 {
```

```

3   fSpeed = (float)valeur;
4   // fSpeed etant une variable de type float utilisee dans le script entier
5 }
6 else if( label == "PASSAGERS" )
7 {
8   iNbPassagers = (integer)valeur;
9   if( iNbPassagers < 0 ) iNbPassagers = 0;
10  if( iNbPassagers > 10 ) iNbPassagers = 10;
11  // iNbPassagers etant une variable de type integer utilisee dans le script entier
12 }

```

4.4 Tout en un

```

1 key   kQuery;
2 integer iLine   = 0;
3 string sNCName  = "Config";
4
5 // Parametres de mon script
6 float fSpeed    = 2.0;
7 integer iNbPassagers = 3;
8
9 default
10 {
11
12   state_entry()
13   {
14     llOwnerSay( "Lecture de la note..." );
15
16     // Lit la note "Config" (si possible)
17     // Lecture a partir de la ligne 0
18     kQuery = llGetNotecardLine( sNCName, iLine );
19   }
20
21   touch_start( integer p )
22   {
23     llOwnerSay( "Vitesse = +(string)fSpeed+ " \n Nombre de passagers = +(string)iNbPassagers );
24   }
25
26   dataserver( key query_id, string data )
27   {
28     // Verification sur l'identifiant de la requete
29     if ( query_id == kQuery )
30     {
31       // Si on est ici c'est qu'on est bien train de lire la notecard
32
33       // Si on est a la fin du document ...
34       if ( data == EOF )
35       {
36         // On peut stopper la lecture
37         llOwnerSay( "Plus de lignes dans la note : " + (string)iLine + " ligne(s) lue(s).");
38       }
39       // Sinon ...
40       else
41       {

```

```

42 // On continue a lire !
43
44 // Premier nettoyage
45 data = llStringTrim( data, STRING_TRIM);
46
47 // Si la ligne n'est pas vide et si le premier caractere n'est pas "#" :
48 if( data != "" && llGetSubString( data, 0, 0) != "#" )
49 {
50 // Creation de la liste des 2 termes pour la ligne courante
51 list duo = llParseString2List( data, ["="], []);
52
53 // nom de la donnee a configurer
54 string label = llStringTrim( llList2String( duo, 0), STRING_TRIM);
55 // valeur de configuration pour cette donnee
56 string valeur = llStringTrim( llList2String( duo, 1), STRING_TRIM);
57
58 if( label == "VITESSE" )
59 {
60 fSpeed = (float)valeur;
61 // fSpeed etant une variable de type float utilisee dans le script entier
62 }
63 else if( label == "PASSAGERS" )
64 {
65 iNbPassagers = (integer)valeur;
66 if( iNbPassagers < 0 ) iNbPassagers = 0;
67 if( iNbPassagers > 10 ) iNbPassagers = 10;
68 // iNbPassagers etant une variable de type integer utilisee dans le script
69 // entier
70 }
71 }
72 // Puis on incremente le numero de la ligne de maniere a aller lire la ligne suivante
73 ++iLine;
74
75 // Lecture de la ligne ce qui va refaire appel a dataserver
76 kQuery = llGetNotecardLine( sNCName, iLine);
77 }
78 }
79 }
80 }

```

Correspondant à une notecard de la forme :

```

1 # Notecard de configuration de script
2 # Tout ce texte est commente et ne sert pas pour la configuration
3
4 # L utilisateur peut rajouter ses propres commentaires
5 # il peut meme decaller son texte !
6
7 # Premier parametre : vitesse du bolide (en m/s)
8 VITESSE = 30
9 # La ligne precedente n est pas commentee et n est pas vide, elle sert donc pour la configuration.
10
11 # Second parametre : nombre max de passagers (< 11)
12 PASSAGERS = 5

```

Lorsqu'on cliquera sur l'objet après la lecture de la notecard, voici ce qui s'affichera :

*Vitesse = 30.0
Nombre de passagers = 5*

5 Aller plus loin ... pourcentage de progression !

Voilà une section courte pour vous montrer comment afficher le pourcentage de progression dans la lecture de la notecard (ai je besoin de préciser que cela ne sert à rien pour des notecards très petites?) :

```
1 key    kQuery;
2 key    kQueryLines;
3 integer iLine = 0;
4 string sNCName = "Note";
5 integer iNbOfLines;
6
7 default
8 {
9
10    state_entry()
11    {
12        llOwnerSay( "Lecture de la note...");
13        llSetText("C'est parti !!", <1,1,1>, 1.0);
14
15        kQueryLines = llGetNumberOfNotecardLines( sNCName);
16    }
17
18    dataserver(key query_id, string data)
19    {
20        // Si je cherche le nombre de lignes de la notecard
21        if( query_id == kQueryLines )
22        {
23            iNbOfLines = (integer)data;
24
25            // Le nombre de lignes est connu, on peut commencer a lire la notecard :
26            kQuery = llGetNotecardLine( sNCName, iLine);
27        }
28        // Sinon si il s'agit de ma lecture de notecard
29        else if (query_id == kQuery)
30        {
31            // Si on est ici c'est qu'on est bien train de lire la notecard
32
33            // On effectue le calcul de progression (produit en croix) et on affiche le resultat
34            integer progression = (integer)(iLine * 100.0 / iNbOfLines);
35            llSetText("Progression ... "+(string)progression+" %", <1,1,1>, 1.0);
36
37            // Si on est a la fin du document ...
38            if (data == EOF)
39            {
40                // On peut stopper la lecture
41                llOwnerSay( "Plus de lignes dans la note : " + (string)iLine + " ligne(s) lue(s).");
42                llSetText("", <1,1,1>, 1.0);
43            }
44            // Sinon ...
45            else
46            {
```

```

47         // On continue a lire !
48
49         // On affiche ici le numero de la ligne courante ainsi que son contenu
50         llOwnerSay( "Ligne " + (string)iLine + ": " + data);
51
52         // Puis on incremente le numero de la ligne de maniere a aller lire la ligne suivante
53         ++iLine;
54
55         // Lecture de la ligne ce qui va refaire appel a dataserver
56         kQuery = llGetNotecardLine( sNCName, iLine);
57     }
58 }
59 }
60 }

```

Notez bien ici l'importance d'identifier ses requêtes.

Maintenant je vous laisse en exercice de rassembler les deux derniers principes : pourcentage et paramétrisation! Cela va faire un script bien complet qui vous resservira des tonnes de fois!

6 Récapitulatif

Lire simplement une notecard :

- key llGetNotecardLine(string nom, integer ligne); \Leftarrow Lire la ligne d'une notecard donnée.
- dataserver(key idrequete, string donnee){ ; } \Leftarrow Évènement appelé par la fonction précédente présentant le contenu de la ligne courante.
- donnee vaut EOF à la fin de la notecard.
- ne pas oublier d'incrémenter la ligne à lire à chaque tour.
- ne pas oublier de tester la validité de la requête.

Paramétrer son script :

- nettoyage des lignes.
- test de validité des lignes.
- découpage des lignes pour en extraire la donnée à paramétrer et sa valeur.
- ne pas oublier de nettoyer les données extraites.

Obtenir le pourcentage de progression son script :

- key llGetNumberOfNotecardLines(string nom); \Leftarrow Obtenir le nombre de lignes d'une notecard.
- calcul et affichage du pourcentage.
- ne pas oublier de tester l'identifiant de la requête.