Comment installer et programmer des scripts Python dans Blender ?

Vincent Vansuyt

v 1.02

Table des matières

1 Introduction

Cet article indique comment ajouter et programmer des scripts Python dans Blender.

En pré-requis il faut savoir un peu manipuler les fenêtres de Blender. Vous pouvez lire par exemple le "Mini-tutoriel Blender" (seulement 5 pages), dans mes pages "Tutoriels Blender".

Si vous souhaitez seulement ajouter des scripts existant dans Blender, vous pouvez aller directement à la section 5 "Liaison de scripts Python avec les menus de Blender", page 5.

2 Installation

2.1 Vérifications après l'installation de Blender

Suivant les distributions de Blender, l'installation de Blender pose parfois des problèmes d'exécution des scripts.

Le script ci-contre, listing 1, doit fonctionner dans une fenêtre "éditeur de texte" de Blender.

Listing 1 – listing test

```
1 import Blender
print Blender.Get('uscriptsdir')
```

Les variables "path" qui fonctionnent sur Ubuntu et Windows doivent ressembler à ceci :

Sous Linux Ubuntu 8.04 :

```
'/home/magoo/Softs/blender-2.46-linux-glibc236-py25-i386',
'/usr/lib/python25.zip',
'/usr/lib/python2.5',
'/usr/lib/python2.5/plat-linux2',
'/usr/lib/python2.5/lib-tk',
'/usr/lib/python2.5/lib-dynload',
'/usr/local/lib/python2.5/site-packages',
'/usr/lib/python2.5/site-packages',
'/usr/lib/python2.5/site-packages/Numeric',
'/usr/lib/python2.5/site-packages/PIL',
'/usr/lib/python2.5/site-packages/gst-0.10',
'/var/lib/python-support/python2.5',
'/usr/lib/python2.5/site-packages/gtk-2.0',
'/var/lib/python-support/python2.5/gtk-2.0',
'/home/magoo/scripts_python',
'/home/magoo/Softs/blender-2.46-linux-glibc236-py25-i386/.blender/scripts',
'/home/magoo/Softs/blender-2.46-linux-glibc236-py25-i386/.blender/scripts/bpymodules']
```

Sous Windows :

```
C:\Python25
C:\Python25\Lib
C:\Python25\Lib\lib-tk
C:\Python25\lib\site-packages
C:\Python25\lib\site-packages\wx-2.8-msw-unicode
C:\Python25\DLLs
C:\Program Files\Blender
C:\PROGRA~1\Blender
C:\Program Files\Blender\python25.zip
C:\Program Files\Blender\.blender\scripts
C:\Program Files\Blender\.blender\scripts\bpymodules
```

2.2 Extensions Python compatibles avec Blender

Blender nécessite l'installation au minimum de Python, mais on bénéficie également dans les scripts Blender, de toutes les librairies additionnelles que l'on installe pour Python. On trouve l'installation de la dernière version de Python ici : http://www.python.org/

L'installation de Python par défaut permet de parser des fichiers XML, mais pas de faire de calcul matriciel. Pour avoir accès aux matrices et leurs fonctions de base, il faut installer NumPy. NumPy donne accès aux matrices et vecteurs.

Pour installer toutes les fonctions d'algèbre linéaire, traitement du signal, statistiques, il faut installer Scipy.

Les dernières version de NumPy et SciPy se trouvent ici : http://www.scipy.org/

(Ce site contient également une excellente documentation sur ces librairies)

On peut aussi avoir accès aux bases de données mySQL avec mySQLDb, créer des interfaces graphiques avec Qt grâce à pyQt, etc.

3 Avec quel éditeur développer en Python ?

Il existe le plugin PyDev pour Eclipse, qui fonctionne très bien. Malheureusement, je n'ai pas réussi à l'utiliser avec les librairies Blender. La question est actuellement sans réponse sur le forum www.developpez.com.

En attendant, pour les petits projets, n'importe quel éditeur de texte peut convenir.

Il existe un énorme choix d'autres éditeurs ici : http://wiki.python.org/

4 Comment écrire et exécuter un premier programme Python ?

4.1 Sans Blender

Python est un langage plus simple que le C++ (mais néanmoins objet) qui ressemble un peu au C. Voici un exemple très simple de programme Python :

Listing 2 – Exemple simple de programme Python "C :\tmp\Hello.py"

```
print "Hello"
a = 1.5
b = 1.2
print "a + b = %.3f"%( a + b )
```

Dans ce cas, l'installation de Python est dans le dossier `C:\Python25`.

Pour exécuter le programme précédente "Hello.py", il suffit de taper dans une console :

```
C:\tmp>c:\python25\python hello.py
```

Résultat :

```
Hello
a + b = 2.700
```

4.2 Avec Blender

Dans Blender, l'éditeur de texte permet d'écrire et d'exécuter des scripts (voir figures 1 et 2).

FIG. 1 – Configuration d'une fenêtre Blender en éditeur de texte

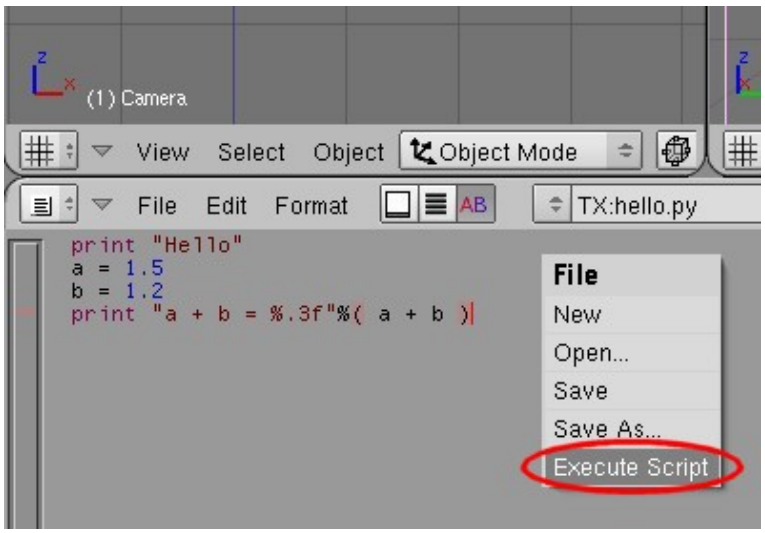

FIG. 2 – Exécution d'un script dans Blender

Le résultat suivant s'affiche dans la console de Blender :

```
Hello
a + b = 2.700
```

4.3 Comment afficher la console (ou terminal) ?

Effectivement, sous Linux Ubuntu et sur Macintosh, l'exécution de Blender par défaut n'affiche pas de seconde fenêtre en mode texte (la "console" ou "terminal").

Pour remédier à ce problème, on peut écrire un petit fichier de lancement, comme ceci (à adapter, bien sûr selon les dossiers que vous utilisez). Voici la solution que j'ai utilisé pour Linux Ubuntu 8.04, ce n'est peut-être pas la meilleure :

```
cd /home/Toto/Soft/Blender-2.46-linux-glibc236-py25-i386
./blender
```

FIG. 3 – Script "Launch_Blender" de lancement de Blender avec la console

Dans les propriétés de ce fichier, choisir l'onglet "Permissions" et cocher la case "Autoriser l'exécution du fichier comme un programme".

Ensuite, on peut créer un lien vers ce fichier et le placer sur le bureau.

Lorsque l'on double-clique dessus, une fenêtre de choix s'affiche ; choisir "Lancer dans un terminal".

Normalement Blender s'exécute normalement et derrière, il y a la fenêtre "Console" qui affiche tous les sorties "print" des programmes Python et erreurs survenues.

Sous Linux Ubuntu, la console affiche dès le début :

```
Compiled with Python version 2.5.2
```

Avec Linux Kubuntu, si une application a été installée avec un package, on peut modifier les propriétés de l'exécutable pour qu'il se lance dans un terminal. On évite donc la manipulation précédente.

5 Liaison de scripts Python avec les menus de Blender

Dans Blender, les scripts Python sont signalés par un petit logo vert représentant un python (le serpent). On en trouve dans les menus `File \ Import`, `File \Export`, `Object`, `Mesh` (voir copies d'écran 4 à 6).

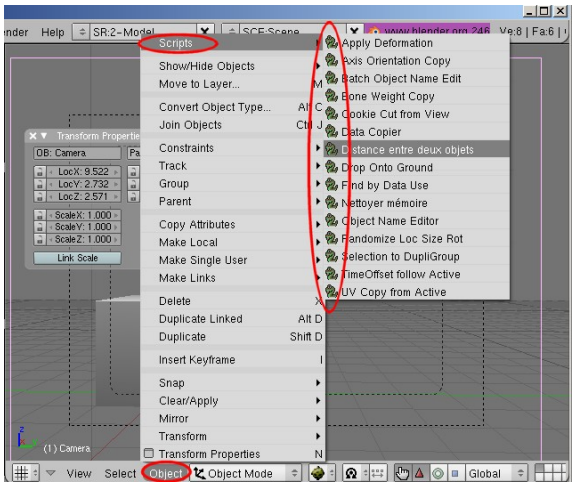

FIG. 4 – Scripts dans le menu "object"

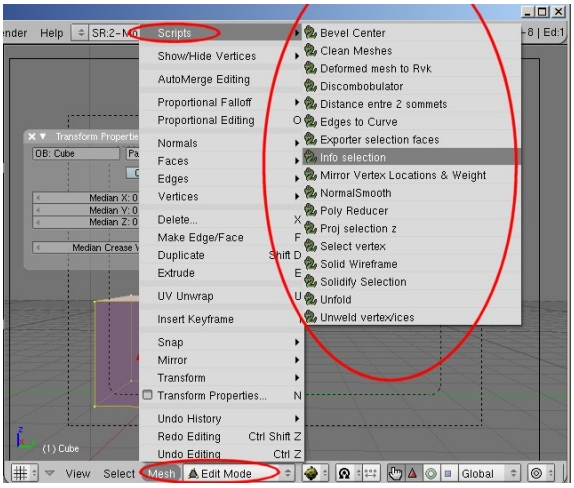

FIG. 5 – Scripts dans le menu "mesh"

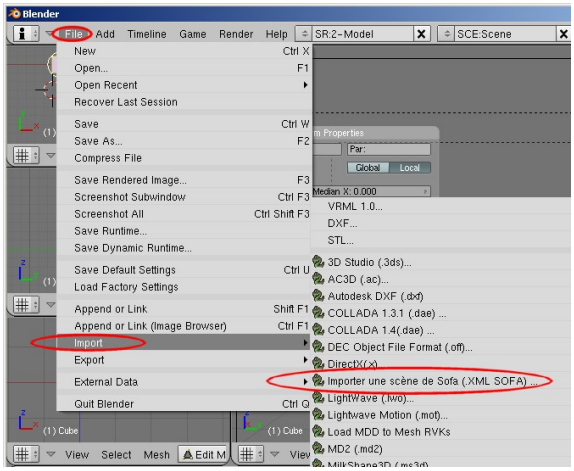

FIG. 6 – Scripts dans le menu "File - import"

Dans la figure ci-contre (6), on remarque le script d'import des scènes SOFA, entouré en rouge.

Sous Windows, tous les scripts livrés par défaut avec Blender sont dans le chemin :

`C:\Program Files\Blender\.blender\scripts`

Chaque script a l'extension `.py`. Pour savoir à quel menu il se rapporte, il suffit de l'ouvrir (en général, le nom du fichier donne déjà l'indication). Voici par exemple le début du script "`import_obj.py`" :

Listing 3 – exemple du script "import_obj.py" livré avec Blender

```
#!BPY

"""
Name: 'Wavefront (.obj)...'
Blender: 242
Group: 'Import'
Tooltip: 'Load a Wavefront OBJ File, Shift: batch import all dir.'
"""

__author__= "Campbell Barton", "Jiri Hnidek"
__url__= ["blender.org", "blenderartists.org"]
__version__= "2.0"

__bpydoc__= """\
This script imports a Wavefront OBJ files to Blender.

```

```
Usage:
Run this script from "File->Import" menu and then load the desired OBJ file.
Note, This loads mesh objects and materials only, nurbs and curves are not supported.
"""
```

On remarque ligne 6 du listing 3 (`Group: 'Import'`), ci-dessus, que ce script est rattaché au menu "`File Import`".

Pour ajouter un script dans un menu Blender, il suffit donc d'ajouter un fichier ".py" avec le "bon" entête dans le dossier des scripts Python de Blender. Cela fonctionne, mais ce n'est pas la bonne solution.

Pour éviter de mélanger ses scripts avec ceux de Blender, il faut configurer l'endroit où sont les scripts utilisateurs dans Blender. Pour cela, ouvrir un une fenêtre Blender en mode "Préférences utilisateur" :

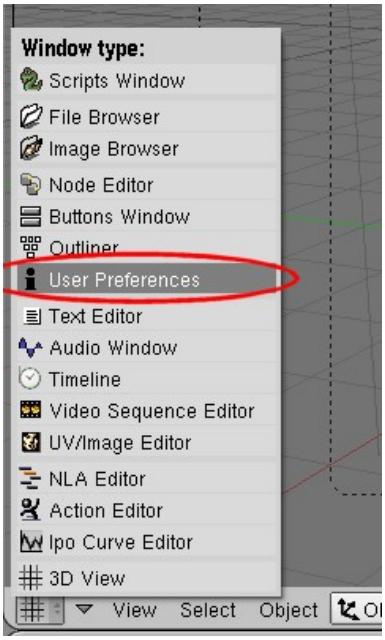

FIG. 7 – Accès à la fenêtre des préférences de l'utilisateur

Dans le menu "File Path", configurer le chemin des scripts de l'utilisateur (voir figure 8).

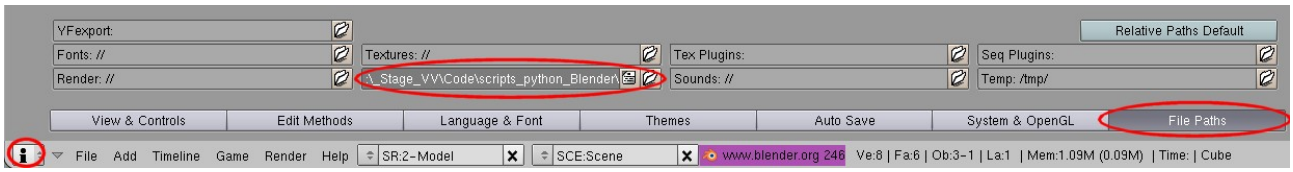

FIG. 8 – Configuration des scripts de l'utilisateur

Et donc mes scripts pour Blenders sont isolés dans le chemin
"`C:\_Stage_VV\Code\scripts_python_Blender`".

Enfin, pour que Blender se souvienne de ce choix, il faut enregistrer ce changement dans les préférences utilisateur en tapant "`CTRL-U`" ou par le menu `File \ Save Default Settings`.

6 Compatibilité des scripts Python pour Blender : attention à la “BOM”

Il peut arriver qu’un script Python soit visible dans les menus Blender-Windows et pas dans ceux de Blender-Linux (j’ai rencontré avec Linux Ubuntu v8.04, python 2.5 et Blender 2.46).

Le script n’est pas visible dans les menus de Blender-Linux lorsque le fichier texte du script contient une “BOM”. La “B.O.M.” est la “Byte Order Mark”. C’est une marque de quelques octets en début des fichiers UTF-(8, 16 ou 32).

Cette marque indique si le fichier est codé en “big endian” ou “little endian”, c’est à dire si les octets de poids fort sont en premier ou pas.

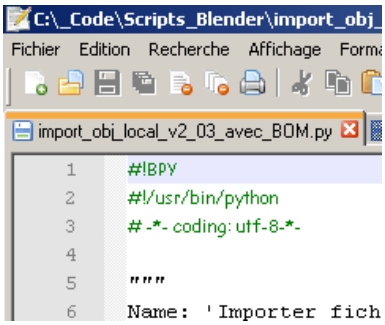

FIG. 9 – Texte utf-8 avec BOM ouvert sous Windows avec Notepad++

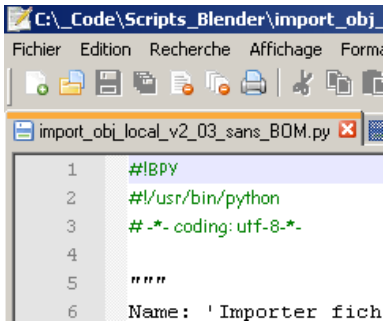

FIG. 10 – Texte utf-8 sans BOM ouvert sous Windows avec Notepad++

Ci-dessus, figures 9 et 10, le même fichier avec ou sans la “BOM”, ouverts dans l’éditeur de texte Notepad++. On ne remarque strictement aucune différence.

Par contre si on passe ces deux fichiers en mode hexadécimal, on obtient ceci (figures 11 et 12) :

+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f	Dump
23	21	42	50	59	0d	0a	23	21	2f	75	73	72	2f	62	69	#!BPY..#!/usr/bi
6e	2f	70	79	74	68	6f	6e	0d	0a	23	20	2d	2a	2d	20	n/python..# -*-
63	6f	64	69	6e	67	3a	20	75	74	66	2d	38	2d	2a	2d	coding: utf-8-*-

FIG. 11 – Texte sans BOM, mode hexadécimal

+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f	Dump
ef	bb	bf	23	21	42	50	59	0d	0a	23	21	2f	75	73	72	ï»¿#!BPY..#!/usr
2f	62	69	6e	2f	70	79	74	68	6f	6e	0d	0a	23	20	2d	/bin/python..# -
2a	2d	20	63	6f	64	69	6e	67	3a	20	75	74	66	2d	38	*- coding: utf-8

FIG. 12 – Texte avec BOM, mode hexadécimal

On voit, figure 12, que la BOM est la suite d’octets `0x0EF, 0x0BB, 0x0BF`.

Si on ouvre ces mêmes fichiers sous Linux Ubuntu 8.0.4 avec gedit, on obtient les copies d’écran suivantes :

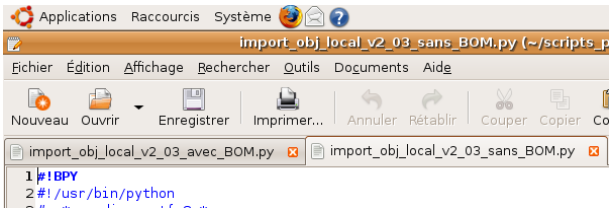

FIG. 13 – Texte utf-8 sans BOM ouvert sous Linux Ubuntu avec gedit

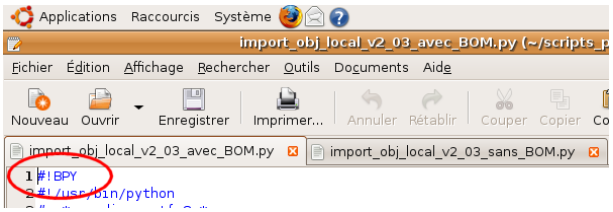

FIG. 14 – Texte utf-8 avec BOM ouvert sous Linux Ubuntu avec gedit

Figure 13, on remarque que la première ligne est en gras et qu’elle ne l’est pas sur la figure 14. C’est le seul moyen de détecter la différence avec gedit.

6.1 Changement de format

Avec Notepad++, il est facile de passer d’un format à l’autre (voir figure 15).

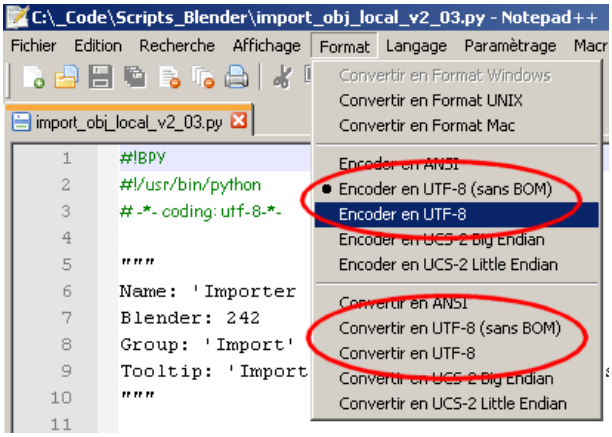

FIG. 15 – Changement de format avec Notepad++ sous Windows

6.2 En résumé

Pour que les scripts Blender codés en UTF-8 apparaissent dans les menus “scripts” de Blender aussi bien sous Windows que sous Linux, il ne faut pas que la “BOM” soit présente dans les fichiers “.py”.

6.3 Tous les détails sur la “BOM”

Pour plus d’informations, voir la FAQ du site de l’unicode (http://unicode.org).

7 Modification des chemins configurés dans l’environnement Python

En C/C++, on utilise la fonction `#include <...>` pour inclure d’autres fichiers sources dans un projet. En Python, il faut utiliser `import nom_fichier`.

Dans ce cas, il faut que le fichier `nom_fichier.py` soit présent dans la liste des chemins définis dans l’environnement Python.

Pour connaître les chemins configurés par défaut dans Python 2.5, il suffit de taper le code suivant :

```
import sys
print sys.path
```

```
['C:\\Python25\\Lib\\idlelib',
 'C:\\WINDOWS\\system32\\python25.zip',
 'C:\\Python25\\DLLs',
 'C:\\Python25\\lib',
 'C:\\Python25\\lib\\plat-win',
 'C:\\Python25\\lib\\lib-tk',
 'C:\\Python25',
 'C:\\Python25\\lib\\site-packages']
```

Chemins configurés par défaut dans Blender 2.46 :

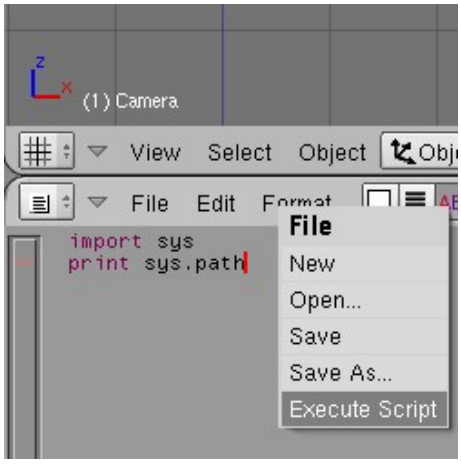

FIG. 16 – Récupération de la valeur de la variable sys.path avec Blender

On récupère la liste des chemins suivants dans la console :

```
"['C:\\Python25\\Lib',
  'C:\\Python25\\DLLs',
  'C:\\Python25\\Lib\\lib-tk',
  'C:\\Python25',
  'C:\\Python25\\lib\\site-packages',
  'C:\\_Stage_VV\\Code\\scripts_python_Blender',
  'C:\\PROGRA~1\\Blender',
  'C:\\Program Files\\Blender',
  'C:\\Program Files\\Blender\\python25.zip',
  'C:\\Program Files\\Blender\\.blender\\scripts',
  'C:\\Program Files\\Blender\\.blender\\scripts\\bpymodules']"
```

Mes programmes pour Blender sont donc exclusivement dans le dossier :

`C:\\_Stage_VV\\Code\\scripts_python_Blender\`

Pour appeler des programmes qui sont dans des sous-dossiers de “scripts_python_Blender”, j’ajoute le dossier voulu dans le chemin de sys.path et je le retire en fin de programme pour ne pas polluer les autres scripts qui utiliseraient des modules de même nom. Ce qui donne en Python :

Listing 4 – Exemple de modification et rétablissement du chemin

```
# Nom du dossier des scripts utilisateurs
strNomDossierScriptsUser = Blender.Get('uscriptsdir')
# Modification du sys.path
strNomDossierInclude = os.path.join( strNomDossierScriptsUser , \
                       "Import_de_SOFA_v0_01" )# <<< CONFIGURER ICI <<<<
if not strNomDossierInclude in sys.path:
  sys.path.append( strNomDossierInclude )

# Appel des fonctions de "Import_de_SOFA_v0_01"
import main_CImport_depuis_SOFA
reload( main_CImport_depuis_SOFA )
from main_CImport_depuis_SOFA import *

# Rétablissement du chemin original pour ne pas "polluer" d'autres fonctions
while strNomDossierInclude in sys.path:
  sys.path.remove( strNomDossierInclude )
```

8 Utilisation de la documentation de l'API Blender pour Python

A la ligne 2 du listing 4 ci-dessus, on note l'instruction :

`Blender.Get('uscriptsdir')`

Pour trouver la documentation sur l'interface Blender il faut lire la documentation Blender à cet endroit :

http://www.blender.org/documentation/246PythonDoc/

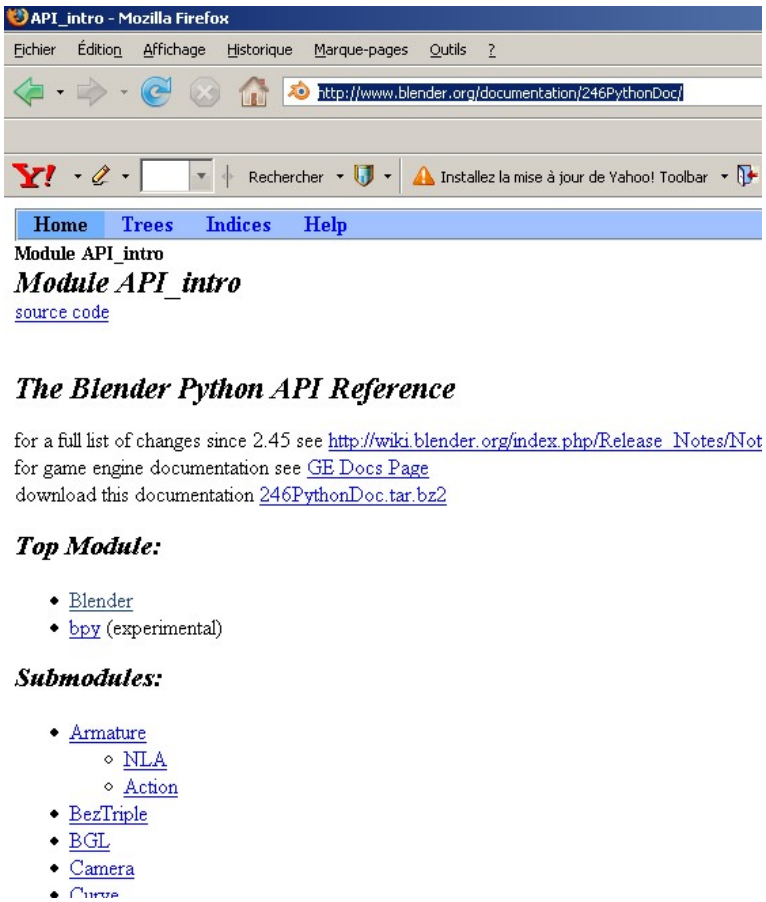

FIG. 17 – Page d'accueil de la documentation de l'API Blender

Ensuite, si on clique sur "Blender", puis "Get", on trouve cette page :

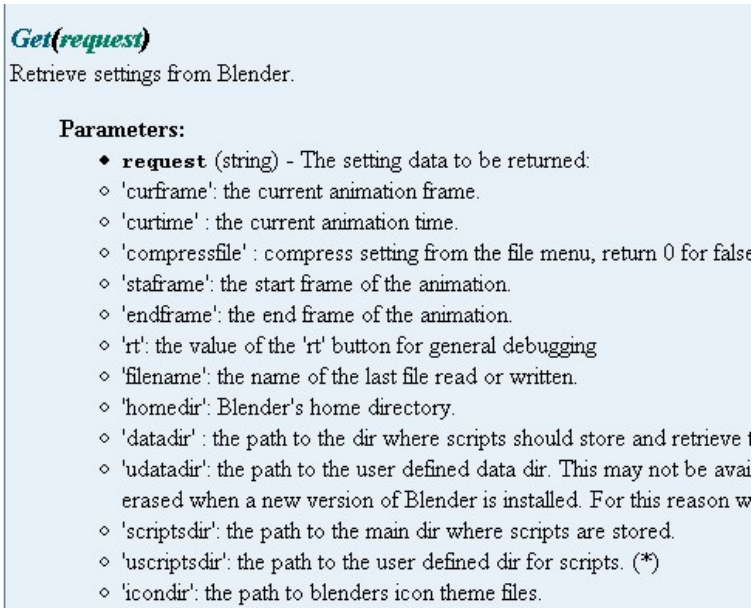

FIG. 18 – Détail de la fonction "Get" du module Blender

Avec les scripts livrés par défaut avec Blender, on trouve de nombreux exemples qu'il est facile de récupérer et adapter à ses besoins.

9 Scripts Python pour Blender et codage des caractères

Dans un script Python classique, on spécifie le jeu de caractère utilisé en ajoutant les lignes suivantes en début de script :

Listing 5 – Codage utf-8

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

Listing 6 – Codage Latin1 (ou iso8859-1)

```
#!/usr/bin/python
# -*- coding: latin-1 -*-
```

Par contre, les scripts Blender commencent par `#!BPY` à la première ligne :

Listing 7 – Listing blender

```
#!BPY
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""
Name: 'Distance entre objets'
Blender: 240
Group: 'Object'
Tip: 'Calcule la distance entre deux objets'
"""
```

Le commentaire `#!BPY"` indique à Blender que ce script s'utilise dans un menu. La spécification du menu est indiquée à la ligne “Group :”.

Si on souhaite utiliser un tel script à partir d'un autre script (par la commande “import ...”, le codage des caractères (utf-8) ne sera pas reconnu et si il contient des caractères accentués, un erreur se produira. Dans ce cas, une solution est de copier ce script et de supprimer uniquement la ligne `#!BPY`.