Morph-M - Anomalie #18

Inclusion problématique de arrayobject.h dans pythonExt/src/pymNumeric.cpp

12/01/2008 09:42 AM - Serge Koudoro

Status:	Nouveau	**Start date:**	02/02/2007
Priority:	Bas	**Due date:**	
Assignee:	Serge Koudoro	**% Done:**	80%
Category:	old plone Bugs	**Estimated time:**	0.00 hour
Target version:			

Description

Inclusion problématique de arrayobject.h dans pythonExt/src/pymNumeric.cpp

Description:

#include <numarray/arrayobject.h> ne passe pas avec certaines configurations.
J'utilise python2.3 .
// Python 2.3:
#if PY_MINOR_VERSION < 4
#include <numarray/arrayobject.h>

#5 15/06/2007 15:08 (Tibs)

Change: status: "pending" -> "accepted"
Comment:
Bon en fait j'avais oublié de l'accepter celui la.
Le problème est toujours semblable, sauf que NumPy est vraiment en train de
s'imposer.
Je pense que lorsque le passage à >boost-1.34 et >python-2.5 se fera ce sera
l'occasion d'imposer l'utilisation par défaut de NumPY dans le code et de fermer
ce bug.

#4 15/06/2007 15:08 (Tibs)

Change: assignees: "[]" -> "['raffi', 'Thomas', 'Tibs']"

#3 comment 09/02/2007 12:07 (Anonymous User)

Comment:
Ce problème ne vient malheureusement pas de Morphee.
Un petit résumé de la situation: deux librairies avec des fonctionalites très
proches ont été développées pour Python * Numeric qui a changé plusieurs fois de nom et est donc aussi connues sous
"NumPy" ou "scipy_core" * Numarray
Les librairies telles que Boost.Python ont rapidement offert une interface
censée fonctionner avec ces "deux" librairies, et permettant d'utiliser les
tableau (array, ndarray) définis dans ces modules Python lors de l'interfacage
d'une librairie C++ avec le langage Python.
C'est dans ce contexte que Morphée utilise ces librairies.
Les problèmes actuels découlent de deux faits: * ces librairies sont supportées différemment entre les divers OS et mm entre
les diverses distibutions Linux (certaines considérant que l'une ou l'autre des
librairies comme instable et ne proposant que l'autre par default etc etc) * ces disparité ont engendré le fait que les chemins vers
les includes varient
aussi bcp suivant les systèmes (certains sont dans le path d'autres non etc...)
Une note d'espoir:
On peut noter sur le site de NumPy (numpy.scipy.org) que la décision a été prise
paar les deux camps de fusionner les librairies (et si on a de la chance de ne
plus en changer le nom tous les 6mois...). Ceci est confirmé sur le site des
développeur de Numarray à Stanford.
L'avenir appartient donc à Numpy.
Les problèmes restants: * D'après mes experiences récentes le code d'interfacage de Boost ne fonctionne
plus avec NumPy et j'ai donc du improviser une code d'interfacage qui se trouve
dans Morphee mais n'est activé à la compilation qu'avec le flag USE_NUMPY
(étonnement ce code semble marcher et passe les tests MorpheePython) * le repertoire ou se trouvent les en-têtes à inclure ne

me semble pas etre dans
un emplacement standard (cf le path que j'ai mis et qui est specifique a ma
distribution Linux (Gentoo)), c'est très moche mais nous n'y sommes pour rien.
Ce qu'il faut faire: * passer à NumPy sur toutes les installations où c'est possbile * continuer a faire sa propre tambouille là où ce n'est pas possible (de toute
facon il FAUDRA passer à Numpy) * surveiller l'uniformisation EN COURS des librairies Numpy et numarray pour
pouvoir aussi unifier leur utilisation dans Morphée
Thibauld

#2 09/02/2007 11:32 (etienne)

Change: solution: "Voici la rustine que j'utilise:

//#include <numarray/arrayobject.h>
#include <Numeric/arrayobject.h>

Il faudrait trouver une solution plus générale.

" -> "Voici la rustine que j'utilise:

// Python 2.3:
#if PY_MINOR_VERSION < 4
// numarray replaced by Numeric (Etienne Decenciere)
//#include <numarray/arrayobject.h>
#include <Numeric/arrayobject.h>

Il faudrait trouver une solution plus générale.

"
Change: description: "#include <numarray/arrayobject.h> ne passe pas avec certaines configurations (lequelles?)." -> "#include <numarray/arrayobject.h> ne passe pas avec certaines configurations. J'utilise python2.3 .

// Python 2.3:
#if PY_MINOR_VERSION < 4
#include <numarray/arrayobject.h>
"

#1 09/02/2007 11:24 (etienne)

Change: topic: "" -> "UI"
Change: solution: "" -> "Voici la rustine que j'utilise:

//#include <numarray/arrayobject.h>
#include <Numeric/arrayobject.h>

Il faudrait trouver une solution plus générale.

"
Change: importance: "medium" -> "low"
Change: title: "" -> "Inclusion problématique de arrayobject.h dans pythonExt/src/pymNumeric.cpp"
Change: description: "" -> "#include <numarray/arrayobject.h> ne passe pas avec certaines configurations (lequelles?)."

History

#1 - 12/01/2008 09:46 AM - Serge Koudoro

- Category set to old plone Bugs

- Priority changed from Normal to Bas

- Start date set to 02/02/2007

#2 - 12/01/2008 09:49 AM - Serge Koudoro

- % Done changed from 0 to 80

#3 - 06/03/2009 08:49 PM - Jean Felder

- Assignee set to Serge Koudoro

Pour mes propres besoins, il me faut pouvoir convertir une image en un array numpy multidimensionnel.
D'après ce que j'ai compris le ImageToArray existant renvoie un tableau unidimensionnel.

J'ai implémenté (très rapidement) une nouvelle fonction imageToArrayFredo (private joke) disponible (à ce jour) dans
users/felder/estimages/pythonExt/src/imageToArray.cpp
Elle donne en sortie un array numpy de dimensions largeurxhauteur

Un exemple en python :

```
imIn = createImage(dataCategory.dtScalar, scalarDataType.sdtFloat)
imIn.setSize(3, 2)
imIn.allocateImage()
L = [1., 2., 3., 4., 5., 6.]
imIn.fromList(L)
foo = ImageToArrayFredo(imIn)
print foo
Out: [[ 1.  2.  3.]
      [ 4.  5.  6.]]
```

Une note au passage : attention à la convention numpy

```
print imIn.getPixel(GetOffsetFromCoords(imIn, (1, 0, 0)))
Out: 2.0

print foo[0, 1]
Out: 2.0
```

Tout ceci n'est par contre pas satisfaisant pour (au moins) deux raisons :

- ImageToArrayFredo ne gère (pour l'instant) que les images de float -> nécessité de rajouter un template

- La fonction appelle au niveau c++ le module pytohn de numpy via : *PyImport_Import(boost::python::object("numpy").ptr());*
Est-il possible de passer outre via le header de numpy : *numpy/arrayobject.h* ?
Il est à noter que la fonction ImageToArray existante dans pymNumeric.cpp fait aussi cet appel dans la fonction *object demand_resize_function(const char* module_name)*

Voici ce que l'on pourrait faire :

1. numarray est obsolète, il faut le retirer des sources
2. coder une fonction ImageToArray gérant les différents types d'images et renvoyant un array multidimensionnel en utilisant le header de numpy
3. coder une fonction miroir ArrayToImage
4. tests unitaires