TP Imagerie 3D nº 1 (3 heures)

Lecture, stockage d'images 3D et visualisation volumique

- Le TP est **individuel**.
- Les programmes doivent être écrits en **C/C++** en n'utilisant que des **bibliothèques classiques** (stdio.h, stdlib.h, math.h).
- **Le TP est noté : le compte-rendu doit être envoyé** sous **forme électronique à : gerard.subsol@lirmm.fr le jour même avant 13h00.**
- La participation active pendant le TP pourra aussi être prise en compte.
- Le compte-rendu doit inclure **votre nom**, quelques lignes d'explication sur l'algorithme, des captures d'écran (ici, la visualisation en Volume Rendering des différentes images 3D, en particulier *whatisit*) et le code source intégral (avec quelques commentaires). Le tout doit être sous la forme d'un **unique fichier pdf**.
- **Tout plagiat sera lourdement sanctionné**.

Découvrir les images 3D *orange*, *INCISIX* et *t1-head* (format Analyze) à l'aide de Fiji (voir annexe).

1**. Ecrire en C/C++ un programme de lecture et stockage d'images 3D** :

- Lecture de l'image img en format brut (sans en-tête) codée en unsigned short (2 octets).
  En entrée, on pourra utiliser les paramètres suivants : <imageFile> <dimX> <dimY> <dimZ>
  avec <dimX> <dimY> <dimZ>= dimensions de l'image en voxels dans le nom du fichier.
- Stockage en mémoire de l'image
- getValue(i,j,k) qui renvoie la valeur du voxel (i,j,k).
  Attention par convention (et en accord avec Fiji) : (i,j,k) ∈ [0,dimX-1,dimY-1,dimZ-1]
- Affichage de la valeur minimale et maximale des voxels de l'image
- Affichage de l'intensité d'un voxel de coordonnées rentrées par l'utilisateur.

- Tester sur les images 3D (*voir annexe pour vérifier les valeurs suivantes sous Fiji) :*
  - *orange* : min=0 max=228 I(128,128,32)=9
  - *INCISIX* : min=0 max=4095 I(184,343,83)=1225
  - *T1-head* : min=0 max=885 I(158,143,64)=242

Si vous ne trouvez pas les bonnes valeurs, réfléchissez au balayage (voir annexe) ou au stockage d'une variable sur 2 octets.

2. Ecrire en C/C++ un programme de Volume Rendering (MIP, AIP, MinIP) suivant les directions axiales x, y et z.

Le résultat sera une image qui pourra être sauvegardée dans un format simple déjà vu en cours ou au format brut (sans en-tête et codée par exemple en unsigned short (2 octets)). Elle pourra ensuite être lue par Fiji (voir annexe).

En entrée, on pourra utiliser les paramètres suivants : <imageFile> <dimX> <dimY> <dimZ> <resultFile> <visuAxis : 1=x, 2=y, 3=z> <visuMode : 1=mip, 2=aip, 3=minip>

4. Tester sur les images 3D : *orange / INCISIX* / *T1-head* et visualisez quelques résultats.

5. **Défi** : qu'est-ce que *whatisit* ?

A. Données disponibles à : *http://www.lirmm.fr/~subsol/HAI804I/*

B. Fiji : *http://fiji.sc/*

- Lire une image 3D au format Analyze : *File/Import/Analyze* puis le fichier.hdr
- Lire une image 2D brute : *File/Import/Raw* puis rentrer les paramètres (width, height, image type…).
- ctrl+H en incluant toutes les images vous donne le min/max de l'image 3D.

C. Stockage de l'image

On supposera que l'image est lue coupe par coupe et qu'elle est balayée par ligne suivant le schéma suivant qui correspond à l'affichage sous Fiji :

```
y\x --------------> (dimX=4 voxels by line)
 | 0  1  2  3
 | 4  5  6  7
 v 8  9 10  11

(dim Y=3 voxels by column)
```

Pour voir l'intensité d'un voxel dans Fiji, se placer dessus (éventuellement en zoomant avec + et -) et regarder la valeur dans le bandeau (se fier aux valeurs entières des coordonnées entre parenthèses qui sont entre 0 et dimX/dimY/dimZ -1).

D. Rappels de fonction C de lecture/écriture de données binaires

FILE *fopen(const char *_filename_, const char *_mode_);
Opens the filename pointed to by filename. The mode argument may be one of the following constant strings:

rb read binary mode
wb write binary mode (truncates file to zero length or creates new file)

size_t fread(void *_ptr_, size_t _size_, size_t _nmemb_, FILE *_stream_);
Reads data from the given stream into the array pointed to by *ptr*. It reads *nmemb* number of elements of size *size*. The total number of bytes read is (**size*nmemb**).
On success, the number of elements read is returned. On error or end-of-file the total number of elements successfully read (which may be zero) is returned.

size_t fwrite(const void *_ptr_, size_t _size_, size_t _nmemb_, FILE *_stream_);
Writes data from the array pointed to by *ptr* to the given stream. It writes *nmemb* number of elements of size *size*. The total number of bytes written is (**size*nmemb**).
On success, the number of elements written is returned. On error the total number of elements successfully written (which may be zero) is returned.