Introduction

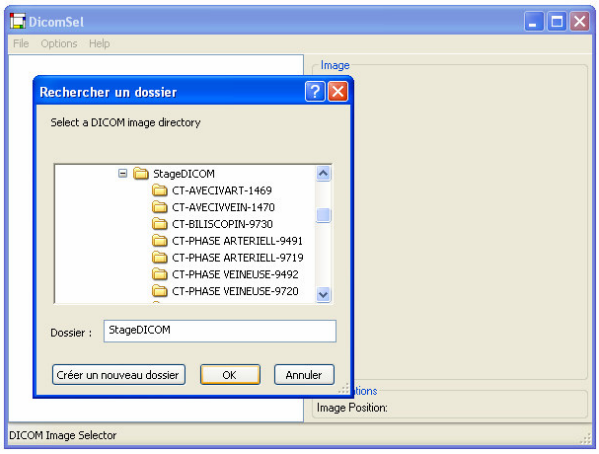
Nous avons développé une application de traitement de et visualisation d'images Dicom répondant au sujet suivant :

« Développement d'un outil de traitement d'informations des images DICOM issues d'acquisitions scanner, dans le cadre d'une application de simulation d'opérations chirurgicales de radiofréquence sur tumeurs hépatiques ».

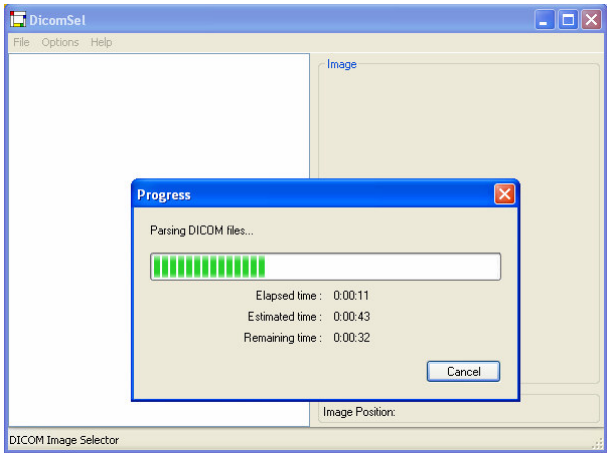
Le travail s'est orienté autours de la librairie Dicom ainsi que de l'interface WxWidgets.

Sommaire

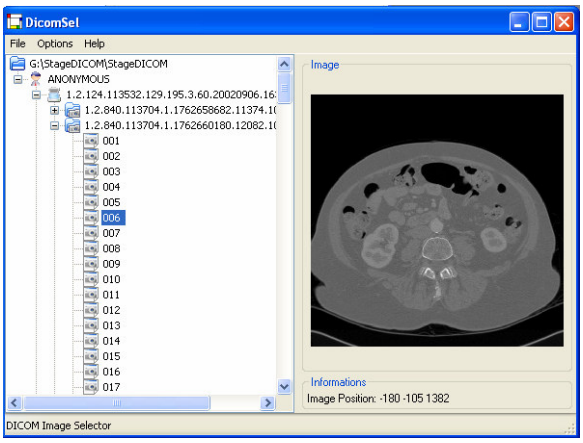
Prise en main rapide :



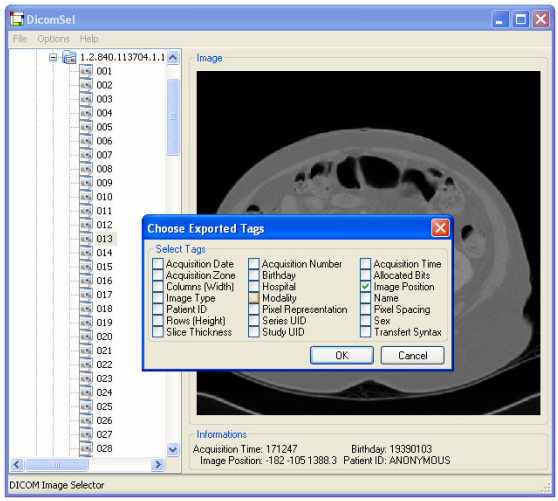
Il faut tout d'abord **importer** un dossier contenant des images Dicom.
L'importation peut prendre du temps, surtout si le temps d'accès aux données est long.



Une fois l'importation terminée, on peut trouver les images dans l'arbre, à gauche de l'interface.
Il s'agit donc a présent de trouver la bonne image parmi les séries d'images.



Il est possible de choisir quel tag DICOM on doit afficher a l'écran, et quel tag on doit pouvoir exporter. Tout cela est dans le menu option.



Enfin, on peut exporter les tags dans un fichier texte (fichier ->export tags…) en précisant la destination du fichier.

Structure du programme :

La libdicom fournie par Johan Moreau de l'IRCAD possède une classe abstraite.
La plus grande partie du travail a été de créer des classes qui en héritent.
Pour faire hériter nos classes, nous avons dû surcharger toutes les méthodes définies dans la classe abstraite.

On va utiliser la librairie pour extraire des données. Chaque fois qu'elle va "rencontrer" un fichier, elle va appeler une fonction, et chaque fois qu'elle va "rencontrer" un tag dans le fichier elle va appeler la fonction correspondante au tag rencontré.
C'est ce qu'on appelle le design pattern "visiteur".

Un visiteur permet de passer récursivement une structure, pour en extraire certaines données.
Un visiteur est ce que l'on appelle un Design pattern, c'est un concept issu de la programmation orientée objet, destiné à résoudre les problèmes récurrents.

Ce sont donc des solutions standard pour répondre à des problèmes d'architecture et de design des logiciels. On peut considérer les designs patterns comme un outil de capitalisation de l'expérience appliquée à la conception logicielle.

La librairie fournit des visiteurs qui permettent de parcourir récursivement des structures composites. Que se soit un répertoire d'images DICOM ou une image Dicom en elle même.

Le programme se déroule en deux étapes:

La class **dicomcollection** parcourt un répertoire d'images DICOM, pour construire un arbre de données (classes qui héritent de dicom::visitor::CDicom). Le parcours du dossier ne se fait pas en lisant la totalité des données contenues dans chaque image.

Le visiteur reconnaît et extrait seulement quelques informations clés pour chaque image, soit 3 tags (tag = élément permettant le stockage d'information sous le format biomédical DICOM) : le nom du patient, les id d'étude et de série. Cela permet de construire un arbre de données représentants les images. Les données images ne sont pas lues à ce moment, ce qui

permet de gérer des séries d'images importantes. Ainsi il est possible de représenter un CD-ROM d'image Dicom sans pour autant dépasser les 10 Mo de mémoire ram utilisée, une fois que l'arbre est construit, on peut partir à la recherche de l'image désirée.

Pour cela on va faire appel a une seconde classe : dicomfile.
Dicomfile permet de récupérer les informations contenues dans un fichier.

A partir de l'arbre précédemment construit, on récupère l'identifiant de l'image qui nous intéresse de visualiser. Dicomfile se charge de recueillir les données telle que la position de la coupe ou encore l'image en elle même. Cette image est affichée grâce à la classe BitmapPanel qui gère l'affichage.

L'utilisation de ces deux classes complémentaire est judicieuse car elle permet d'avoir un accès aux données optimales, à la fois on peut gérer de grandes quantités de données, et le tout en un temps très court.

Le protocole DICOM.

DICOM (Digital Imaging and Communications in Medecine) est la norme internationale pour l'imagerie médicale dans son ensemble (radiologie, endoscopie, microscopie…). La norme décrit ce qui est nécessaire pour gérer et communiquer des images dans le cadre d'une gestion d'un dossier médical. Cette norme est compliquée car elle se veut universelle, adaptable et orientée objet. DICOM permet un mode d'identification très précis de chacune des images émises par les appareils d'imagerie numérique car à chaque image sont associées des **informations techniques**, **démographiques** et **médico-légales** ainsi qu'une **clé** d'identification unique. Cela permet d'envisager le diagnostic et le travail à partir de documents numériques, ce qui à terme permettra d'abandonner le film, grâce à sa capacité à gérer un grand nombre d'images sans risque de mélange ou de perte d'informations.

DICOM concerne des **"objets"**, les images par exemple, dont il existe de multiples variétés en fonction :

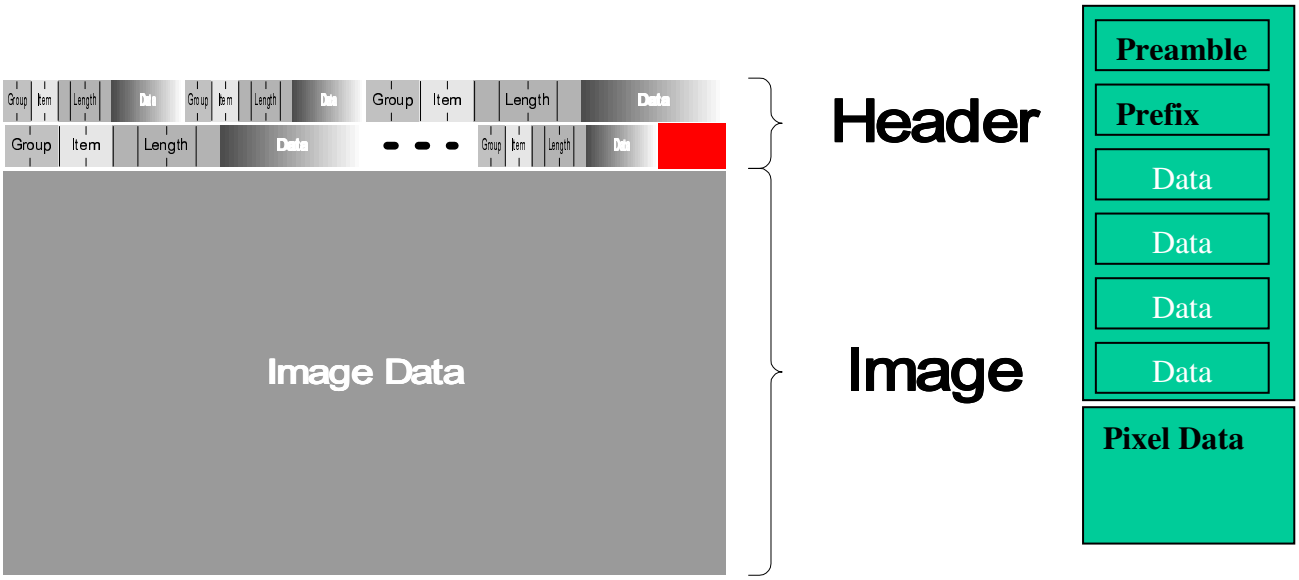
1. de la modalité : images CT (scanner), MR (IRM), US (ultra-sons direct), SC (capture secondaire du signal vidéo souvent utilisé en ultra-sons), ...
2. du mode de représentation des informations des différents champs (y compris des pixels). Il existe 3 modes possibles : little endian explicite, little endian implicite, big endian explicite.

Dicom décrit aussi l'utilisation ou non d'une compression JPEG des pixels de l'image, en sachant qu'il y a plus de 20 types de compression possible : Fractale ou wavelet par exemple, il suffit juste que les constructeurs gèrent le type de compression (ce qui n'est toujours le cas). La compression sans perte est presque toujours utilisée dans le monde du médical.

Une image DICOM comporte de nombreux **champs** définis dans le "DICOM standard", dont certains sont obligatoires. Un constructeur peut aussi définir ses propres champs pour y mettre (en théorie) des informations qui ne peuvent pas être placées dans les champs définis par le comité DICOM.

Les **services** (en particulier les protocoles de communication) s'appliquent aux objets pour réaliser des "Service-Object pair (SOP) classes".

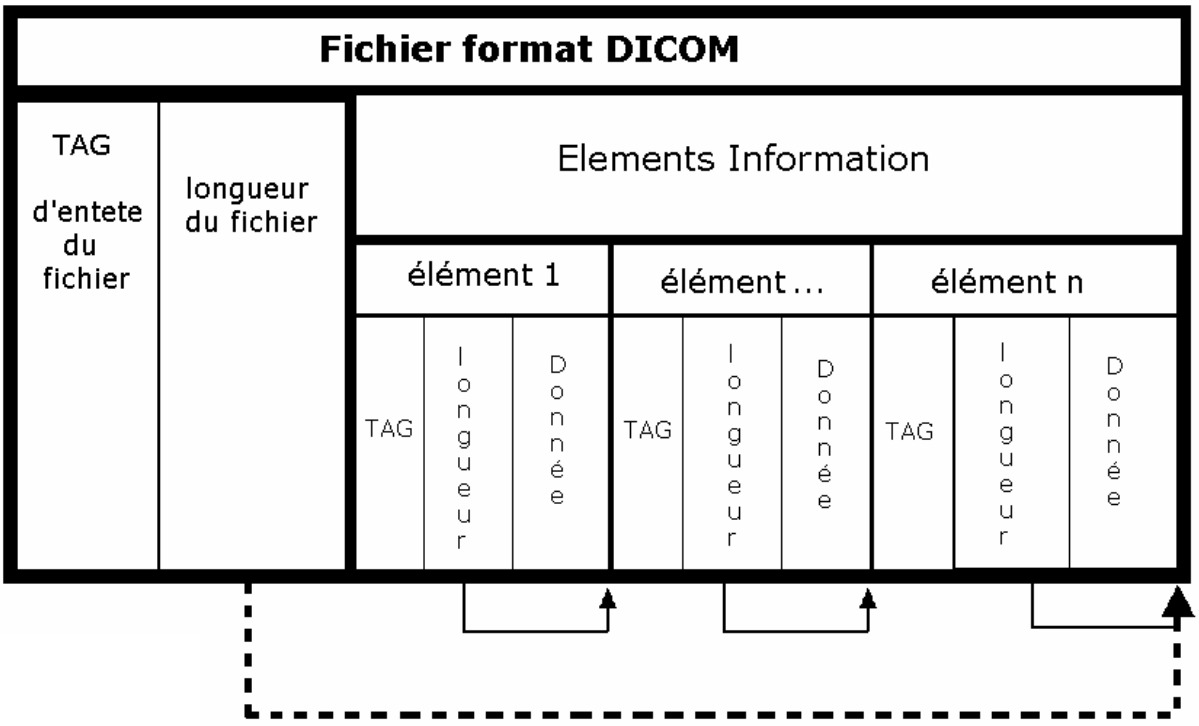
Fichier DICOM



Il faut savoir que dans un fichier DICOM, on peut stocker plusieurs informations et plusieurs images ;elles sont alors mises bout à bout avec d'abord l'en-tête du fichier, puis les informations et les images avec leur en-tête respectives.

En-tête DICM	Elément donnée constructeur	Elément donnée materiel	• • •	Elément donnée nom du patient	• • •	Elément donnée image	• • •	Elément donnée image

TAG **Code de l'information (4octets)**	VR **Format de la donnée (2octets)**	VL **Longueur de la donnée (~2octets)**	**Data Donnée (taille variable selon la donnée)**



Les TAGS

Un fichier DICOM commence toujours avec 4 octets 'D' 'I' 'C' 'M' (44 49 43 4D en hexa) suivi par la longueur du fichier.
Puis les informations sont regroupées en éléments, qui sont mis bout à bout ; en derniers éléments on a souvent la ou les images.

La nature de l'information stockée dans un élément est identifiée sur 4 octets .on appelle cette balise « un TAG. »
Selon les deux premiers octets du TAG , on sait s'il s'agit de :

Code hexa	Explications
0000	Commande
0008	Identification du centre
0010	Information sur le patient
0018	Type d'acquisition (épaisseur de coupe, inclinaison du statif, temps, etc..)
0020	Positionnement et information relatives à l'acquisition (Relationship)
0028	Présentation de l'image (dimensions, niveaux de gris, tables de couleurs)
4000	Texte
6000-601E	(even) Overlay, type de compression....
7FE0	Pixel Data, Pixels de l'image.

Exemple d'info éditée en hexa d'un fichier DICOM :

08 00 70 00 4C 4F 18 00 50 68 69 6C 69 70 73 20 4D 65 64 69 63 61 6C 20 53 79 73 74 65 6D 73 20

08 00 70 00	**4C 4F**	**18 00**
TAG : Manufacturer	Type LO :Long String	Longueur = 24 octets

50 68 69 6C 69 70 73 20 4D 65 64 69 63 61 6C 20 53 79 73 74 65 6D 73 20
Information : Philips Medical Systems.

Pour identifier un matériel biomédical, la norme DICOM a un dictionnaire. Avec une référence on peut retrouver quel type de matériel a été utilisé, mais il faut savoir qu'en plus de spécifier le matériel, ce code comprend des instructions, comme « Storage » par exemple, ou encore « FIND », « MOVE » ou « GET » qui correspondent à une instruction matérielle sur le fichier.

Exemple :

Computed Radiography Image Storage 1.2.840.10008.5.1.4.1.1.1

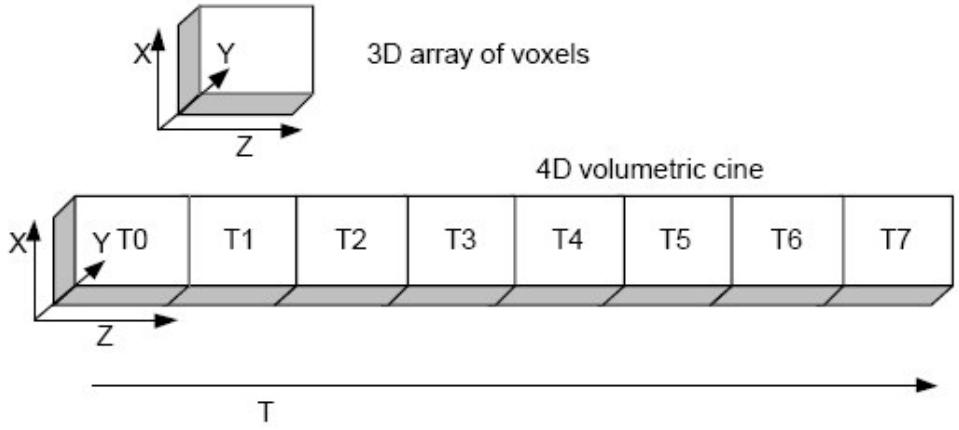
VR :

Les VR sont des balises définissant la nature de information (codé sur deux octets) ainsi que la magnére dont elle est stoquée : S' il n'y a pas de VR, alors est utilisée la forme par défaut (implicite).

AE :Application Entity	**OW** :Other Word String
AS :Age String	**PN** :Person Name
AT :Attribute Tag	**SH** :Short String
CS :Code String	**SL** :Signed Long
DA :Date	**SQ** :Sequence of Items
DS :Decimal String	**SS** :Signed Short
DT :Date Time	**ST** :Short Text
FL :Floating Point Single	**TM** :Time
FD :Floating Point Double	**UI** :Unique Identifier(UID)
IS :Integer String	**UL** :Unsigned Long
LO :Long String	**UN** :Unknown
LT :Long Text	**US** :Unsigned Short
OB :Other Byte String	**UT** :Unlimited Text
OF :Other Float String	

Représentation de voxel dans Dicom :

Le groupe de travail 17 du comité du standard DICOM a proposé fin 2004 un supplément à la norme DICOM, qui propose une représentation supplémentaire de l'acquisition multidimensionnelle qui peut s'appliquer à toute une gamme de modalités et d'analyses.
Il est à noter que ces modifications seront certainement prises en compte pour la fin de l'année 2005. Ainsi la représentation par voxel est envisagée. Il sera possible de stoker une scène d'objet volumétrique avec le format DICOM.

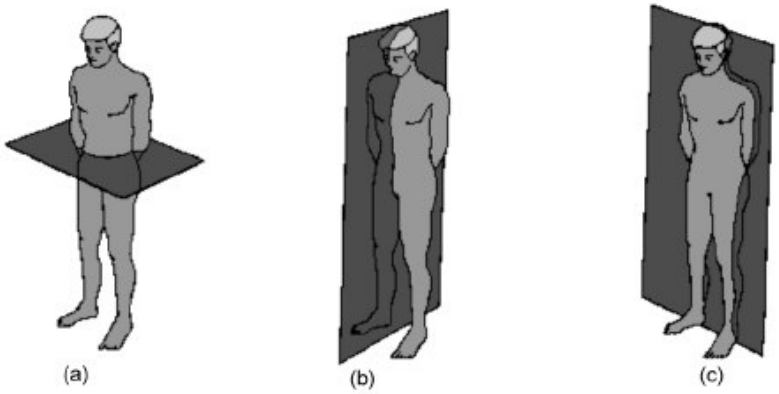


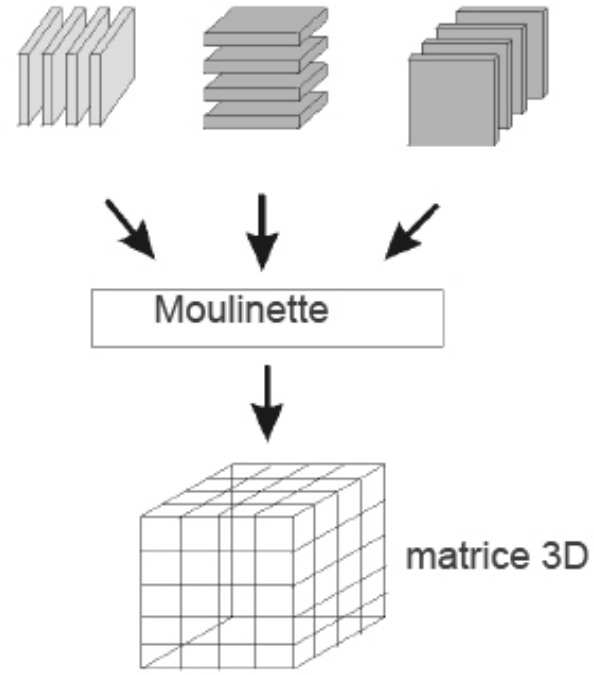
Il est à présent possible de décrire la structure des éléments stockés.
Il suffit, pour adresser n'importe quel valeur de données, de remplir ces tags :

- Le nombre de dimensions (gggg, 6320)
- La valeur maximum pour chaque dimension (gggg, 6321)
- La taille mémoire allouée (0028.0100).

Les données sont stockées sous forme de matrice.

Construction de bloc DICOM 3D a partir de série d'image 2D :
Il n'existe pas encore de convertisseur gratuit (bien souvent très cher).





Le principe est d'entrecouper plusieurs vues différentes pour construire un bloc 3D. Il faut disposer de trois séries d'images sous trois angles différents pour cela.

Quelques TAGs DICOM

(nnnn,0000) BD S Group Length # longueur du groupe en octets nnnn/ l
(nnnn,4000) AT M Comments #commentaires

(0008,0010) AT S Recognition Code # ACR-NEMA 1.0 or 2.0
(0008,0020) AT S Study Date # yyyy.mm.dd
(0008,0021) AT S Series Date # yyyy.mm.dd
(0008,0022) AT S Acquisition Date # yyyy.mm.dd
(0008,0023) AT S Image Date # yyyy.mm.dd
(0008,0030) AT S Study Time # hh.mm.ss.frac
(0008,0031) AT S Series Time # hh.mm.ss.frac
(0008,0032) AT S Acquisition Time # hh.mm.ss.frac
(0008,0033) AT S Image Time # hh.mm.ss.frac
(0008,0060) AT S Modality # CT,NM,MR,DS,DR,US,OT

(0010,0010) AT S Patient Name
(0010,0020) AT S Patient ID
(0010,0030) AT S Patient Birthdate # yyyy.mm.dd
(0010,0040) AT S Patient Sex # M, F, O pour les autres
(0010,1010) AT S Patient Age # xxxD or W or M or Y

(0018,0010) AT M Contrast/Bolus Agent # ou RIEN
(0018,0030) AT M Radionuclide
(0018,0050) AN S Slice Thickness # mm
(0018,0060) AN M KVP

(0018,0080) AN S Repetition Time # ms
(0018,0081) AN S Echo Time # ms
(0018,0082) AN S Inversion Time # ms
(0018,1120) AN S Gantry Tilt # angle d'inclinaison.

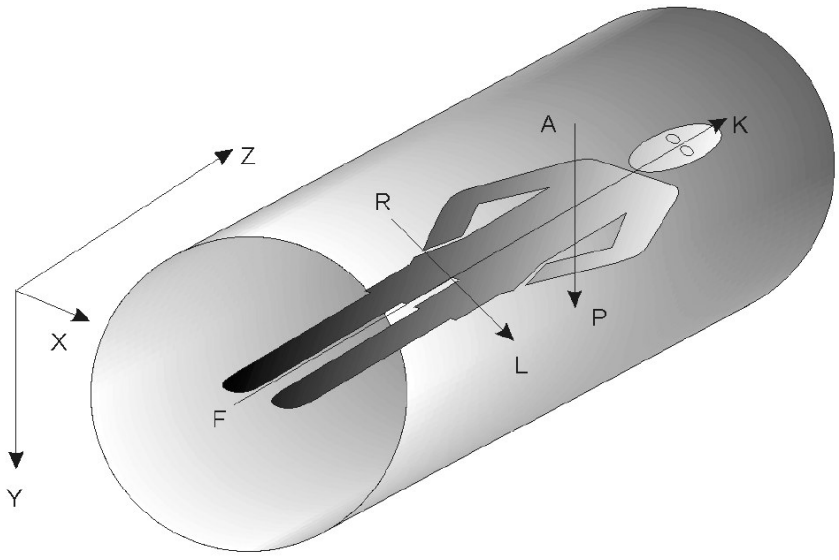
(0020,1040) AT S Position Reference # exemple crête iliaque
(0020, 0032) Position de l'image (relative au patient)
(0020, 0037) Orientation de l'image

(0020,1040) AN S Slice Location # en mm (signé)

(0028,0010) BI S Rows
(0028,0011) BI S Columns
(0028,0030) AN M Pixel Size # rangées\colonnes en mm
(0028,0100) BI S Bits Allocated # ex. 12 bits pour la TDM
(0028,0101) BI S Bits Stored # ex. 16 bits
(0028,0102) BI S High Bit # ex. 11
(0028,0102) BI S Pixel Representation # 1 signé, 0 non signé

(7FE0,0010) BI M Pixel Data # données image, comme décrites au groupe 0028

Les axes Dicom



Glossaire DICOM :

AE : Entité applicative ou programme
CR Computed Radiography Radiolographie numérisée
DICOM : Digital Imaging and Communications in Medicine.
DIMSE : DICOM Message Service Element (partie de dicom qui gére la communication).
HL7 : « Health Level 7, » Protocole de communication de santé.
LITTLE ENDIAN : L'octet de poids faible est stocké avant l'octet de poids fort.
LUT Look-up table Table de valeurs
PDV : « Presentation Data Value »
PMS Print Management Service Service de gestion de reprographe
RIS/HIS Radiology or Hospital Information System
Santec : organisme de normalisation médicale
SCNS Study Content Notification Service Service de notification du contenu d'une étude
SCP : « Service Class Provider », rôle de serveur. Fournisseur d'une classe de service
SCU : » Service Class User », rôle de client. Utilisateur d'une classe de service
SDG : "Santec Dicom Gate"
SOP « Service-Object Pair » Classe de parité service-objet (union de la donnée et de ce que l'on doit en faire)
SSC « Storage Service Class » Classe de service de sauvegarde
UID : identifiant unique
VL : longueur des données
VM : indique le nombre de valeurs contenues dans la donnée.
VR : « VALUE REPRESENTATION » Indique le type de données et le format de l'information. (détermine si c'est une chaîne de caractères, binaire, texte , alphanumérique un nombre …etc)

Sources :

http://www.dclunie.com/dicom-status/status.html

Cite web regroupant les « DICOM Standard Status ». Il s'agit de la norme DICOM en anglais.

http://www.xray.hmc.psu.edu/physresources/dicom/dicom_intro/index.html

Cite web riche sur le standard DICOM. En anglais.

http://medical.nema.org/

Le site dicom

L'interface graphique

Nous expliquerons dans cette partie les grandes lignes de l'implémentation de notre interface graphique en WxWidgets. Cette partie est également une introduction à WxWidgets. Pour plus de détails, ce référer au code.

Origine de WxWidgets

C'est en 1992 que WxWidgets fut crée par Julian Smart un chercheur en Intelligence Artificielle qui avait besoin d'une bibliothèque multi plateforme pour qu'il puisse programmer sous Windows et Unix. Julian Smart jugeait les outils existant trop chers, il décida donc de créer WxWidgets. Cet outil connu un franc succès, et il y a eu de nombreuses contributions pour améliorer cet outil.

Pourquoi WxWidgets ?

WxWidgets (anciennement appelé wxWindows) apporte une API simple et facile d'utilisation pour écrire des applications avec interface graphique (GUI) sur de multiples plates-formes. Liée avec les librairies appropriées à votre plate-forme (Windows/Unix/Mac) et compilée (la plupart des compilateurs C++), les application adopteront l'apparence et le fonctionnement adaptés à cette plate-forme. En plus, sur la grande majorité des fonctionnalités GUI, WxWidget fournit une aide en ligne, la programmation réseau, les flux (streams), l'ancrage et le glisser-déposer, le multitâche, le chargement et la sauvegarde d'images dans une large gamme de formats, le support des bases de données, les cadres HTML avec les impressions, et bien plus encore.

Il est également très facile d'intégrer WxWidgets dans des environnement de développement tel le logiciel Dev-cpp.

WxWidgets permet donc de gérer à peu près tout ce qui peut arriver dans une fenêtre. Quelque soit le système ou le compilateur utilisé.

L'intérêt majeur de WxWidgets vient de sa richesse, son vécu (plus de 10 ans) et donc de la quantité impressionnante de documentation, de tutoriaux et d'outils annexes existants.

La compatibilité de WxWidgets repose sur le fait que l'API est «au-dessus » des couches systèmes.

Architecture WxWidgets

wxWindows application framework								
Un même code source pour toutes les plates-formes								
WxWidgets API								
wxMSW	wxX11	wxGTK	wxMotif	wxMac	wxBase			
GDI	**Xlib/X11**	**GTK+**	**Motif**	**Mac**	*without GUI*			
Windows	**Unix (1)**			**MacOS**	**Windows**	**Unix (1)**	**MacOS**	**OS/2**

(1) Windows, Linux, Solaris, AIX

- Un programme GUI WxWidgets consiste en:
  - Un objet application - une instance de classe wxApp.
  - Un objet cadre (une instance de classe wxFrame). Un cadre peut avoir des objets comme une barre de menu, une barre d'état, une icône, etc.
  - Le cadre peut contenir beaucoup d'autres objets comme un contrôle de texte, des boutons, des fenêtres scindées ou comme dans notre interface des « splitters » .

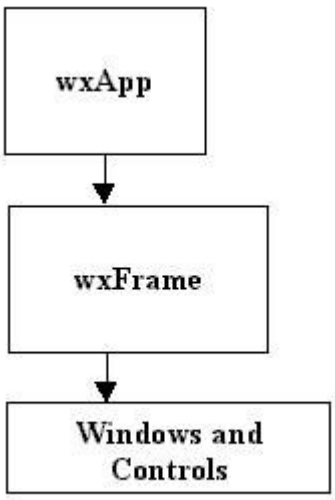


Figure 1

Les Sizer

Les sizers permettent l'ancrage par rapport aux quatre cotés de la fenêtre.
Cela va nous permettre de créer plusieurs zone, contenant chacun un objet différent (un arbre de fichier, un zone texte, etc)

La librairie wxWidgets possède plusieurs types de sizers, nous utilisons un sizer simple, le **wxStaticBoxSizer**.
wxBoxSizer et wxStaticBoxSizer sont tous deux sont dérivés de la classe abstraite wxSizer.
Nous utiliserons des staticbox sizer qui peuvent contenir des textes statiques,alors que le boxsizer est juste une boîte.

Un sizer se crée comme un objet fenêtré à l'aide de l'opérateur new et sa destruction est du ressort de l'application.

Dans l'exemple nous créons d'abord un objet wxStaticBox infoBox. Son constructeur reçoit comme argument un entier indiquant comment seront alignés les éléments qu'il contient, deux possibilités **wxVERTICAL** pour les aligner les uns sous les autres ou **wxHORIZONTAL** pour les aligner les uns à coté des autres.

exemple:

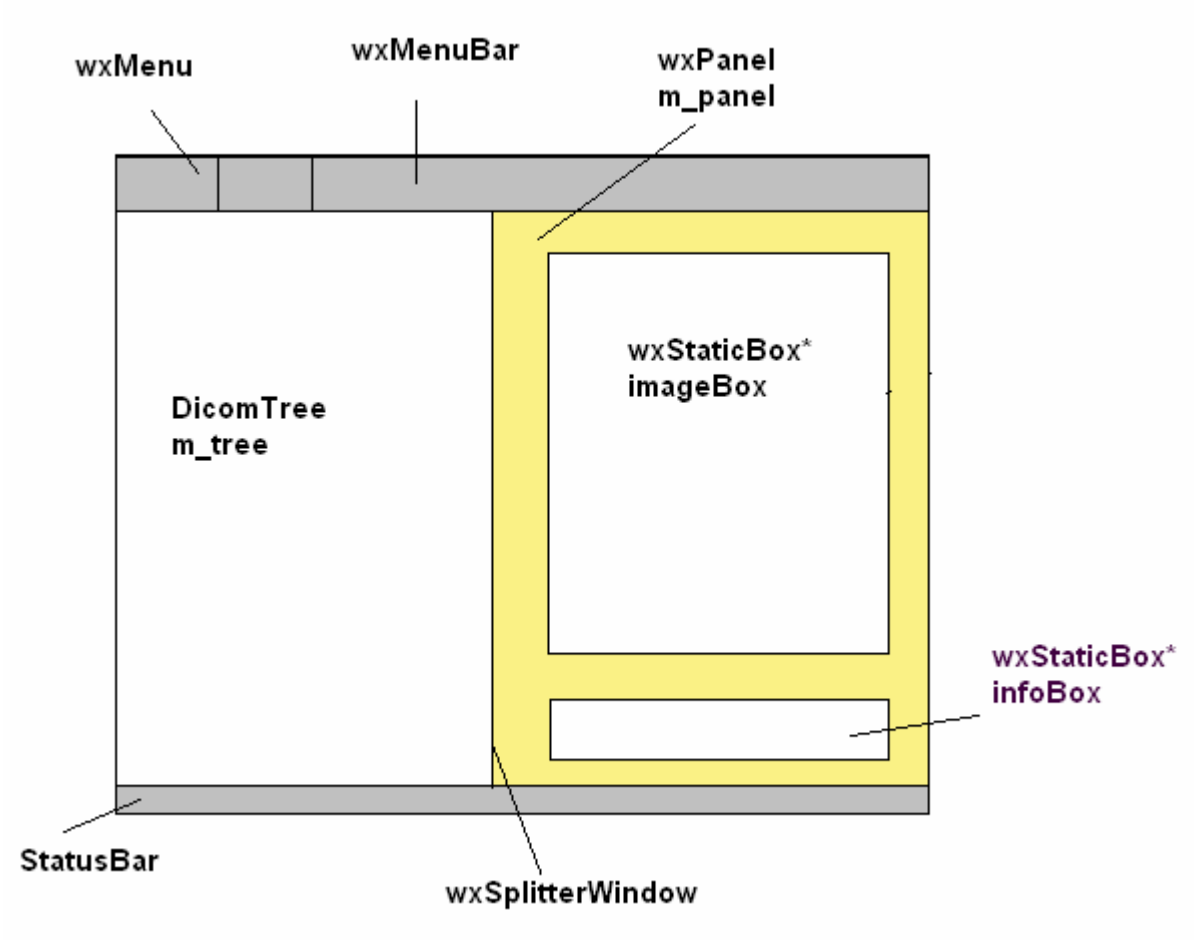
```
// Static box containing informations (tags)
wxStaticBox* infoBox =
        new wxStaticBox( m_panel, WID_INFOBOX, wxT( "Informations" ) );
m_infoSizer = new wxStaticBoxSizer( infoBox, wxVERTICAL );
```

Nous pouvons utiliser autant de sizers que nous voulons pour achever une présentation . Nous pouvons aussi y ajouter d'autres fonctionnalités de WxWidgets pour faire pour une présentation uniforme à l'écran en utilisant des objets à contrainte de disposition et des scindements de fenêtre (splitter).

Exemple avec splitter :

```
wxSplitterWindow *splitter =
        new wxSplitterWindow( this, WID_SPLITTER, wxDefaultPosition,
                              wxDefaultSize, wxSP_LIVE_UPDATE );
splitter->SetMinimumPaneSize( 150 );
// Tree on the left, panel on the right
m_tree = new DicomTree( splitter, WID_DICOMTREE );
m_panel = new wxPanel( splitter, WID_PANEL );
splitter->SplitVertically( m_tree, m_panel, 300 );
```

Schéma récapitulatif des sizers et splitters de notre interface :



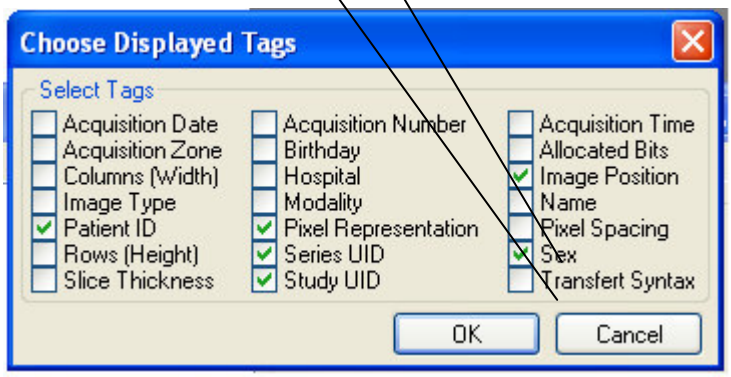
Autre exemple, pour le choix des tags, avec une grille :
(dans chacune des cases de la grille, on va placer une checkBox)

exemple:

```
const int count = TagSet::GetTagCount();
wxFlexGridSizer* const grid = new wxFlexGridSizer( (count + 9) / 10,
                                                   0, 12 );
boxSizer->Add( grid );

for( int i = 0; i < count; i++ )
{
      wxCheckBox* const checkBox =new wxCheckBox( this, -1,
                        TagSet::GetTagName(static_cast< TagSet::TagID >( i ) ) );
      m_checkBoxes[i] = checkBox;
      grid->Add( checkBox, 0, wxEXPAND );
}
```

wxCheckbox	wxCheckbox	wxCheckbox
wxCheckbox	wxCheckbox	wxCheckbox
wxCheckbox	wxCheckbox	wxCheckbox
wxCheckbox	wxCheckbox	wxCheckbox
wxCheckbox	wxCheckbox	wxCheckbox
wxCheckbox	wxCheckbox	wxCheckbox
wxCheckbox	wxCheckbox	wxCheckbox



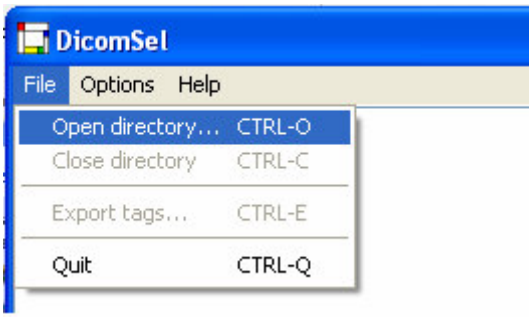
Les Menus

Venons-en à la construction du menu. WxWidgets permet la gestion automatique des menu : pour créer un menu File, il suffit de déclarer un pointeur sur un menu fileMenu et un pointeur sur une barre de menu menuBar.

Exemple:
```
wxMenu* fileMenu = new wxMenu;
fileMenu->Append( MENU_OPENDIR, wxT( "&Open directory...\tCTRL-O" ),
                  wxT( "Scan a directory for DICOM images" ) );
wxMenuBar* menuBar = new wxMenuBar;
```

Il faut ensuite ajouter le fileMenu à la menuBar
```
menuBar->Append( fileMenu, wxT( "&File" ) );
```

La gestion du raccourci pour les menuest faite automatiquement par WxWidgets, il suffit de désigner la touche pour le raccourci *(CTRL-O ci-dessus).*



menuBar l'interface

Gestion des Evénements

Un évènement est un fait d'une certaine importance qui est survenu en dehors du programme. Il peut avoir été provoqué par un utilisateur en sélectionnant un élément (item) du menu ou par un raccourci clavier. Il peut avoir été provoqué par quelque chose dans l'environnement de l'ordinateur mais en dehors du programme, un « timer » (ou chrono) qui s'arrête, des données arrivant sur un port de communication, un message d'arrêt du système etc...

Comme vous pouvez le voir ci-dessus, la table des évènements wxWindows est très simple. C'est une collection de macros, la première annonce le début de la table, les suivantes une constante de plusieurs types avec une méthode de classe, et la dernière annonce la fin de la table. La table des évènements relie les évènements identifiés par des constantes comme MENU_OPENDIR, aux fonctions comme MainFrame::OnMenuOpenDir.

Exemple, notre table d'événement :

```
BEGIN_EVENT_TABLE( MainFrame, wxFrame )
   EVT_CLOSE( MainFrame::OnClose )
   EVT_MENU( MENU_OPENDIR,        MainFrame::OnMenuOpenDir       )
   EVT_MENU( MENU_CLOSEDIR,       MainFrame::OnMenuCloseDir      )
   EVT_MENU( MENU_EXPORT,         MainFrame::OnMenuExport        )
   EVT_MENU( MENU_QUIT,           MainFrame::OnMenuQuit          )
   EVT_MENU( MENU_DISPLAYED_TAGS, MainFrame::OnMenuDisplayedTags )
   EVT_MENU( MENU_EXPORTED_TAGS,  MainFrame::OnMenuExportedTags  )
   EVT_MENU( MENU_ABOUT,          MainFrame::OnMenuAbout
   EVT_TREE_SEL_CHANGED( WID_DICOMTREE,
MainFrame::OnSelectionChanged )
END_EVENT_TABLE()
```

Il peux y avoir plusieurs tables d' évènements dans un programme, aussi nous devons préciser quelle classe doit organiser les évènements pour chaque table fournie. La macro BEGIN_EVENT_TABLE déclare que la table des événements appartient à la classe MainFrame qui est dérivé de la classe wxFrame. Nous avons besoin aussi de préciser que la classe va utiliser une table d'évènements et cela est fait avec la macro DECLARE_EVENT_TABLE() qui apparaît dans le prototype de la classe dans le fichier d'en-tête MainFrame.h).

Ce sont les fonctions membres qui vont gérer les évènements. Elles utilisent entre autre des common dialogs.

Utilisation des «common dialogs»

Beaucoup d'opérations dans un GUI sont répétitives, par exemple l'ouverture des fichiers, l'impression, le changement de répertoire. Cela aide les utilisateurs à condition que les outils qu'ils utilisent pour ces opérations soient pratiques d'utilisation pour le but recherché et que la plupart des API (Application Programming Interfaces) fournissent les opérations courantes. Cela aide très souvent le programmeur et assure aussi une interface fiable.
L'utilisateur peut faire une sélection depuis le menu et s'attendre à voir un dialogue familier fonctionnant de façon logique sans se préoccuper de l'application exécutée.
wxWindows pourvoit aux usages de dialogue courant à travers un certain nombre de classes:

3. wxColourDialog
4. wxFontDialog
5. wxPrintDialog
6. wxFileDialog
7. wxDirDialog
8. wxTextEntryDialog
9. wxMessageDialog
10. wxSingleChoiceDialog
11. wxMultipleChoiceDialog

Nous utilisons par exemple wxDirDialog pour ouvrir u répertoire quand l'utilisateur a cliqué sur le menu Open Directory:

```
MainFrame::OnMenuOpenDir( wxCommandEvent& WXUNUSED( event ) )
{
   wxDirDialog* dialog = new
         wxDirDialog( this, wxT( "Select a DICOM image directory" ),
                     m_dirPath );
}
```

Quand on utilise wxDirDialog comme un dialogue d'ouverture de dossier on voit cette boite de dialogue familière comme indiquée ici.
C'est le Dialog “répértoire” pour la plate-forme Windows, elle se présente différemment sur Linux ou Mac.

Puisque wxWidgets est un cadre multiple plate-forme, cela reste omniprésent si bien que le dialogue courant est celui qui correspond à une plate-forme donnée ou, si le dialogue courant est inexistant, wxWidgets lui substitue un dialogue générique.

Pour chaque Dialog il est possible de spécifier un type de fenêtre de dialogue

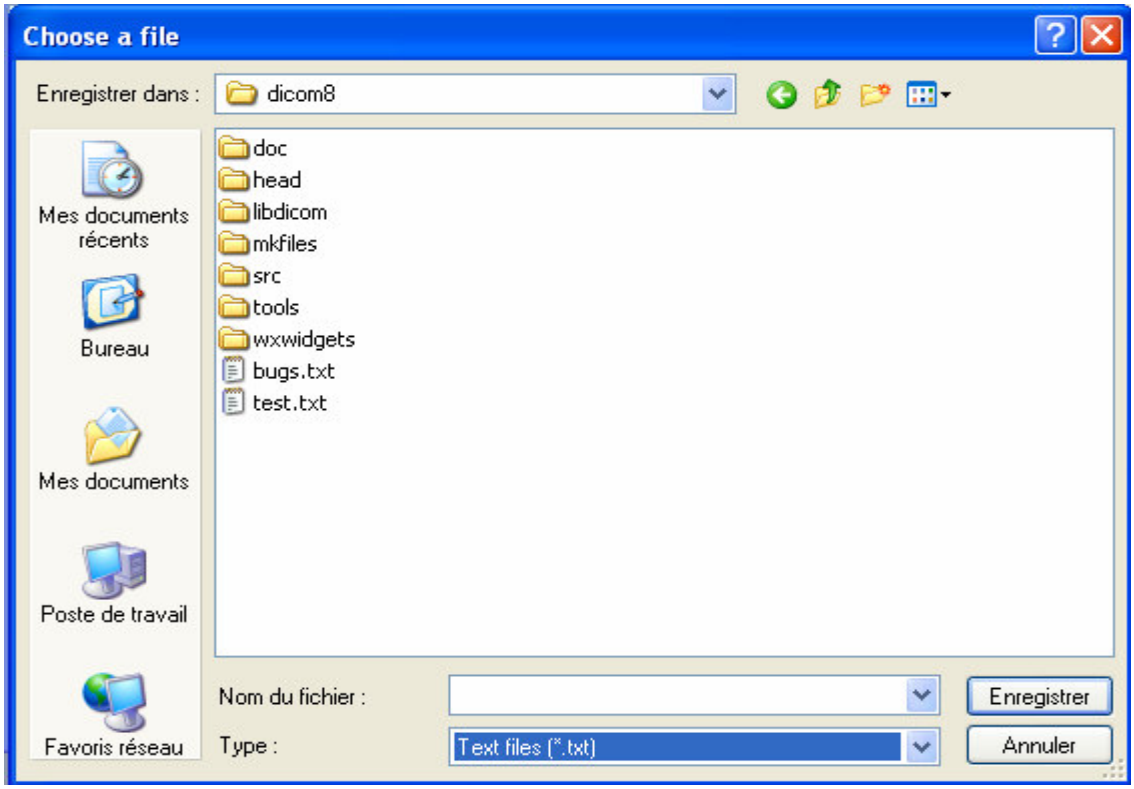
- wxOPEN Dialogue d'ouverture.
- wxSAVE Dialogue de sauvegarde.
- wxHIDE_READONLY Fichiers cachés lecture seule.
- wxOVERWRITE_PROMPT Demande de confirmation pour écraser un fichier.
- wxMULTIPLE Pour dialogue d'ouverture seul: permet la sélection multiple de fichiers

On peut également spécifier un masque pour l’enregistrement des fichier (*.*, *.txt ,*.jpg)

Exemple du dialog de sauvegarde des Tags :

```
wxFileDialog* dialog = new
        wxFileDialog( this, wxT( "Choose a file" ), *wxEmptyString,
                *wxEmptyString, wxT( "Text files (*.txt)|*.txt" ),
                wxSAVE | wxOVERWRITE_PROMPT );
```

Nous pouvons juste avoir "*.*" comme chaîne de type de fichier mais le dialogue de fichier acceptera une chaîne structurée comme l'exemple. Cela représente ma liste des types de fichiers qui apparaîtra la boite composite d'exportation des Tags.



On accède ensuite très simplement au clique du bouton du Dialog par :
exemple pour un bouton OK :
if(dialog->ShowModal() == wxID_OK)
{...}

Lors de la création de l'interface graphique de notre logiciel, nous avons pu constater toute la puissance et la simplicité de la librairie WxWidgets.
L'utilisation de « common dialogs » pour l'ouverture/fermeture/sauvegarde de fichiers ou répertoires allège le code et permet de combiner efficacité et simplicité. WxWidgets permet d'obtenir une interface à la fois fonctionnelle et esthétique qui s'adapte aussi bien à un environnemnt Windows que à un environnement Linux.

Conclusion

Ce projet nous a permis de réaliser un logiciel fonctionnel qui pourra être intégré ensuite à d'autres programmes selon les besoins. Nous avons pu appliquer nos connaissances, notamment pour le coté algorithmique du projet, et découvrir d'autres aspect de la programmation tel les interfaces WxWidgets. Nous tenons à remercier en particulier Caroline Villard notre tutrice, pour nous avoir confié ce projet, et l'ensemble de l' IRCAD Strasbourg.

Merci à Johan Moreau pour son aide lors du développement des librairies préexistantes. Nous espérons que notre logiciel donnera entière satisfaction à ses utilisateurs.