RAPPORT STAGE CAMBA 2021

Affichage du plugin PyDBS sur un écran Holographique zSpace

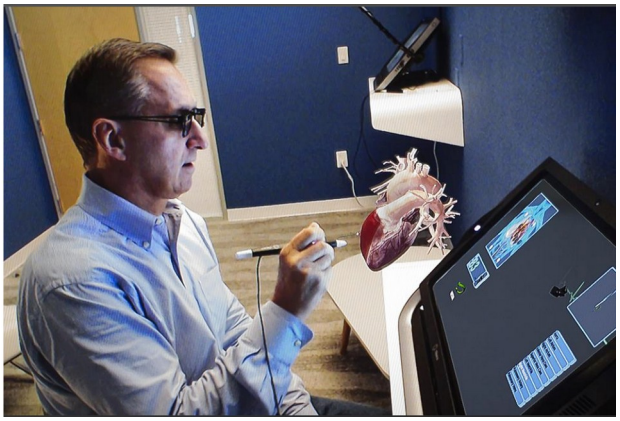
1. Le zSpace

- Introduction

Le zSpace est un écran de réalité virtuelle de type **holographique.**

Ils utilisent la méthode d'**affichage stereoscopique "QuadBuffer"** (ou "Crystal Eyes", voir plus bas) combiné à des lunettes polarisés : on traque la position des lunettes pour calculer l'image en fonction du déplacement de la personne et de son orientation. Cela permet de donner l'impression d'image holographique : l'objet est devant nous ! Sans **tracking** on verrait en 3D comme au cinéma donc plus "plate" et sans possibilité d'intéragir avec l'environnement (décalage à gauche à droite etc).

Le porteur des lunettes voit comme ceci sur un zSpace:



- Technologies

À l'ICM et l'ISIR ce sont des ZSpace AIO 300, c'est-à-dire des **ordinateurs complets** (pas simples écrans) utilisés pour afficher en 3D. Ils ont des propriétées physiques particulière comme une carte graphique compatible avec le mode quadbuffer et un écran 120Hz.

- Programmer des applications compatibles

Pour programmer des applications compatibles avec le zSpace, il faut se référer au Guide du développeur, et à la documentation du zSpace SDK (qui possède des exemples déjà fait voir C:/zspace/zSpaceSdks/.../Samples).

L'API du zSpace est très bien détaillé et heureusement car il n'y a aucun rapport avec les techniques d'affichage de **réalité virtuelle** classique comme les casques VR et les fish tanks.

Il est donc uniquement possible de se référer à la documentation constructeur pour construire une application fonctionnelle et non pas réutiliser des modules déjà existant comme le module VR de Slicer par exemple ou le module Looking Glass.

Looking Glass

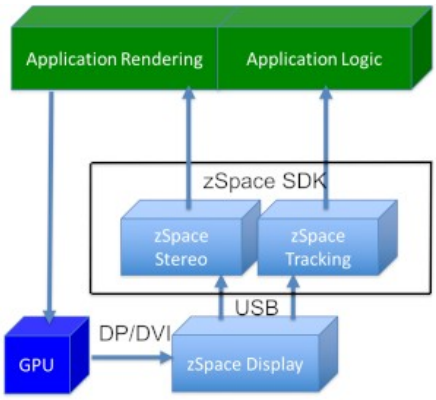
- Structure du code

C'est notre application qui calcule les images stéréoscopiques et non le zSpace lui-même. L'application doit être capable :

- D'**afficher en stereo** en utilisant pour notre cas OpenGL

- D'utiliser des valeurs calculables par zSpace
- Récupérer les infos du stylet

Avec le SDK, on récupère les fonctions nécessaires au tracking des lunettes, du stylet... et au calcul des images stereo, et on envoit tout ça à l'application pour faire le rendu (voir schéma à droite).



L'avantage d'avoir un SDK est que des applications non constructeurs peuvent s'ajouter à la bibliothèque des applications utilisables sous zSpace. Nous allons voir comment rendre compatible 3D Slicer avec cette technologie.

2. Rendre Slicer compatible avec le zSpace: état de l'art

- Plugin Paraview

Paraview est un logiciel de visualisation et de traitement d'image **basé sur VTK**. Il utilise OpenGL pour dialoguer avec les composants physiques (l'API de zSpace a déjà des options implémentées pour réaliser cela) et sa structure est assez proche de celle de Slicer (voir photo à droite tiré du VTK developper guide).

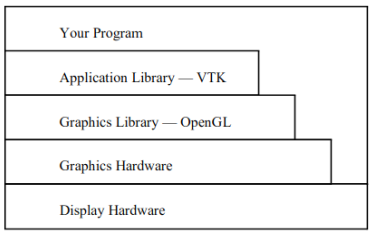


Figure 3–18 Typical graphics interface hierarchy.

Sorti en février 2021, le plugin permet d'afficher en 3D sur le ZSpace avec un suivi du stylet. Pour se faire, il faut son propre build du logiciel car il n'est pas encore disponible dans le binary (téléchargement classique): il le sera dans le release Paraview 5.10 en automne 2021. L'idée pour plus tard (pas de date fixé) est d'intégrer directement le zSpace au module OpenXR propre à VTK (mélange des modules OpenVR déjà dans VTK, de OpenXR lui-même en un seul plugin VTK).

- Module Slicer 4.3

Créé en 2013 par un développeur de Slicer et un de zSpace, le **module permettait d'utiliser la vue 3D du zSpace et le stylet.** Il n'est **pas à jour** avec la nouvelle API de Slicer et n'a jamais vu le jour dans le extension manager de Slicer. Cependant, on peut s'inspirer de sa structure côté Slicer (notamment comment se connecter à la scène etc) car c'est une structure assez classique comme si on utilisait l'extension wizard, et écrit en python.

La documentation est disponible ici : Medical Visualization and Clinical Applications

- Mode QuadBuffer

Slicer possède un **mode stereo QuadBuffer** (dans la vue 3d et le view controller) qui **fonctionne sur Slicer 4.8**, mais plus après, dû à des problèmes de compatibilité Qt et VTK. Le mode stereo QuadBuffer de Slicer permet seulement d'afficher les images en 3D, il ne fait **pas de suivi de tête** ni du stylet. Il nécessite le port de lunette pour voir correctement.

3. Rendre Slicer compatible avec le zSpace: étapes retenues

- Plugin paraview

J'ai commencé par regarder le plugin Paraview pour voir a quel point le plugin pouvait être compatible avec Slicer. Peu de méthodes venaient de la librairies de Paraview directement que j'ai recensés ici :

MODULES

= Repertoires avec des classes à importer

Décrit dans le fichier vtk.module. On importe 5 modules de VTK pour le rendu (rendering core, rendering open gl...) et 2 du zSpace (zspace et zspace view) en plus des 4 de paraview.

Module du plugin venant de la librairie Paraview
ParaView::RemotingViews
ParaView::VTKExtensionsInteractionStyle
ParaView::RemotingMisc
ParaView::RemotingCore

Ces modules permettent d'utiliser ces fonctions propres à Paraview, qui héritent de VTK ou Qt.

CLASSES UTILISÉES

Ces imports sont pour le corps des classes/méthodes créés, il y a problème **si les méthodes n'ont pas d'équivalent** entre PV et Slicer (ou VTK). Il faudra alors modifier les classes du plugin pour trouver des méthodes similaires.

Classes utilisées venant de Paraview	Hérite de	Dans Slicer	What it does	Notes	Liens
PqView	pqProxy	vtkView	This is a PQ abstraction of a generic view module.	Slicer n'utilise pas le proxy	https://kitware.github.io/paraview-docs/latest/cxx/classpqView.html
PqPipelineSource	pqProxy	?	PQ representation for a vtkSMProxy that can be involved in a pipeline		https://kitware.github.io/paraview-docs/latest/cxx/classpqPipelineSource.html#details
pQApplicationC			This class is the crux		https://kitware.github.io/paraview-docs/latest/cxx/

pqRenderView	pQView	vtkRenderView	This is a PQ abstraction of a render view	Core de Qt dans paraview	https://kitware.github.io/paraview-docs/latest/cxx/classpqRenderView.html
pqServerManagerModel	QObject		Model for server manager	Slicer n'utilise pas le proxy	https://kitware.github.io/paraview-docs/latest/cxx/classpqServerManagerModel.html
vtkPVRenderView	vtkPVView>VTKView	vtkRenderView	Render View for ParaView.	Spécifique à PV	https://kitware.github.io/paraview-docs/latest/cxx/classvtkPVRenderView.html#details

CLASSES CRÉÉES AVEC LE PLUGIN

Explication de ce que fait la classe

CLASSES	What is it
vtkZSpaceSDKManager	zSpace SDK manager class
VtkZSpaceRayActor	An actor for displaying a ray. Modifie la tête du rayo
pqZSpaceManager	Autoload class that enable input independent update of the ZSpace render views.

CLASSES MODIFIÉS POUR ETRE UTILISÉES DANS LE PLUGIN

CLASSES	What is it
VtkRenderWindowInteractor	Implements zSpace specific functions required by vtkRenderWindowInteractor.
vtkZSpaceInteractorStyle	vtkZSpaceInteractorStyle extends vtkInteractorStyle3D to override command methods.
VtkZSpaceCamera	Extends vtkOpenGLCamera to use custom view and projection matrix given by zSpace SDK.
VtkPVZSpaceView	vtkPVZSpaceView extends vtkPVRenderView (which extends vtkRenderView, this class is usable by others than PV) to render the scene using Crystal Eyes stereo and interact with it through a zSpace device.

Après avoir trier les infos du plugin Paraview j'ai comparé au zSpace Developper Guide, la structure de rendu de l'affichage est bien entendue la même il fallait donc chercher comment adapter cela à 3D Slicer.

- Module 2013 Slicer

La structure d'affichage en 3D adapté au zSpace est proche de ce qui est donné dans le Developper Guide et dans Paraview, bien entendu beaucoup de fonctions manquent ou n'existent plus car l'**API à beaucoup changé**, il faut donc mettre cela à jour et ajouter ce qui est nécessaire.

Pour connecter une scène MRML de Slicer à l'affichage du zSpace on peut utiliser l'exemple de **l'Extension Wizard** de Slicer 4.13, la méthode sera la même et ressemble beaucoup à ce qu'il y a dans ce module.

Nous avons constaté que dans le module il fallait créer un **ThreeDWidget** : *slicer.qMRMLWidget()*, et que le type de widget créée dépend du fichier **ctkVTKOpenGLNativeWidget**. Dans ce fichier on remarque que le widget créer par défaut est un **QVTKOpenGLNativeWidget** et que ce widget ne supporte pas de faire de la stereo en mode QuadBuffer. Il fallait donc essayer de faire fonctionner Slicer 4.13 avec la stereo QuadBuffer, donc changer le type de widget par défaut par un qui accepte ce mode comme le **QVTKOpenGLStereoWidget**.

L'idée était de construire un plugin pour faire le QuadBuffer moi-même, en reprenant la structure d'anciens code pour le QuadBuffer et en utilisant les nouvelles classes disponibles. Une fois ce plugin créé, voir pour passer directement au zSpace en s'inspirant des deux plugins précédents.

4. Rendre Slicer compatible avec le zSpace: réalisation

- Application VTK puis utilisation dans Slicer

J'ai commencer par créer une **application VTK** que j'ai adapté dans Slicer pour espérer obtenir un rendu QuadBuffer une fois que le widget par défaut auraut été changer.

En effet, pour activer le mode quadbuffer il faut au minimum :

- Utiliser un QVTKOpenGLStereoWidget car le **native n'est pas compatible avec le mode QuadBuffer**, ce sera le nouveau widget par défaut appelé par qMRMLThreeDWidget
- **Afficher la surface** du widget en mode stereo avec "QSurfaceFormat().setStereo(True)"
- **Activer le mode stereo crystal eyes** (=QuadBuffer) de la renderWindow en la rendant capable d'afficher en stereo : "SetStereoCapableWindow(1) " et en mettant le bon mode "SetStereoType(1)" (1 pour crystal eyes).

La méthode "SetStereoType(3) " associé au nœud qui observe la scène active le mode QuadBuffer comme si l'on cliquait sur le bouton dans la fenêtre 3D.

Mon plugin terminé et fonctionnant avec les autres types de stereo il fallait maintenant changer de widget par défaut.

Voir le plugin ici, et son mode d'utilisation plus bas (annexe 2). Voir le tableau pour le nom des fonctions liées au QuadBuffer (plusieurs renommages) (annexe 3).

On supposait alors que l'on avait besoin d'avoir Slicer qui acceptais de faire du QuadBuffer pour utiliser le zSpace. Effectivement, le plugin de 2013 de Slicer utilise un widget qui à l'époque pouvait afficher en QuadBuffer alors que maintenant le widget par défaut ne peut pas le faire.

- Premièrement quand on clique sur le mode QuadBuffer stereo dans le fenêtre 3D de Slicer 4.13 on lit dans les output:

Warning: In C:\S4\S4R\VTK\Rendering\Core\vtkRenderWindow.cxx, line 240

vtkGenericOpenGLRenderWindow (00000185F9811E80): Adjusting stereo mode on a window that does not

support stereo type CrystalEyes is not possible.

Donc on a besoin d'un widget/fenêtre capable de faire du stereo QuadBuffer, comme le widget QVTKOpenGLStereoWidget et non celui de base QVTKOpenGLNativeWidget.

- J'ai essayé de changer le type de widget dans Slicer :

On se met sous slicer 4.13 et en mettant Slicer_VTK_VERSION_MAJOR=9 (car pas dispo en 4.11). En prenant exemple sur l'extension Wizard avec un scripted plugin. Pour le build from source voir : https://slicer.readthedocs.io/en/latest/developer_guide/build_instructions/windows.html

Problème : pour modifier le type de widget il y a beaucoup de dépendances au sein du cœur de Slicer avec des anciennes classes maintenant obsolètes et changer uniquement les options de build ne suffisait pas :

- Où en est actuellement :

J'ai essayé de voir dans le cœur de slicer comment passer à un autre widget par défaut.

Selon les développeur du logiciel (voir ma question sur le forum ici : https://discourse.slicer.org/t/quadbuffer-view-in-slicer-4-11/17508), le fichier **ctkVTKOpenGLNativeWidget** permet de **choisir entre deux options de build** : avec le widget natif ou stereo. En effet dans ce fichier la différentiation est possible via les options de build **CTK_HAS_QVTKOPENGLNATIVEWIDGET_H =OFF** et **CTK_USE_QVTKOPENGLWIDGET = ON**, mais après plusieurs essais cela ne fonctionnait pas. J'ai donc enlever les options possibles et laisser uniquement le QVTKOpenGLWidget (ancien nom de QVTKOpenGLStereoWidget) pour créer les widget. Après quelques autres modifications dans ctkVTKWidgetUtils.

Après vérifications dans Slicer 4.13, nous sommes bien en widget stereo mais il n'affiche rien dans la vue 3D. Cela vient d'un autre problème dans le core. C'est un **problème d'héritage** lors de la création des widgets :

  - De base par défaut on a QVTKOpenGLNativeWidget qui hérite de QOpenGLWidget
  - Celui que l'on veut est QVTKOpenGLStereoWidget qui hérite de QWidget = on perd des signaux + méthodes à cause de l'héritage différent

- Solution?

Modifier encore le corps de Slicer pour résoudre les erreurs liées au passage en stereo widget par défaut, notamment ici, modifier le signal émit ou faire hériter les fonctions différemment. Ou voir avec les développeur de Slicer une "amélioration du logiciel" pour que le changement de widget par défaut soit plus simple.

Conclusion

Nous n'avons pas réussi à changer le widget par défaut et pensons que sans ce type de widget, Slicer ne sera pas compatible avec le zSpace.

Par ailleurs, le développeur du plugin Paraview n'a pas répondu à nos mails concernant l'utilisation de widget (du même style que notre ThreeDWidget) dans son plugin, ainsi, nous n'avons pas su si le type de widget lui a posé un problème ou non dans son code.

De plus, le code que nous avons fait et mis plus haut, permettra d'afficher en mode stereo QuadBuffer une fois le changement de widget par défaut faisable.

Dans l'idée pour le futur il faudrait:

- Rester sur **Slicer 4.13 avec VTK 9** pour ne pas poser de problèmes de compatibilité et être le plus à jour possible.
- Regarder les étapes obligatoire du **zSpace developper guide**, en suivant la structure du code proposé et en s'aidant des **samples** dans zSpace directement.
- Beaucoup de méthodes du plugin de Paraview peuvent presque directement être utilisés dans Slicer grâce à VTK (voir tableau 2), il doit être possible de s'en inspirer et d'utiliser la structure de ce code, même si je privilégierai une lecture approfondie de ce code puis un retour à 0 pour le plugin Slicer en se basant uniquement sur le zSpace developper guide pour ne pas se laisser bloquer par des fonctions uniquement disponibles sur Paraview.
- Le plugin de 2013 donne un rapide aperçu de la structure nécessaire pour faire fonctionner Slicer (connexion à la scène etc) et utiliser l'extension wizard. Attention zSpace dit que les méthodes sont en C donc le plugin Slicer devra sûrement être un loadable module et non un scripted.

Il n'est sûrement pas nécessaire de passer par l'activation du mode QuadBuffer dans Slicer, seul le type de widget rentrera en compte. Nous avons fait une future request à Slicer pour ne pas bloquer les futurs travaux si il se trouve avoir réellement besoin d'un widget qui fait du stereo QuadBuffer.

Kitware pense **intégrer le zSpace à son module OpenXR** mais il n'y a pas de date de sortie actuellement. Sachant que le plugin zSpace sortira dans le binary en automne 2021, l'intégration n'arrivera peut être pas avant 2022.

Annexe

1. Définition de QuadBuffer

Quadbuffer est le fait de calculer 4 images au lieu de 2 pour donner un effet de réalisme et de netteté. Rapidement : pour afficher une image un ordinateur la calcule, la stocke dans un tampon (buffer) puis l'affiche, en temps normal il y a une image qui s'affiche quand l'autre est en train d'être calculée (dual buffer). Pour plus de réalisme et de fluidité, et parce que pour voir en 3d l'œil humain doit voir une image "gauche" et une "droite" pour créer la profondeur, on va calculer 4 images (quadbuffer) : 2 par œil, et pour chaque œil une image sera en train d'être calculé et l'autre sera affichée. Ils faut donc avoir 4 images en simultané dans la mémoire (en train d'etre calculé ou déjà calculté qui attend d'etre affiché) d'où le QuadBuffer!

2. Fonctionnement du plugin:

Aller dans la catégorie zSpace, cliquer sur le module.

Quand clique sur le bouton affiche une fenêtre à l'extérieur du layout classique, capable d'afficher en QuadBuffer stereo mode. Pas besoin d'activer le mode à la main le plugin s'en charge.

Fermer la fenêtre directement avec la croix et relancer le plugin autant de fois que l'on veut.

Les vues 3D classiques et dans la fenêtre du QuadBuffer sont indépendantes, j'ai choisis de créer une nouvelle fenêtre mais garder la fenêtre 3d classique et passer au layout 3d uniquement ferait la même chose.

3. Classes VTK et versions

Classes de VTK 8 et 9 pour créer des widgets dans Slicer avec QMRMLThreeDWidget:

Cette classe permet de créer différents types de widget, avec le renommage de ces classes sur VTK il faut

faire attention à qui nous manipulons.

Classes	Version VTK	Utilisation
QWIDGET	Deprecated depuis 8	QWidget for displaying a vtkRenderWindow in a Qt Application. Deprecated Please use QVTKOpenGLNativeWidget instead, https://vtk.org/doc/nightly/html/QVTKOpenGLWidget_8h_source.html
QVTKOpenGLWidget	Deprecated depuis 8	Wrapper for QVTKOpenGLStereoWidget for legacy support purpose. QVTKOpenGLWidget is only a wrapper for QVTKOpenGLStereoWidget so old code can still be built. Peut être utilisé pour le quad buffer à voir dans son .h
QVTKOpenGLStereoWidget	Nouvelle QVTKOpenGLWidget (renommage) depuis 9	If Quad-buffer stereo rendering capabilities are needed better choice than Native (and cannot be used as native widget) !! If use QScrollArea or in a QMDIArea : will force it to be native and this is *NOT* supported Couche la plus haute ?? https://github.com/Kitware/VTK/blob/master/GUISupport/Qt/QVTKOpenGLStereoWidget.h#L69
QVTKOpenGLNativeWidget	Depuis 8	for a more versatile implementation (pas le premier choix car difficile et pas super pour quadbuffer)
QVTKOpenGLWindow	9	One of the mechanisms for displaying VTK rendering, results in a Qt application. QVTKOpenGLWindow extends QOpenGLWindow to display the rendering results of a vtkGenericOpenGLRenderWindow. *Friend class QVTKOpenGLStereoWidget*