École Nationale Supérieure
d'Électrotechnique, d'Électronique, d'Informatique,
d'Hydraulique et des Télécommunications

Institut National Polytechnique

Guide développeur

Alexandre Bernard
Victor Delaveau
Jean Matthieu Khoury
Lucile Schneider
Nadia Voedts
10 Mars 2017

Summary

1 Informations générales

1.1 Environnement

L'application a été développée en langage C++ à l'aide de Visual Studio 2015. Afin de pouvoir compiler et tester l'application, il est nécessaire d'installer plusieurs outils :

- Package boost : afin de pouvoir gérer les threads
- SDK Oculus Rift 0.8 : afin de pouvoir utiliser les méthodes de communcation avec l'Oculus
- Runtime Oculus Rift 0.8

Nous avons travaillé en utilisant la version 0.8 car il s'agit de la dernière version du Runtime proposant un simulateur d'Oculus permettant de tester sans disposer du casque; cependant, le runtime 1.11 est installé sur l'ordinateur du client. Il faut donc remplacerle SDK par la version 1.11 (disponible ici) avant de tester le logiciel sur cet ordinateur.

1.2 Mise en place

SDK : Télécharger le SDK. Dans les propriétés du projet :

1. VC++ Directories -> Include Directories -> Edit : ajouter le dossier Include (..\OculusSDK\LibOVR\Include) ;
2. Library Directories -> Edit -> ajouter les Lib (..\OculusSDK\LibOVR\Lib\Windows\Win32\Release \VS2015)
3. Linker -> Input -> Additional Dependencies -> Edit : écrire "LibOVR.lib"
4. Appliquer les modifications

Boost : Télécharger et dézipper l'archive boost. Exécuter bootstrap puis bjam puis b2. Dans les propriétés du projet :

1. C++ -> General -> Additional Include Directories -> Edit : ajouter le dossier ..\boost_1_60_0\boost_1_60_0
2. C++ -> changer "use precompiled headers" en "not using precompiled headers"
3. Linker -> General -> Additional Library Directories -> Edit : ajouter le dossier ..\boost_1_60_0\boost_1_60_0\stage\lib

1.3 Limitations

Un des problèmes que présente la version la plus récente du Runtime/SDK de l'Oculus Rift est qu'il n'est pas possible de faire tourner simultanément deux applications accédant au casque, par exemple celle-ci et Vestibulus, logiciel utilisé par Denis Ducommun pour afficher des images dans le casque. Afin de pouvoir afficher quelque chose dans le casque, il faut donc le programmer dans le projet même, sans quoi ne sera affichée qu'une grille.

1.4 Contact

En cas de questions relatives au projet, vous pouvez contacter un des membres de notre groupe aux adresses suivantes :
Alexandre Bernard : alexandre.bernard94@gmail.com
Victor Delaveau : victor.delaveau@gmail.com
Jean Matthieu Khoury : jean.khoury@etu.enseeiht.fr
Lucile Schneider : schneider.lucile@gmail.com
Nadia Voedts : nadia.voedts@gmail.com

2 Architecture globale

Nous avons principalement utilisé des Windows Forms pour élaborer notre projet. Ils permettent de disposer d'un éditeur graphique permettant d'ajouter des graphiques, boutons, zones de textes, listes à dérouler (ce sont les composants que nous avons utilisés). Les deux fichiers principaux de notre projet sont Home (fenêtre s'ouvrant lors du lancement de l'executable) et AcqForm, la fenêtre s'affichant lors de l'appui sur acquisition. Il y a également un fichier généré WProject décrit ci-dessous.

Pour les forms, le code se trouve dans les fichiers d'extension .h, disponibles dans 2 versions : texte et design. Pour le reste des fichier, le code est présent dans les fichiers .cpp et les fichiers .h associés constituent leurs interfaces (afin de pouvoir les importer dans d'autres fichiers).

3 Architecture détaillée

3.1 WProject

Classe principale de l'application générée par Visual Studio lors de la création du projet. Les modifications effectuées ont été de commenter la ligne faisant apparaître cette fenêtre (ligne 116) et de faire apparaître Home (ligne 41) après avoir appelé la méthode EnableVisualStyles() afin d'avoir un style Windows 10 au lieu de 98 par défaut.

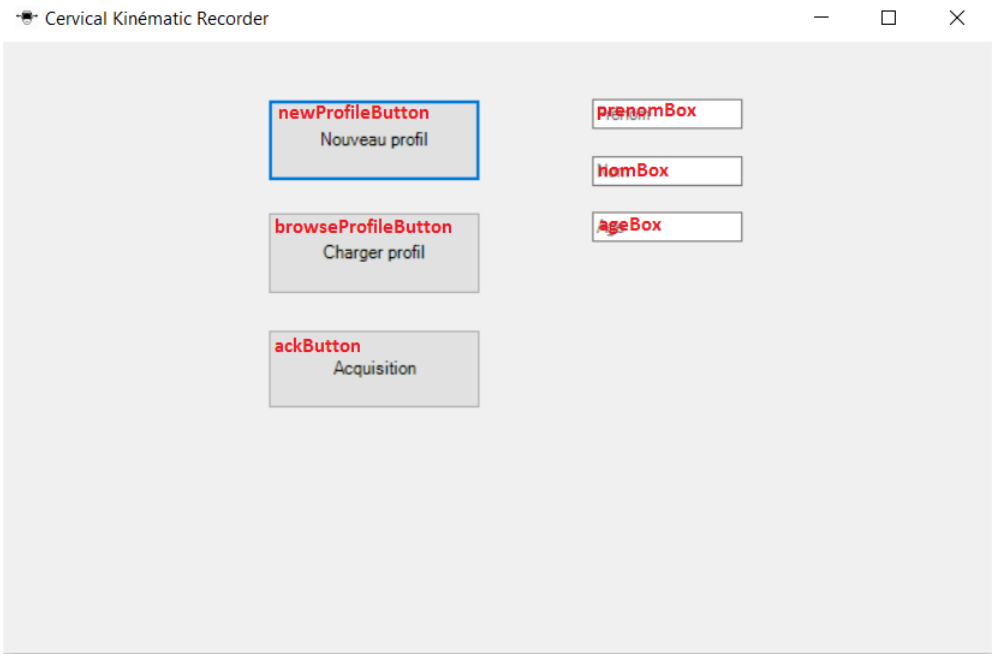
3.2 Home

Le code est contenu dans le fichier Home.h. Il s'agit donc d'un windows form dont le squelette est généré automatiquement. Rajouter un accès à d'autres form que AckForm ou d'autres informations concernant la gestion des profils des patients pourra se faire dans ce fichier

3.2.1 Attributs

Nous avons mis en place les composants graphiques suivants (en gras figurent les types des objets) :

- **TextBox** prenomBox : zone de texte où l'utilisateur entre le prénom du patient
- **TextBox** nomBox : zone de texte où l'utilisateur entre le nom du patient
- **TextBox** ageBox : zone de texte où l'utilisateur entre l'age du patient
- **TextBox** infoBox : zone de texte où sont affichées les informations du patient actuellement chargé
- **Button** ackButton : bouton permettant d'accéder à la fenêtre d'acquisition
- **Button** browseProfileButton : bouton ouvrant un explorateur de dossiers permettant de charger le profil d'un patient déjà existant
- **Button** newProfileButton : bouton permettant de créer un nouveau profil patient après avoir entré ses nom et prénom.



3.2.2 Méthodes

- **InitializeComponent** : Cette méthode est générée et met en place les différents composants et leurs propriétés et notamment leurs gestionnaires d'événements détaillés dans la suite.
- **newProfileButton_Click** : Cette méthode est appelée lorsque l'utilisateur clique sur le bouton "Nouveau Profil". Elle effectue un test sur les prenomBox et nomBox : si celles-ci sont vides ou contiennent les caractères "Prenom" et "Nom" (valeurs par défaut), on inscrit dans infoBox le texte "Aucun nom saisi". Sinon, les attributs nom et prenom prennent la valeurs des boxs, un dossier est crée (de la forme nom_prenom) et on inscrit dans infoBox le texte "Profil crée : nom + prenom".
- **Home_FormClosing** : Gestionnaire de l'événement de fermeture de Home (clic sur la croix rouge) : on appelle la méthode exit afin de fermer également MyForm1.
- **prenomBox_GotFocus** : Lorsque l'utilisateur clique dans prenomBox pour commencer à y inscrire un prénom, le texte par défaut ("Prénom") s'efface et la couleur de la police passe au noir (au lieu du gris précedemment).
- **nomBox_GotFocus** : Identique à la méthode précédente pour nomBox.
- **ageBox_GotFocus** : Identique à la méthode précédente pour ageBox.
- **openExplorer** : Fonction annexe permettant de lancer un thread en mode STA (nécessaire pour certains threads).
- **openFolder** : Fonction annexe permettant d'ouvrir un explorateur windows pour selectionner un dossier existant correspondant au profil d'un patient et en mettre le path dans la variable path (si l'utilisateur clique sur OK).
- **ackButton_Click** : Méthode appelée lorsque l'utilisateur clique sur le bouton acquisition. Si les champs nom et prenom ne sont pas vides (c'est-à-dire qu'un profil vient d'être crée ou chargé), il instancie un objet MyForm1 et lui envoie les valeurs de nom et prenom à l'aide des méthodes setPrenom et setNom de MyForm1. Elle appelle également la méthode printNoms de MyForm1 qui

permet d'afficher les nom et prénom du patient dans la fenêtre d'acquisition. Ensuite, la méthode "réinitialise" la fenêtre home : prenomBox et nomBox reprennent les valeurs par défaut et une police grise, les champs nom et prénom sont remis à "" et infoBox est vidée.

Si la condition n'est pas remplie, le texte d'infoBox devient "Aucun profil selectionné".

- **browseProfileButton_Click** : Méthode appelée lorsque l'utilisateur clique sur le bouton charger profil. Elle lance un nouveau thread (en mode STA grâce à openExplorer) qui ouvre un explorateur windows pour que l'utilisateur puisse saisir le dossier correspondant au profil voulu (grâce à la fonction openFolder) puis à la fin du thread charge ce profil en mettant à jour les texte des Box nom, prenom et age ainsi que leur couleur et l'infoBox affiche "Profil Chargé :" suivi des informations du profil.

  Un test encadre le tout pour vérifier que le path a bien été changé (et que donc l'utilisateur n'a pas cliqué sur annuler lors du chargement du profil)

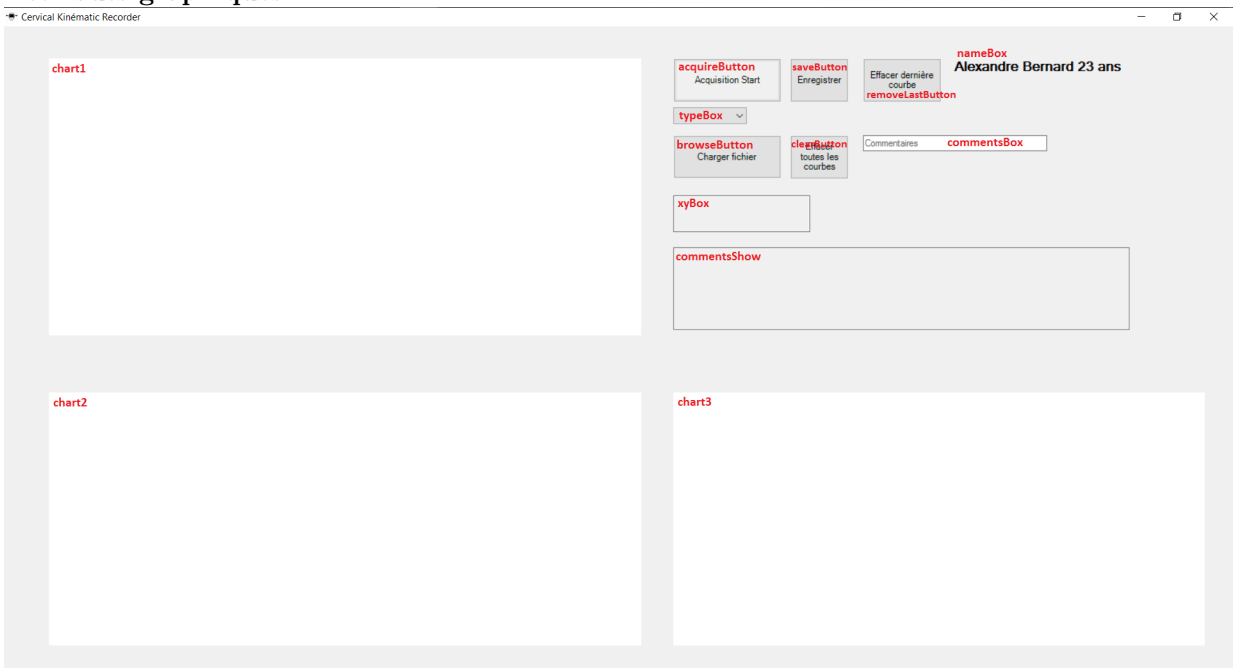
- **splitPath** : Fonction annexe permettant de récupérer uniquement le nom du dernier dossier d'un path.

3.3 AcqForm

Ce fichier est le noyau de l'application, tout ajout de fonctionnalité pourra se faire dans ce form ou tout du moins être appelé dans ce form par l'ajout de boutons et de fonctions annexes.

3.3.1 Attributs

Attributs graphiques :



Autres attributs :

- **bool** canSave : indique s'il existe une acquisition non enregistrée
- **bool** canCursor : indique si une courbe est présente sur les charts et donc s'il faut y afficher un curseur

- **bool** fromCharge : indique si la courbe à tracer provient d'un chargement (utile pour la génératon de titre de série)
- **String** path : chemin de la courbe à charger
- **String** prenom : prénom du patient courant
- **String** nom : nom du patient courant
- **String** age : âge du patient courant
- **String** nomCharge : nom du patient dont on charge une courbe
- **String** fileName : nom du fichier chargé lors du chargement d'une courbe
- **Thread** ackThr : thread d'acquisition démarré lors d'un clic sur Acquisition Start
- **Icon** icon : icône de l'application

3.3.2 Méthodes

- **acquireButton_Click** : Cette méthode est appelée lorsque l'utilisateur clique sur le bouton Acquisition (Start ou Stop). Elle commence par vérifier qu'un mouvement a bien été sélectionné; si ce n'est pas un message "Aucun mouvement sélectionné" s'affiche dans xyBox. Sinon, typeBox est bloquée et on affiche le nom des courbes dans les charts selon le mouvement sélectionné. Ensuite, si l'utilisateur a cliqué sur "Acquistion Start", on change le texte de la box en "Acquistion Stop" et on lance le thread d'acquisition. S'il a cliqué sur "Acquisition Stop", on procède à l'opération inverse et on écrit l'acquistion dans un fichier temporaire en vue d'un eventuel enregistrement. Le booléen canSave passe à true, indiquant qu'il y a une acquisition non enregistrée.
- **browseButton_Click** : Cette méthode est appelée lorsque l'utilisateur clique sur le bouton "Charger". Comme pour l'acquisition, on commence par vérifier qu'un mouvement a été selectionné. Si c'est le cas, on lance dans un thread la méthode openFile qui ouvre un explorateur de fichiers. On récupère alors le chemin selectionné par l'utilisateur dans path. On vérifie que le fichier chargé est compatible avec ceux déjà affichés (même type de mouvement). Dans ce cas, on extrait du path le nom du patient afin de pouvoir l'afficher en titre de série. On cherche ensuite les commentaires et on les affiche dans commentsShow. Enfin on convertir le fichier en retour et on trace les courbes correspondantes avec leurs titres selon le type de mouvement, et on réinitialise le path.
- **clearButton_Click** : Cette Méthode est appelée lorsque l'utilisateur clique sur le bouton "Effacer toutes les courbes". On vérifie la valeur de canSave : s'il est à true, il y a une acquistion qui n'a pas encore été enregistrée; on affiche donc une pop-up d'avertissement qui propose d'enregistrer, de ne pas enregistrer ou d'annuler l'opération. Si l'utilisateur choisit d'enregistrer, on le fait puis on appelle clear() (détaillée plus loin). On efface le fichier temporaire et canSave repasse à false. S'il choisit de ne pas enregistrer, on effectue les mêmes opérations sauf celle d'enregistrement.

  Si canSave était à false, on appelle clear(), on efface tmp et canSave passe à false.
- **saveButton_Click** : Cette méthode est appelée lorsque l'utilisateur clique sur le bouton "Enregistrer". Si canSave est à true (une acquistion n'a pas été enregistrée, un fichier temporaire a été écrit), elle appelle save().
- **removeLastButton_Click** : Cette méthode est appelée lorsque l'utilisateur clique sur le bouton "Effacer dernière courbe". Si canSave est à true (il y a une acquisition non enregistrée)et qu'il y a que cette série présente sur les charts (ce serait donc celle-ci que l'on effacerait), une boite de dialogue s'ouvre alertant l'utilisateur. S'il choisit d'enregistrer, on appelle save() puis on retire la courbe; sinon on retire juste la courbe. Si l'utilisateur annule, on desactive typeBox.

- **clear** : Méthode de "nettoyage" du form : retire les séries des charts, passe canCursor à false, rend typeBox éditable, vide la zone d'affichage de commentaires, reinitialise celle d'édition des commentaires à "Commentaires" et vide le path.
- **plot(Retour_t retour)** : Méthode de tracage des courbes : prend en paramètre un retour. Crée les 3 séries de données, les ajoute aux charts. Si fromCharge est à faux, c'est-à-dire si le retour provient d'une acquisition et non d'un chargement, on génère le nom de la courbe avec la date et le nom du patient courant. On récupère ensuite les données yaw,pitch et roll du retour et on les ajoute aux séries précédemment créées selon le type de mouvement choisi. canCursor passe à true, indiquant qu'une courbe est présente.
- **openExplorer** : ouvre un explorateur de fichiers
- **openFile** : Ouvre un explorateur de fichiers dans le dossier du patient courant en lui donnant la possibilité de sélectionner un fichier d'extension .orpl. S'il choisit un fichier, on récupère le chemin de celui-ci d
- **printTitles** : Teste la valeur de type (texte de typeBox) et modifie les titres des charts en conséquence.
- **printXY(Chart c, MouseEvent e)** : Imprime les coordonnées de la souris dans la chart passée en paramètre.
- **chart[i]_MouseMove** : Cette méthode est appelée lorsque la souris passe dans la zone d'une des 3 charts. Si canCursor est à true, c'est-à-dire qu'une courbe est affichée, elle récupère les coordonnées de la souris dans le graphe et les affiche dans xyBox en appelant printXY.
- **AcqForm_FormClosing** : Cette méthode est appelée lorsque l'utilisateur ferme la fenêtre d'acquistion. Elle empêche la fermeture et cache la fenêtre à la place, sinon le form ne pourra plus être ouvert par la suite.
- **commentsBox_GotFocus** : Cette méthode est appelée lorsque l'utilisateur clique dans le champ d'écriture de commentaires : elle vide commentsBox (où est initialement écrit "Commentaires" et change la couleur de la police en noir.
- **xyBox_MouseEnter** : Transforme le curseur d'édition de texte en curseur flèche lorsque la souris entre dans la zone d'affichage des coordonnées.
- **commentsShow_MouseEnter** : Similaire à la méthode précédente pour la zone d'affichage des commentaires.
- **xyBox_MouseClick** : Cache le curseur d'édition lors d'un clic dans la zone d'affichage des coordonnées.
- **commentsShow_MouseClick** : Similaire à la méthode précédente pour la zone d'affichage des commentaires.

3.4 Acquire

Tous les appels à l'oculus se font depuis ce fichier, la récupération des positions x y z de l'oculus à déjà été implémentée (puis commentée) en prévision d'une possible utilisation, elles pourront donc être stockées dans le fichier tmp ou un autre pour être utilisées ou envoyées.

3.4.1 Attributs

Le code est contenu dans le fichier acquire.cpp et correspond à la fonction d'accès à l'Oculus pour la récupération des données de rotation et leur envoi dans un fichier temporaire.

- **int** duree : duree maximale de l'acquisition (intitialisée à 1 minute)
- **int** nBperSec : Nombre de Hertz à laquelle l'aquisition se fait (intitialisé à 40 Hz)

3.4.2 Méthodes

- **acquire** : Initialise la connexion avec l'Oculus Rift (et l'éteint si elle ne se fait pas correctement), et récupère au rythme prédéfini les informations d'orientation de l'oculus, les convertit au format eulérien, les tare en prenant la première valeur mesurée comme zéro, et les inscrit au fur et à mesure dans un fichier temporaire tmp.orpl. Une fois terminé il ferme l'accès au fichier, detruit la session en cours avec l'oculus rift et éteint l'oculus.
- **Quat_To_Pitch** : Convertit un quaternion en angle eulérien pitch.
- **Quat_To_Yaw** : Convertit un quaternion en angle eulérien yaw.
- **Quat_To_Roll** : Convertit un quaternion en angle eulérien roll.

3.5 ConvertToRetour

3.5.1 Type

- **Retour_t** : structure représentant une acquisition : contient 3 vecteurs de données yaw,pitch, roll ainsi que la taille de l'acquistion (taille d'un des vecteurs)

3.5.2 Méthodes

- **convertToRetour** : Prend en paramètre un fichier (en l'occurence un fichier .orpl issu d'une acquisition) et le lit ligne par ligne afin de ranger chaque donnée (yaw, pitch,roll) dans une structure de type Retour_t.
- **splitLine** : Prend en paramètre une chaîne de caractères (en l'occurence une ligne d'un fichier .orpl) et range dans un vecteur les coordonnées yaw,pitch,roll. Par exemple, la ligne 1.0 2.0 3.0 donnera le vecteur (1.0,2.0,3.0).

3.6 WriteInFile

3.6.1 Méthode

writeInFile : Prend en paramètre un retour de type Retour_t et une chaîne de caractères contenant le path (chemin dans lequel on veut enregistrer le retour). Crée le fichier à l'endroit souhaité et écrit dedans les données contenues dans le retour avec en première ligne "yaw pitch roll" puis pour chaque ligne i yaw(i) pitch(i) roll(i).

3.7 saveAcq

3.7.1 Méthode

saveAcq : Prend en paramètres le nom, le prénom et l'âge d'un patient ainsi que le type du mouvement effectué. Récupère la date et l'heure pour créer le nom du fichier à enregistrer avec le type du mouvement, puis écrit le chemin auquel enregistrer le fichier à partir des nom,prénom, âge. Convertit ensuite le fichier temporaire d'acquisition en Retour_t pour appeler writeInFile avec le retour et le path.

3.8 sstringToChar

3.8.1 Méthode

sstringToChar : Prend en paramètre un String et le convertit en const char*, opération fréquemment effectuée dans AcqForm.

4 Mode d'emploi d'ajout de fonctionnalités

4.1 Ajout d'un champ

Si vous souhaitez ajouter un champ à un profil (ville, taille...), les étapes à effectuer sont :

- Créer l'attribut dans Home.h et AcqForm.h : String champ
- à l'aide de l'éditeur graphique, ajouter une TextBox à Home, la nommer champBox.
- Dans Home.h créer la méthode getChamp sur le même modèle que getNom
- Dans Home.h ajouter dans newProfileButton_Click: champ = champBox->Text; et modifier System::IO::Directory::CreateDirectory(nom + "_" + prenom + "_" + age); en System::IO::Directory::CreateDirectory(nom + "_" + prenom + "_" + age + "_" + champ);
- Dans Home.h ajouter dans ackButton_Click: form->setChamp(this->getChamp());, champBox->Text = "Champ"; , champBox->ForeColor = System::Drawing::SystemColors::WindowFrame;, champ="";
- Dans AcqForm.h ajouter la méthode setChamp sur le modèle de setNom

4.2 Ajout d'un bouton

La démarche est la même pour l'ajout d'un bouton dans AcqForm ou Home :

- Ouvrir le fichier en mode design
- Dans la toolbox, chercher button et faire glisser à l'endroit souhaité du form, puis rendimensionner si nécessaire
- Dans les propriétés du bouton, aller dans l'onglet des événements et double-cliquer sur buttonClick : ceci va générer la méthode dans le code du form
- Remplir la méthode selon l'action souhaitée