POLYTECH LILLE

Université Lille1 Sciences et Technologies

OUT OF SPACE

IMA 2013/2014

Zohour ASSAIEB / Adnane JAOUI

Remerciement

Nous tenons d’abord à exprimer nos profonds remerciements à toute personne nous ayant aidé de près ou de loin à la réalisation de ce projet à ses différents étapes.

Nous tenons aussi à remercier Mr Laurent GRISONI pour la confiance qui nous a accordée, ainsi que sa disponibilité et le temps qui nous accordé tout au long du projet ainsi que Matthias DE BIE, un des étudiants IMA5 qui travaillait sur le même projet, pour toutes les informations qu’il nous a fourni.

Sommaire

Introduction

I. Présentation du projet
   a. Contexte
   b. Objectif
   c. Cahier des charges
   d. Outils logiciel et matériel

II. Travail effectué
   a. Présentation de l'outil de Calibration
   b. Travail préliminaire
   c. Développement de l'outil sur QT

Conclusion

Annexes

Introduction:

Ce rapport présente le travail que nous avons effectué dans le projet de 4ème année de cycle d'ingénieur à Polytech'Lille, pour notre formation en Informatique Microelectronique et Automatique, nous avons réalisé ce projet dont l'intitulé est "Out of space", proposé par l'équipe MINT du LIFL, dans l'optique de la collaboration avec une artiste et une société de jeux vidéo, afin de réaliser un outil d'interaction homme machine en créant une expérience d'exploration et de manipulation dans un univers 3D. Ce projet est la continuité d'autres projets qui ont été réalisé par des étudiants IMA5 pour leurs projets de fin d'étude, ou dans le cadre d'un stage.

Étant nous-même de filière Systèmes Autonomes, nous avons choisi ce sujet à prédominance en systèmes communiquant afin de prolonger notre formation en informatique, d'autant plus que le domaine dans lequel s'inscrit ce projet est intriguant, et c'était pour nous une opportunité de pouvoir travailler sur un sujet qui tourne autour du fonctionnement et de la réalisation de projets de graphisme 3D, et qui nous amènera à chercher et comprendre plus de choses sur le sujet et les mettre en pratique.

Dans notre rapport, nous essaierons d'abord de présenter notre projet et son contexte, nous rentrerons ensuite nous allons expliquer les méthodes utilisées pour la réalisation du sujet, et enfin établir les résultats obtenus et difficultés rencontrées.

I- Présentation du projet

1- Contexte

”Out of space” met en collaboration une artiste et une société de production de jeu vidéo 3D. Il vise à mettre en place une application interactive en utilisant une classe de dispositifs d'interaction orienté "maquette holographique". La maquette holographique est un écran 3D ou l’on affichera une scène créée par l’artiste.

Le projet global vise donc à réaliser une application interactive sur un affichage stéréoscopique, et qui consiste à piloter un objet (une bille) dans un certain environnement, tout en déroulant l’évolution de cet environnement sur une vidéo de fond. Le mouvement de cet objet est contrôlé par les gestes des mains qui sont détectées par une Kinect et traduits sur un écran 3D.

De là où les étudiants ayants travaillé sur le projet se sont arrêté, la calibration de la Kinect avec l’écran se faisait de manière manuelle, sauf que l’utilisateur de l’application n’est pas forcément un informaticien. Nous avons donc intervenus sur ce point-là afin de rendre le calibrage automatique et faciliter l’utilisation du matériel.

2- Objectifs

La tâche qui nous a été incombée est celle de créer un outil de calibration qui est censé permettre à l’artiste ou n’importe quel autre utilisateur non informaticien de pouvoir ajuster les réglages et le calibrage des Kinects par rapport à l’écran stéréoscopique sans avoir à modifier les lignes de code de l’application.

Notre objectif est donc celui de développer une interface graphique qui servira à guider l’utilisateur à réaliser le calibrage du matériel de manière autonome.

3- Cahier des charges

- Réaliser une application graphique sur QT qui se déroulera sur 3 étapes et par l’affichage de 3 fenêtres.
- L’application exécutera en même temps des applications nécessaires au calibrage.
- Faire un programme qui sera lancé aussi et qui fera le transfert des coordonnées enregistrés.

4- Outils logiciels et matériels utilisés :

- 3Gear possède une technologie qui permet, à partir des caméras 3D de la Kinect, de reconstruire une représentation précise des doigts et de ce que les mains font, en prenant les données 3D brutes et les transformant en informations utilisables sur l'état des mains. Les interfaces de programmation sont basées sur le pointage et le pincement et fournissent les angles des articulations approximatives, elles sont disponibles en C ++, C # et Java

- Unity 3D est un logiciel middleware de création de contenu interactif comportant de la vidéo et des objets 3D/2D. Il permet de créer des scènes supportant des éclairages, des terrains, des caméras, des textures ... . L'intérêt de ce logiciel est que celui-ci dispose d'une interface intuitive d'intégration d'objets et de scripts ; l'éditeur d'Unity intègre des composants préconfigurés évitant le développement de code fastidieux.

- QT est une interface de programmation orientée objet et développée en C++. Elle offre des composants d'interface graphique (widgets), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc. Par certains aspects c'est aussi un Framework lorsqu'on l'utilise pour concevoir des interfaces graphiques ou que l'on architecture son application en utilisant les mécanismes des "signaux" et "slots" par exemple.

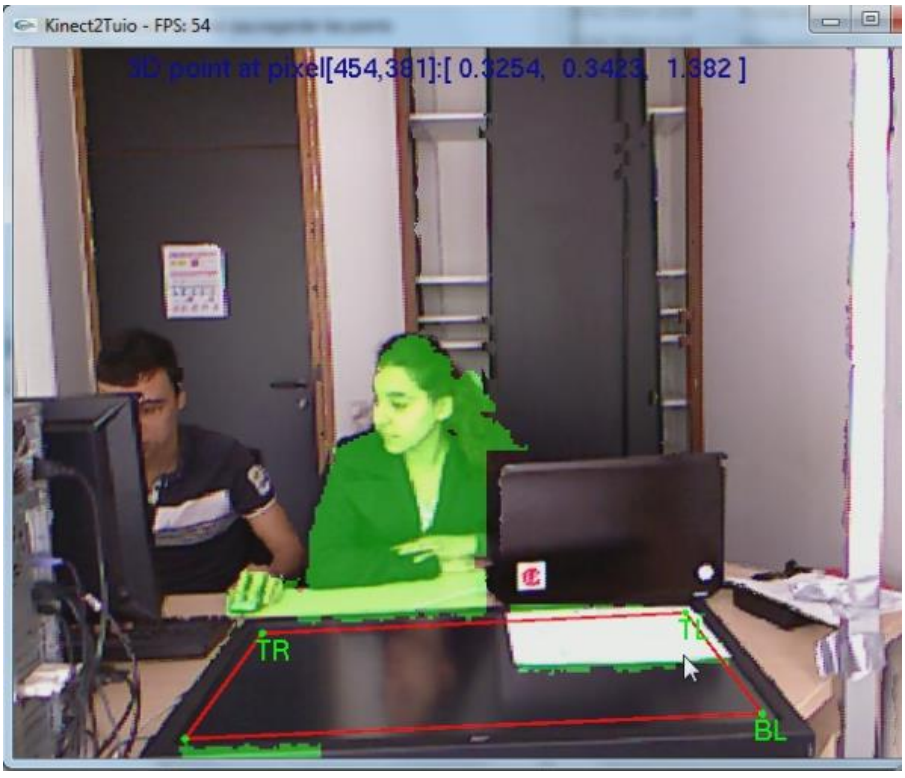
  QT est donc constituée d'un ensemble de bibliothèque, appelées « modules ». On peut y trouver entre autres ces fonctionnalités :
  - Module GUI : c'est toute la partie création de fenêtres. Nous nous concentrerons surtout, dans ce cours, sur le module GUI.
  - Module OpenGL : Qt peut ouvrir une fenêtre contenant de la 3D gérée par OpenGL.
  - Module de dessin : pour tous ceux qui voudraient dessiner dans leur fenêtre (en 2D), le module de dessin est très complet !
  - Module réseau : Qt fournit une batterie d'outils pour accéder au réseau, que ce soit pour créer un logiciel de Chat, un client FTP, un client Bittorent, un lecteur de flux RSS…
  - Module SVG : Qt permet de créer des images et animations vectorielles, à la manière de Flash.
  - Module de script : Qt prend en charge le Javascript (ou ECMAScript), que vous pouvez réutiliser dans vos applications pour ajouter des fonctionnalités, par exemple sous forme de plugins.

- Module XML : pour ceux qui connaissent le XML, c'est un moyen très pratique d'échanger des données à partir de fichiers structurés à l'aide de balises, comme le XHTML.
- Module SQL : permet d'accéder aux bases de données (MySQL, Oracle, PostgreSQL…).

II- Travail effectué

1- Présentation de l'outil :

Afin de réaliser un calibrage, il faut d'abord lancer les programmes d'installation de la librairie 3gear de la Kinect (nimble_calib et nimble_server, ces applications sont fournies avec le kit de la librairie) Il faut ensuite lancer l'application Kinect_MSFT_Calibration. Cette application a déjà été développée pour ce projet et qui consiste en l'affichage de l'image reçue par la Kinect afin de sélectionner dessus les points des 4 coins de l'écran stéréoscopique (l'écran étant visible sur l'image), l'application enregistre alors ces coordonnées dans un fichier texte, et donc, ces coordonnées doivent être repris et mis dans un autre fichier texte en effectuant quelques modifications dessus. Ce dernier est celui qui est utilisé par l'application d'affichage de la scène 3D sur l'écran 3D, et qui utilise ces points pour régler l'affichage.



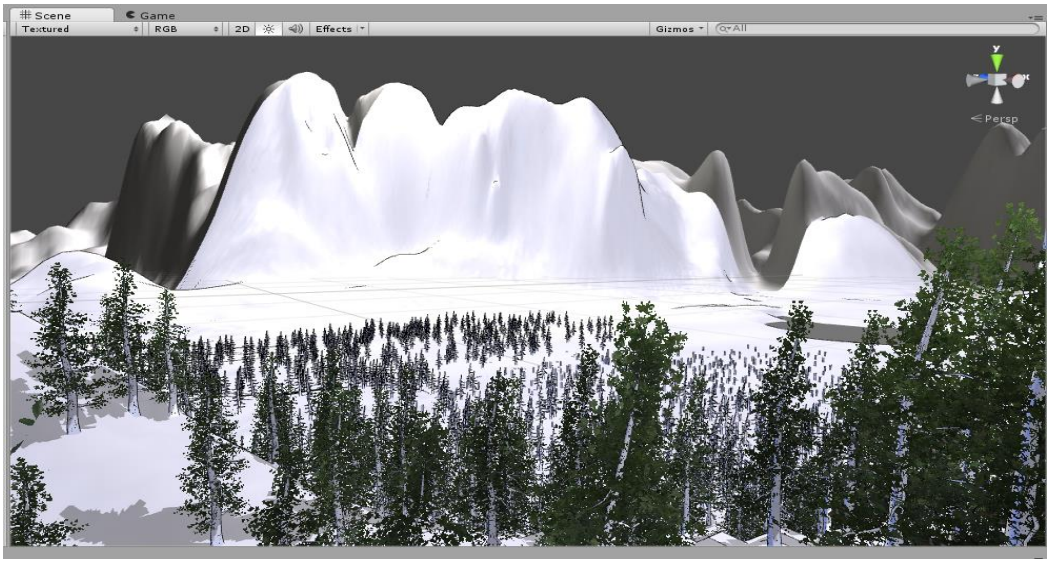
L'APPLICATION KINECT_MSFT_CALIBRATION

2- Travail préliminaire :

a. Réglage de la scène sous Unity :

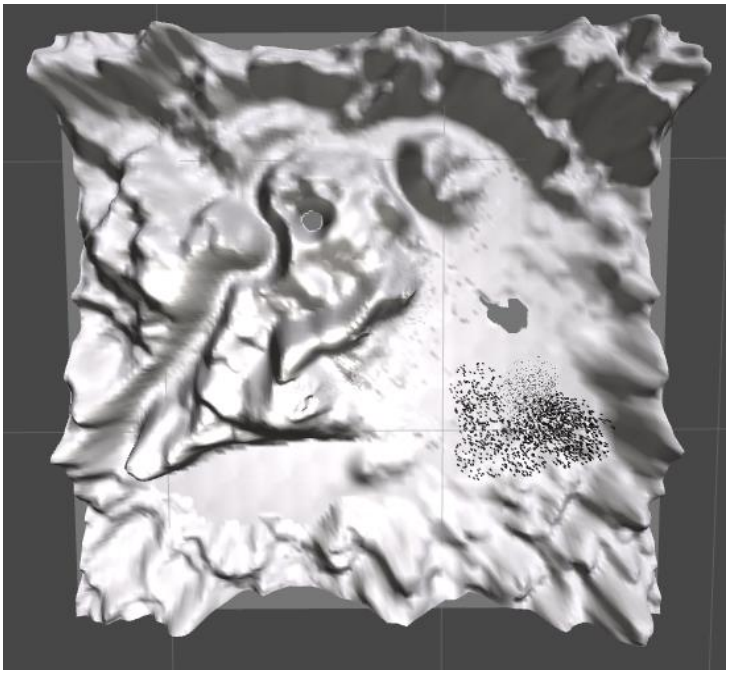
Au début, il fallait ajuster la scène à la taille de l'écran stéréoscopique sous Unity. Unity est un logiciel 3D temps réel et multimédia ainsi qu'un moteur 3D/2D et physique utilisé pour la création de jeux en réseau, d'animation en temps réel , de contenu interactif comportant de l'audio, de la vidéo et des objets 3D/2D.

La scène se présente comme suit :



La scène était perpendiculaire à l'écran stéréoscopique. Il fallait l'ajuster de façon à ce qu'on est un 'ground' (qui correspond à l'écran stéréoscopique) qui soit parallèle à la scène et aillant les mêmes dimensions de cette dernière afin qu'il n'y ait pas de dépassement de bord.

A travers plusieurs manipulations sous Unity, nous sommes arrivés au résultat suivant :



b- Mini-Programme de transfert des coordonnées:

Ce mini programme est développer en langage C, il sert principalement à réaliser le transfert des coordonnées enregistrés par l'application dans un premier fichier texte, et les placés dans un second fichier auquel le programme sur Unity y a accès.

Dans ce programme on utilise principalement la librairie « string.h » pour la gestion des chaines de caractères, il commence par ouvrir les deux fichiers textes puis récupère les 4 dernières lignes du premier fichier en utilisant des « fscanf » (les coordonnées qui nous intéressent se trouve sur ces 4 lignes là). Ensuite, comme ces coordonnées-là ne sont pas enregistrées correctement, on y effectue quelques modifications pour enfin les mettre sur le fichier source via des « fprintf ».

Le code de ce programme se retrouve en annexe.

c- Analyse de l'interface graphique:

L'application se présentera comme suit :

- Une première fenêtre pour l'introduction à l'outil et l'exécution des deux programmes nimble_calib et nimble_server
- La deuxième fenêtre servira au lancement l'application de calibration Kinect_MSFT_Calibration, avec un guide d'utilisation de celle-ci. Là, les coordonnées de la position de l'écran par rapport à la Kinect sont enregistrées par l'application sur un premier fichier texte.

- En troisième étape, les coordonnées seront transférées, du premier fichier texte au second, via le mini programme Transfert_Points.c que nous avons réalisé. Et qui affichera la fin de la calibration

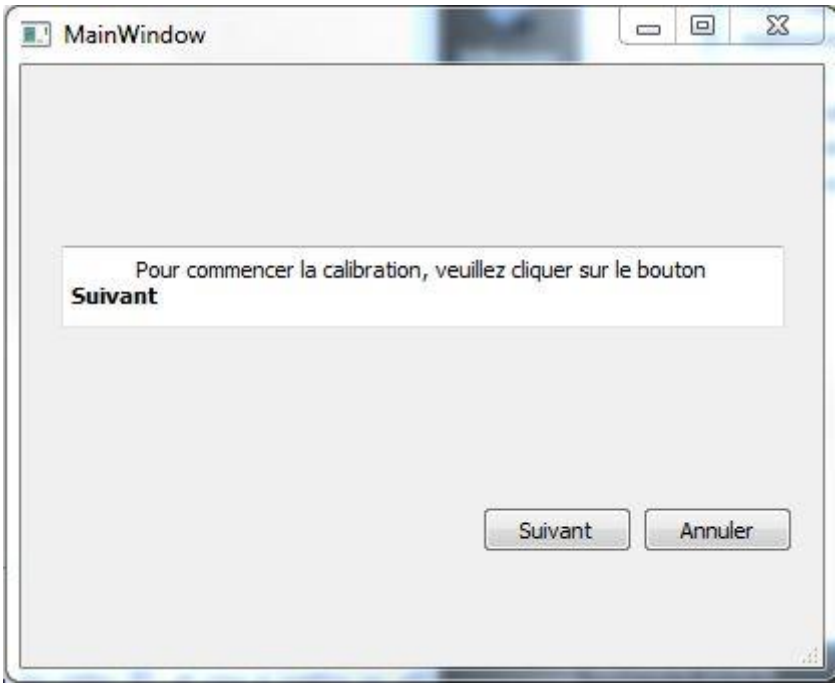
3- Développement de l'outil :

Dans cette partie, nous avons utilisé le logiciel de développement d'interface graphique QT qui est un API orientée objet et développé en C++ par QT development Frameworks. QT offre des composants d'interface graphiques (widgets), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc.

L'interface graphique de calibration comporte 3 fenêtres. On peut se déplacer d'une fenêtre à l'autre grâce à des boutons.

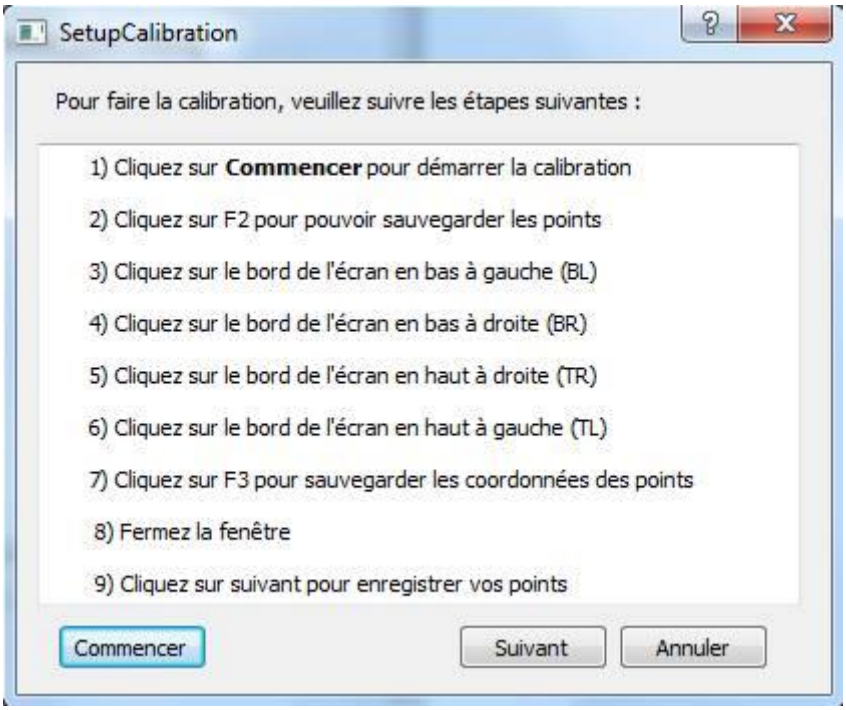
a- Première fenêtre:

La première fenêtre est la fenêtre typique de chaque installation, qui demande à l'utilisateur s'il veut calibrer sa Kinect ou bien c'est juste qu'il s'est trompé d'application et qu'il voudrait annuler.

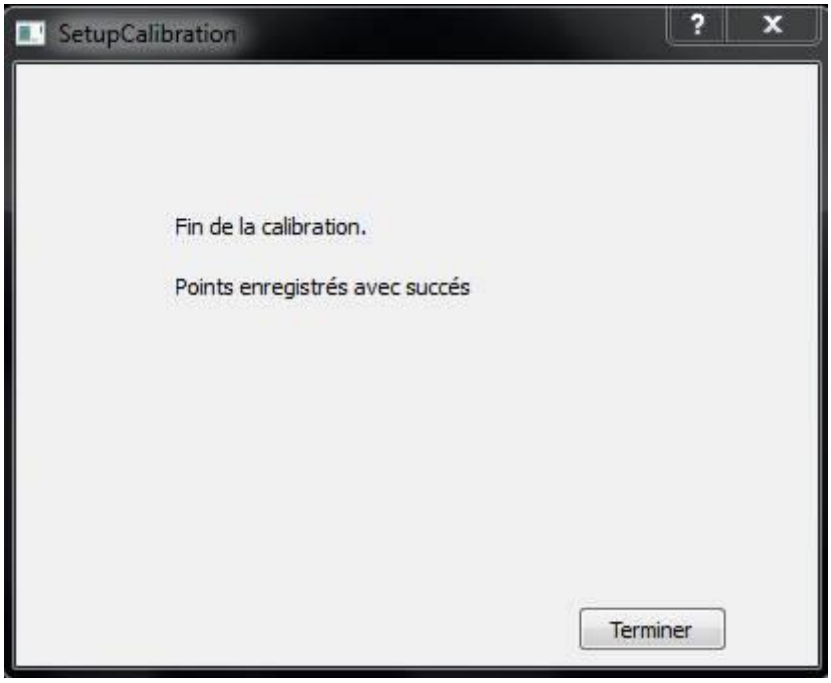


b- Deuxième fenêtre:

Le bouton ‘suivant’ permet de continuer l’installation et ainsi de passer à la deuxième fenêtre où on a listé les différentes étapes nécessaires pour la calibration. La calibration se fait via l’application ‘KinectMSFTCalibration’. Cette application permet de faire un calibrage de la Kinect de manière à définir Pour lancer cette dernière, l’utilisateur a besoin de cliquer sur le bouton ‘commencer’ qui permet d’afficher la vue de la Kinect qui est dirigé vers l’écran stéréoscopique. On pourra par la suite définir les points en cliquant sur F2 puis sur les bords de l’écran en commençant par le bas de l’écran à gauche puis à droite ensuite en haut à droite puis à gauche dans le sens antihoraire. Quand on place les 4 points au bord de l’écran, on définit le plan de définition. F3 nous permet de sauvegarder les coordonnées des points saisie dans un fichier texte. Quand on clique sur le bouton 'suivant', on exécute un programme qui nous permet de copier ces coordonnées vers un notre fichier texte. Pour pouvoir revenir à la fenêtre calibsetup, il suffit de fermer l’application.



c- Troisième fenêtre:



En ce qui concerne la 3éme fenêtre, elle permet juste de dire à l'utilisateur que la calibration a été bien faite et que les points ont bien été enregistrés dans le fichier texte

Pour le bouton 'annuler' qui est présent dans toutes les fenêtres, quand on appui dessus, il demande à l'utilisateur s'il veut vraiment quitter la calibration en lui proposant deux boutons « Yes » et « No ». Le bouton « no » permet de revenir à la fenêtre précédente et le bouton « Yes » permet de quitter l'application.

Maintenant je vais vous parler de la fonction MAIN du programme qui est la première fonction appelée par le système d'exploitation lors de l'ouverture d'une application. Elle est la fonction marquant le début de chaque programme, en appelant d'autre ou non. Elle est constituée comme toute fonction d'un en-tête et d'un bloc pouvant en contenir d'autres.

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Je vais essayer dans la suite d'expliquer quelques lignes de code.

#include <QDialog> est la classe de base des boîtes de dialogue. Une boîte de dialogue est une fenêtre de premier niveau souvent utilisée pour les tâches de court terme et les

communications brèves et avec l'utilisateur. Les QDialog peuvent être modales ou non modales. Elles peuvent fournir une valeur de retour et peuvent aussi posséder un bouton par défaut.

void on_pushButton_clicked(); Le bouton poussoir « on_pushButton » est peut-être le widget le plus couramment utilisé dans une interface graphique. Il permet, en cliquant dessus, d'effectuer une action, ou de répondre à une question. Les boutons typiques sont: Ok, Appliquer, Annuler, Fermer, Oui, Non et Aide. Il est normalement de bouton rectangulaire et affiche généralement un texte décrivant son action.

system("C:/Users/Demo.ALCOVE-HP-MINT4/Desktop/OOS/tool/KinectMSFTCalibration.exe"); cette ligne permet d'ouvrir une application ou autre fichier quand on clique sur un bouton.

this->hide(); permet de fermer la fenêtre sans la détruire.

Conclusion

Ce projet, malgré la mauvaise répartition du temps qu'on a eu, et la définition des tâches assez tardive, nous a permis de découvrir de nouveaux outils tels que Unity et QT, et nous a permis de nous familiariser plus avec la programmation C++, et le développement d'applications graphiques.

Cependant, nous avons rencontré plusieurs difficultés lors de notre projet que nous pouvons résumer comme suit :

- L'installation de 3Gear vu que ce n'était pas le matériel que les IMA5 avait utilisé lors de leur projet. On avait du mal à installer 3Gear Nimble SDK parce qu'il nécessitait un ordinateur 64 bits et qui n'était pas à notre disposition.

- On n'arrivait pas à ouvrir les 2 applications de calibrage lorsqu'on cliquait sur le bouton 'commencer' dans la deuxième fenêtre. Au début, on croyait que c'était un problème de code, du coup on l'avait changé plusieurs fois avant de savoir que c'était des fichiers qui manquait de notre fichier source.

- Pour la définition des coordonnées des points de calibration, on n'arrivait pas à les prendre sans mettre une feuille blanche au-dessus de l'écran stéréoscopique pour la détection. Du coup on devait la déplacer autour de l'écran pour chaque point.

Annexes :

1- Annexe1 :

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main ( )
{
        FILE * f1;
        FILE * f2;
        //Opening files
        f1=fopen("tool/KinectDisplay.txt","r");
        if(f1==NULL) {printf("Impossible d'ouvrir le fichier source\n"); exit(1);}
        f2=fopen("Unity/DisplayPoints.txt","w");
        if(f2==NULL) {printf("Impossible d'ouvrir le fichier destination\n"); exit(1);}

        //Récupération des 4 dernières lignes
        char C;
        int i,nb_lignes=0;
        float n1,n2,n3;
         while( !feof(f1) )
         {
           if(nb_lignes<4)
                {
                  nb_lignes++;
                  do
                    { fscanf(f1,"%c",&C); }
                  while(C!='\n');
        }
```

```
    else
        {
         i=fscanf(f1,"%f %f %f %f",&n1,&n2,&n3);
         n1=-n1;
         if(i>0) fprintf(f2,"%f %f %f %f\n",n1,n2,n3);

        }

  }

 //Closing files
 if(fclose(f1) == EOF){ printf("Probleme de fermeture du fichier source");  exit(1);}
 if(fclose(f2) == EOF){ printf("Probleme de fermeture du fichier dest");  exit(1);}

return 0;
}
```

2- Annexe2

MainWindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QApplication>
#include <QProcess>
#include <QtGui>
#include "calibsetup.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
```

```
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();

private slots:

    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

private:
    Ui::MainWindow *ui;
    calib *calibsetup;
};



#endif // MAINWINDOW_H
```

MainWindow.cpp :

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "calibsetup.h"

#include <QFileDialog>
#include <QMessageBox>
#include <QDesktopServices>
#include <QUrl>
#include <QApplication>
#include <QDebug>
#include <QObject>



MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    calibsetup = new calib;
    calibsetup->show();
    this->hide();

}
```

```
void MainWindow::on_pushButton_2_clicked()
{
   //QMessageBox::information(this,"QUIT","Voulez-vous quitter
l'application ?","Non","Oui");
    QMessageBox msgBox;
      msgBox.setWindowTitle("Exit setup");
      msgBox.addButton(QMessageBox::Yes);
      msgBox.addButton(QMessageBox::No);
      msgBox.setText("Voulez-vous vraiment quitter la calibration ?");

      int selection = msgBox.exec();
      if(selection == QMessageBox::Yes)
      {
        QCoreApplication::quit();
      }
}
```

3- Annexe3

CalibSetup.h

```
#ifndef CALIB_H
#define CALIB_H

#include <QDialog>
#include "fincalibration.h"

namespace Ui {
class calib;
}

class calib : public QDialog
{
    Q_OBJECT

public:
    explicit calib(QWidget *parent = 0);
    ~calib();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_pushButton_3_clicked();

private:
    Ui::calib *ui;
    FinCalibration *fincalib ;
};

#endif // CALIB_H
```

Calibsetup.cpp :

```
#include "calibsetup.h"
#include "ui_calibsetup.h"
#include "fincalibration.h"
#include <QMessageBox>

calib::calib(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::calib)
{
    ui->setupUi(this);
}
calib::~calib()
{
    delete ui;
}

void calib::on_pushButton_clicked()
{
    system("C:/Users/Demo.ALCOVE-HP-MINT4/Desktop/OOS/nimble_calib.bat");
    system("C:/Users/Demo.ALCOVE-HP-MINT4/Desktop/OOS/nimble_server.bat");
    system("C:/Users/Demo.ALCOVE-HP-
MINT4/Desktop/OOS/tool/KinectMSFTCalibration.exe");
}

void calib::on_pushButton_2_clicked()
{
        fincalib = new FinCalibration;
        fincalib->show();
        this->hide();
        system("C:/Users/Demo.ALCOVE-HP-
MINT4/Desktop/OOS/Transfert_Points.exe")
;}

void calib::on_pushButton_3_clicked()
{
    QMessageBox msgBox;
      msgBox.setWindowTitle("Calibration setup");
      msgBox.addButton(QMessageBox::Yes);
      msgBox.addButton(QMessageBox::No);
      msgBox.setText("Voulez-vous vraiment quitter la calibration ?");

      int selection = msgBox.exec();
      if(selection == QMessageBox::Yes)
      {
        QCoreApplication::quit();
      }
}
```

4- Annexe4

Fincalibration.h

```
#ifndef FINCALIBRATION_H
#define FINCALIBRATION_H

#include <QDialog>
```

```
namespace Ui {
class FinCalibration;
}

class FinCalibration : public QDialog
{
    Q_OBJECT

public:
    explicit FinCalibration(QWidget *parent = 0);
    ~FinCalibration();

private slots:
    void on_pushButton_clicked();

private:
    Ui::FinCalibration *ui;

};

#endif // FINCALIBRATION_H
```

Fincalibration.cpp

```
#include "fincalibration.h"
#include "ui_fincalibration.h"

#include <QMessageBox>

FinCalibration::FinCalibration(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::FinCalibration)
{
    ui->setupUi(this);
}

FinCalibration::~FinCalibration()
{
    delete ui;
}

void FinCalibration::on_pushButton_clicked()
{
      QCoreApplication::quit();

}
```