# ETUDES ET REALISATION

Remplacer les commandes physiques par des commandes virtuelles

2013/2014
PORTO/REVIRON
Utilisation de Kinect for Windows

# Nos objectifs

- Capturer une vidéo avec la Kinect
- Récupérer les points du squelette
- Capter un mouvement
- Lui attribuer une action
- Utiliser cette action avec une application type PowerPoint

# Table des matières



# Introduction

Nous souhaitons réaliser un système permettant de contrôler seulement avec son corps des fonctions du logiciel PowerPoint. Pour cela, nous utilisons la caméra perfectionnée « Kinect for Windows » de la société *Microsoft* qui permet d'accéder à des informations sur notre corps qu'aucune autre caméra ne peut nous livrer. Nous allons donc expliquer dans ce rapport comment accéder aux fonctionnalités de cette caméra, des plus simples à celle que nous souhaitons exploiter, et comment les intégrer dans un logiciel comme PowerPoint.

# 1 .Connecter Kinect

Pour utiliser la Kinect Windows sur un PC, nous devons:

- Installer le SDK v1.8
- Installer Visual Studio 2013
- Créer un projet WPF
- Inclure la bibliothèque « Microsoft.Kinect.dll » dans notre programme
- Utiliser les méthodes et attributs de la bibliothèque

# 2. Récupérer Informations Kinect

Pour réaliser un programme afin de récupérer les informations principales que nous délivre Kinect, nos premières pensées sont de récupérer des exemples de code sur internet. Or l'accès à des codes existants est difficile, car la bibliothèque que nous utilisons est récente et celle-ci contient de nombreux changements par rapport aux anciennes versions des SDK précédents.

Nous devons donc traduire les codes récupérés en fonction des nouvelles méthodes et attribues de la dernière bibliothèque.

Ces programmes sont écrits en langage C#, c'est pourquoi nous utiliserons ce langage dans nos futurs réalisations.

## 2.1) Initialiser Kinect

```
KinectSensor nui;

private void setupKinect()
 {
     if (KinectSensor.KinectSensors.Count == 0)
     {
         this.Title = "No Kinect Detected";
     }

     else
     {
         nui = KinectSensor.KinectSensors[0];
         nui.Start();
     }
 }
```

La fonction "setupKinect()" nous sert à regarder si la Kinect est bien branchée et si elle nous renvois quelque chose. Si on reçoit bien ces informations, alors on peut l'initialiser (« nui.start() ») et écrire à la suite ce que l'on veut récupérer.

## 2.2) Récupérer vidéo couleur

```
<Window Title="MainWindow" Height="480" Width="640">
    <Image x:Name="ImgKinect" HorizontalAlignment="Left" Height="440" Margin="-
4,4,0,0" VerticalAlignment="Top" Width="630"/>

</Window>
```

Pour afficher une vidéo couleur, nous devons tout d'abord, dans le fichier .xaml de notre programme (fichier contenant tous les éléments de notre fenêtre), créer une image, qui servira à stocker les données récupérées, et rafraichir celle-ci en continu afin de créer une vidéo.

Dans notre exemple, notre image porte le nom de « ImgKinect » et est très légèrement plus petite que notre fenêtre de 480*640 pixels.

```
else
            {
                nui = KinectSensor.KinectSensors[0];
                nui.Start();//Initialisation de la Kinect
                nui.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
                nui.ColorFrameReady += new     //Appel de l'événement ColorImageReady
EventHandler<ColorImageFrameReadyEventArgs>(ColorImageReady);

            }
```

Pour récupérer l'image de la caméra couleur de Kinect, nous devons appeler la méthode « ColorStream.Enable » de « nui » qui est un élément KinectSensor, afin d'indiquer à la caméra kinect que nous voulons utiliser la vidéo couleur.

Nous devons donc, en paramètre, sélectionner le format de l'image qui est ici en 640*480.

La dernière ligne permet d'appeler un évènement. L'évènement est ici « ColorImageReady », de type « ColorImageFrameReadyEventArgs », fonction que nous allons créer dans la suite du programme :

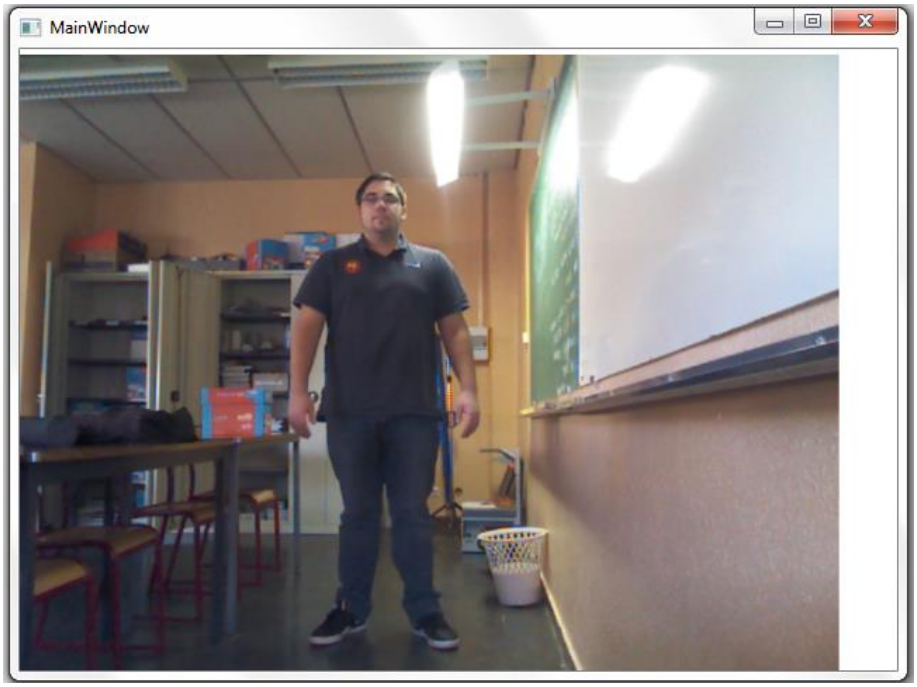
```
byte[] pixelData;

void ColorImageReady(object sender, ColorImageFrameReadyEventArgs e)
        {
            bool receivedData = false; //booléen pour vérifié que données bien reçu

            using (ColorImageFrame colorImageFrame = e.OpenColorImageFrame())
            {
                if (colorImageFrame != null) //Si le cadre reçoit des informations
                                             //alors :
                {
                    if (pixelData == null) //On initialise notre octet pixelData
                    {
                        pixelData = new byte[colorImageFrame.PixelDataLength];
                    }
                    colorImageFrame.CopyPixelDataTo(pixelData);
                    receivedData = true;
                }
```

```
                if (receivedData) //Vérification que données reçues
                {

                    ImageSource imgSrc = BitmapSource.Create(colorImageFrame.Width,
colorImageFrame.Height, 96, 96, PixelFormats.Bgr32, null, pixelData,
colorImageFrame.Width * colorImageFrame.BytesPerPixel);
                    ImgKinect.Source = imgSrc;


                }
            }

        }
```

Les éléments que nous avons créés sont :

- **pixelData** : un octet dans lequel nous allons stocker les données reçues par la caméra couleur.
- **e** : un argument de l'évènement nous signalant qu'une trame d'image est prête.
- **colorImageFrame** : une trame couleur à laquelle on affecte celle de l'argument « **e** ».
- **receivedData** : un simple booléen servant à nous indiquer si l'on a reçu des données et si on peut les envoyer dans notre image (ImgKinect).
- **imgSrc** : nous devons créer une source d'image Bitmap pour ImgKinect et lui indiquer tous ces paramètres. C'est à celle-ci que nous affectons notre octet pixelData.

**Résultat :**



## 2.3) Récupérer points du squelette

```
<Canvas Name="skeletonCanvas" Width="480" Height="480" HorizontalAlignment="Left"
Margin="250,-40,0,434">
        <Ellipse Name="rightHandEllipse" Width="20" Height="20" Fill="Red"/>
        <Ellipse Name="headEllipse" Width="20" Height="20" Fill="Red"/>
        <Ellipse Name="rightShoulderEllipse" Width="20" Height="20" Fill="Red"/>
        <Ellipse Name="centerShoulderEllipse" Width="20" Height="20" Fill="Red"/>
      […]

    </Canvas>
```

Nous devons créer dans notre .xaml, pour cet exemple, un canvas,  c'est-à-dire une toile.
Cette toile va nous servir à y placer des ellipses, qui représenteront chacun des points de notre corps.

```
else
            {
                nui = KinectSensor.KinectSensors[0];
                nui.Start();
                nui.SkeletonStream.Enable(new TransformSmoothParameters()
                    {
                        Smoothing = 0.5f,
                        Correction = 0.5f,
                        Prediction = 0.5f,
                        JitterRadius = 0.05f,
                        MaxDeviationRadius = 0.04f
                    });
                nui.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(nui_SkeletonFrameReady);


            }
```

De la même manière que pour la vidéo couleur, nous devons indiquer à la caméra Kinect que nous voulons utiliser sa fonction de récupération du squelette. A la place du format d'image, nous devons lui indiquer ici des paramètres de diffusion qui vont nous permettre de l'affiner, comme le temps de réponse par exemple (facultatif).

Nous devons encore également appeler un évènement dans notre fonction setupKinect() , appelé ici « nui_SkeletonFrameReady ».

```
void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
            using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
                if (skeletonFrame != null)
                {
                    Skeleton[] skeletonData = new
Skeleton[skeletonFrame.SkeletonArrayLength];
                    skeletonFrame.CopySkeletonDataTo(skeletonData);
                    Skeleton playerSkeleton = (from s in skeletonData where
s.TrackingState == SkeletonTrackingState.Tracked select s).FirstOrDefault();









 if (playerSkeleton != null) //Si l'on perçoit un squelette
                    {
ProjectAndMoveEllipse(headEllipse, playerSkeleton.Joints[JointType.Head]);
ProjectAndMoveEllipse(rightWristEllipse, playerSkeleton.Joints[JointType.WristRight]);
ProjectAndMoveEllipse(leftWristEllipse, playerSkeleton.Joints[JointType.WristLeft]);
[…]

                    }

                }
            }

            }
```

Cet évènement comprend le même type d'éléments que dans l'exemple précédent, mais en remplaçant ici « ColorImage » par « Skeleton ».

Le tableau d'octet pixelData est remplacé par un tableau de type Skeleton (skeletonData), qui va nous permettre d'affecter ses données à notre squelette « playerSkeleton ».

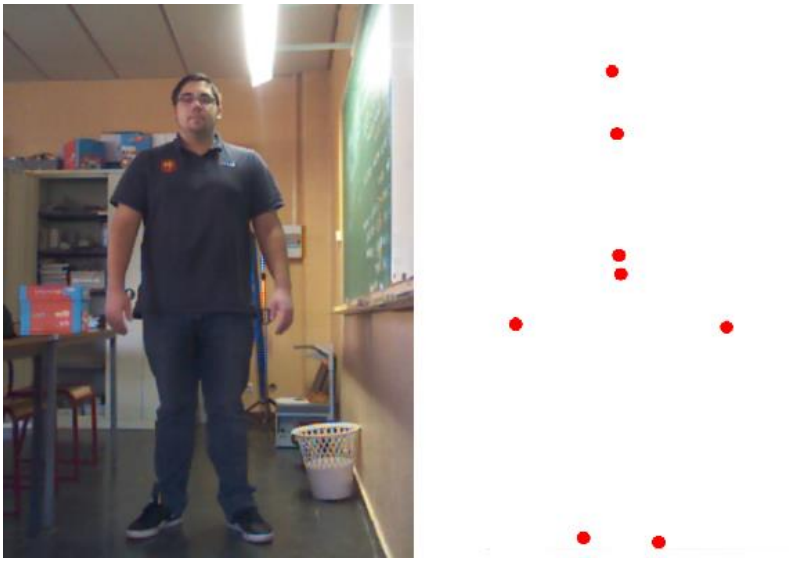
Si ce dernier n'est pas nul, que l'on perçoit un squelette, alors on appelle une fonction « ProjectAndMoveEllipse() » que nous allons créer :

```
private void ProjectAndMoveEllipse(UIElement ellipse, Joint joint)
        {
            Vector vecteurPosition = new Vector(joint.Position.X, joint.Position.Y);


            // On place l'ellipse avec les coordonnées projetées trouvées
            Canvas.SetLeft(ellipse, vecteurPosition.X * skeletonCanvas.Width);
            Canvas.SetBottom(ellipse, vecteurPosition.Y * skeletonCanvas.Height);
        }
```

Un élément de type Joint nous renvoie une position entre -1 et 1 de points de notre corps sur 3 axes (X, Y, Z).
Dans cette fonction, nous allons récupérer les coordonnées, sur seulement 2 axes, de ces « Joints » et les affecter à un vecteur (vecteurPosition).

Grâce aux fonctions SetLeft() et SetBottom() de notre Canvas, nous affectons les coordonnées de notre vecteur à l'ellipse choisit afin qu'elle représente la position du point de notre corps voulu. On multiplie ces coordonnées par la largeur et la hauteur de notre toile pour que la position de notre ellipse soit proportionnelle à la taille de notre fenêtre.

**Résultat :**



## 2.4) Détecter un mouvement

```
if (playerSkeleton != null) //Si l'on perçoit un squelette
                    {
ProjectAndMoveEllipse(headEllipse, playerSkeleton.Joints[JointType.Head])
[…]

  GestionDesMouvements(playerSkeleton.Joints[JointType.HandRight],
playerSkeleton.Joints[JointType.HipLeft]);
```

Pour détecter un mouvement, nous devons créer une nouvelle fonction, nommée ici « GestionDesMouvements », que nous allons lancer dans l'événement « nui_SkeletonFrameReady ».

Cette fonction prend en paramètres notre main droite, ainsi que notre anche gauche.

```
int compt;

        private void GestionDesMouvements (Joint handRight, Joint hipLeft)
        {

            if(handRight.Position.X < hipLeft.Position.X)
            {
                compt++;
            }

            if (compt == 10)
            {
                System.Windows.MessageBox.Show("Changement de diapo");
                compt = 0;
            }
        }
```

Nous allons inscrire essentiellement dans cette fonction les conditions nous permettant de savoir où se trouvent nos points du corps les uns par rapport aux autres.

La première ligne : « if(handRight.Position.X < hipLeft.Position.X) » indique seulement que nous pouvons lancer l'action si notre main droite est à la gauche de notre anche gauche.

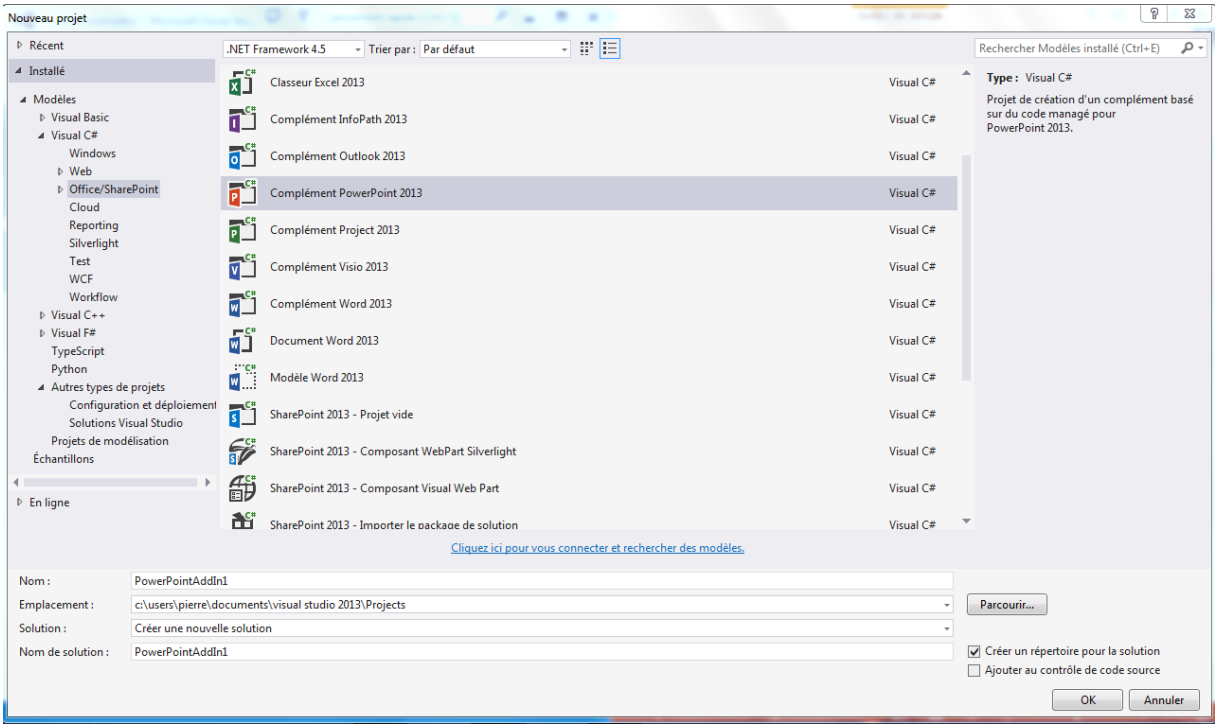
L'entier « compt » que nous avons déclaré juste avant la fonction nous sert simplement de tempo, afin que la Boite de dialogue (MessageBox) ne s'exécute pas plusieurs fois  lors d'un seul mouvement.

Maintenant que nous savons comment lancer une action à la suite d'une détection de mouvement, nous pouvons appliquer ce procédé pour n'importe quelle action. C'est ce que nous allons donc faire pour utiliser PowerPoint avec notre corps.

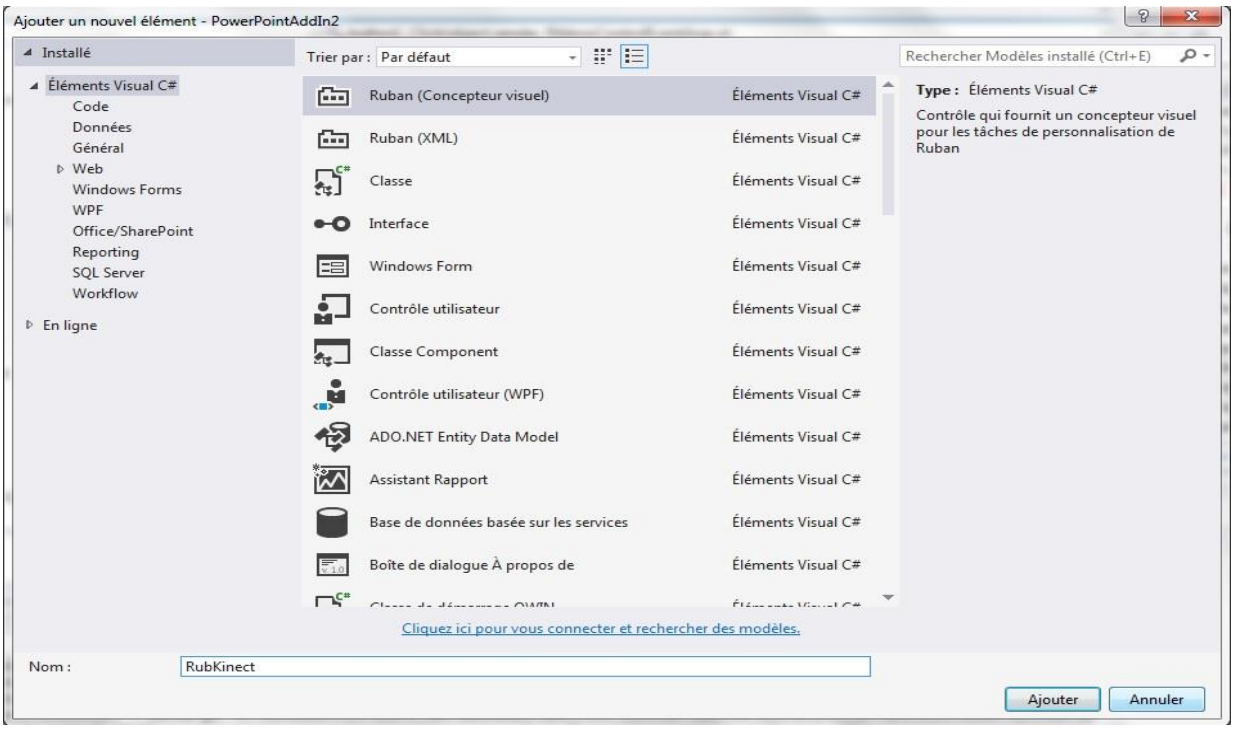
# 3. Programmer un plug-in pour logiciel de type PowerPoint

## 3.1) Créer le programme

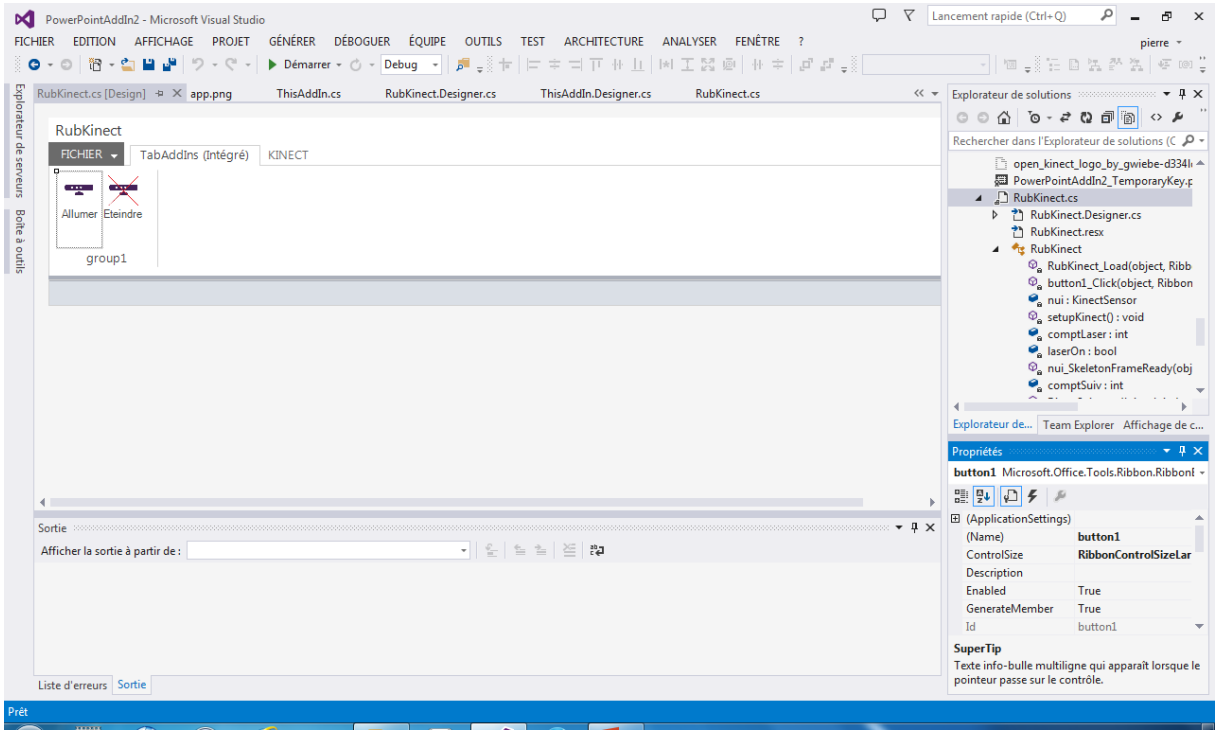
- Créer un projet complément pour Office,
  plus précisément complément pour PowerPoint 2013



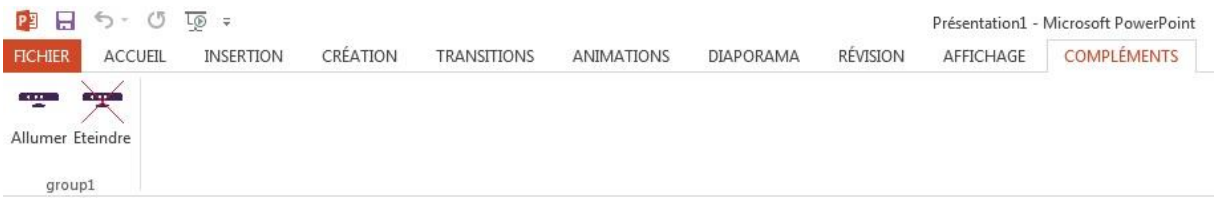

- Ajouter un ruban : PROJET/Ajouter un composant/Ruban (Concepteur Visuel)



- Ajouter des boutons dans « group1 »




- Résultat :




## 3.2) Introduire le programme précédent

Pour plus de facilité, nous allons introduire toutes nos fonctions dans le fichier RubKinect.cs, qui est le fichier de création du nouveau ruban, au lieu du programme principal ThisAddIn.cs, car ces fonctions vont être exécutés à partir des boutons du ruban.

```
private void button1_Click(object sender, RibbonControlEventArgs e)
        {
            setupKinect();
        }
```

Nous devons simplement créer une fonction « Bouton1_Click » dans laquelle nous introduisons notre fonction « setupKinect() » .

Toutes nos actions vont encore se retrouver dans notre événement « nui_SkeletonFrameReady » dans la partie « if (playerSkeleton != null) », mais nous n'avons plus besoin des précédentes (ProjectAndMoveEllipse et GestionDesMouvements). Nous allons donc créer de nouvelles fonctions pour chaque mouvement.

```
if (playerSkeleton != null)
                    {
                        DiapoSuivante(playerSkeleton.Joints[JointType.HandRight],
playerSkeleton.Joints[JointType.HipLeft]);
                        DiapoPrécédente(playerSkeleton.Joints[JointType.HandLeft],
playerSkeleton.Joints[JointType.HipRight]);
                        LancerDiapo(playerSkeleton.Joints[JointType.HandRight],
playerSkeleton.Joints[JointType.Head]);
                        QuitterDiapo(playerSkeleton.Joints[JointType.HandLeft],
playerSkeleton.Joints[JointType.KneeLeft]);
                        ActiverLaser(playerSkeleton.Joints[JointType.HandLeft],
playerSkeleton.Joints[JointType.ShoulderLeft]);
```

A chaque fonction est attribué ses propres paramètres (les points du corps) et ses propres actions à exécuter.
Prenons l'exemple de la fonction « DiapoSuivante » qui prend en paramètre notre main droite ainsi que notre hanche gauche :

```
private void DiapoSuivante(Joint handRight, Joint hipLeft)
        {
            if (handRight.Position.X < hipLeft.Position.X)
            {

                comptSuiv++;
            }

            if (comptSuiv == 20)
            {
                System.Windows.Forms.SendKeys.SendWait("{Right}");
                comptSuiv = 0;
            }
        }
```

Les conditions sont exactement les mêmes que dans notre ancienne fonction « GestionDesMouvements », nous allons seulement remplacer l'ouverture de la fenêtre dans laquelle nous avions inscrit « Changement de Diapo » par un réelle changement de diapositive.
Le moyen le plus simple que nous avons trouvé pour réaliser cette action est d'envoyer à l'ordinateur l'information d'un appuie sur la touche « Droite ». Nous utilisons donc le code suivant :

```
System.Windows.Forms.SendKeys.SendWait("{Right}");
```

Nous utiliserons donc le même principe pour toutes les actions que nous voulons provoquer dans PowerPoint, nous remplacerons toutes les commandes du clavier par le geste que l'on voudra faire.

La seule action que nous n'avons pas écrite dans une fonction séparée de l'événement principal et qui est légèrement différente des autres est la fonction qui nous permet de remplacer la souris par notre main droite. Cette fonction nous provient d'une bibliothèque téléchargée sur internet, et nous devons l'exécuter en continue pour qu'elle enregistre le moindre déplacement de notre main à chaque instant, c'est pourquoi nous devons déclarer un booléen et lancer la fonction lorsque le booléen est activé :

```
      if (playerSkeleton.Joints[JointType.HandLeft].Position.Y >
playerSkeleton.Joints[JointType.ShoulderLeft].Position.Y)
                        {
if(laserOn==false)laserOn = true;
else if (laserOn==true)laserOn = false;
                        }

                        if(laserOn==true)
                        {
Joint scaledRight =
playerSkeleton.Joints[JointType.HandRight].ScaleTo((int)SystemInformation.PrimaryMonit
orSize.Width, (int)SystemInformation.PrimaryMonitorSize.Height, -
playerSkeleton.Position.X, -playerSkeleton.Position.Y);
int CursorX = (int)scaledRight.Position.X;
int CursorY = (int)scaledRight.Position.Y;

KinectMouseController.KinectMouseMethods.SendMouseInput(CursorX, CursorY,
SystemInformation.PrimaryMonitorSize.Width,
SystemInformation.PrimaryMonitorSize.Height, false);

                        }
```

Cette fonction consiste à récupérer les coordonnées X et Y de notre main droite et de les introduite dans la méthode « SendMouseInput » qui prend en paramètre les coordonnées d'un nouveau curseur et la largeur et la hauteur de l'écran.

Les autres fonctions sont inscrites en Annexe.

## CONCLUSION

A ce stade de notre projet, nous avons réussi à comprendre comment fonctionne la caméra Kinect, et comment utiliser la bibliothèque concernant le squelette. Nous ne maitrisons pas encore l'utilisation du capteur de profondeur, mais pour détecter un mouvement ce que nous avons utilisé est déjà suffisant. Cette bibliothèque comprend de grandes possibilités de développement et maintenant que nous savons l'utiliser, nous pourrons nous en servir de nouveau dans d'autres projets.

Nous aurions pu ajouter plus de mouvements attribués à autant plus d'actions, ainsi qu'un menu permettant à l'utilisateur de choisir quelles sont les mouvements qu'il souhaite attribuer à chaque action. Ceci serait légèrement plus complexe à réaliser mais réellement intéressant pour l'utilisateur du logiciel.

## Annexe Couleur

```
using Microsoft.Kinect;

namespace WpfApplication1
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            setupKinect();
        }

        KinectSensor nui;

        private void setupKinect()
        {
            if (KinectSensor.KinectSensors.Count == 0)
            {
                this.Title = "No Kinect Detected";

            }
            else
            {
                nui = KinectSensor.KinectSensors[0];
                nui.Start();
                nui.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
                nui.ColorFrameReady += new
EventHandler<ColorImageFrameReadyEventArgs>(ColorImageReady);
            }
        }

        byte[] pixelData;

        void ColorImageReady(object sender, ColorImageFrameReadyEventArgs e)
        {
            bool receivedData = false;
            using (ColorImageFrame colorImageFrame = e.OpenColorImageFrame())
            {
                if (colorImageFrame != null)
                {
                    if (pixelData == null)                    {
                        pixelData = new byte[colorImageFrame.PixelDataLength];
                    }
                    colorImageFrame.CopyPixelDataTo(pixelData);
                    receivedData = true;
                }

                if (receivedData)
                {
                    ImageSource imgSrc = BitmapSource.Create(colorImageFrame.Width,
colorImageFrame.Height, 96, 96, PixelFormats.Bgr32, null, pixelData,
colorImageFrame.Width * colorImageFrame.BytesPerPixel);
                    ImgKinect.Source = imgSrc;

                }
            }

        }
    }
}
```

## Annexe Squelette

```
namespace WpfApplication3
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            setupKinect();
        }
        KinectSensor nui;

        private void setupKinect()
        {
            if (KinectSensor.KinectSensors.Count == 0)
            {
                this.Title = "No Kinect Detected";

            }

            else
            {
                nui = KinectSensor.KinectSensors[0];
                nui.Start();
                nui.SkeletonStream.Enable(new TransformSmoothParameters()
                    {
                        Smoothing = 0.5f,
                        Correction = 0.5f,
                        Prediction = 0.5f,
                        JitterRadius = 0.05f,
                        MaxDeviationRadius = 0.04f
                    });
                nui.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(nui_SkeletonFrameReady);
            }
        }

        private void ProjectAndMoveEllipse(UIElement ellipse, Joint joint)
        {
            Vector vecteurPosition = new Vector(joint.Position.X, joint.Position.Y);
            Canvas.SetLeft(ellipse, vecteurPosition.X * skeletonCanvas.Width);
            Canvas.SetBottom(ellipse, vecteurPosition.Y * skeletonCanvas.Height);
        }


        void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {

            using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {

                if (skeletonFrame != null)
                {
                    Skeleton[] skeletonData = new
Skeleton[skeletonFrame.SkeletonArrayLength];
                    skeletonFrame.CopySkeletonDataTo(skeletonData);
                    Skeleton playerSkeleton = (from s in skeletonData where
s.TrackingState == SkeletonTrackingState.Tracked select s).FirstOrDefault();

                    if (playerSkeleton != null)
                    {
```

```
                        ProjectAndMoveEllipse(headEllipse,
playerSkeleton.Joints[JointType.Head]);
                        ProjectAndMoveEllipse(rightWristEllipse,
playerSkeleton.Joints[JointType.WristRight]);
                        ProjectAndMoveEllipse(leftWristEllipse,
playerSkeleton.Joints[JointType.WristLeft]);
                        ProjectAndMoveEllipse(centerShoulderEllipse,
playerSkeleton.Joints[JointType.ShoulderCenter]);
                        ProjectAndMoveEllipse(spineEllipse,
playerSkeleton.Joints[JointType.Spine]);
                        ProjectAndMoveEllipse(centerHip,
playerSkeleton.Joints[JointType.HipCenter]);
                        ProjectAndMoveEllipse(rightAnkleEllipse,
playerSkeleton.Joints[JointType.AnkleRight]);
                        ProjectAndMoveEllipse(leftAnkleEllipse,
playerSkeleton.Joints[JointType.AnkleLeft]);

                    }
                }



            }
        }
    }
}
```

## Annexe Plug-In PowerPoint

fichier RubKinect.cs

```
using Microsoft.Kinect;
using KinectMouseController;
using Coding4Fun.Kinect.Wpf;

namespace PowerPointAddIn2
{
public partial class RubKinect
{
private void RubKinect_Load(object sender, RibbonUIEventArgs e){}

private void button1_Click(object sender, RibbonControlEventArgs e)
{
    setupKinect();
}

KinectSensor nui;            //Déclaration d'un élément Capteur

private void setupKinect()
{
    if (KinectSensor.KinectSensors.Count == 0){}
                                    //Si non détection de la caméra
                                    //Alors Rien

    else
    {
    nui = KinectSensor.KinectSensors[0];
                                    //Attribution du premier capteur détecté
    nui.Start();
                                    //Démarrer le capteur kinect
    nui.SkeletonStream.Enable(new TransformSmoothParameters()
    {
                                    //Réglage des paramètres de la caméra
       Smoothing = 0.5f,            //Pour une meilleure fluidité
       Correction = 0.5f,
       Prediction = 0.5f,
       JitterRadius = 0.05f,
       MaxDeviationRadius = 0.04f
     });
       nui.SkeletonFrameReady += new
       EventHandler<SkeletonFrameReadyEventArgs>(nui_SkeletonFrameReady);
            }
                                    //Appel de l'événement Création d'un squelette
     }


    bool laserOn = false;           //Booléen pour l'activation de la main
                                    //en tant que curseur
    void nui_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
                                    //Création de l'événement
    {
    using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
                                   //Déclaration d'une trame pour squelette
    {
        if (skeletonFrame != null)
        {
                                  //Déclaration d'un tableau de type Squelette
                                  //Copie du tableau dans la trame
                                  //Déclaration d'un squelette dans lequel
```

```
                        //on introduit le premier squelette détecté
        Skeleton[] skeletonData = new
        Skeleton[skeletonFrame.SkeletonArrayLength];
        skeletonFrame.CopySkeletonDataTo(skeletonData);
        Skeleton playerSkeleton = (from s in skeletonData where s.TrackingState
        ==   SkeletonTrackingState.Tracked select s).FirstOrDefault();

    if (playerSkeleton != null)
    {
                        //Appel des fonctions "Détecter un mouvement"
        DiapoSuivante(playerSkeleton.Joints[JointType.HandRight],
        playerSkeleton.Joints[JointType.HipLeft]);
        DiapoPrécédente(playerSkeleton.Joints[JointType.HandLeft],
        playerSkeleton.Joints[JointType.HipRight]);
        LancerDiapo(playerSkeleton.Joints[JointType.HandRight],
        playerSkeleton.Joints[JointType.Head]);
        QuitterDiapo(playerSkeleton.Joints[JointType.HandLeft],
        playerSkeleton.Joints[JointType.KneeLeft]);
        ActiverLaser(playerSkeleton.Joints[JointType.HandLeft],
        playerSkeleton.Joints[JointType.ShoulderLeft]);


                        //Détection d'un mouvement servant à
                        //activer le booléen
         if (playerSkeleton.Joints[JointType.HandLeft].Position.Y >
         playerSkeleton.Joints[JointType.ShoulderLeft].Position.Y)
         {
              if(laserOn==false)laserOn = true;
              else if (laserOn==true)laserOn = false;
         }

                        //Si booléen activé alors détection de la main
                        //droite en tant que curseur
          if(laserOn==true)
          {
               Joint scaledRight =
               playerSkeleton.Joints[JointType.HandRight].ScaleTo((int)SystemInfo
               rmation.PrimaryMonitorSize.Width,
               (int)SystemInformation.PrimaryMonitorSize.Height, -
               playerSkeleton.Position.X, -playerSkeleton.Position.Y);
               int CursorX = (int)scaledRight.Position.X;
               int CursorY = (int)scaledRight.Position.Y;

               KinectMouseController.KinectMouseMethods.SendMouseInput(CursorX,
               CursorY, SystemInformation.PrimaryMonitorSize.Width,
               SystemInformation.PrimaryMonitorSize.Height, false);

                   }
               }
          }
     }
}

int comptSuiv;            //Entier servant de tempo entre deux gestes

private void DiapoSuivante(Joint handRight, Joint hipLeft)
{
```

```
        if (handRight.Position.X < hipLeft.Position.X)
                            //Si main droite à gauche de hanche gauche
        {

            comptSuiv++;      //Alors incrémentation du compteur
        }

        if (comptSuiv == 20) //Si compteur arrivé à 20
    {
        System.Windows.Forms.SendKeys.SendWait("{Right}");
        comptSuiv = 0;      //Alors envoie de la touche "Droite"
    }                       //Et remise à 0 du compteur
}

int comptPrec;              //Entier servant de tempo entre deux gestes

private void DiapoPrécédente(Joint handLeft, Joint hipRight)
{
    if (handLeft.Position.X > hipRight.Position.X)
                            //Si main gauche à droite de hanche droite
    {

        comptPrec++;      //Alors incrémentation du compteur
    }

    if (comptPrec == 20)  //Si compteur arrivé à 20
    {
        System.Windows.Forms.SendKeys.SendWait("{Left}");
                            //Alors envoie de la touche "Gauche"
        comptPrec = 0;    //Et remise à 0 du compteur
    }
}

int comptLanc;
private void LancerDiapo(Joint handRight, Joint head)
{
    if(handRight.Position.Y > head.Position.Y)
                            //Si main droite en haut de tête
    {
        comptLanc++;
    }

    if(comptLanc==20)
    {
        System.Windows.Forms.SendKeys.SendWait("{F5}");
                             //Envoie de touche F5
        comptLanc = 0;
    }
}

int comptQuit;
private void QuitterDiapo(Joint handLeft, Joint kneeLeft)
{
    if(handLeft.Position.Y < kneeLeft.Position.Y)
                            //Si main gauche en bas du genou gauche
    {
        comptQuit++;
    }
```

```
            if(comptQuit==20)
            {
                System.Windows.Forms.SendKeys.SendWait("{ESC}");
                                //Evoie de touche echap
                comptQuit = 0;
            }
        }

        int comptLaser;
        private void ActiverLaser(Joint handLeft, Joint shoulderLeft)
        {
            if(handLeft.Position.Y > shoulderLeft.Position.Y)
                                //Si main gauche en haut de épaule gauche
            {
                comptLaser++;
            }

            if(comptLaser==20)
            {
                System.Windows.Forms.SendKeys.SendWait("^l");
                                //Envoie de "CTRL+L"
                comptLaser = 0;
            }
        }



        private void button3_Click(object sender, RibbonControlEventArgs e)
        {
            nui.Stop();
                                //Arrêt du capteur
        }
    }
}
```