Francisco Gea Martínez y Francisco Pérez Hernández

NPI – Kinect v2.0

Los objetivos de esta práctica han sido:

1. Visualización del esqueleto.
2. Mostrar al usuario la posición y el gesto correcto antes de comenzar la detección.
3. Fijar una posición inicial y ayudar al usuario a situarse en la misma.
4. Ayudar al usuario mediante marcas virtuales para realizar acciones.
5. Establecer un margen de error que pueda ser modificable en la ejecución.

Los pasos iniciales han sido instalar los drivers del SDK de Kinect 2.0 [1.sdk]. Hemos instalado Visual Studio para Windows y dentro de este, hemos creado un nuevo proyecto. Para crear el nuevo proyecto, sería dar a Nuevo Proyecto -> Visual C# -> Aplicación WPF -> Dar nombre.

En el siguiente paso, sería mezclar los proyectos que proporciona el SDK de Kinect de BodyBasic y ColorBasic, pero al tener problemas a la hora de unirlos, hemos encontrado por internet un código que ya ha realizado esa tarea. [2zubariahmed][3codigo]. El problema que tuvimos a la hora de hacer la unión, fue que no éramos capaces a mostrar los dos flujos, pero una vez conseguido, teníamos desplazado el esqueleto con respecto al cuerpo en color.

Ahora pasaríamos a mezclar el código descargado (MainWindow.xaml.cs) junto con el nuestro. Para ello:

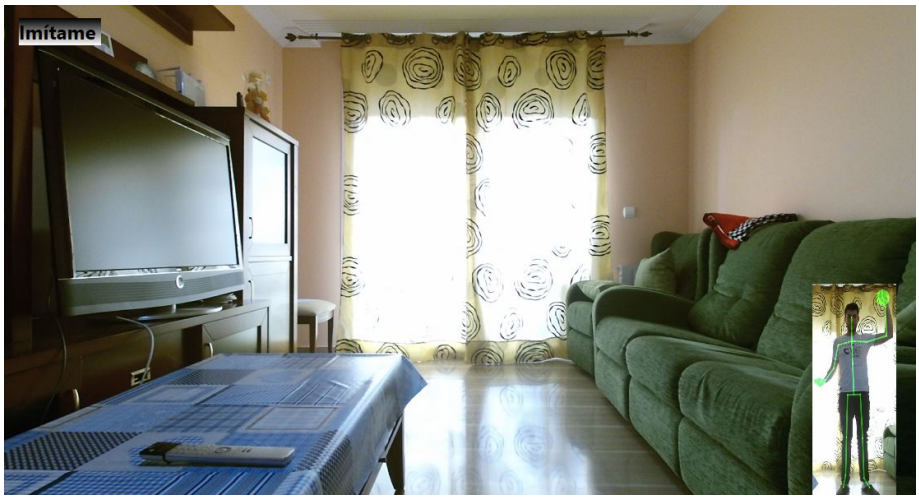
- Primero añadimos los using restantes.
- Añadimos la referencia: Proyecto -> Añadir referencia -> Ensamblados -> Extensiones -> Microsoft.Kinect
- Copiamos del código descargado desde “public partial class MainWindow: Window, INotifgyPropertyChanged” en Adelante.
- Añadimos la librería WriteableBitmapEx.Wpf.dll [4WriteableBitmap]
- Creamos la carpeta lib dentro del proyecto, copiamos lo descargado. En el proyecto nos vamos a Proyecto -> Examinar -> Reciente -> Buscamos “WriteableBitmapEx.Wpf.dll” y la añadimos.
- Ahora en el .cs vamos a eliminar, en la línea 223: “if (this.bodies ¡= null)” todo ese if, ya que esa comprobación la realizamos antes.
- En la línea 112 del .cs, tendremos que cambiar “kinectSensor.Default” por “KinectSensor.GetDefault()”
- Ahora en el fichero MainWindow.xaml, copiamos desde “Title” hasta el final.
- Comentamos el código en español.

El resultado será:

Lo siguiente a realizar es mostrar la posición inicial al usuario, para ello:

- Añadimos una imagen en el borde inferior de la ventana, con la posición a realizar. Para ello solo debemos arrastrar la imagen al .xaml y colocarla y escalarla.
- En el .cs hacemos que cuando se inicie el programa sea visible con: "posicionInicio.Visibility = System.Windows.Visibility.Visible" cuando comienzan las tareas.
- Añadimos un cuadro de texto, para ir indicando instrucciones.

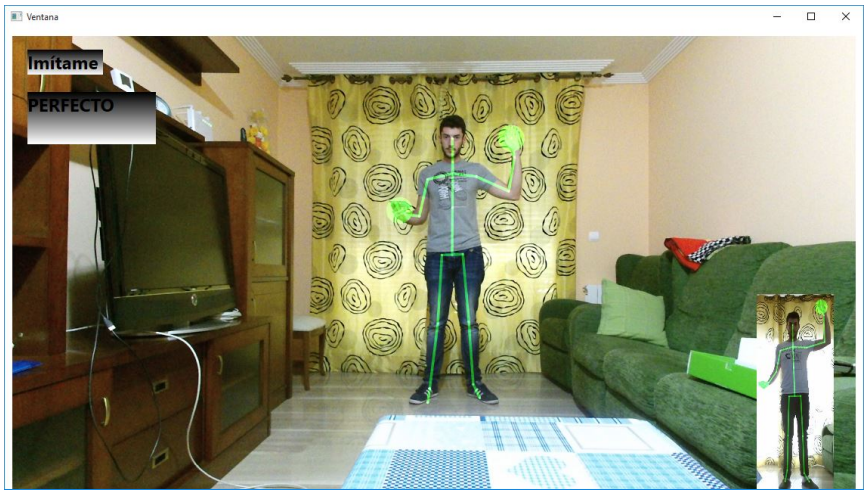
En este punto el resultado sería:



Para fijar la posición y ayudar al usuario:

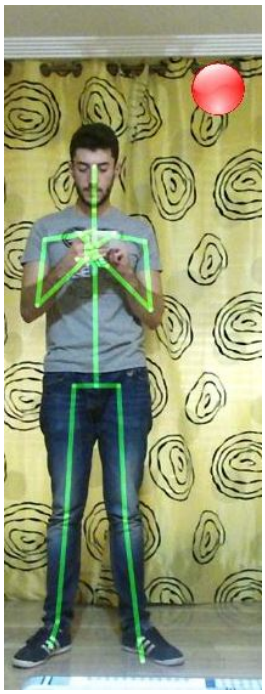
- Primero nos basamos en en BodyBasic.Wpf para mostrar si el usuario se sale de la franja con el método "DrawClippedEdges"
- Creamos los Points para guardar los puntos de las partes del cuerpo importantes como: cabeza, hombros, pies y manos.
- Creamos la función para controlar los movimientos que debe hacer el usuario, como: moverse a algún lado, alejarse, levantar la mano…
- Crear el text en el .xaml para mostrar el movimiento.
- Hemos usado esta imagen para ver los puntos importantes [5joints]

En este punto:



Para realizar la aplicación mucho más intuitiva para el usuario vamos a ayudarle mediante marcas virtuales:

- Para ello, vamos a usar círculos de colores.
- Cuando el usuario se haya puesto en la distancia adecuada, aparecerá un círculo rojo para que suba la mano izquierda hasta el círculo.
- En el momento que la suba, el círculo será verde.


Otra ayuda para el usuario es el uso de flechas indicando que se tiene que mover a la izquierda o derecha, ya que es mucho más intuitivo que estar leyendo:

Por último, hemos añadidos hotkeys, para poder modificar los porcentajes de error en la detección:

- Para realizar los hotkeys, hemos hecho uso de unos ejemplos. [6hotkeys]
- Los márgenes a modificar son el horizontal, vertical y la precisión a al hora de tocar la pelota e iniciar la aplicación.

Los problemas que se nos han presentado han sido:

- No sabíamos mostrar los flujos de color y esqueleto juntos.
- Hemos encontrado para documentación específica para desarrollar en Kinect One.
- Una vez mostrados ambos flujos, el esqueleto no estaba correctamente coordinado con el color.
- No sabíamos como añadir elementos al viewport, tales como figuras planas o modelos 3D.
- Las coordenadas del flujo de esqueleto y el de color no estaban correctamente coordinados.
- Hemos tenido errores al fijar el margen por defecto de la aplicación debido a que somos muy altos.

Referencias:

- [1 sdk] http://www.microsoft.com/en-us/download/details.aspx?id=44561
- [2zubairahmed] http://www.zubairahmed.net/?p=1682
- [3codigo] https://onedrive.live.com/?id=2A7FDB962EA8E245%217908&cid=2A7FDB962EA8E245&group=0&parId=root&authkey=%21AIyBKaaipmZ8VDQ&action=locate
- [4WriteableBitmap] https://writeablebitmapex.codeplex.com/
- [5joints] https://i-msdn.sec.s-msft.com/dynimg/IC741371.png
- [6hotkeys] http://stackoverflow.com/questions/1361350/keyboard-shortcuts-in-wpf
- [7github] https://github.com/PacoPollos
- [8github] https://github.com/Exea
- [9starkoverflow] http://stackoverflow.com
- [10youtube] https://www.youtube.com/watch?v=F58uoEz861M&feature=youtu.be