# Programación para Estudiantes

Blog de programacion. El nivel de estas entradas será orientado a estudiantes.
Muchas de las entradas estan orientadas a la ayuda de compañeros de universidad.
Curiosidades y otros.

## SDK Kinect for Dummies-1

De Fernando - abril 10, 2012

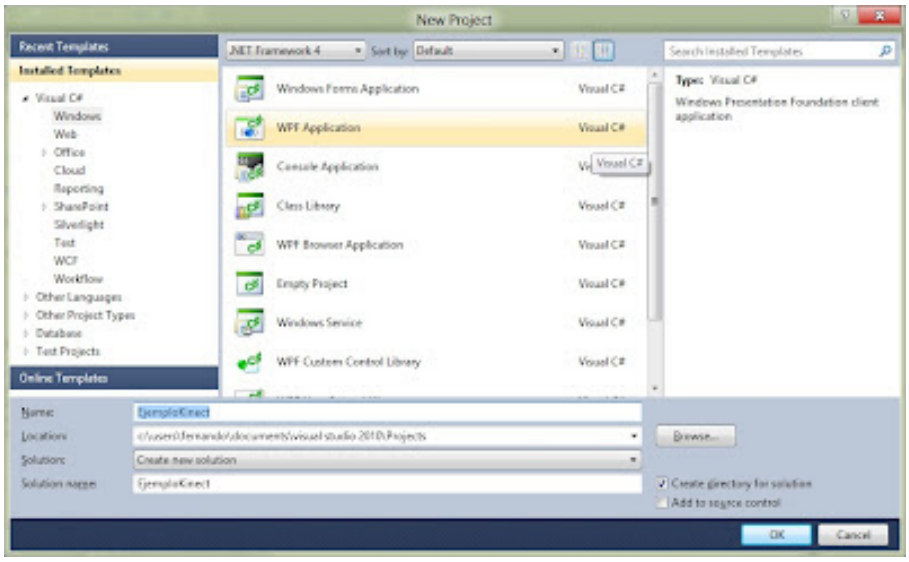
Todos conocemos Kinect, unos lo habréis probado en casa de algún amigo, otros solo lo habéis visto en la tele. Sea como sea Kinect nos puede servir como herramienta de desarrollo y de investigación o simplemente como entretenimiento. Al poco de aparecer Kinect hubo un SDK no oficial para poder emplear Kinect en nuestros PCs. Pasado un tiempo Microsoft lanzó el SDK de Kinect para Windows. Pero en los detalles de hardware o como ha ido evolucionando Kinect no es en lo que me fijaré en el blog. Intentaré orientarme más a la programación.


Ahora viene mi opinión. Pues bien me he puesto con el SDK de Kinect y me gusta. He estado haciendo cosillas con el reconocimiento de voz aunque de momento no tengo material para subir ninguna entrada. Por otra parte como primer paso con el SDK de Kinect veremos como pintar lo que muestra la cámara. Es decir, tan solo la cámara sin Skeleton ni nada por el estilo.

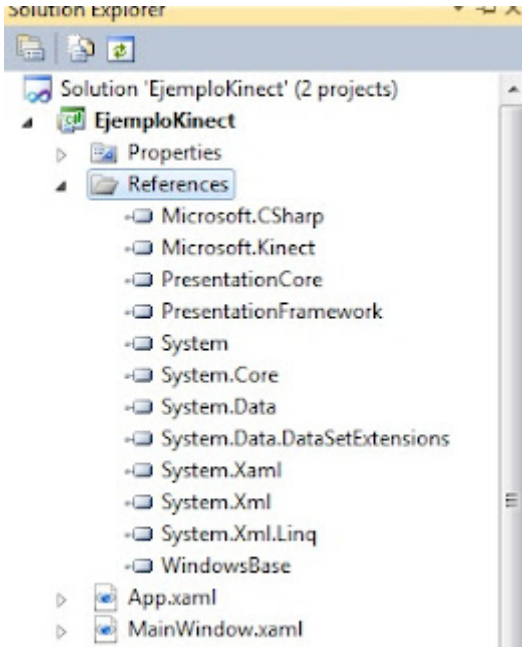
Necesitamos:
- Visual Studio: versión Express o si somos estudiantes desde www.dreamspark.com
- SDK de Kinect: http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx
- Sensor Kinect.

# Programación para Estudiantes

Abrimos Visual Studio y creamos un nuevo proyecto: WPF application. Le damos un nombre a nuestro proyecto, en mi caso lo he llamado ejemploKinect.

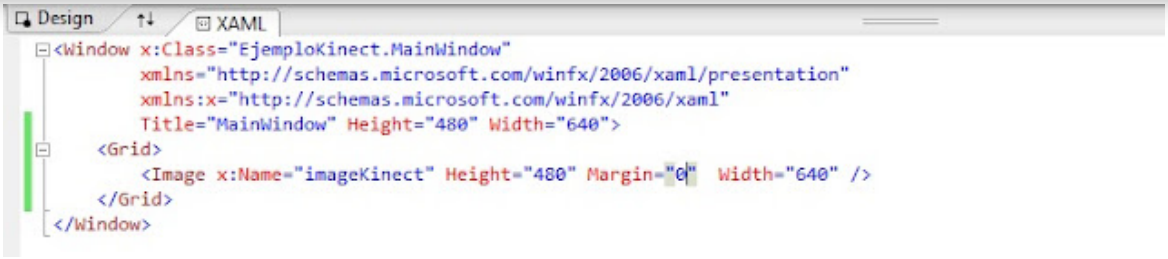



Una vez agregado el proyecto debemos agregar la referencia de Kinect a nuestro proyecto. Vamos al Solution Explorer y en nuestro proyecto en 'Reference' hacemos clic derecho y seleccionamos 'Add Reference..' y en la pestaña de .NET buscamos la **referencia Microsoft.Kinect**.




Una vez agregada la referencia debemos ir al *"MainWindow.xaml"* y agregar un control del tipo imagen. Le cambiaremos el nombre al que queramos y modificaremos su ancho y su alto. Debido a que la imagen de Kinect va a 640x480 recomiendo poner 480 de alto y 640

# Programación para Estudiantes




A continuación vamos al .cs asociado a MainWindow. Esto es "MainWindow.xaml.cs". El código esta comentado por tanto no será necesario explicaciones extra. Tan solo apuntar que se debe añadir "**using Microsoft.Kinect;**"

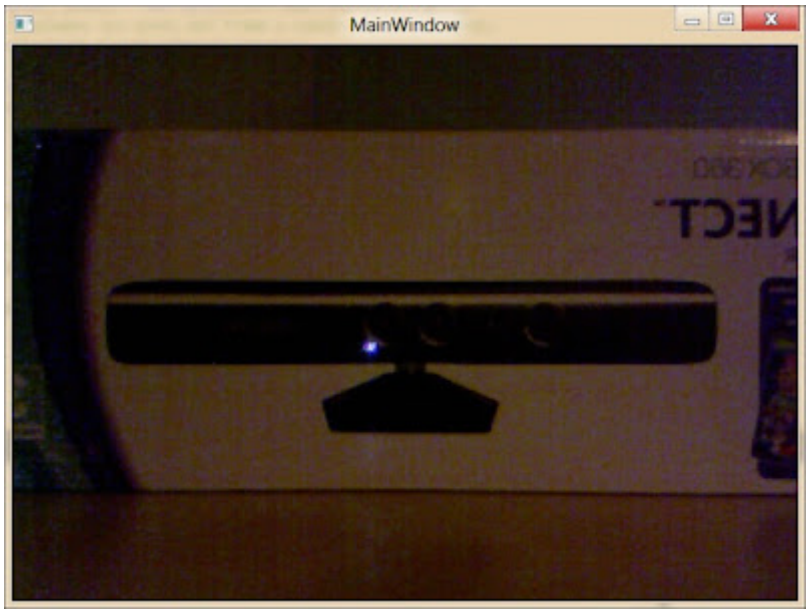
```
///

    /// Interaction logic for MainWindow.xaml
    ///

    public partial class MainWindow : Window
    {
        KinectSensor miKinect;
        public MainWindow()
        {
            InitializeComponent();
            // Agregamos el evento de carga de la ventana
            Loaded += MainWindowLoaded;
        }

        void MainWindowLoaded(object sender, RoutedEventArgs e)
        {
            // Comprobamos que tenemos un sensor conectado
            if (KinectSensor.KinectSensors.Count > 0)
            {
                //Evento ejecutado al cerrar
                Closing += MainClosing;
                // Escogemos el primer sensor kinect que tengamos conectac
                miKinect = KinectSensor.KinectSensors[0];
                // Habilitamos la cámara elegiendo el formato de imagen.
                miKinect.ColorStream.Enable(ColorImageFormat.RgbResolutior
```

# Programación para Estudiantes

```
            // Nos suscribimos al metodo
            miKinect.AllFramesReady += KinectAllFramesReady;
        }

    }

    ///

    /// Obtenemos los frames y los pintamos en la imagen.
    ///

    ///
    ///

    void KinectAllFramesReady(object sender, AllFramesReadyEventArgs e
    {
        //Obtenemos el frame de imagen de la camara
        using (var colorFrame = e.OpenColorImageFrame())
        {
            // Si este es null no continuamos
            if (colorFrame == null) return;
            // Creamos un array de bytes del tamaño de los pixel del f
            byte[] pixels = new byte[colorFrame.PixelDataLength];
            //Copiamos los pixel del frame a nuestro array de bytes.
            colorFrame.CopyPixelDataTo(pixels);
            // Colocamos los pixel del frame en la imagen del xml
            int stride = colorFrame.Width * 4;
            imageKinect.Source = BitmapSource.Create(colorFrame.Width,
            //imageKinect es el objeto imagen colocado en el xaml
        }
    }

    void MainClosing(object sender, EventArgs e)
    {
        // Al cerrar paramos Kinect
        if (miKinect != null)
        {
            miKinect.Stop();
        }
```

# Programación para Estudiantes

Y ahora veamos como queda:




**Las imágenes se ven como un espejo** debido a que Kinect esta preparado para que nosotros nos veamos en él.

Intentaré subir el proyecto proximamente.

C# Kinect

---

**Anónimo** *1 de febrero de 2013, 6:46*

Gracvias amigo, esto me interesa y es bueno que hayas puesto un ejemplo como este de claro

**RESPONDER**

**Tony** *2 de marzo de 2013, 0:11*

muy buen aporte yo tambien acabo de iniciarme a la programacion de mismo

**RESPONDER**

**GianMarkin** *14 de abril de 2013, 2:59*

# Programación para Estudiantes

**Unknown** *23 de septiembre de 2013, 6:55*

```
namespace WpfApplication1
{
///
/// Lógica de interacción para MainWindow.xaml
///
public partial class MainWindow : Window
{
Microsoft.Kinect. KinectSensor miKinect;
public MainWindow()
{
InitializeComponent();
// Agregamos el evento de carga de la ventana
Loaded += MainWindowLoaded;
}

void MainWindowLoaded(object sender, RoutedEventArgs e)
{
// Comprobamos que tenemos un sensor conectado
if (Microsoft.Kinect.KinectSensor.KinectSensors.Count > 0)
{
//Evento ejecutado al cerrar
Closing += MainClosing;
// Escogemos el primer sensor kinect que tengamos conectado. Puede
haber más de un kinect conectado
miKinect =Microsoft.Kinect.KinectSensor.KinectSensors[0];
// Habilitamos la cámara elegiendo el formato de imagen.
miKinect.ColorStream.Enable(Microsoft.Kinect.ColorImageFormat.RgbReso
lution640x480Fps30);
// Arrancamos Kinect.
miKinect.Start();
// Nos suscribimos al método
miKinect.AllFramesReady += KinectAllFramesReady;
}

}

        void KinectAllFramesReady(object sender,
Microsoft.Kinect.AllFramesReadyEventArgs e)
{
//Obtenemos el frame de imagen de la camara
using (var colorFrame = e.OpenColorImageFrame())
{
// Si este es null no continuamos
if (colorFrame == null) return;
// Creamos un array de bytes del tamaño de los pixel del frame.
byte[] pixels = new byte[colorFrame.PixelDataLength];
//Copiamos los pixel del frame a nuestro array de bytes.
colorFrame.CopyPixelDataTo(pixels);
// Colocamos los pixel del frame en la imagen del xml
int stride = colorFrame.Width * 4;
imageKinect.Source = BitmapSource.Create(colorFrame.Width,
colorFrame.Height, 96, 96, PixelFormats.Bgr32, null, pixels, stride);
```

# Programación para Estudiantes

```
void MainClosing(object sender, EventArgs e)
{
// Al cerrar paramos Kinect
if (miKinect != null)
{
miKinect.Stop();
}
}
}
}
```

codigo en vb2012
**RESPONDER**

**Anónimo** *14 de noviembre de 2013, 12:00*

gracias por el tutorial está muy bien explicado¡¡

**RESPONDER**

**Anónimo** *16 de febrero de 2015, 7:00*

Hola soy nuevo en C# y no entiendo porque el código me arroja estos errores

Error 1: The name 'InitializeComponent' does not exist in the current context
Error 2: The name 'imageKinect' does not exist in the current context

> **Anónimo** *16 de febrero de 2015, 10:46*
>
> InitializeComponent inicializa los componentes de la interfaz gráfica. imageKinect es el objeto Image de la interfaz. Parece que no estás accediendo al .cs asociado al .xaml de la interfaz.

**RESPONDER**

Escribe tu comentario

**Entradas populares de este blog**

## Procesos Linux - exec y fork

*De Fernando - noviembre 13, 2011*

# Programación para Estudiantes

LEER MÁS

## Función __doPostBack

*De Fernando - agosto 30, 2012*

A veces en ASP.NET tenemos la necesidad de realizar un 'post back' desde JavaScript. Si analizamos el HTML que se genera con ASP.NET encontramos que los controles de ASP.NET llaman a la función __doPostBack para realizar esta acción. Esta es ...

LEER MÁS

## malloc vs calloc

*De Fernando - octubre 31, 2011*

Anteriormente en la segunda parte de punteros hablé acerca de cómo crear arrays dinámicos. Vimos la importancia de la función malloc() en este proceso. Sin embargo existen otras dos funciones complementarias a malloc() . Estas funciones sc ...

LEER MÁS