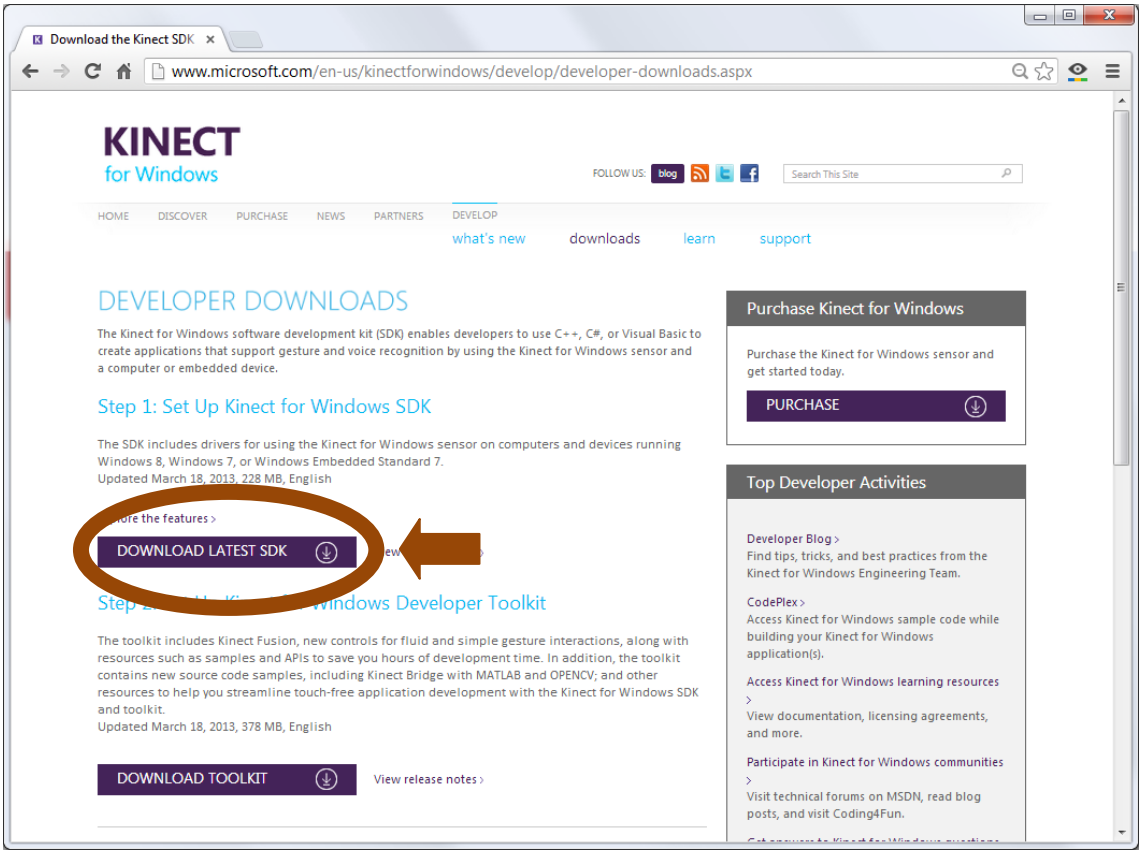
TUTORIAL KINECT

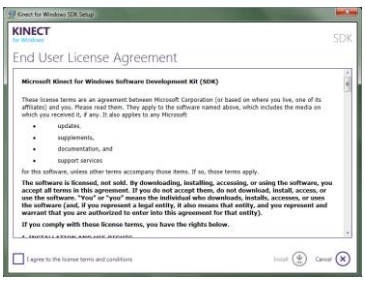
1. Instalando o Sensor

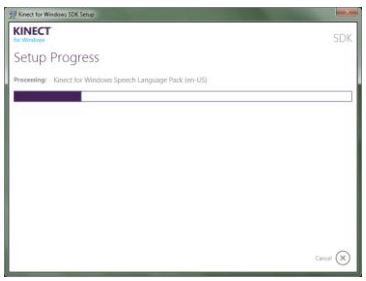
- Primeiro deve-se instalar a Kinect for Windows SDK. Vá até a página: http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx e clique em “Download Latest SDK”:

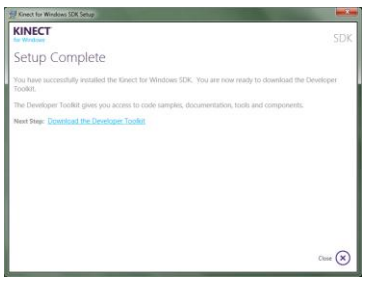


Nesta etapa pode ser feito também o download da Toolkit de desenvolvimento clicando em “Download Toolkit”. Sua instalação segue os mesmos passos da instalação da SDK.

- A instalação é simples e rápida:







- O próximo passo é conectar o sensor ao computador. Ele deverá instalar automaticamente e ao final uma luz verde no sensor indicará seu funcionamento.

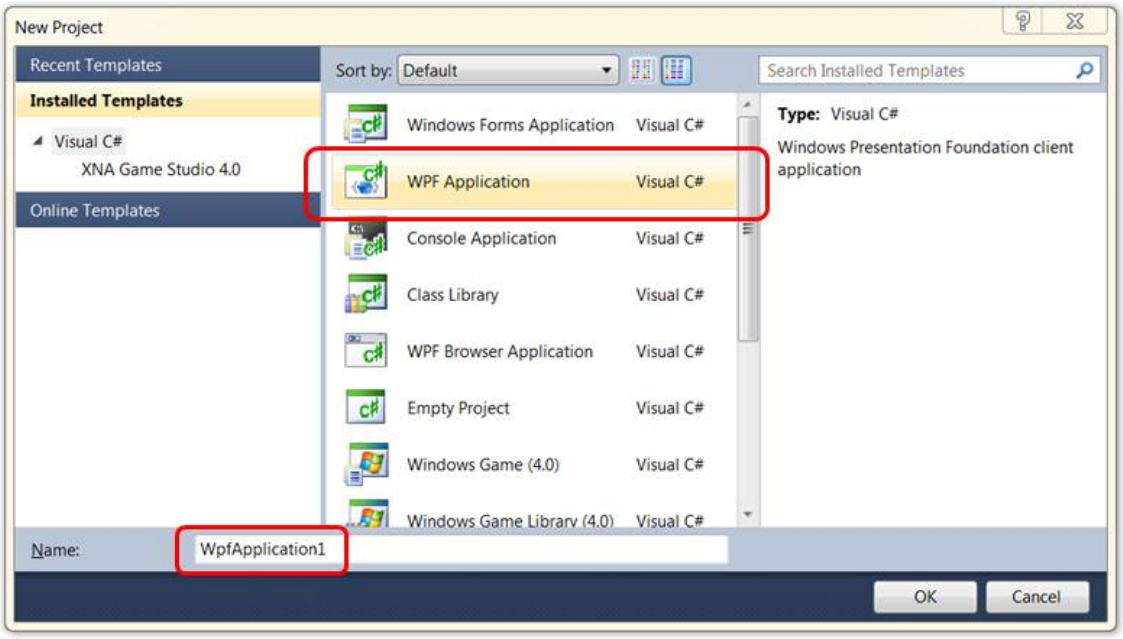
- Ao final, clique em *Control Panel -> Device Manager*. Você deverá ver o Kinect como na imagem a seguir:

2. Configurando o Ambiente de Desenvolvimento

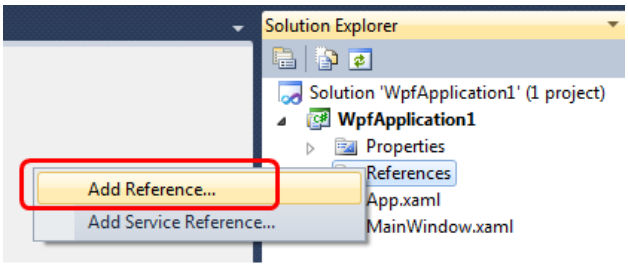
Aqui será apresentada a utilização do Visual Studio 2010 (você deverá ter instalado previamente). A linguagem utilizada foi C#.

- Inicie o Visual Studio 2010 (Express ou superiores);

- Selecione *File –> New Project –> Windows Presentation Foundation* e nomeie seu projeto:



- Na janela *Solution Explorer*, clique com o botão direito em *References* e clique em *Add Reference*:



- Na janela que se abrirá, vá até a aba .NET e procure pelo componente Microsoft.Kinect e clique em OK.

- Agora em seu projeto você pode referenciar os namespaces do Kinect:

```
using Microsoft.Research.Kinect.Nui;
using Microsoft.Research.Kinect.Audio;
```

Nas demonstrações foi utilizada a Toolkit Coding4Fun. Esta biblioteca contém facilita a realização de diversas tarefas, como converter dados vindos do Kinect em um array ou Bitmap. Para utilizá-la vá até o site (http://c4fkinect.codeplex.com) e baixe a sua última versão. Descompacte o arquivo em alguma pasta e adicione às referências do seu projeto como a toolkit do Kinect. Desta vez, ao invés da aba .NET, navegue pela aba Browse até a pasta com a dll.

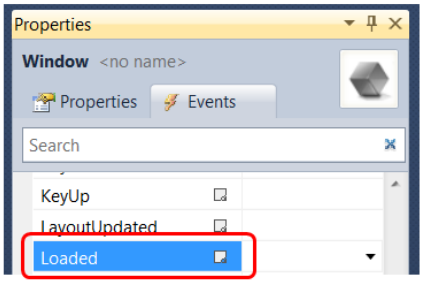
Caso esteja trabalahndo com WPF Applications, use:

```
using Coding4Fun.Kinect.Wpf;
```

Caso esteja trabalhando com Windows Form Applications, use:

```
using Coding4Fun.Kinect.WinForm;
```

- Para criar o evento *Window_Loaded* onde o sensor será inicializado, vá à janela *Properties* (F4), selecione a *MainWindow*, selecione a aba Events e dê um duplo clique no evento *Loaded* para criar o evento *Window_Loaded*:



Siga os mesmos passos, desta vez procurando pelo evento Closed e crie o evento Window_Closed.

- Fora do evento *Window_Loaded*, digite:

```
KinectSensor sensor = KinectSensor.KinectSensors[0];
```

- Dentro do evento *Window_Loaded*:

```
sensor.Start();
```

-Dentro do evento *Window_Closed*:

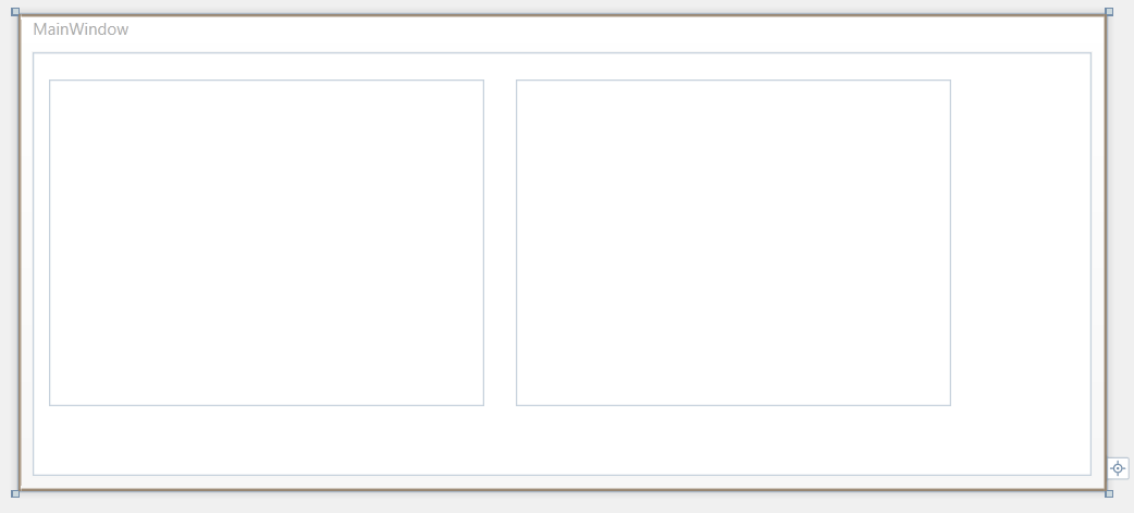
```
sensor.Stop();
```

3. Demonstração 1: Exibindo Imagem da Camera RGB e Dados de Profundidade

- Configurado o ambiente de desenvolvimento, adicione duas imagens 320x240 arrastando-as da Toolbox, ou digitando o código XAML:

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="<a
href="http://schemas.microsoft.com/winfx/2006/xaml/presentation"">http://schem
as.microsoft.com/winfx/2006/xaml/presentation"</a>
        xmlns:x="<a
href="http://schemas.microsoft.com/winfx/2006/xaml"">http://schemas.microsoft.
com/winfx/2006/xaml"</a>
        Title="MainWindow" Height="350" Width="800">
    <Grid>
        <Image Height="240" HorizontalAlignment="Left" Margin="12,20,0,0"
Name="image1" Stretch="Fill" VerticalAlignment="Top" Width="320" />
        <Image Height="240" HorizontalAlignment="Left" Margin="355,20,0,0"
Name="image2" Stretch="Fill" VerticalAlignment="Top" Width="320" />
    </Grid>
</Window>
```

- O resultado deverá ser:



- No evento *Window_Loaded*, habilite a utilização dos streams da Camera RGB e da de profundidade, antes de iniciar o sensor:

```
sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
sensor.DepthStream.Enable(DepthImageFormat.Resolution320x240Fps30);
sensor.Start();
```

- Ainda no evento *Window_Loaded*, assine o evento *sensor_ColorFrameReady*. Você pode fazê-lo digitando `sensor.ColorFrameReady +=` e depois pressionando a tecla <tab> duas vezes. Deverá surgir o evento:

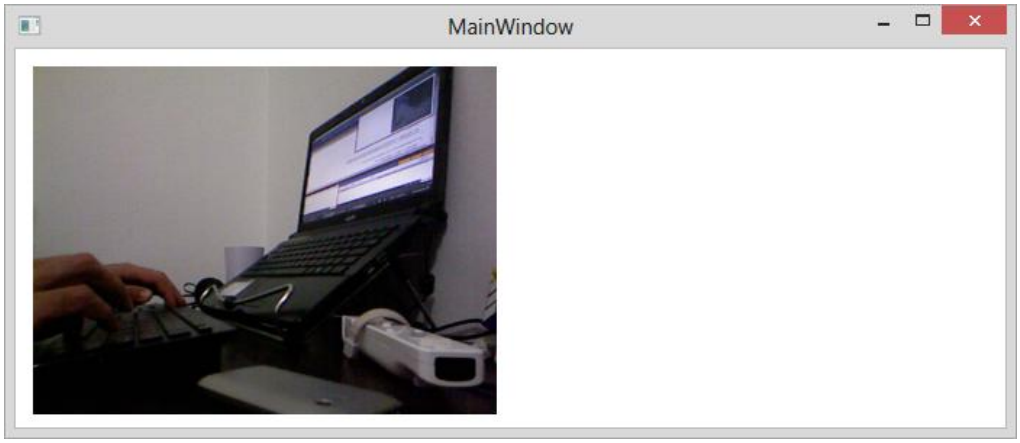
```
void sensor_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e) {

        //…

}
```

- Em `sensor_ColorFrameReady`, adicione:

```
ColorImageFrame imgFrame = e.OpenColorImageFrame();
image1.Source = imgFrame.ToBitmapSource();
```

O método ToBitmapSource() provém da Toolkit Coding4Fun.

- Rode a aplicação e veja o resultado:



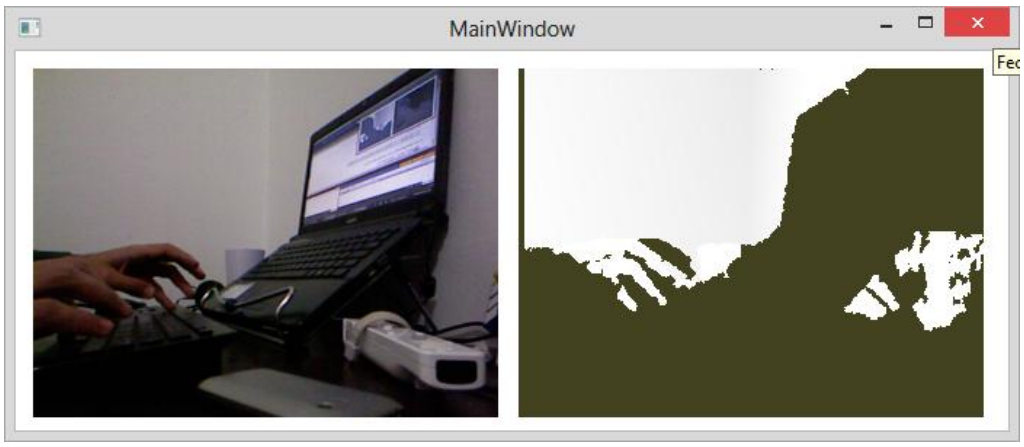
- De volta ao evento *Window_Loaded*, assine também o evento *sensor_DepthFrameReady*. Você pode fazê-lo digitando `sensor.DepthFrameReady +=` e depois pressionando a tecla <tab> duas vezes. Deverá surgir o evento:

```
void sensor_DepthFrameReady(object sender, DepthImageFrameReadyEventArgs e) {

    //…

}
```

- Em `sensor_DepthFrameReady`, adicione:

```
DepthImageFrame imgFrame = e.OpenDepthImageFrame();
image2.Source = imgFrame.ToBitmapSource();
```

- Rode novamente a aplicação e veja o resultado, agora com os dados de profundidade:



Código da Demonstração 1:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;

namespace WpfApplication6{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window{
        public MainWindow(){
            InitializeComponent();
        }

        KinectSensor sensor = KinectSensor.KinectSensors[0];

        private void Window_Loaded(object sender, RoutedEventArgs e){

sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
sensor.DepthStream.Enable(DepthImageFormat.Resolution320x240Fps30);
            sensor.Start();

            sensor.ColorFrameReady += new
EventHandler<ColorImageFrameReadyEventArgs>(sensor_ColorFrameReady);
            sensor.DepthFrameReady += new
EventHandler<DepthImageFrameReadyEventArgs>(sensor_DepthFrameReady);
        }

        void sensor_DepthFrameReady(object sender,
DepthImageFrameReadyEventArgs e){
            DepthImageFrame imgFrame = e.OpenDepthImageFrame();
            image2.Source = imgFrame.ToBitmapSource();
        }

        void sensor_ColorFrameReady(object sender,
ColorImageFrameReadyEventArgs e){
            ColorImageFrame imgFrame = e.OpenColorImageFrame();
            image1.Source = imgFrame.ToBitmapSource();
        }

        private void Window_Closed(object sender, EventArgs e){
            sensor.Stop();
        }
    }
}
```

4. Demonstração 2: Rastrear movimentos das mãos e cabeça

- Configurado o ambiente de desenvolvimento, adicione três elipses dentro de um Canvas nomeado *MainCanvas*, como no código XAML a seguir:

```
<Canvas Name="MainCanvas">
    <Ellipse Canvas.Left="0" Canvas.Top="0" Height="50" Name="headEllipse"
Stroke="Black" Width="50" Fill="Orange" />
    <Ellipse Canvas.Left="50" Canvas.Top="0" Height="50" Name="rightEllipse"
Stroke="Black" Width="50" Fill="SlateGray" />
    <Ellipse Canvas.Left="100" Canvas.Top="0" Fill="SpringGreen" Height="50"
Name="leftEllipse" Stroke="Black" Width="50" />
</Canvas>
```

Note que as elipses foram nomeadas também.

- O resultado deverá ser:

- Crie a váriável `Skeleton[] skeleton;` para armazenar os corpos rastreados. E no evento *Window_Loaded*, habilite a utilização da stream:

```
sensor.SkeletonStream.Enable();
sensor.Start();
```

- Ainda no evento *Window_Loaded*, assine o evento *sensor_SkeletonFrameReady*. Você pode fazê-lo digitando `sensor. SkeletonFrameReady +=` e depois pressionando a tecla <tab> duas vezes. Deverá surgir o evento:

```
void sensor_SkeletonFrameReady (object sender, SkeletonFrameReadyEventArgs e){

    //…

}
```

- Para manipular os corpos identificados, em `sensor_SkeletonFrameReady`, adicione:

```
using(SkeletonFrame skeletonFrame = e.OpenSkeletonFrame()){
    if (skeletonFrame != null) {
        skeleton = new Skeleton[skeletonFrame.SkeletonArrayLength];
        skeletonFrame.CopySkeletonDataTo(skeleton);
    }
}
```

Antes de posicionar as elipses de acordo as posições das mãos e da cabeça, é necessário escalar. O método ScaleTo() provém da Toolkit Coding4Fun.

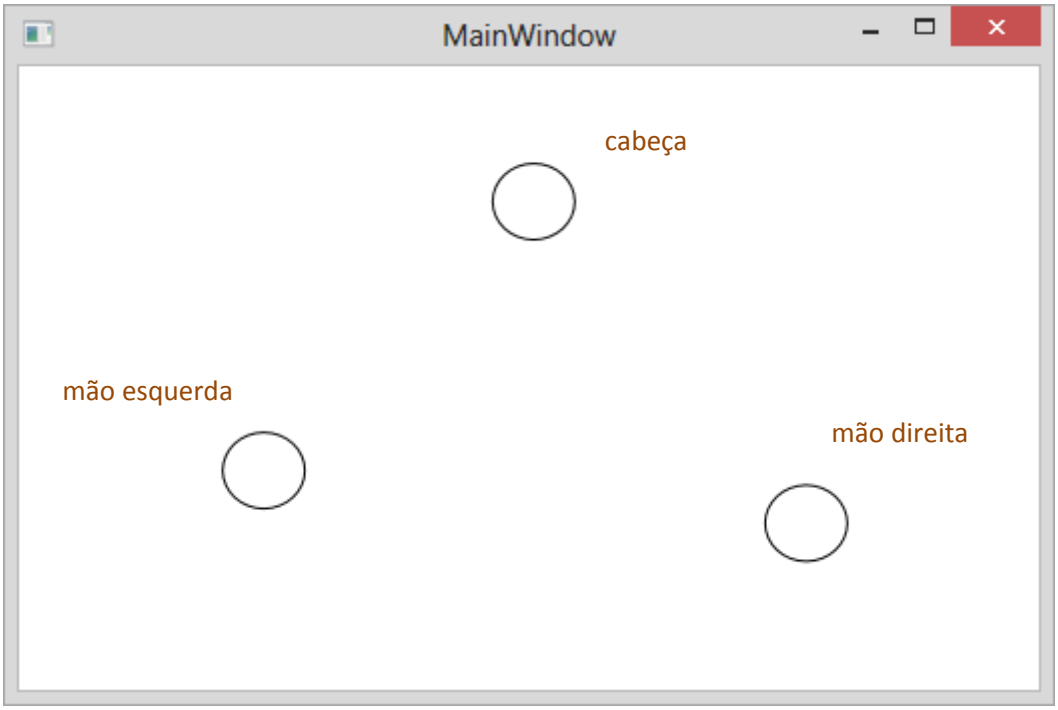
- Crie a seguinte função para receber uma elipse e uma junta, escalar sua posição e reposicionar as elipses:

```
private void SetEllipsePosition(FrameworkElement ellipse, Joint joint){
       var scaledJoint = joint.ScaleTo(640, 480, .5f, .5f);

       Canvas.SetLeft(ellipse, scaledJoint.Position.X);
       Canvas.SetTop(ellipse, scaledJoint.Position.Y);
}
```

- De volta ao evento *sensor_SkeletonFrameReady*, percorra a variável skeleton buscando o player rastreado:

```
if (skeleton.Length != 0) {
    foreach (Skeleton s in skeleton) {
        if (s.TrackingState == SkeletonTrackingState.Tracked) {
            SetEllipsePosition(headEllipse, s.Joints[JointType.Head]);
            SetEllipsePosition(leftEllipse, s.Joints[JointType.HandLeft]);
            SetEllipsePosition(rightEllipse, s.Joints[JointType.HandRight]);
        }
    }
}
```

- Rode a aplicação, se posicione em frente ao Kinect e veja o resultado. As elipses deverão se mover de acordo suas mãos e cabeça.



Código da Demonstração 2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Kinect;
using Coding4Fun.Kinect.Wpf;

namespace WpfApplication5{
    public partial class MainWindow : Window{
        public MainWindow(){
            InitializeComponent();
        }
        KinectSensor sensor = KinectSensor.KinectSensors[0];
        Skeleton[] skeleton;

        private void Window_Loaded(object sender, RoutedEventArgs e){
            sensor.SkeletonStream.Enable();
            sensor.Start();

            sensor.SkeletonFrameReady += new
EventHandler<SkeletonFrameReadyEventArgs>(sensor_SkeletonFrameReady);
        }

 void sensor_SkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e){
            using(SkeletonFrame skeletonFrame = e.OpenSkeletonFrame()){
                if (skeletonFrame != null) {
                 skeleton = new Skeleton[skeletonFrame.SkeletonArrayLength];
                 skeletonFrame.CopySkeletonDataTo(skeleton);
                }
            }
            if (skeleton.Length != 0) {
             foreach (Skeleton s in skeleton) {
               if (s.TrackingState == SkeletonTrackingState.Tracked) {
             SetEllipsePosition(headEllipse, s.Joints[JointType.Head]);
             SetEllipsePosition(leftEllipse, s.Joints[JointType.HandLeft]);
             SetEllipsePosition(rightEllipse, s.Joints[JointType.HandRight]);
               }
              }
             }
        }

      private void SetEllipsePosition(FrameworkElement ellipse, Joint joint){
            var scaledJoint = joint.ScaleTo(640, 480, .5f, .5f);
            Canvas.SetLeft(ellipse, scaledJoint.Position.X);
            Canvas.SetTop(ellipse, scaledJoint.Position.Y);
      }

      private void Window_Closed(object sender, EventArgs e){
            sensor.Stop();
        }
    }
}
```