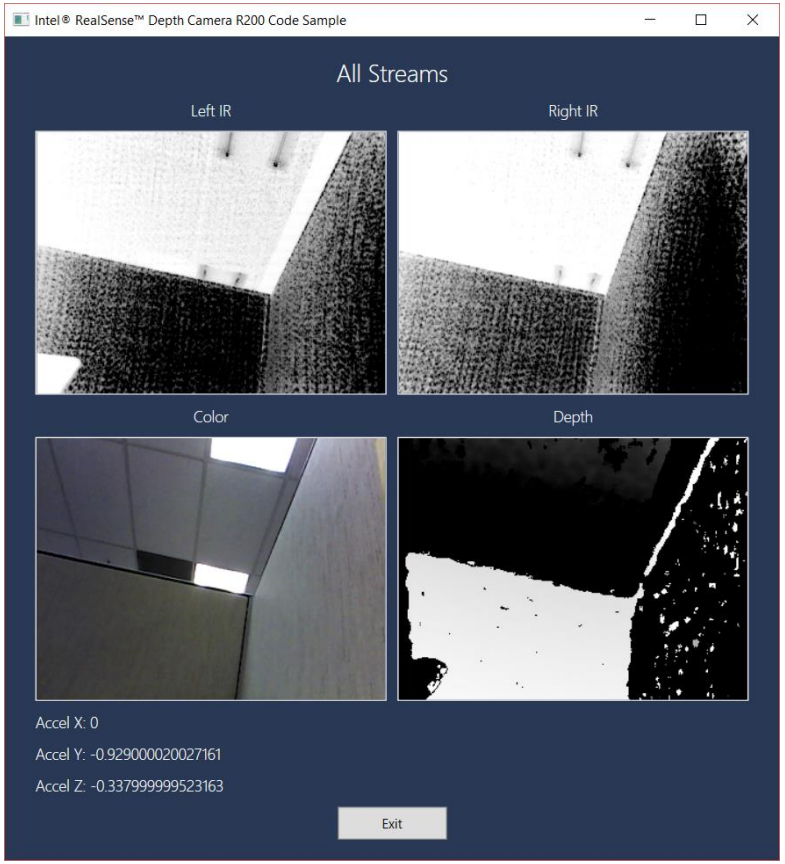

1. Download the R200 sample code located here: https://software.intel.com/en-us/articles/intel-realsense-depth-camera-code-sample-r200-camera-streams

2. Set AllStreams as the Startup Project.

3. Unload the AllStreams project and modify the AllStreams .csproj file as described here: https://software.intel.com/en-us/articles/using-winrt-apis-from-desktop-applications. NOTE: specify `<TargetPlatformVersion>8.1</TargetPlatformVersion>` (instead of 8.0).

4. Modified the AllStreams R200 sample code as shown below (changes are highlighted):

MainWindow.xaml.cs

```
//------------------------------------------------------------------------------------
// Copyright 2015 Intel Corporation
// All Rights Reserved
//
// Permission is granted to use, copy, distribute and prepare derivative works of this
// software for any purpose and without fee, provided, that the above copyright notice
// and this statement appear in all copies.  Intel makes no representations about the
// suitability of this software for any purpose.  THIS SOFTWARE IS PROVIDED "AS IS."
// INTEL SPECIFICALLY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, AND ALL LIABILITY,
// INCLUDING CONSEQUENTIAL AND OTHER INDIRECT DAMAGES, FOR THE USE OF THIS SOFTWARE,
// INCLUDING LIABILITY FOR INFRINGEMENT OF ANY PROPRIETARY RIGHTS, AND INCLUDING THE
// WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.  Intel does not
// assume any responsibility for any errors which may appear in this software nor any
// responsibility to update it.
//------------------------------------------------------------------------------------
using System;
using System.Windows;
using System.Windows.Media.Imaging;
using System.IO;
using System.Drawing;
using System.Threading;

// Added to access accelerometer
using Windows.Devices.Sensors;


namespace CameraStreams
{
    public partial class MainWindow : Window
    {
        private PXCMSession session;
        private PXCMSenseManager senseManager;
        private Thread update;
```

```
        // Added to access accelerometer
        private Accelerometer accel;
        private double accelX;
        private double accelY;
        private double accelZ;

        public MainWindow()
        {
            InitializeComponent();

            accel = Accelerometer.GetDefault();

            // Configure RealSense session and SenseManager interface
            session = PXCMSession.CreateInstance();
            senseManager = session.CreateSenseManager();
            senseManager.EnableStream(PXCMCapture.StreamType.STREAM_TYPE_LEFT, 320, 240, 30);
            senseManager.EnableStream(PXCMCapture.StreamType.STREAM_TYPE_RIGHT, 320, 240, 30);
            senseManager.EnableStream(PXCMCapture.StreamType.STREAM_TYPE_COLOR, 320, 240, 30);
            senseManager.EnableStream(PXCMCapture.StreamType.STREAM_TYPE_DEPTH, 320, 240, 30);
            senseManager.Init();

            // Start Update thread
            update = new Thread(new ThreadStart(Update));
            update.Start();
        }

        private void Accel_ReadingChanged(Accelerometer sender, AccelerometerReadingChangedEventArgs args)
        {
            throw new NotImplementedException();
        }

        private void Update()
        {
            // Start AcquireFrame-ReleaseFrame loop
            while (senseManager.AcquireFrame(true) >= pxcmStatus.PXCM_STATUS_NO_ERROR)
            {
                // Added to access accelerometer
                if (accel != null)
                {
                    AccelerometerReading reading = accel.GetCurrentReading();
                    accelX = reading.AccelerationX;
                    accelY = reading.AccelerationY;
```

```
                    accelZ = reading.AccelerationZ;
                }

                PXCMCapture.Sample sample = senseManager.QuerySample();

                // Get Left IR image data
                PXCMImage.ImageData irLeftData;
                Bitmap irLeftBitmap;
                sample.left.AcquireAccess(PXCMImage.Access.ACCESS_READ, PXCMImage.PixelFormat.PIXEL_FORMAT_RGB32, out
irLeftData);
                irLeftBitmap = irLeftData.ToBitmap(0, sample.left.info.width, sample.left.info.height);

                // Get Right IR image data
                PXCMImage.ImageData irRightData;
                Bitmap irRightBitmap;
                sample.right.AcquireAccess(PXCMImage.Access.ACCESS_READ, PXCMImage.PixelFormat.PIXEL_FORMAT_RGB32, out
irRightData);
                irRightBitmap = irRightData.ToBitmap(0, sample.right.info.width, sample.right.info.height);

                // Get Color image data
                PXCMImage.ImageData colorData;
                Bitmap colorBitmap;
                sample.color.AcquireAccess(PXCMImage.Access.ACCESS_READ, PXCMImage.PixelFormat.PIXEL_FORMAT_RGB32, out
colorData);
                colorBitmap = colorData.ToBitmap(0, sample.color.info.width, sample.color.info.height);

                // Get Depth image data
                PXCMImage.ImageData depthData;
                Bitmap depthBitmap;
                sample.depth.AcquireAccess(PXCMImage.Access.ACCESS_READ, PXCMImage.PixelFormat.PIXEL_FORMAT_RGB32, out
depthData);
                depthBitmap = depthData.ToBitmap(0, sample.depth.info.width, sample.depth.info.height);

                // Update UI
                // Modified to access accelerometer
                Render(irLeftBitmap, irRightBitmap, colorBitmap, depthBitmap, accelX, accelY, accelZ);

                // Release resources
                irLeftBitmap.Dispose();
                sample.left.ReleaseAccess(irLeftData);

                irRightBitmap.Dispose();
```

```
                sample.right.ReleaseAccess(irRightData);

                colorBitmap.Dispose();
                sample.color.ReleaseAccess(colorData);

                depthBitmap.Dispose();
                sample.depth.ReleaseAccess(depthData);

                // Release frame
                senseManager.ReleaseFrame();
            }
        }

        private void Render(Bitmap bitmapLeft, Bitmap bitmapRight, Bitmap bitmapColor, Bitmap bitmapDepth, double x, double
y, double z)
        {
            BitmapImage bitmapImageLeft = ConvertBitmap(bitmapLeft);
            BitmapImage bitmapImageRight = ConvertBitmap(bitmapRight);
            BitmapImage bitmapImageColor = ConvertBitmap(bitmapColor);
            BitmapImage bitmapImageDepth = ConvertBitmap(bitmapDepth);

            // Update the WPF Image control
            this.Dispatcher.Invoke(System.Windows.Threading.DispatcherPriority.Normal, new Action(delegate()
            {
                imgLeftStream.Source = bitmapImageLeft;
                imgRightStream.Source = bitmapImageRight;
                imgColorStream.Source = bitmapImageColor;
                imgDepthStream.Source = bitmapImageDepth;

                // Added to access accelerometer
                lblX.Content = string.Format("Accel X: {0}", x);
                lblY.Content = string.Format("Accel Y: {0}", y);
                lblZ.Content = string.Format("Accel Z: {0}", z);

            }));
        }

        private BitmapImage ConvertBitmap(Bitmap bitmap)
        {
            BitmapImage bitmapImage = null;

            if (bitmap != null)
```

```
            {
                MemoryStream memoryStream = new MemoryStream();
                bitmap.Save(memoryStream, System.Drawing.Imaging.ImageFormat.Bmp);
                memoryStream.Position = 0;
                bitmapImage = new BitmapImage();
                bitmapImage.BeginInit();
                bitmapImage.StreamSource = memoryStream;
                bitmapImage.CacheOption = BitmapCacheOption.OnLoad;
                bitmapImage.EndInit();
                bitmapImage.Freeze();
            }

            return bitmapImage;
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            ShutDown();
        }

        private void btnExit_Click(object sender, RoutedEventArgs e)
        {
            ShutDown();
            this.Close();
        }

        private void ShutDown()
        {
            // Stop the Update thread
            update.Abort();

            // Dispose RealSense objects
            senseManager.Dispose();
            session.Dispose();
        }
    }
}
```

MainWindow.xaml

```
<Window x:Class="CameraStreams.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Intel® RealSense™ Depth Camera R200 Code Sample" Height="786.82" Width="720" Background="#FF293955"
WindowStartupLocation="CenterScreen" Closing="Window_Closing">
    <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
        <Label Foreground="White" Content="All Streams" FontFamily="Segoe UI Light" FontSize="22"
HorizontalAlignment="Center"/>
        <StackPanel Orientation="Horizontal">
            <StackPanel>
                <Label Foreground="White" Content="Left IR" FontFamily="Segoe UI Light" FontSize="14"
HorizontalAlignment="Center"/>
                <Border BorderBrush="White" Height="240" Width="320" BorderThickness="1" Margin="5">
                    <Image x:Name="imgLeftStream"/>
                </Border>
            </StackPanel>
            <StackPanel>
                <Label Foreground="White" Content="Right IR" FontFamily="Segoe UI Light" FontSize="14"
HorizontalAlignment="Center"/>
                <Border BorderBrush="White" Height="240" Width="320" BorderThickness="1" Margin="5">
                    <Image x:Name="imgRightStream"/>
                </Border>
            </StackPanel>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <StackPanel>
                <Label Foreground="White" Content="Color" FontFamily="Segoe UI Light" FontSize="14"
HorizontalAlignment="Center"/>
                <Border BorderBrush="White" Height="240" Width="320" BorderThickness="1" Margin="5">
                    <Image x:Name="imgColorStream"/>
                </Border>
            </StackPanel>
            <StackPanel>
                <Label Foreground="White" Content="Depth" FontFamily="Segoe UI Light" FontSize="14"
HorizontalAlignment="Center"/>
                <Border BorderBrush="White" Height="240" Width="320" BorderThickness="1" Margin="5">
                    <Image x:Name="imgDepthStream"/>
                </Border>
```

```
            </StackPanel>
        </StackPanel>
        <Label x:Name="lblX" Foreground="White" Content="-" FontFamily="Segoe UI Light" FontSize="14"/>
        <Label x:Name="lblY" Foreground="White" Content="-" FontFamily="Segoe UI Light" FontSize="14"/>
        <Label x:Name="lblZ" Foreground="White" Content="-" FontFamily="Segoe UI Light" FontSize="14"/>

        <Button x:Name="btnExit" Content="Exit" Height="31" FontSize="12" Click="btnExit_Click" Width="100" Margin="5"/>
    </StackPanel>
</Window>
```