**Contoso Jobs**

**Lab 4. Add Camera Support**

In this Lab you will add support for capturing an image from the camera on either side of the Surface Hub.

The camera will be used to capture an image to place in the background of the Job.

This Lab continues from the previous Lab, or you can start from the start point in the folder named *4. Start Contoso Jobs Camera*

1. Open the existing **Contoso Jobs.sln** solution file in **Visual Studio 2015**
2. In order to support the camera input in a UWP App, the App Manifest file must request access to the camera (and microphone) from the user.
3. In Microsoft Visual Studio, in **Solution Explorer**, open the designer for the application manifest by double-clicking the **package.appxmanifest** item.
4. Select the **Capabilities** tab.
5. Check the box for **Webcam** and the box for **Microphone**.

If you have built a UWP app to use a camera before you may have used the **CameraCaptureUI**. This is great for a phone or personal computer app, but it displays a modal dialog. A great Surface Hub app will not display a modal dialog and so you are going to add camera support into the JobControl directly.

1. Open the JobControl.xaml file and replace the Edit button with a Camera button. In the previous tutorial you removed the need for the edit button.

<Button Grid.Row="2" VerticalAlignment="Bottom" HorizontalAlignment="Right"

Background="Transparent" ToolTipService.ToolTip="Take picture"

FontFamily="Segoe MDL2 Assets" Content="&#xE722;" Click="Camera\_Click" />

1. In the Grid behind the Description TextBox add an Image control to host the photo.

<Grid Grid.Row="1">

<Image Name="Photo" />

<TextBox x:Name="DescriptionTextBlock" Margin="0" BorderThickness="0"

TextWrapping="Wrap" Text="{Binding Description, Mode=TwoWay}" Height="150"

VerticalAlignment="Center" FontSize="{StaticResource TextFontSize}"/>

<InkCanvas x:Name="Annotation" Height="150" IsHitTestVisible="False" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right"

Fill="Red" Margin="5, 0" PointerPressed="Ellipse\_PointerPressed" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right"

Fill="Blue" Margin="30, 0" PointerPressed="Ellipse\_PointerPressed" />

<Ellipse Width="20" Height="20" VerticalAlignment="Top"

HorizontalAlignment="Right"

Fill="Black" Margin="55, 0" PointerPressed="Ellipse\_PointerPressed" />

</Grid>

1. After that Grid and before the first Button add a new Grid to host a preview video feed from the camera.

<Grid Name="CameraControl" Visibility="Collapsed" Grid.Row="1" Grid.RowSpan="2">

<CaptureElement Name="PreviewControl" Stretch="UniformToFill" />

<Button VerticalAlignment="Bottom" HorizontalAlignment="Right" Background="White"

ToolTipService.ToolTip="Take picture"

FontFamily="Segoe MDL2 Assets" Content="&#xE722;" Click="Camera\_Click" />

</Grid>

1. Open the JobControl.xaml.cs file and add fields to the class for the **MediaCapture**, a flag for checking if the camera is in preview and a **DisplayRequest** object.

MediaCapture mediaCapture;

bool isPreviewing;

DisplayRequest displayRequest;

You will need to include the namespaces to support the MediaCaputre and DisplayRequest classes

using Windows.Media.Capture;

using Windows.System.Display;

1. Then add the code for the Camera\_Click event handler. If the camera is already showing a preview of it’s video feed then this method will capture a photo from that camera. If the camera is not displaying a preview of the video feed then it will start showing the preview of the video feed.

private async void Camera\_Click(object sender, RoutedEventArgs e)

{

if (isPreviewing)

{

CameraControl.Visibility = Visibility.Collapsed;

LowLagPhotoCapture lowLagCapture = await

mediaCapture.PrepareLowLagPhotoCaptureAsync(

ImageEncodingProperties.CreateUncompressed(MediaPixelFormat.Bgra8));

CapturedPhoto capturedPhoto = await lowLagCapture.CaptureAsync();

SoftwareBitmap softwareBitmap = capturedPhoto.Frame.SoftwareBitmap;

if (softwareBitmap.BitmapPixelFormat != BitmapPixelFormat.Bgra8

|| softwareBitmap.BitmapAlphaMode == BitmapAlphaMode.Straight)

{

softwareBitmap = SoftwareBitmap.Convert(softwareBitmap,

BitmapPixelFormat.Bgra8, BitmapAlphaMode.Premultiplied);

}

var source = new SoftwareBitmapSource();

await source.SetBitmapAsync(softwareBitmap);

Photo.Source = source;

await lowLagCapture.FinishAsync();

await CleanupCameraAsync();

isPreviewing = false;

}

else

{

CameraControl.Visibility = Visibility.Visible;

try

{

mediaCapture = new MediaCapture();

await mediaCapture.InitializeAsync();

PreviewControl.Source = mediaCapture;

await mediaCapture.StartPreviewAsync();

isPreviewing = true;

displayRequest.RequestActive();

DisplayInformation.AutoRotationPreferences =

DisplayOrientations.Landscape;

}

catch (UnauthorizedAccessException)

{

// This will be thrown if the user denied access to the camera

Debug.WriteLine("The app was denied access to the camera");

}

catch (Exception ex)

{

Debug.WriteLine(

"MediaCapture initialization failed. {0}", ex.Message);

}

}

}

You will also need to add the following namespaces to the class file.

using System.Diagnostics;

using Windows.Graphics.Display;

using Windows.Graphics.Imaging;

using Windows.Media.MediaProperties;

using Windows.UI.Xaml.Media.Imaging;

1. To ensure the camera resources are released add a method called CleanupCamraAsync

private async Task CleanupCameraAsync()

{

if (mediaCapture != null)

{

if (isPreviewing)

{

await mediaCapture.StopPreviewAsync();

}

await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>

{

PreviewControl.Source = null;

if (displayRequest != null)

{

displayRequest.RequestRelease();

}

});

mediaCapture.Dispose();

mediaCapture = null;

}

}

You will need to add two namespaces for this to complile

using System.Threading.Tasks;

using Windows.UI.Core;

1. You have added the code needed to capture an image from the default camera, if you run this on your development machine (assuming it has a webcam) you should be able to preview a video feed from the default camera and capture a photo.

For a Surface Hub app you need ot be able to specify which camera to use. Start with the right camera, add a **Tag** in the Button to identify the *Right* camera (jobcontrol.xaml)

<Button Grid.Row="2" VerticalAlignment="Bottom" HorizontalAlignment="Right"

Background="Transparent" ToolTipService.ToolTip="Take picture"

Tag="Right"

FontFamily="Segoe MDL2 Assets" Content="&#xE722;" Click="Camera\_Click" />

1. In the JobControl.xaml.cs fille add code at the top of the try block to select the camera to use for the video feed and therefore the picture, based on the Tag text in the Button.

try

{

MediaCaptureInitializationSettings mediaInitSettings =

new MediaCaptureInitializationSettings();

DeviceInformationCollection devices =

await DeviceInformation.FindAllAsync(DeviceClass.VideoCapture);

foreach (var device in devices)

{

EnclosureLocation loc = device.EnclosureLocation;

Debug.WriteLine($"Location: {loc.Panel.ToString()} ID: {device.Id}");

if (sender is Button

&& ((Button)sender).Tag.ToString() == loc.Panel.ToString())

{

mediaInitSettings.VideoDeviceId = device.Id;

break;

}

}

mediaCapture = new MediaCapture();

await mediaCapture.InitializeAsync(mediaInitSettings);

PreviewControl.Source = mediaCapture;

await mediaCapture.StartPreviewAsync();

isPreviewing = true;

displayRequest.RequestActive();

DisplayInformation.AutoRotationPreferences = DisplayOrientations.Landscape;

}

Add the namespace for the Devices Enumeration

using Windows.Devices.Enumeration;

1. The Progress button on the bottom left of the JobControl is no longer needed as a Job can be dragged and dropped into the location desired. Replace this Progress Button with another Camera Button that will use the Left camera.

<Button Grid.Row="2" VerticalAlignment="Bottom" HorizontalAlignment="Left"

Background="Transparent" ToolTipService.ToolTip="Take picture" Tag="Left"

FontFamily="Segoe MDL2 Assets" Content="&#xE722;" Click="Camera\_Click" />

1. Build and run the app (F5). If you are remotely deploying to a Surface Hub in Dev Mode you should be able to capture from either the left or the right camera.

If you are running on a development machine you might need to change the Left and Right Tag strings to Front and Back to match the location of your cameras.

On a Surface Hub you will find a bug if you try to use both cameras at the same time. This is because this code is previewing both the Left and Right camera at the highest resolution and the Surface Hub serial bus cannot cope with two simultaneous high definition video feeds.

1. To fix this bug with the two high resolution video feeds you are going to the lower the resolution of the camera video preview feed. In the try block where the MediaCapture is initialized change the code as follows

try

{

MediaCaptureInitializationSettings mediaInitSettings =

new MediaCaptureInitializationSettings();

DeviceInformationCollection devices =

await DeviceInformation.FindAllAsync(DeviceClass.VideoCapture);

foreach (var device in devices)

{

EnclosureLocation loc = device.EnclosureLocation;

Debug.WriteLine($"Location: {loc.Panel.ToString()} ID: {device.Id}");

if (sender is Button

&& ((Button)sender).Tag.ToString() == loc.Panel.ToString())

{

mediaInitSettings.VideoDeviceId = device.Id;

mediaInitSettings.StreamingCaptureMode = StreamingCaptureMode.Video;

mediaInitSettings.PhotoCaptureSource = PhotoCaptureSource.VideoPreview;

break;

}

}

mediaCapture = new MediaCapture();

await mediaCapture.InitializeAsync(mediaInitSettings);

var resolutions =

mediaCapture.VideoDeviceController.GetAvailableMediaStreamProperties(

MediaStreamType.Photo).Select(x => x as VideoEncodingProperties);

var minRes = resolutions.OrderBy(x => x.Height \* x.Width).FirstOrDefault();

await mediaCapture.VideoDeviceController.SetMediaStreamPropertiesAsync(

MediaStreamType.VideoPreview, minRes);

PreviewControl.Source = mediaCapture;

await mediaCapture.StartPreviewAsync();

isPreviewing = true;

}

1. Build and run the app (F5). If you are not running on a Surface Hub your cameras might not support multiple resolutions and this might create a problem. On a Surface Hub this will select the lowest resolution video feed available for the camera.

In this tutorial you have learned how to use the two cameras in Surface Hub to preview a video feed and capture a photo. To be useful for the Contoso Jobs application the Job object should store the photo. This is left as an exercise for you solve yourself.