Surface Dial

Lab 2. Adding a menu item to the Surface Dial menu

When you push the button on a Surface Dial a menu is displayed. This menu is called a RadialControllerMenu. In your application you can extend this RadialControllerMenu by adding new menu items and handling the selection of those new menu items.

It is recommended that you leave the default operating system menu items in pace when modifying the RadialControllerMenu. These items include the volume control and the scroll control. Leaving the default operating system menu items on the RadialControllerMenu creates a consistency of experience for customers working with the Surface Dial product.

In this lab you will learn how to add a new menu item to the RadialControllerMenu and handle the menu item being invoked.

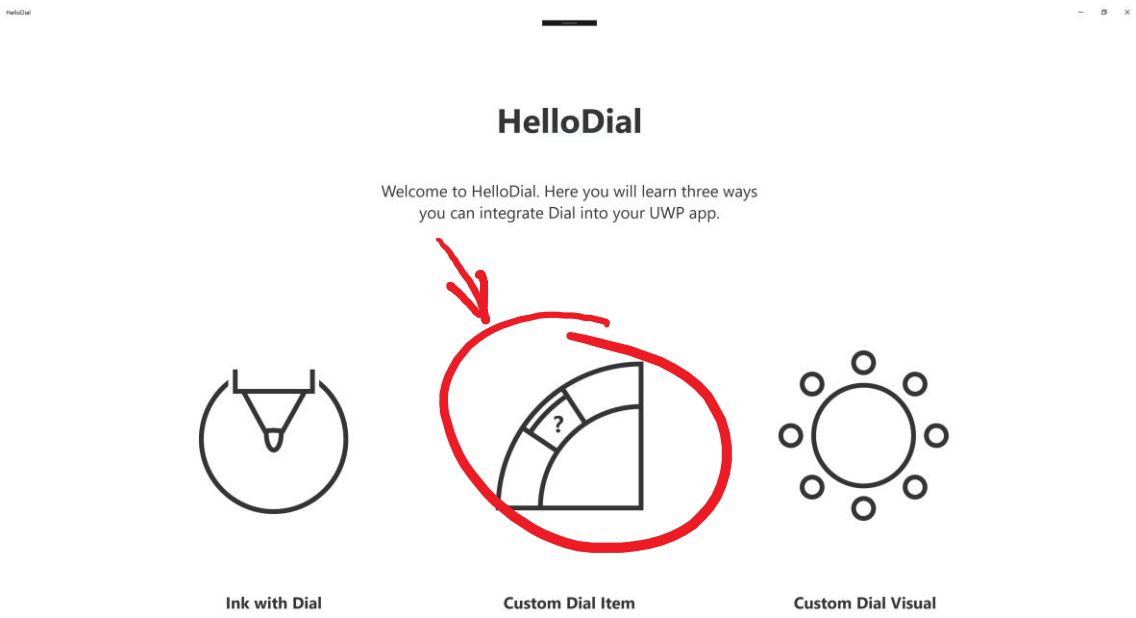
This lab assumes you have a working knowledge of C# and the Visual Studio IDE. To test the code in this lab you will need a Surface Dial.

Note: This lab has been built on a Surface Studio running a resolution of 4500 x 3000. You might need to change the layout on the Pages to work on your display.

1. Open the existing application:

    a. Find the folder named **3. Start HelloDial Menu Item**
    b. In this folder open the **HelloDial.sln** solution file in **Visual Studio 2015 or Visual Studio 2017**

2. Build and Run the application (F5) and you will see the start screen. In this hands on lab you will build the second scenario ‘Custom Dial Item’



3. In **Solution Explorer** open the **Pages** folder. You will see a Page called InkPage.xaml which was created in Lab 1: Dial with Ink. Instead open the **MenuItemPage.xaml** file. It contains:
    a. a back **Button**,
    b. a large **Image** placed centrally on the screen and
    c. a horizontally oriented **StackPanel** of images

4. Open the **MenuItemPage.xaml.cs** code file. In this file you will see the handler for the back **Button** already implemented.

5. At the top of the **MenuItemPage** class add:

```
using Windows.UI.Input;
```

6. Within the class, add member variables to hold the **RadialController** instance and the new **RadialControllerMenuItem** you are going to create.

```
private RadialController myController;
private RadialControllerMenuItem selectImageMenuItem;
```

7. Add a **DependecyProperty** to the **MenuItemPage** that can maintain the index of the currently selected image, call it **SelectedImage**

```
public int SelectedIndex
   {
```

```
        get { return (int)GetValue(SelectedIndexProperty); }
        set { SetValue(SelectedIndexProperty, value); }
    }

public static readonly DependencyProperty SelectedIndexProperty =
DependencyProperty.Register("SelectedIndex", typeof(int),
typeof(MenuItemPage), new PropertyMetadata(0));
```

8. At the top of the **MenuItemPage** class add:

```
using Windows.Storage.Streams;
```

9. Modify the constructor of the **MenuItemPage** class to retrieve the **RadialController** for this view, create a new menu item and add it to the **RadialControllerMenu**.

```
public MenuItemPage()
{
    this.InitializeComponent();

    RadialControllerConfiguration myConfiguration =
        RadialControllerConfiguration.GetForCurrentView();
    //determine which operating system menu items should be shown
    myConfiguration.SetDefaultMenuItems(new[]
    {
        RadialControllerSystemMenuItemKind.Volume,
        RadialControllerSystemMenuItemKind.Scroll
    });

    // Create a reference to the RadialController.
    myController = RadialController.CreateForCurrentView();

    // Create an icon for the custom tool.
    RandomAccessStreamReference icon =
        RandomAccessStreamReference.CreateFromUri(
            new Uri("ms-appx:///Assets/dial_icon_custom_item.png"));

    // Create a menu item for the custom tool.
    selectImageMenuItem =
        RadialControllerMenuItem.CreateFromIcon("Select Image", icon);

    // Add the custom tool to the RadialController menu.
    myController.Menu.Items.Add(selectImageMenuItem);
```

Continue by writing the code to call the methods MyController_RotationChanged and SelectImageItem_Invoked. These methods are created in the next step.

```
    // Declare input handlers for the RadialController.
    myController.RotationChanged += MyController_RotationChanged;

    selectImageMenuItem.Invoked += SelectImageItem_Invoked;
}
```

10. At the top of the **MenuItemPage** class add:

```
using System.Diagnostics;
```

11. Add an event handler method for the **RotationChanged** event in the **MenuItemPage** class. This will increment or decrement the selected index, and therefore change the selected image.

```
private async void MyController_RotationChanged(RadialController sender,
 RadialControllerRotationChangedEventArgs args)
{
   await Dispatcher.RunAsync(Windows.UI.Core.CoreDispatcherPriority.Normal, () =>
   {
     try
     {
         if (args.RotationDeltaInDegrees > 0
         && SelectedIndex < 6)
         {
             SelectedIndex++;
         }
         else if (args.RotationDeltaInDegrees < 0
             && SelectedIndex > 0)
         {
             SelectedIndex--;
         }

     }
     catch (Exception ex)
     {
         Debug.WriteLine(ex.Message);
     }
   });
}
```

12. Add the handler for the new MenuItem Invoked event. This will simply output a Debug message for now.

```
 private void SelectImageItem_Invoked(RadialControllerMenuItem sender, object args)
 {
     Debug.WriteLine("Button Invoked");
 }
```

13. Add a line in the BackButton_Click event handler method to remove the menu item from the RadialControllerMenu when the page is left.

```
private void BackButton_Click(object sender, RoutedEventArgs e)
{
   myController.Menu.Items.Remove(selectImageMenuItem);
   Frame.GoBack();
}
```

14. Build and Run the application (F5).

Go to the Custom Dial Item page and push down on the Surface Dial. You will see your new menu item in the **RadialControllerMenu**.
Rotate the Surface Dial to select the new menu item and invoke it by pushing down on the Surface Dial.
Now when you rotate the Surface Dial, you will change the selected image on the page.

In this Lab you have learned that the Surface Dial device is represented as a **RadialController** object to your view. In your code you can then create a new **RadialControllerMenuItem** and add it to the **RadialControllerMenu** for your view.

You should also have learned that until your **MenuItem** is invoked, the **RotationChanged** event is not raised.