# How to Add a Button and Gesture to GestureBar

Use this tutorial to add a new button to the example shell application included in StarPad called AnApp (./Apps/AJournal/AnApp).

**Add the Button**

1. Open Tab\_Home.xaml in Microsoft Expression Blend 2
2. Add a new GestureBarLib.GestureButton object to Tab\_Home.xaml
3. Position the button where you would like it to be
4. Give the button a name, such as “CircleButton”
5. Set the property Title, under Miscellaneous, to “Circle”
6. Set the TooltipTitle property to “Circle Tool”
7. Set the TooltipDescription property to “Use this tool to make a circle”

**Add the Icon**

1. Add a new UserControl to your project using Blend, call it “Icon\_Circle.xaml”
2. Make the control 50 x 35 pixels
3. Make an icon using any of the tools in Blend you like (e.g. make a Circle and add a gradient fill)
4. Build the project
5. Go back to Tab\_Home.xaml, and select the button you added
6. In the properties window, expand Miscellaneous
7. Next to the property DisplayIcon click the New button and in the ensuing dialog box, browse for Icon\_Circle and click OK
8. The icon should appear in the designer

**Add the Gesture Animation**

1. Add a new UserControl to your project using Blend, call it “Gesture\_Circle.xaml”
2. Make the control 65 x 65 pixels
3. Select the pencil tool in Blend
4. Draw a sequence of one or more marks, slowly and carefully; these marks will be procedurally converted by GestureBar into a demonstration animation; make one mark per stroke in the gesture – (e.g. for a single stroke gesture of a circle, you might draw a circle in a single smooth motion)
5. We will now add the detail tips. Add a canvas to the control, and name it DetailsLayer (it must be named details layer).
6. In this canvas you can put anything you want as children and they will appear after the demonstration animations; we recommend using the detail tips that have already been created; you can copy and paste those into the DetailsLayer canvas (for example, see Gestures/Gesture\_Ellipse.xaml).
7. Build the project
8. Go back to Tab\_Home.xaml, and select the button you added
9. In the properties window, expand Miscellaneous
10. Next to the property Stroke1 click the New button and in the ensuing dialog box, browse for Gesture\_Circle and click OK
11. Give a title to this gesture variation by setting Stroke1Title
12. Give a feature description using Stroke1FeatureDescription
13. Repeat the previous section to make an icon for this gesture variation, and then set it under Stroke1Icon
14. Repeat this section for any additional gesture variations that you want to make

**Add the Try It Area Overlay**

1. Add a new User Control to your project called Overlay\_Circle.xaml
2. Make the control be about 110 x 110 pixels
3. Use the Pencil tool to draw a tracing overlay for the user to trace in the Try It Area
4. Set its opacity to 25%
5. Build the project
6. Go back to Tab\_Home.xaml, and select the button you added
7. In the properties window, expand Miscellaneous
8. Next to the property Stroke1TryItOverlay click the New button and in the ensuing dialog box, browse for Overlay\_Circle and click OK

**Test Your Work So Far**

1. Run your project to see the results, and make changes as necessary (no practice area should appear as you have not defined it yet)
2. Change AnimationSpeed (or AnimationSpeed2, etc.) to speed up or slow down your demonstration animation

**Add the Practice Area**

This part requires a little bit of code in a callback to notify the practice area of the user’s performance

1. Add a new function to Window1.xaml.cs that can tell whether a fired gesture matches the one described by this GestureBar Button:

protected void ConfigureCircleDrawingCanvas(GestureExplorer sender,

FrameworkElement elt)

{

APage page = (APage)elt;

// don't want to see the gesture control, but it has to be non-transparent to get events

page.Background = new SolidColorBrush(Color.FromArgb(1, 255, 255, 255));

// install a function that reports gesture success/failure to the GestureExplorer (remove it first in case this gets called multiple times)

page.Commands.GestureRecognizer.GestureRecognizedEvent -=

new Gesturizer.GestureRecognizedHandler((object s, Gesture g, Stroq[] strokes) => sender.ReportTryItResult(g is ScribbleTapCommand, ""));

page.Commands.GestureRecognizer.GestureRecognizedEvent +=

new Gesturizer.GestureRecognizedHandler((object s, Gesture g, Stroq[] strokes) => sender.ReportTryItResult(g is ScribbleTapCommand, ""));

// optionally install a callback method that reports non-gestures to the GestureExplorer as gesture failures

// page.Commands.NonGestureEvent -= new Gesturizer.StrokeUnrecognizedHandler((object sender, Stroq stroke) => { sender.ReportTryItResult(false, ""); return true; });

// page.Commands.NonGestureEvent += new Gesturizer.StrokeUnrecognizedHandler((object sender, Stroq stroke) => { sender.ReportTryItResult(false, ""); return true; });

}

1. Add the following code to the end of Window\_Initialized:  
     
     
   GestureButton button = gestureBar.FindButtonByTitle("Circle");

button.ConfigureDrawingCanvasFunction1 += new GestureButton.ConfigureDrawingCanvasFunc(ConfigureCircleDrawingCanvas);

**Test Your Work So Far**

1. Run your project to see the results, and make changes as necessary (no practice area should appear as you have not defined it yet)
2. If everything works you are done!