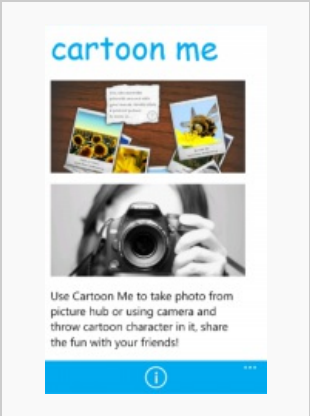
CartoonMe - Gestures on Windows Phone

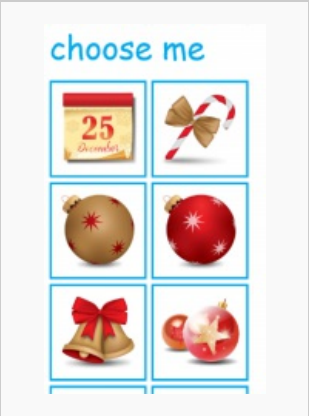
This article demonstrates the use of Gestures, Camera and Photo chooser tasks for Windows Phone.

Introduction

CartoonMe is an application that allows user to take a photo from Picture Hub or from Camera using Choosers for Windows Phone and throw in cartoon items/objects. To adjust the selected object on top of the photo we used multi-touch gestures from Windows Phone Toolkit, which allows Drag, Pinch, and Rotation of the object.



Home Screen



Choose item from list

Add item on top of photo

Steps

- Create an empty Windows Phone project.
- Selecting photo from Picture Hub
- Taking picture from Camera
- Adding object on the photo
- Implementing Gesture Listener
- Saving and sharing the image.

Implementation

Create an empty Windows Phone Project

- Launch Visual Studio
- Click on File
- New Project
- Select Windows Phone Application (Visual C# Template)
- Add Name and Location of the project
- Click OK to create the project.

Selecting photo from Picture Hub

To select a photo from phone picture hub first we launch the picture gallery using `PhotoChooserTask` Class and select a photo. When the task is completed, an event is generated and the event handler receives a photo in the result. We get the photo from the event argument and display on the screen in an Image element.

```
void photoChooserTask_Completed(object sender, PhotoResult e)
        {
            if (e.TaskResult == TaskResult.OK)
            {
                App.isFunnyPhotoSelected = true;
                  e.ChosenPhoto.Position = 0;
                JpegInfo info = ExifReader.ReadJpeg(e.ChosenPhoto, e.OriginalFileName);
                _width = info.Width;
```


(C) Copyright Nokia 2014. All rights reserved.

```
                _height = info.Height;
                _orientation = info.Orientation;
                switch (info.Orientation)
                {
                    case ExifOrientation.TopLeft:
                    case ExifOrientation.Undefined:
                        _angle = 0;
                        break;
                    case ExifOrientation.TopRight:
                        _angle = 90;
                        break;
                    case ExifOrientation.BottomRight:
                        _angle = 180;
                        break;
                    case ExifOrientation.BottomLeft:
                        _angle = 270;
                        break;
                }
                if (_angle > 0d)
                {
                    capturedImage = RotateStream(e.ChosenPhoto, _angle);
                }
                else
                {
                    capturedImage = e.ChosenPhoto;
                }
                BitmapImage bmp = new BitmapImage();
                bmp.SetSource(capturedImage);
                myMainImage.Source = bmp;
            }
            else
            {
                App.isFunnyPhotoSelected = false;
            }
        }
```

While debugging the application you might experience that the `PhotoChooserTask` Class doesn't launch the Picture Library of the phone when it's connected with Zune, to overcome this use Connect Tool to connect the device with the PC. To learn more about `PhotoChooserTask` see MSDN.

Taking picture from Camera

Alternatively user can also take photo from camera using `CameraCaptureTask` Class. `CameraCaptureTask` launches the built-in camera application of the phone, when user completes the task an event is raised and the event handler receives an image in the result. We get the photo from the event argument and displays on to the screen in an Image element. To learn more on `CameraCaptureTask` see MSDN.

```
void cameraCaptureTask_Completed(object sender, PhotoResult e)
        {
            if (e.TaskResult == TaskResult.OK)
            {
                App.isFunnyPhotoSelected = true;
                              e.ChosenPhoto.Position = 0;
               JpegInfo info = ExifReader.ReadJpeg(e.ChosenPhoto, e.OriginalFileName);
                _width = info.Width;
                _height = info.Height;
```


(C) Copyright Nokia 2014. All rights reserved.

```
                _orientation = info.Orientation;
                 switch (info.Orientation)
                 {
                     case ExifOrientation.TopLeft:
                     case ExifOrientation.Undefined:
                         _angle = 0;
                         break;
                     case ExifOrientation.TopRight:
                         _angle = 90;
                         break;
                     case ExifOrientation.BottomRight:
                         _angle = 180;
                         break;
                     case ExifOrientation.BottomLeft:
                         _angle = 270;
                         break;
                 }
                if (_angle > 0d)
                {
                    capturedImage = RotateStream(e.ChosenPhoto, _angle);
                }
                else
                {
                    capturedImage = e.ChosenPhoto;
                }

                BitmapImage bmp = new BitmapImage();
                bmp.SetSource(capturedImage);
                myMainImage.Source = bmp;
            }
            else
            {
                App.isFunnyPhotoSelected = false;
            }
        }
```

Adding object on the photo

We burn some images on to the application folder and created an xml file for tracking. When user clicks to add object it parse the xml and display the images on to a ListBox. User can select any object from the list to be added on top of the main photo.

```
private void LoadChoosePhotos()
        {
            try
            {
                XElement xmlData = XElement.Load("Xml/ChoosePhoto.xml");
                XNamespace ns = "";
                string LocalIconsPath = "/Images/Characters/";
                ListBoxPhotos.Items.Clear();
                foreach (var item in xmlData.Descendants("item"))
                {
                    RssPhotos rssfunnyphotos = new RssPhotos();
                    rssfunnyphotos.Image = LocalIconsPath + (string)item.Element(ns +
"image").Value;
                    ListBoxPhotos.Items.Add(rssfunnyphotos);
                }
```


(C) Copyright Nokia 2014. All rights reserved.

```
            }
            catch (System.Xml.XmlException ex)
            {
            }
        }
```

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,0,0">
            <ListBox   x:Name="ListBoxPhotos" Margin="0,0,0,0"
Loaded="ListBoxPhotos_Loaded" SelectionChanged="ListBoxPhotos_SelectionChanged" >
                <ListBox.ItemsPanel>
                    <ItemsPanelTemplate>
                        <toolkit:WrapPanel ItemHeight="220" ItemWidth="220"/>
                    </ItemsPanelTemplate>
                </ListBox.ItemsPanel>
                <ListBox.ItemTemplate>
                    <DataTemplate>
                        <StackPanel Margin="0,0,0,0">
                            <Border BorderBrush="#00ABEE" BorderThickness="5"
Height="210" Width="210" HorizontalAlignment="Center"  VerticalAlignment="Center"   >
                                <Image Stretch="Fill" Source="{Binding Image}"
Height="170" Width="170"  HorizontalAlignment="Center" VerticalAlignment="Center" />
                            </Border>
                        </StackPanel>
                    </DataTemplate>
                </ListBox.ItemTemplate>
            </ListBox>
        </Grid>
```

Implementing Gesture Listener

Till now we will see the selected object on top of the main picture. But to adjust the size and space of the object with the selected photo we need to put it in the right place and space it perfect. Windows Phone Toolkit provides `GestureService` which allows developers to implement Drag, Pinch etc. gestures on an element, in this case the selected image. To begin with let's add `Microsoft.Phone.Controls.Toolkit.dll` assembly in your Windows Phone project. And then add the following namespace declaration in XAML.

```
xmlns:toolkit="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
```

The `GestureListener` is an event listener which allows developers to detect touch gestures on to the application screen. We add the `GestureListener` component on to the selected object.

```
<Image  x:Name="imageSelectedCartoonPhoto" Visibility="Collapsed">
                    <Image.RenderTransform>
                        <CompositeTransform x:Name="ImageTransformation" />
                    </Image.RenderTransform>
                    <toolkit:GestureService.GestureListener>
                        <toolkit:GestureListener
                                    PinchStarted="OnPinchStarted"
                                    PinchDelta="OnPinchDelta"
                                    DragDelta="OnDragDelta"/>
                    </toolkit:GestureService.GestureListener>
             </Image>
```


(C) Copyright Nokia 2014. All rights reserved.

In this application we used:

- **DragDelta** : The `DragDelta` event is raised when there is a change in movement. It gives the value of change in horizontal and vertical position of the element.

```
  private void OnDragDelta(object sender, DragDeltaGestureEventArgs e)
        {
            Image rect = sender as Image;
            TranslateTransform transform = rect.RenderTransform as TranslateTransform;
            // Move the rectangle
            ImageTransformation.TranslateX += e.HorizontalChange;
            ImageTransformation.TranslateY += e.VerticalChange;
        }
```

- **PinchStarted** : It gives the initial angle and scale of the object.

```
private void OnPinchStarted(object sender, PinchStartedGestureEventArgs e)
        {
            _initialAngle = ImageTransformation.Rotation;
            _initialScale = ImageTransformation.ScaleX;
            Point firstTouch = e.GetPosition(imageSelectedCartoonPhoto, 0);
            Point secondTouch = e.GetPosition(imageSelectedCartoonPhoto, 1);
            Point center = new Point(firstTouch.X + (secondTouch.X - firstTouch.X) /
2.0,
                                     firstTouch.Y + (secondTouch.Y - firstTouch.Y) /
2.0);
            ImageTransformation.CenterX = center.X;
            ImageTransformation.CenterY = center.Y;
        }
```

- **PinchDelta** : It allows two touch input and provides the value of change in angle and scale of the object.

```
private void OnPinchDelta(object sender, PinchGestureEventArgs e)
        {
            ImageTransformation.Rotation = _initialAngle + e.TotalAngleDelta;
            ImageTransformation.ScaleX = _initialScale * e.DistanceRatio;
            ImageTransformation.ScaleY = ImageTransformation.ScaleX;
        }
```

Saving and sharing the image

After we place and scale the object we are ready to save and share it with our friends. To save the picture in phone's Saved Pictures album we call `Save_Click()` method.

```
        private void Save_Click(object sender, EventArgs e)
        {
            WriteableBitmap bmp = new WriteableBitmap((int)this.ActualWidth,
(int)this.ActualHeight);
            bmp.Render(this, null);
            byte[] bb = imageBits = EncodeToJpeg(bmp);
            bmp.Invalidate();
            MemoryStream mem = new MemoryStream();
            bmp.SaveJpeg(mem, bmp.PixelWidth, bmp.PixelHeight, 0, 100);
            mem.Seek(0, System.IO.SeekOrigin.Begin);
```


(C) Copyright Nokia 2014. All rights reserved.

```
            if (mem != null)
            {
                  MediaLibrary library = new MediaLibrary();
                try
                {
                    Picture pic = library.SavePicture(Guid.NewGuid().ToString(), mem);
                    MessageBox.Show("Your picture is now accessible through the Saved
Picture album.", "Saved successfully.", MessageBoxButton.OK);
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Unable to save the photo.");
                }
            }
        }
```

We can also save it to phone local storage for future use. ShareMediaTask Class provides an easy way of media sharing on Windows Phone 8. As `ShareMediaTask` doesn't supports on Windows Phone 7, we can upload the media file to a remote location and post the remote URL on to our social sites using ShareLinkTask Class. To learn more see Launchers and Choosers for Windows Phone.

Note: Sharing for Windows Phone 7 doesn't explain in this example.

Summary

During the course of the development you might come across to a place where the camera captured picture always display in landscape mode, no matter how you held the device (portrait or landscape) while taking the photo. To overcome see this article. You can also find a discussion of this article here.

Source Code

- The full source code of the example is available here: File:CartoonMe Wp8.zip : File:CartoonMe Wp7.zip
- Updated source code : File:CartoonMe Updated v1.1.zip

(C) Copyright Nokia 2014. All rights reserved.