ComponentOne

Imaging for UWP

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of GrapeCity, Inc. All other trademarks used herein are the properties of their respective owners.

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

Table of Contents

Copyright © 2017 GrapeCity, inc. All rights reserved.

Imaging for UWP

Load images (PNG and JPG), edit pixel by pixel, and show in an image tag or save to a stream with **Bitmap for UWP (**C1Bitmap**)**.

Display animated GIF images on your Windows store application as you would in traditional Web applications with **Image for UWP (**C1Image**)**. Animated GIFs are compact and allow you to add attractive visual elements to your applications with minimal effort.

Help with UWP Edition

Getting Started

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with ComponentOne Studio UWP Edition.

Bitmap for UWP

Load images (PNG and JPG), edit pixel by pixel, and show in an image tag or save to a stream with **Bitmap for UWP (**C1Bitmap**)**.

Bitmap for UWP Features

Reduce/Crop Images

Editing the pixels enables you to resize images and reduce the resolution, which reduces the file size and results in faster upload time. Also, you can crop users' images in order to upload only part of them as you do in Facebook or any other web user account.

Edit Images Programmatically

The C1Bitmap class allows you to access individual pixels to create special effects, crop, resize, or transform images in any way you want.

Save Generated Images as JPG/PNG

You can take screen shots using the **WritableBitmap**, pass it to **C1Bitmap**, and save the result into a new PNG/JPG file on the fly.

Bitmap for UWP Quick Start

The following quick start guide is intended to get you up and running with **Bitmap for UWP**. In this quick start, you'll create a new Windows Store application that allows users to load a default image and then crop it.

Step 1 of 4: Creating a Windows Store Application

In this step you'll create a Windows Store application in Visual Studio using Bitmap for UWP.

To set up your project, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. Select Templates | Visual C# | Windows | Universal. From the templates list, select Blank App (Universal Windows).
3. Enter a Name and click **OK** to create your project. Open the XAML view of the MainPage.xaml file; in this quick

Copyright © 2017 GrapeCity, inc. All rights reserved.

start you'll add controls using XAML markup.
4. Right-click the project name in the Solution Explorer and select **Add Reference**.
5. In the **Reference Manager** dialog box, select **ComponentOne Studio UWP Edition** and click **OK** to close the dialog box and add the reference.

In the next step, you'll set the styles and add an image to the project.
Step 2 of 4: Adding an Image

Step 2 of 4: Adding an Image

In this step, you will add the following XAML to set the image styles and create a new image.

1. Add the XAML within the <UserControl> tags and overwrite the default <Grid> tags.

Markup

```
<UserControl.Resources>
  <SolidColorBrush Color="#66FFFFFF" x:Key="MaskBrush" />
 </UserControl.Resources>
 <Grid>
  <Grid.RowDefinitions>
   <RowDefinition Height="Auto" />
   <RowDefinition />
  </Grid.RowDefinitions>
  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">
   <Button Content="Load your own image" Click="LoadImage" Margin="0 0 10 0"
Width="180" HorizontalAlignment="Left" />
   <Button Content="Export selection" Click="ExportImage" Grid.Column="1"
Width="140" />
  </StackPanel>
  <Grid Name="imageGrid" Grid.Row="1" HorizontalAlignment="Center"
VerticalAlignment="Center">
   <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
    <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
    <ColumnDefinition Width="Auto" />
   </Grid.ColumnDefinitions>
   <Image Stretch="None" Name="image" Grid.RowSpan="3" Grid.ColumnSpan="3" />
   <Grid Name="topMask" Grid.ColumnSpan="2" Background="{StaticResource
MaskBrush}" />
   <Grid Name="bottomMask" Grid.Column="1" Grid.Row="2" Grid.ColumnSpan="2"
Background="{StaticResource MaskBrush}" />
   <Grid Name="leftMask" Grid.RowSpan="2" Grid.Row="1" Background="
{StaticResource MaskBrush}" />
   <Grid Name="rightMask" Grid.Column="2" Grid.RowSpan="2" Background="
{StaticResource MaskBrush}" />
  </Grid>
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
</Grid>
```

2. Add an image to the project:

   - Select **Project | Add Existing** Item.
   - Browse to find an image. In this example, we use the Lenna.jpg image from the sample - **C1.UWP.Imaging** sample provided with **ComponentOne Studio UWP Edition**.
   - Select the image and click **Add**.

In the next step, you'll add the code used to crop the image.

Step 3 of 4: Adding Code for Image Cropping

The code in this step will load the default image and allow the user to crop it.
Follow these steps:

1. Open the MainPage.xaml.cs file and add the following using (Imports in Visual Basic) statements.

C#

```
using C1.Xaml;
using C1.Xaml.Imaging;
using System.IO;
```

2. Add the following code to load a default image and define cropping:

C#

```
public partial class MainPage : UserControl
    {
        C1Bitmap bitmap = new C1Bitmap();
        Rect selection;

        public MainPage()
        {
            InitializeComponent();
            LoadDefaultImage();
            image.Source = bitmap.ImageSource;
            var mouseHelper = new C1DragHelper(imageGrid);
            mouseHelper.DragStarted += OnDragStarted;
            mouseHelper.DragDelta += OnDragDelta;
        }
        void OnDragDelta(object sender, C1DragDeltaEventArgs e)
        {
            var transform = Window.Current.Content.TransformToVisual(image);
            var start = transform.TransformPoint(_startPosition);
            var end = transform.TransformPoint(e.GetPosition(null));
            start.X = Math.Min((double)Math.Max(start.X, 0), bitmap.Width);
            end.X = Math.Min((double)Math.Max(end.X, 0), bitmap.Width);
            start.Y = Math.Min((double)Math.Max(start.Y, 0), bitmap.Height);
            end.Y = Math.Min((double)Math.Max(end.Y, 0), bitmap.Height);
            selection = new Rect(new Point(
                Math.Round(Convert.ToDouble(Math.Min(start.X, end.X))),
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
                Math.Round(Convert.ToDouble(Math.Min(start.Y, end.Y)))),
                new Size(Convert.ToDouble(Math.Round(Math.Abs(start.X -
end.X))),
                    Convert.ToDouble(Math.Round(Math.Abs(start.Y - end.Y)))));
            UpdateMask();
}
        void UpdateMask()
        {
            topMask.Height = selection.Top;
            bottomMask.Height = bitmap.Height - selection.Bottom;
            leftMask.Width = selection.Left;
            rightMask.Width = bitmap.Width - selection.Right;
        }

        void LoadDefaultImage()
        {
            Assembly asm = typeof(Crop).GetTypeInfo().Assembly;
            Stream stream =
asm.GetManifestResourceStream("ImageSamplesLib2012.Resources.Lenna.jpg");
            LoadImageStream(stream);
        }
       void LoadImageStream(Stream stream)
        {
            bitmap.SetStream(stream);

            imageGrid.Width = bitmap.Width;
            imageGrid.Height = bitmap.Height;
            selection = new Rect(0, 0, bitmap.Width, bitmap.Height);
            UpdateMask();
        }
    }
```

In the next step you will run the application.

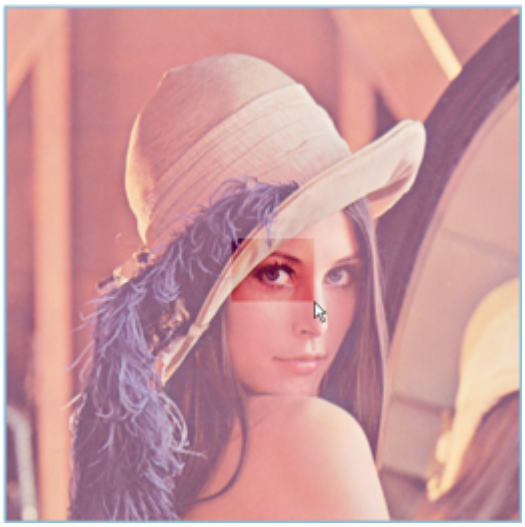
Step 4 of 4: Running the Application

Run the application.

1. From the Debug menu, select Start Debugging to view the image.
2. Click on the image and keep the left mouse button pressed while dragging the cursor. The ImagingSamples2015 sample provided with UWP Edition shows you how to export the cropped image and save it to a file.

   In the following image, notice the eye area is cropped.

Copyright © 2017 GrapeCity, inc. All rights reserved.



Congratulations! You have successfully completed the Image for UWP quick start.

Working with Bitmap for UWP

Bitmap for UWP allows you to complete many different tasks. By combining **Bitmap for UWP** with some of the baked-in UWP capabilities, you can crop images, warp images, and render simple XAML framework elements. The following topics will also discuss exporting your image, loading your own image, and restarting an event .

We'll take images and code from the ImageSamples2012 sample for the following topics. You can find this sample located in **C:\Users\<username>\Documents\ComponentOne Samples\UWP**.

Cropping with a Draggable Crop Box

Being able to crop an image entirely on the client is extremely useful. With C1Bitmap or the **WriteableBitmap** class, this is achievable in your Windows Store applications. The C1Bitmap component provides an API that is easier to work with when doing any bitmap related manipulation. This is primarily because it can get and set simple colors and it gives more direct access to pixels with the GetPixel and SetPixel methods.

Here is the XAML that defines the elements needed to create our crop area. To create the crop area, we'll create an image mask in XAML through adding <Grid> elements to the XAML markup and apply it to the image based on user selection in the code behind.

Markup

```
<UserControl.Resources>
            <SolidColorBrush Color="#66FFFFFF" x:Key="MaskBrush" />
      </UserControl.Resources>
<Grid>
            <Grid.RowDefinitions>
                  <RowDefinition Height="Auto" />
                  <RowDefinition />
            </Grid.RowDefinitions>
<Grid Name="imageGrid" Grid.Row="1" HorizontalAlignment="Center"
VerticalAlignment="Center">
                  <Grid.RowDefinitions>
                        <RowDefinition Height="Auto" />
                        <RowDefinition Height="*" />
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="*" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Stretch="None" Name="image" Grid.RowSpan="3"
Grid.ColumnSpan="3" />
                <Grid Name="topMask" Grid.ColumnSpan="2" Background="
{StaticResource MaskBrush}" />
                <Grid Name="bottomMask" Grid.Column="1" Grid.Row="2"
Grid.ColumnSpan="2" Background="{StaticResource MaskBrush}" />
                <Grid Name="leftMask" Grid.RowSpan="2" Grid.Row="1" Background="
{StaticResource MaskBrush}" />
                <Grid Name="rightMask" Grid.Column="2" Grid.RowSpan="2"
Background="{StaticResource MaskBrush}" />
            </Grid>
        </Grid>
```

The code below implements the cropping effect for your image. It uses the OnDragStarted and OnDragDelta events to capture the horizontal and vertical changes as a user chooses the section of the image to crop. The code also uses some math to get the values of the start and end points and to convert the points into a Rect object.

C#

```
Point _startPosition;
        void OnDragStarted(object sender, C1DragStartedEventArgs e)
        {
            _startPosition = e.GetPosition(null);
        }

        void OnDragDelta(object sender, C1DragDeltaEventArgs e)
        {
            var transform = Window.Current.Content.TransformToVisual(image);
            var start = transform.TransformPoint(_startPosition);
            var end = transform.TransformPoint(e.GetPosition(null));
            start.X = Math.Min((double)Math.Max(start.X, 0), bitmap.Width);
            end.X = Math.Min((double)Math.Max(end.X, 0), bitmap.Width);
            start.Y = Math.Min((double)Math.Max(start.Y, 0), bitmap.Height);
            end.Y = Math.Min((double)Math.Max(end.Y, 0), bitmap.Height);

            selection = new Rect(new Point(
                Math.Round(Convert.ToDouble(Math.Min(start.X, end.X))),
                Math.Round(Convert.ToDouble(Math.Min(start.Y, end.Y)))),
                new Size(Convert.ToDouble(Math.Round(Math.Abs(start.X - end.X))),
                    Convert.ToDouble(Math.Round(Math.Abs(start.Y - end.Y)))));

            UpdateMask();
        }
```

We apply some logic for the bounds of each portion of the mask.

Copyright © 2017 GrapeCity, inc. All rights reserved.

C#

```
void UpdateMask()
        {
            topMask.Height = selection.Top;
            bottomMask.Height = bitmap.Height - selection.Bottom;
            leftMask.Width = selection.Left;
            rightMask.Width = bitmap.Width - selection.Right;
        }
```

The **UpdateMask()** method updates the position of all the **Grid** mask elements based on the **Left**, **Top**, **Width** and **Height** properties of the **Grid Rect**.

C#

```
void UpdateMask()
        {
            topMask.Height = selection.Top;
            bottomMask.Height = bitmap.Height - selection.Bottom;
            leftMask.Width = selection.Left;
            rightMask.Width = bitmap.Width - selection.Right;
         }
```

Exporting an Image

Using some simple code and a general button control, you can export your cropped image. Here's the XAML markup for the general button control:

Markup

```
<Button Content="Export selection" Click="ExportImage" Grid.Column="1" Width="140" />
```

The code you'll use to control the export function allows you to block the export option if there's no cropped section to export. If there is a cropped section to export, then you can choose the file type and destination.

C#

```
private async void ExportImage(object sender, RoutedEventArgs e)
        {
            if(selection.Width == 0 || selection.Height == 0)
            {
                MessageDialog md = new MessageDialog("Can't export, selection is
empty");
                md.ShowAsync();
                return;
            }
            var picker = new FileSavePicker();
            picker.FileTypeChoices.Add("png", new List<string>{".png"});
            picker.DefaultFileExtension = ".png";
            StorageFile file = await picker.PickSaveFileAsync();
            if (file != null)
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
            {
                var saveStream = await file.OpenStreamForWriteAsync();
                var crop = new C1Bitmap((int)selection.Width, (int)selection.Height);
                crop.BeginUpdate();
                for (int x = 0; x < selection.Width; ++x)
                {
                    for (int y = 0; y < selection.Height; ++y)
                    {
                        crop.SetPixel(x, y, bitmap.GetPixel(x + (int)selection.X, y +
(int)selection.Y));
                    }
                }
```

Loading Your Own Image

You can also load your own image to crop. This can be accomplished easily with a general button control and some code behind.

Markup

```
<Button Content="Load your own image" Click="LoadImage" Margin="0 0 10 0" Width="180"
HorizontalAlignment="Left" />
```

The code that that responds to the click event will open a File Picker and allow you to choose the image file you wish to display and then crop.

C#

```
private async void LoadImage(object sender, RoutedEventArgs e)
        {
            var picker = new FileOpenPicker();
            picker.FileTypeFilter.Add(".png");
            picker.FileTypeFilter.Add(".jpg");
            picker.FileTypeFilter.Add(".gif");
            picker.FileTypeFilter.Add(".jpeg");
            StorageFile file = await picker.PickSingleFileAsync();

            if (file != null)
            {
                using (var fileStream = await file.OpenStreamForReadAsync())
                {
                    try
                    {
                        LoadImageStream(fileStream);
                    }
                    catch (Exception ex)
                    {
                        LoadDefaultImage();
                        MessageDialog md = new MessageDialog("Image format not
supported, error: \n" + ex.Message, "");
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
                    md.ShowAsync();
                }
            }
        }
    }
```

Warping an Image

Just for fun, you might want to be able to warp an image. The code needed to create the warp effect is somewhat complex, but it uses the C1DragHelpers and some math to apply the warp effect to your image. In the FaceWarp sample, you can load your own image, export your completed image, and reset the image if you don't like how the warp looks.

C#

```
InitializeComponent();
            LoadDefaultImage();
            image.Source = screen.ImageSource;
            var mouseHelper = new C1DragHelper(image, captureElementOnPointerPressed:
true);
            var line = new Line();
            mouseHelper.DragStarted += (s, e) =>
            {
                _position = e.GetPosition(image);
                line = new Line
                {
                    X1 = _position.X,
                    Y1 = _position.Y,
                    X2 = _position.X,
                    Y2 = _position.Y,
                    Stroke = new SolidColorBrush(Colors.Blue),
                    StrokeThickness = 7,
                    StrokeEndLineCap = PenLineCap.Triangle,
                    StrokeStartLineCap = PenLineCap.Round
                };
                imageGrid.Children.Add(line);
            };
            mouseHelper.DragDelta += (s, e) =>
            {
                var pos = e.GetPosition(image);
                line.X2 = pos.X;
                line.Y2 = pos.Y;
            };
            mouseHelper.DragCompleted += (s, e) =>
            {
                imageGrid.Children.Remove(line);
                var start = _position;
                var end = new Point(_position.X + e.CumulativeTranslation.X,
_position.Y + e.CumulativeTranslation.Y);
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
                bitmap = new C1Bitmap(screen);
                Warp(bitmap, screen, start, end);
            };
        }
void Warp(C1Bitmap src, C1Bitmap dst, Point start, Point end)
        {
            dst.BeginUpdate();
            dst.Copy(src, false);
            var dist = Distance(start, end);
            var affectedDist = dist * 1.5;
            var affectedDistSquared = affectedDist * affectedDist;
            for (int row = 0; row < dst.Height; ++row)
            {
                for (int col = 0; col < dst.Width; ++col)
                {
                    var point = new Point(col, row);
                    if (DistanceSq(start, point) > affectedDistSquared)
                    {
                        continue;
                    }
                    if (DistanceSq(end, point) < 0.25)
                    {
                        dst.SetPixel(col, row, src.GetPixel((int)start.X,
(int)start.Y));
                        continue;
                    }
                    var dir = new Point(point.X - end.X, point.Y - end.Y);
                    var t = IntersectRayCircle(end, dir, start, affectedDist);
                    TryT(-end.X / dir.X, ref t);
                    TryT(-end.Y / dir.Y, ref t);
                    TryT((dst.Width - end.X) / dir.X, ref t);
                    TryT((dst.Height - end.X) / dir.X, ref t);
                    var anchor = new Point(end.X + (point.X - end.X) * t, end.Y +
(point.Y - end.Y));
                    var x = start.X + (anchor.X - start.X) / t;
                    var y = start.Y + (anchor.Y - start.Y) / t;
                    dst.SetPixel(col, row, src.GetInterpolatedPixel(x, y));
                }
            }
            dst.EndUpdate();
        }
        static double Distance(Point a, Point b)
        {
            return Math.Sqrt(DistanceSq(a, b));
        }
        static double DistanceSq(Point a, Point b)
        {
            var dx = a.X - b.X;
            var dy = a.Y - b.Y;
            return dx * dx + dy * dy;
        }
```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
        static void TryT(double t2, ref double t)
        {
            if (t2 > 0 && t2 < t)
            {
                t = t2;
            }
        }
        static double IntersectRayCircle(Point rayOri, Point rayDir, Point center,
double radius)
        {
            var a = rayDir.X;
            var b = rayOri.X;
            var c = center.X;
            var d = rayDir.Y;
            var e = rayOri.Y;
            var f = center.Y;
            var g = radius * radius;
            var num1 = Math.Sqrt(d * (2 * a * (b - c) * (e - f) - d * (b * b - 2 * b
* c + c * c - g)) - a * a * (e * e - 2 * e * f + f * f - g));
            var num2 = a * (c - b) + d * (f - e);
            return (num1 + num2 > 0 ? num1 + num2 : num1 - num2) / (a * a + d * d);
        }
```

Restarting an Event

Sometimes, when you're warping an image, you may decide that you don't like how it looks. You can easily clear the warp effect you've applied to the image with a general button control and some code.

Markup

```
<Button Content="Restart" Click="Restart" Grid.Column="2" Width="140"
HorizontalAlignment="Left" />
```

The code attached to the button Click Event should look like the following:

C#

```
private void Restart(object sender, RoutedEventArgs e)
        {
            bitmap.Copy(originalBitmap, false);
            screen.Copy(originalBitmap, false);
        }
```

Image for UWP

Display animated GIF images in your Windows Store application as you would in a traditional Web application with **Image for UWP**. Animated GIFs are compact and allow you to add attractive visual elements to your applications with minimal effort.

Copyright © 2017 GrapeCity, inc. All rights reserved.

Image for UWP Features

The following are some of the main features of **Image for UWP** that you may find useful:

- **Support for Animated GIF Files**

  Image enables you to add animated GIF files to your Windows Store applications. The C1Image control can be used to add GIF images at design time (the regular image control only supports PNG and JPEG formats).

- **Play, Pause, and Stop Methods**

  The image source used with the **C1Image** control is the C1GifImage class, which provides media player-like commands and allows you to control the GIF animations programmatically. You can use these methods to animate GIFs while performing a task, creating interesting progress indicators, or simply better integrating the animations with the state of the application.

Image for UWP Quick Start

The following quick start guide is intended to get you up and running with **Image for UWP**. In this quick start, you'll create a new project in Visual Studio, add a C1Image control to your application, and then run the application.

Step 1 of 3: Creating a Universal Windows Application

In this step you'll create a Windows Store application in Visual Studio using **Image for UWP**.

To set up your project and add a C1Image control to your application, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. Select Templates | Visual C# | Windows | Universal. From the templates list, select Blank App (Universal Windows).
3. Enter a **Name** and click **OK** to create your project. Open the XAML view of the MainPage.xaml file; in this quick start you'll add controls using XAML markup.
4. Right-click the project name in the Solution Explorer and select **Add Reference**.
5. In the **Reference Manager** dialog box, expand Universal Windows and select Extensions; you should see the UWP assemblies in the center pane. Select C1.UWP.Imaging.
6. Place your cursor between the <Grid> and </Grid> tags.
7. Locate the **C1Image** icon in your Visual Studio Toolbox and double-click the image. This will add the C1Image control to your application.

In the next step, you will add an image to the control.

Step 2 of 3: Adding an Image

Next we are going to add an image to the C1Image control.

1. Select the **C1Image** control and in the Visual Studio Properties window, click the ellipsis button next to the C1Image.Source property. The **Choose Image** dialog box opens.
2. Click the **Add** button.
3. In the **Open** dialog box, browse to find an image. It can be a .gif (animated or still), .jpg, .jpeg, or .png.
4. Select the image and click **Open**.
5. Click **OK**.

In the next step you will run the application.

Copyright © 2017 GrapeCity, inc. All rights reserved.

Step 3of 3: Running the Application

Now that you've created a Universal Windows application with a C1Image control, you're ready to run the application.

From the **Debug** menu, select **Start Debugging** to view your image.

Congratulations!

You have successfully completed the **Image for UWP** quick start.

Image for UWP Task- Based Help

The task-based help assumes that you are familiar with programming in Visual Studio and know how to use the C1Image control in general. If you are unfamiliar with the **Image for UWP** product, please see the Image for UWP XAML Quick Start first.

Each topic in this section provides a solution for specific tasks using the **Image for UWP** product. Each topic also assumes that you have created a new Windows Store application

Playing or Stopping an Animated Image

The image source used with the C1Image control is the C1GifImage class, which provides media player-like commands. You can use the **C1GifImage.Play**, **C1GifImage.Stop**, and **C1GifImage.Pause** methods to control GIF animations programmatically. For an example of how to use the **C1GifImage.Play** and **C1GifImage.Stop** methods, follow these steps:

1. In your Windows Store application, double-click the **C1Image** icon in the Visual Studio Toolbox to add the **C1Image** control to MainPage.xaml. The XAML markup will now look similar to the following:

   Markup

   ```
   <Grid x:Name="LayoutRoot" Background="White">
           <c1imaging:C1Image HorizontalAlignment="Left" Margin="10,10,0,0"
   Name="c1Image1" VerticalAlignment="Top" />
       </Grid>
   ```

2. Select the **C1Image** control and in the Properties window, click the ellipsis button next to the **C1Image.Source** property. The **Choose Image** dialog box opens.
3. Click the **Add** button.
4. In the **Open** dialog box, browse to find an animated .gif.
5. Select the image and click **Open**.
6. Click **OK**. You can adjust the size and alignment of the image as necessary.
7. In the Toolbox, double-click the general **CheckBox** control icon.
8. In the XAML markup, set the **Content** to **Play**, set the **HorizontalAlignment** to **Center**, and set the **VerticalAlignment** to **Bottom** so your XAML looks similar to the following:

   Markup

   ```
   <Grid x:Name="LayoutRoot" Background="White" Height="139" Width="384">
   ```

Copyright © 2017 GrapeCity, inc. All rights reserved.

```
        <c1imaging:C1Image HorizontalAlignment="Center" Margin="10,10,0,252"
Name="c1Image1" Source="Images/Butterfly.gif" Width="44" />
        <CheckBox Content="Play" Height="16" HorizontalAlignment="Center"
Margin="10,10,0,0" Name="checkBox1" VerticalAlignment="Bottom" />
    </Grid>
```

9. Open the MainPage.xaml.cs.
10. Add the following using statements (Imports if using Visual Basic):

C#

```
using C1.Xaml.Imaging;
using C1.Xaml;
```

11. Add code for the **C1GifImage.Play** and **C1GifImage.Stop** methods so it looks similar to the following:

C#

```
public MainPage()
        {
            InitializeComponent();
            var gifImage = new C1GifImage(new Uri("/Images/Butterfly.gif",
UriKind.Relative));
            c1Image1.Source = gifImage;
            checkBox1.IsChecked = true;
            checkBox1.Checked += delegate { gifImage.Play(); };
            checkBox1.Unchecked += delegate { gifImage.Stop(); };
        }
```

12. Click **Debug | Start Debugging** to run the application.
13. Select and clear the **Play** check box to play and stop the animated graphic.

Copyright © 2017 GrapeCity, inc. All rights reserved.