Imaging for UWP

2018.04.10 更新

グレープシティ株式会社

Imaging for UWP

目次

Imaging for UWP

Bitmap for UWP（C1Bitmap）を使用して、画像（PNG および JPG）の読み込み、ピクセル単位の編集、イメージタグでの表示、ストリームへの保存を行うことができます。

Image for UWP（C1Image）を使用して、従来の Web アプリケーションで行う場合と同様に、Windows ストアアプリケーションにアニメーション GIF 画像を表示します。アニメーション GIF はコンパクトな上、アプリケーションに最小限の作業で魅力的なビジュアル要素を追加できます。

Bitmap for UWP

Bitmap for UWP（C1Bitmap）を使用して、画像（PNG および JPG）の読み込み、ピクセル単位の編集、イメージタグでの表示、ストリームへの保存を行うことができます。

主な特長

- **画像の縮小/トリミング**
  ピクセルデータを編集することで、画像をサイズ変更したり、解像度を下げることができます。これにより、ファイルサイズが縮小し、アップロード時間が短縮します。また、Facebook などの Web ユーザーアカウントで画像をアップロードする際に、画像をトリミングして一部のみをアップロードできます。
- **プログラムによる画像編集**
  C1Bitmap クラスを使用すると、個々のピクセルにアクセスして、特殊効果を適用したり、画像のトリミング、サイズ変更、任意の変換を行うことができます。
- **生成された画像を JPG/PNG として保存**
  WritableBitmap を使用してスクリーンショットを取得し、それを C1Bitmap に渡して、その結果を直ちに新しい PNG/JPG ファイルに保存できます。

クイックスタート

このクイックスタートガイドは、**Bitmap for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートでは、デフォルトの画像を読み込んでトリミングできる新しい Windows ストアアプリケーションを作成します。

手順1：Windows ストアアプリケーションの作成

この手順では、Visual Studio で、**Bitmap for UWP** を使用して、Windows ストアアプリケーションを作成します。

プロジェクトを設定するには、次の手順に従います。

1. Visual Studio で、[**ファイル**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。
3. **名前**を入力し、[**OK**]をクリックしてプロジェクトを作成します。MainPage.xaml ファイルの XAML ビューを開きます。このクイックスタートでは、XAML マークアップを使用して、いくつかのコントロールを追加します。
4. ソリューションエクスプローラでプロジェクト名を右クリックし、[**参照の追加**]を選択します。
5. [**参照マネージャ**]ダイアログボックスで[**ComponentOne for UWP**]を選択します。[**OK**]をクリックしてダイアログボックスを閉じ、参照を追加します。

次の手順では、スタイルを設定し、プロジェクトに画像を追加します。

手順2：画像の追加

この手順では、画像のスタイルを設定し、新しい画像を作成するために、次の XAML を追加します。

1. 次の XAML を <UserControl> タグ内に追加し、デフォルトの <Grid> タグを上書きします。

   ▶ XAML でマークアップの書き方

   XAML マークアップ

```
<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>
      </Grid>
```

2. プロジェクトに画像を追加します。
   - [プロジェクト]→[既存の項目の追加]を選択します。
   - 画像を参照します。この例では、**ComponentOne for UWP** に付属する C1.UWP.Imagingサンプルの画像 Lenna.jpg を使用します。
   - この画像を選択し、[**追加**]をクリックします。

次の手順では、画像のトリミングに使用するコードを追加します。

手順3：画像のトリミングに使用するコードの追加

この手順のコードは、デフォルトの画像を読み込み、ユーザーがその画像をトリミングできるようにします。次の手順に従います。

1. MainPage.xaml.cs ファイルを開き、次の **using**（Visual Basic の場合は **Imports**）文を追加します。

   ▶ C# コードの書き方

   C#

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

2. デフォルトの画像を読み込み、トリミングを定義するために、次のコードを追加します。

   ▶ C# コードの書き方

   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))),
                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();
        }
    }
```

次の手順では、このアプリケーションを実行します。

手順4：アプリケーションの実行

アプリケーションを実行します。

1. [**デバッグ**]メニューから[**デバッグ開始**]を選択して、画像を表示します。
2. 画像をクリックし、マウスの左ボタンを押しながらポインタをドラッグします。**ComponentOne for UWP** に付属する ImagingSamples2015 サンプルは、トリミングされた画像をエクスポートし、その画像をファイルに保存する方法を示します。
   次の図では、目の領域がトリミングされています。

おめでとうございます。これで、**クイックスタート** は終了です。

C1Bitmap の使い方

Bitmap for UWP では、さまざまなタスクを実行できます。**Bitmap for UWP** を組み込みの UWP 機能と組み合わせて、画像のトリミング、画像のワープ、簡単な XAML フレームワーク要素のレンダリングを行うことができます。以下のトピックでは、画像のエクスポート、独自の画像のロード、イベントの再開についても説明します。

以下のトピックでは、ImageSamples2012 サンプルの画像とコードを使用します。

トリミングボックスのドラッグによるトリミング

クライアント側で画像を完全にトリミングできれば、とても便利です。Windows ストアアプリケーションでは、**C1Bitmap** または WriteableBitmap クラスを使用してこれを実現できます。C1Bitmap コンポーネントには、あらゆるビットマップ関連操作を実行する際に簡単に使用できる API が用意されています。この API は、単純に色を取得および設定することができ、**GetPixel** および **SetPixel** メソッドを使用してピクセルに直接アクセスできます。

次の XAML は、トリミング領域を作成するために必要な要素を定義します。トリミング領域を作成するには、XAML マークアップに `<Grid>` 要素を追加して XAML にイメージマスクを作成し、コードビハインドでユーザーの選択に基づいて画像にイメージマスクを適用します。

▶ XAML でマークアップの書き方

XAML マークアップ

```
<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="*" />
                        <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>
```

以下のコードは、画像にトリミング効果を実装します。このコードは、ユーザーがトリミングする画像の範囲を選択すると、OnDragStarted イベントと OnDragDelta イベントを使用して水平方向と垂直方向の変更をキャプチャします。さらに、計算によって開始点と終了点の値を求め、それらの点を Rect オブジェクトに変換します。

▶ C# コードの書き方

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();
        }
```

マスクの各部の範囲に多少のロジックを適用します。

▶ C# コードの書き方

C#

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

UpdateMask() メソッドは、Grid Rect の **Left**、**Top**、**Width**、および **Height** プロパティに基づいて、すべての Grid マスク要素の位置を更新します。

▶ C# コードの書き方

C#

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

画像のエクスポート

簡単なコードと汎用のボタンコントロールを使用して、トリミングした画像をエクスポートすることができます。汎用のボタンコントロールの XAML マークアップを以下に示します。

XAML マークアップ

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

エクスポート機能の制御に使用するコードは、エクスポート対象になるトリミングした部分がない場合に、エクスポートオプションをブロックします。エクスポート対象になるトリミングした部分がある場合は、ファイルタイプとエクスポート先を選択できます。

▶ C# コードの書き方

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)
            {
                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));
                    }
                }
```

独自の画像のロード

独自の画像をロードしてトリミングすることもできます。これは、汎用のボタンコントロールとコードビハインドを使用して簡単に実現できます。

XAML マークアップ

```
<Button Content="独自のイメージをロードします" Click="LoadImage" Margin="0 0 10 0"
Width="180" HorizontalAlignment="Left" />
```

クリックイベントに応答するコードでファイルピッカーを開き、表示してトリミングする画像ファイルを選択できるようにします。

▶ C# コードの書き方

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("画像形式がサポートされていませんエラー: \n" + ex.Message, "");
                        md.ShowAsync();
                    }
                }
            }
        }
```

画像をワープ

画像をワープできると面白いです。ワープ効果の作成に必要なコードは多少複雑ですが、C1DragHelpers と多少の計算によって画像にワープ効果を適用できます。FaceWarp サンプルでは、独自の画像をロードし、完成した画像をエクスポートし、ワープ効果の見栄えが気に入らない場合は画像をリセットします。

▶ C# コードの書き方

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);

                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;
        }

        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);
        }
```

イベントの再開

画像をワープしてみたが、その見栄えが気に入らない場合もあります。汎用のボタンコントロールと多少のコードを使用して、画像に適用したワープ効果を簡単に除去することができます。

XAML マークアップ

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

クリックイベントボタンに、次のコードを設定します。

C#

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

Image for UWP

Image for UWPを使用して、従来の Web アプリケーションで行う場合と同様に、Windows ストアアプリケーションにアニメーション GIF 画像を表示します。アニメーション GIF はコンパクトな上、アプリケーションに最小限の作業で魅力的なビジュアル要素を追加できます。

主な特長

以下に、**Image for UWP**の便利な機能をいくつか示します。

- **アニメーション GIF ファイルのサポート**
  Image を使用すると、Silverlight アプリケーションにアニメーション GIF ファイルを追加できます。**C1Image** コントロールを使用して、設計時に GIF 画像を追加できます（標準のイメージコントロールは、PNG 形式と JPEG 形式のみをサポート）。
- **再生、一時停止、停止のメソッド**
  C1Image コントロールで使用されるイメージソースは **C1GifImage**クラスです。このクラスは、メディアプレイヤーと同様のコマンドを提供して、GIF アニメーションをプログラムで制御できるようにします。これらのメソッドを使用して、タスクを実行しながら GIFをアニメーション表示することで、面白い進捗状況インジケータを作成したり、アプリケーションの状態に合わせた アニメーションを作成することができます。

クイックスタート

このクイックスタートガイドは、**Image for UWP**を初めて使用するユーザーのために用意されています。このクイックスタートでは、Visual Studio で新しいプロジェクトを作成し、**C1Image** コントロールをアプリケーションに追加してから、アプリケーションを実行します。

手順1：ユニバーサルWindowsアプリケーションの作成

この手順では、Visual Studio で、**Image for UWP**を使用して、Windows ストアアプリケーションを作成します。

プロジェクトを設定し、C1Image コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio で、[**ファイル**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。
3. **名前**を入力し、[**OK**]をクリックしてプロジェクトを作成します。MainPage.xaml ファイルの XAML ビューを開きます。このクイックスタートでは、XAML マークアップを使用して、いくつかのコントロールを追加します。
4. ソリューションエクスプローラでプロジェクト名を右クリックし、[**参照の追加**]を選択します。
5. [**参照マネージャ**]ダイアログボックスで[**ユニバーサルWindows**]を展開し、[**拡張機能**]を選択します。中央ペインで UWPアセンブリが表示されます。C1.UWP.Imagingを選択します。
6. <Grid> タグと </Grid> タグの間にカーソルを置きます。
7. Visual Studio ツールボックスで、C1Image アイコンを探してダブルクリックします。これで、アプリケーションに **C1Image** コントロールが追加されます。

次の手順では、コントロールに画像を追加します。

手順2：画像の追加

次に、**C1Image** コントロールに画像を追加します。

1. C1Image コントロールを選択し、Visual Studio の[プロパティ]ウィンドウで、**Source** プロパティの横にある**省略符**ボタンをクリックします。[**イメージの選択**]ダイアログボックスが開きます。
2. [**追加**]ボタンをクリックします。
3. [**開く**]ダイアログボックスで、画像を参照して見つけます。.gif（アニメーションまたは静止画像）、.jpg、.jpeg、または .png を選択できます。

4. 画像を選択し、[**開く**]をクリックします。
5. [**OK**]をクリックします。

次の手順では、このアプリケーションを実行します。

手順3：アプリケーションの実行

C1Image コントロールを含む ユニバーサルWindowsアプリケーションを作成したので、このアプリケーションを実行します。

[**デバッグ**]メニューから[**デバッグ開始**]を選択して、画像を表示します。

おめでとうございます。これで、**Image for UWP**クイックスタートは終了です。

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio でのプログラミングに精通しており、C1Image コントロールの一般的な使用方法を理解していることを前提としています。**Image for UWP**製品を初めて使用される場合は、まず「**クイックスタート**」を参照してください。

このセクションの各トピックは、**Image for UWP**製品を使用して特定のタスクを実行するための方法を提供します。また、各トピックは、新しい Windows ストアアプリケーションが既に作成されていることを前提としています。

アニメーション画像の再生または停止

C1Image コントロールで使用されるイメージソースは **C1GifImage** クラスです。このクラスは、メディアプレイヤーと同様のコマンドを提供します。Play、Stop、Pause の各メソッドを使用して、GIF アニメーションをプログラムで制御できます。Play メソッドと Stop メソッドの使用例については、次の手順に従います。

1. Windows ストアアプリケーションで、Visual Studio ツールボックスの[C1Image]アイコンをダブルクリックして、C1Image コントロールを **MainPage.xaml** に追加します。XAML マークアップは次のようになります。

XAML マークアップ

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

2. C1Image コントロールを選択し、[プロパティ]ウィンドウで、Source プロパティの横にある**省略符**ボタンをクリックします。[**イメージの選択**]ダイアログボックスが開きます。
3. [**追加**]ボタンをクリックします。
4. [**開く**]ダイアログボックスで、アニメーション .gif を参照して見つけます。
5. 画像を選択し、[**開く**]をクリックします。
6. [**OK**]をクリックします。必要に応じて、画像のサイズと配置を調整できます。
7. ツールボックスで、汎用の **CheckBox** コントロールアイコンをダブルクリックします。
8. XAML マークアップで、**Content** を **Play**、**HorizontalAlignment** を **Center**、および **VerticalAlignment** を **Bottom** に設定します。XAML は、次のようになります。

XAML マークアップ

```
<Grid x:Name="LayoutRoot" Background="White" Height="139" Width="384">
        <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. MainPage.xaml.cs を開きます。
10. 次の **using** ステートメントを追加します（Visual Basic を使用する場合は **Imports**）。

C#

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

11. Play メソッドと Stop メソッドのコードを追加します。次のようになります。

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. [**デバッグ**]→[**デバッグ開始**]をクリックして、アプリケーションを実行します。
13. アニメーショングラフィックを再生または停止するには、[**再生**]チェックボックスをオンまたはオフにします。