Using Nokia Imaging SDK in a Windows Phone game

This article explains how you can use the Nokia Imaging SDK to create a Windows Phone 8 game.

Winner: This article was a winner in the Nokia Imaging Wiki Competition 2013Q3.

11 Aug 2013

Introduction

The new Nokia Imaging SDK gives developers the ability to add filters and effects to images, making it easy to create amazing imaging applications. This article will show you how this SDK can be used in a different scenario such as a game.

The code example demonstrates an image-based quiz game. The idea is that we show the user a picture of a city and present 4 choices. For each correct answer, points are added to the score. To add some difficulty to the game we will use the Imaging SDK so we can add some filters and manipulate the image, making it harder to recognize.

Even though most of the details are covered in this article, be sure to download the source so you can play with it.

Prerequisites

We'll be using the Nokia Imaging SDK, which is available as a NuGet package. Install the package and make sure you remove the *Any CPU* target platform from your solution configuration. If you're having trouble referencing this library, follow this guide: Installing the SDK and including the libraries to a project using NuGet.

We'll also be using the Windows Phone Toolkit in order to add some animations and extra controls to our game.

Adding these NuGet packages is really simple. All you need to do is, in the Solution Explorer, right click the project file and select **Manage NuGet Packages**. Search for the package you want to install, select it and click Install. You can also use the Package Manager Console to install packages.

Creating Picture Quiz

Picture Quiz is a Windows Phone 8 XAML/C# game, based on the Windows Phone App template.

Sample Pictures

In order to have some content in the game, I browsed the Flickr website. I came up with a list of 10 cities I would like to include in the game and searched for great pictures taken in those cities.

I also had to make sure that the authors allowed their pictures to be used in this game. Flickr allows uploaders to add license information to their pictures. So, when choosing these pictures, I had to make sure they were licensed under the Creative Commons Attribution license, meaning that the users allows the use of their work, as well as sharing and modification, as long as the work is attributed to them.

Quiz UI

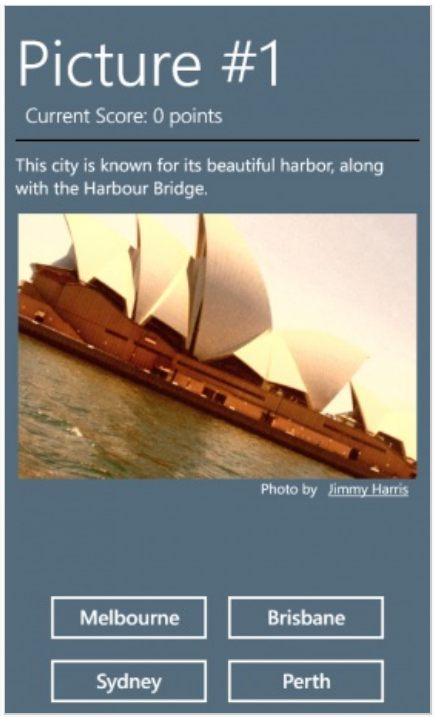
When designing the UI, I wanted it to be simple and informative. When we think about what information needs to be displayed in a quiz game, we can group it in two types: Information about the current game and information about the current round/question.

Information about the game is simply the current round and the current score. Then we have the information related to current question. In this group we can categorize a few more items:

- The current picture to be guessed
- Information about the picture author
- A hint text to help the user
- Four possible answers

After a few experiments, this is the final layout for the Quiz page:

(C) Copyright Nokia 2014. All rights reserved.



Screenshot of the Quiz page

And this is the XAML behind it:

```
<Grid x:Name="LayoutRoot" Background="#FF556C7D">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>

    <!--TitlePanel contains the name of the application and page title-->
    <StackPanel x:Name="ScorePanel" Grid.Row="0" Margin="0,12,0,0">
        <TextBlock x:Name="tbRoundNo" Text="{Binding Round, StringFormat=Picture
#\{0\}}" Style="{StaticResource PhoneTextTitle1Style}" Margin="12,0"/>
        <TextBlock x:Name="tbScore" Text="{Binding Score, StringFormat=Current Score:
\{0\} points}" Margin="23,0,0,0" Style="{StaticResource PhoneTextTitle3Style}" />
        <Border BorderThickness="1" BorderBrush="Black" Margin="12,12" />
    </StackPanel>

    <!--ContentPanel - place additional content here-->
    <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <TextBlock Grid.Row="0" x:Name="tbQuestion" Text="{Binding
CurrentQuestion.QuestionText}" TextWrapping="Wrap" MinHeight="70"/>
        <StackPanel Grid.Row="1">
            <Image x:Name="questionImage" Height="300" Width="450"/>
            <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
                <TextBlock x:Name="Attribuition" Text="Photo by" FontSize="16"
HorizontalAlignment="Right" />
                <HyperlinkButton Content="{Binding CurrentQuestion.Attribution}"
NavigateUri="{Binding CurrentQuestion.AttributionUrl}" FontSize="16" TargetName="_blank"
HorizontalAlignment="Left"/>
            </StackPanel>
```


(C) Copyright Nokia 2014. All rights reserved.

```
        </StackPanel>
    </Grid>

    <StackPanel x:Name="AnswersPanel" Grid.Row="2" HorizontalAlignment="Center">
        <StackPanel Orientation="Horizontal">
            <Button x:Name="btnAnswer1" Width="200" Click="AnswerButton"/>
            <Button x:Name="btnAnswer2" Width="200" Click="AnswerButton"/>
        </StackPanel>
        <StackPanel Orientation="Horizontal">
            <Button x:Name="btnAnswer3" Width="200" Click="AnswerButton"/>
            <Button x:Name="btnAnswer4" Width="200" Click="AnswerButton"/>
        </StackPanel>
    </StackPanel>

</Grid>
```

Models

In order to be able to store the questions, the `Question` class, under the *Models* folder, is used:

```
public class Question
{
    private const string PICTURES_PATH = "Assets/Questions/";

    public string QuestionText { get; set; }
    public string Picture { get; set; }
    public string CorrectAnswer { get; set; }
    public string[] Answers { get; set; }
    public string Attribution { get; set; }
    public string AttributionUrl { get; set; }

    public Question(string questionText, string pictureLocation,
        string correctAnswer,
        string wrongAnswer1,
        string wrongAnswer2,
        string wrongAnswer3,
        string author, string flickrPage)
    {
        QuestionText = questionText;
        Picture = PICTURES_PATH + pictureLocation;
        CorrectAnswer = correctAnswer;
        Answers = new string[3];
        Answers[0] = wrongAnswer1;
        Answers[1] = wrongAnswer2;
        Answers[2] = wrongAnswer3;
        Attribution = author;
        AttributionUrl = flickrPage;
    }
}
```

This class contains all the information we need for a question. The text hint, the path of the picture, all the four possible answers, the author's name and a link to the page where it was obtained.

Information related to the current game also needs to be stored. For that we'll use the `QuizViewModel` class, located in the *ViewModels* folder.

(C) Copyright Nokia 2014. All rights reserved.

```
public class QuizViewModel : INotifyPropertyChanged
{
    private Random _rnd;

    public List<Question> Questions { get; set; }

    private Question _currentQuestion;
    public Question CurrentQuestion
    {
        get { return _currentQuestion; }
        set
        {
            _currentQuestion = value;
            NotifyPropertyChanged("CurrentQuestion");
        }
    }

    private int _score;
    public int Score
    {
        get { return _score; }
        set
        {
            _score = value;
            NotifyPropertyChanged("Score");
        }
    }

    private int _round;
    public int Round
    {
        get { return _round; }
        set
        {
            _round = value;
            NotifyPropertyChanged("Round");
        }
    }



    public QuizViewModel()
    {
        _rnd = new Random();
    }

    public void LoadQuestions()
    {
        Questions = new List<Question>();

        Questions.Add(new Question("This city is known as \"the city that never
sleeps\".", "newyork.jpg",
            "New York", "Las Vegas", "Los Angeles", "Atlanta",
            "Patrick Briggs", "http://www.flickr.com/photos/87241965@N00/4379712440/"));

        Questions.Add(new Question("This city is known for its traditional architecture,
canals, shopping, and many coffeeshops.", "amsterdam.jpg",
```


(C) Copyright Nokia 2014. All rights reserved.

```
            "Amsterdam", "Venice", "Leiden", "Rotterdam",
            "Christian Lendl", "http://www.flickr.com/photos/_dchris/7329055596/"));

        Questions.Add(new Question("This city is known for its beautiful harbor, along
with the Harbour Bridge.", "sydney.jpg",
            "Sydney", "Melbourne", "Brisbane", "Perth",
            "Jimmy Harris", "http://www.flickr.com/photos/jimmyharris/114538159/"));

        Questions.Add(new Question("This is one of the oldest cities in Europe.",
"lisbon.jpg",
            "Lisbon", "London", "Paris", "Madrid",
            "Terence S. Jones ",
"http://www.flickr.com/photos/terence_s_jones/6684473529/"));

        Questions.Add(new Question("This is the capital and largest urban area of both
England and the United Kingdom.", "london.jpg",
            "London", "Manchester", "York", "Leeds",
            "Tim Morris", "http://www.flickr.com/photos/timmorris/3103896345/"));

        Questions.Add(new Question("This city is known for being the fashion capital of
the world.", "paris.jpg",
            "Paris", "Milan", "New York", "London",
            "Terrazzo", "http://www.flickr.com/photos/terrazzo/3958413757/"));

        Questions.Add(new Question("This is the only city in the world to contain in its
interior a whole state.", "rome.jpg",
            "Rome", "Naples", "Milan", "Florence",
            "roamancing", "http://www.flickr.com/photos/roamancing/5795269117/"));

        Questions.Add(new Question("This city is the business and trade hub of United
Arab Emirates.", "dubai.jpg",
            "Dubai", "Abu Dhabi", "Sharjah", "Al Ain",
            "Richard Schneider",
"http://www.flickr.com/photos/picturecorrect/7623566780/"));

        Questions.Add(new Question("This city is referred to as the birthplace of
democracy.", "athens.jpg",
            "Athens", "Rome", "Thessaloniki", "Naples",
            "Ronny Siegel", "http://www.flickr.com/photos/47309201@N02/9259286435/"));

        Questions.Add(new Question("This city has the second largest community of
billionaires in the world.", "moscow.jpg",
            "Moscow", "New York", "London", "Hong Kong",
            "Lori Branham", "http://www.flickr.com/photos/kjunstorm/8337228066/"));

        NextQuestion();
    }

    public void NextQuestion()
    {
        if (Questions.Any())
        {
            CurrentQuestion = Questions[_rnd.Next(Questions.Count)];
            Questions.Remove(_currentQuestion);
        }
    }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    public event PropertyChangedEventHandler PropertyChanged;

    public void NotifyPropertyChanged(string propertyname)
    {
        if (PropertyChanged != null)
        {
            PropertyChanged(this, new PropertyChangedEventArgs(propertyname));
        }
    }
}
```

As you can see, this class is where we store the current Round number and the current Score, as well as the list of Questions in the game. We have some methods to populate the questions list and to select a random question from the list. This class is used as a `DataContext` in the Quiz page so we can bind the interface elements to these properties.

Image Processor

A class called `ImageProcessor` is used to handle all the image processing tasks. Before displaying an image, this class applies some filters and effects to it, using all the magic provided by the Nokia Imaging SDK.

```
public class ImageProcessor
{
    public enum Difficulty
    {
        Easy = 1,
        Moderate,
        Hard,
        VeryHard,
        ReallyHard
    }

    // Filter groups for each difficulty
    private FilterGroup _easyFilterGroup;
    private FilterGroup _moderateFilterGroup;
    private FilterGroup _hardFilterGroup;
    private FilterGroup _veryHardFilterGroup;
    private FilterGroup _impossibleFilterGroup;

    // Masks to be used with the ImageFusion Filter
    private Bitmap _mask1;
    private Bitmap _mask2;
    private Bitmap _mask3;
    private Bitmap _maskBackground;

    public ImageProcessor()
    {
        //Initialize the filter groups for each difficulty level
        SetupDifficultyFilterGroups();
    }

    private void SetupDifficultyFilterGroups()
    {
        // Alpha Masks
        _mask1 = LoadBitmap("mask1.jpg");
        _mask2 = LoadBitmap("mask2.jpg");
        _mask3 = LoadBitmap("mask3.jpg");
```


(C) Copyright Nokia 2014. All rights reserved.

```
        // Alpha mask background
        _maskBackground = LoadBitmap("maskBg.jpg");


        // Easy Difficulty: Antique filter and 23 degree rotation
        _easyFilterGroup = new FilterGroup(new IFilter[]
        {
            FilterFactory.CreateAntiqueFilter(),
            FilterFactory.CreateFreeRotationFilter(23f, RotationResizeMode.FitInside)
        });


        // Moderate Difficulty: Milky and Blur filters and alpha mask 1
        _moderateFilterGroup = new FilterGroup(new IFilter[]
        {
            FilterFactory.CreateMilkyFilter(),
            FilterFactory.CreateBlurFilter(BlurLevel.Blur3),
            FilterFactory.CreateImageFusionFilter(_maskBackground, _mask1, false)
        });

        // Hard Difficulty: Warp, Magic Pen and Gray Scale Negative filters
        _hardFilterGroup = new FilterGroup(new IFilter[]
        {
            FilterFactory.CreateWarpFilter(WarpEffect.SmallNose, 0.8f),
            FilterFactory.CreateMagicPenFilter(),
            FilterFactory.CreateGrayscaleNegativeFilter()
        });

        // Very Hard Difficulty: Cartoon filter and apply alpha mask 2
        _veryHardFilterGroup = new FilterGroup(new IFilter[]
        {
            FilterFactory.CreateCartoonFilter(true),
            FilterFactory.CreateImageFusionFilter(_maskBackground, _mask2, false)
        });

        // Impossible Difficulty: 45 degree rotation, watercolor filer and alpha mask 3
        _impossibleFilterGroup = new FilterGroup(new IFilter[]
        {
            FilterFactory.CreateFreeRotationFilter(45f, RotationResizeMode.FitInside),
            FilterFactory.CreateWatercolorFilter(0.8f, 1),
            FilterFactory.CreateImageFusionFilter(_maskBackground, _mask3, false)
        });

    }

    private static Bitmap LoadBitmap(string fileName)
    {
        Stream maskStream = Application.GetResourceStream(new Uri("Assets/Masks/" +
fileName, UriKind.Relative)).Stream;
        BitmapImage maskImage = new BitmapImage();
        maskImage.SetSource(maskStream);
        return new WriteableBitmap(maskImage).AsBitmap();
    }

    public async Task ApplyDifficultyToImage(Stream image, Difficulty level, Image
```


(C) Copyright Nokia 2014. All rights reserved.

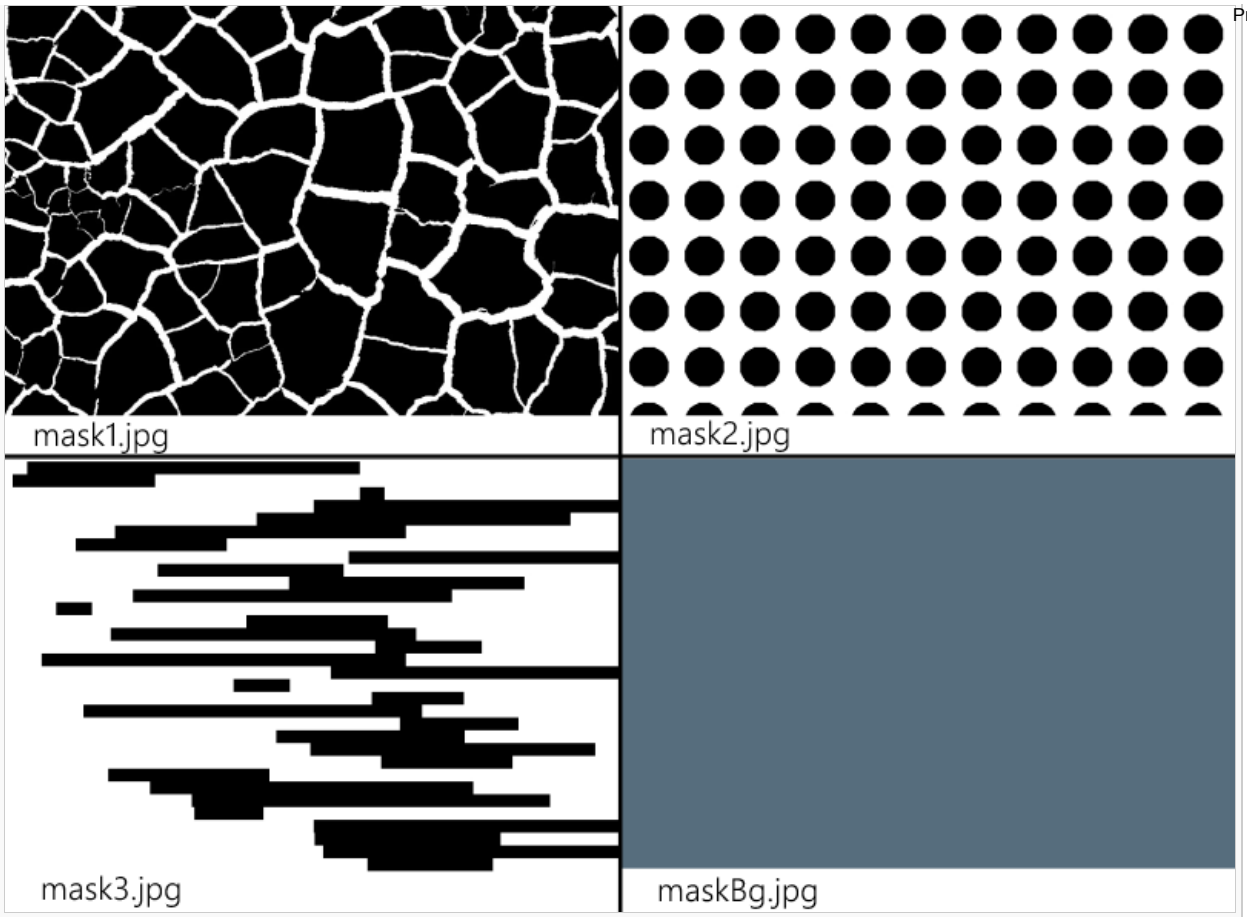
```
resultImage)
    {
        using (EditingSession _session = await
EditingSessionFactory.CreateEditingSessionAsync(image))
 {
         switch (level)
         {
             case Difficulty.Easy:
                 _session.AddFilter(_easyFilterGroup);
                 break;
             case Difficulty.Moderate:
                 _session.AddFilter(_moderateFilterGroup);
                 break;
             case Difficulty.Hard:
                 _session.AddFilter(_hardFilterGroup);
                 break;
             case Difficulty.VeryHard:
                 _session.AddFilter(_veryHardFilterGroup);
                 break;
             case Difficulty.ReallyHard:
                 _session.AddFilter(_impossibleFilterGroup);
                 break;
             default:
                 break;
         }


         //Render the image to the Image control in the Quiz page
         await _session.RenderToImageAsync(resultImage);
        }
    }
}
```

As we can see, this class has five difficulties. For each difficulty settings, a `FilterGroup` is defined. `FilterGroup`s are a really nice way to define set of filters once and apply them as a whole later. Think of this as a way to create your own filter using existing filters. Later, if you want to change the effects and filters for a difficulty setting, all you need to do is redefine the corresponding filter group. Most of the applied filters are simple and you can explore them using the Nokia Imaging SDK Sample Projects.

A filter that is worth mentioning is the Image Fusion filter. This filter allows to combine two different images based on an alpha mask. Using this filter we can create hide some areas of the picture to make it harder for the user to guess.

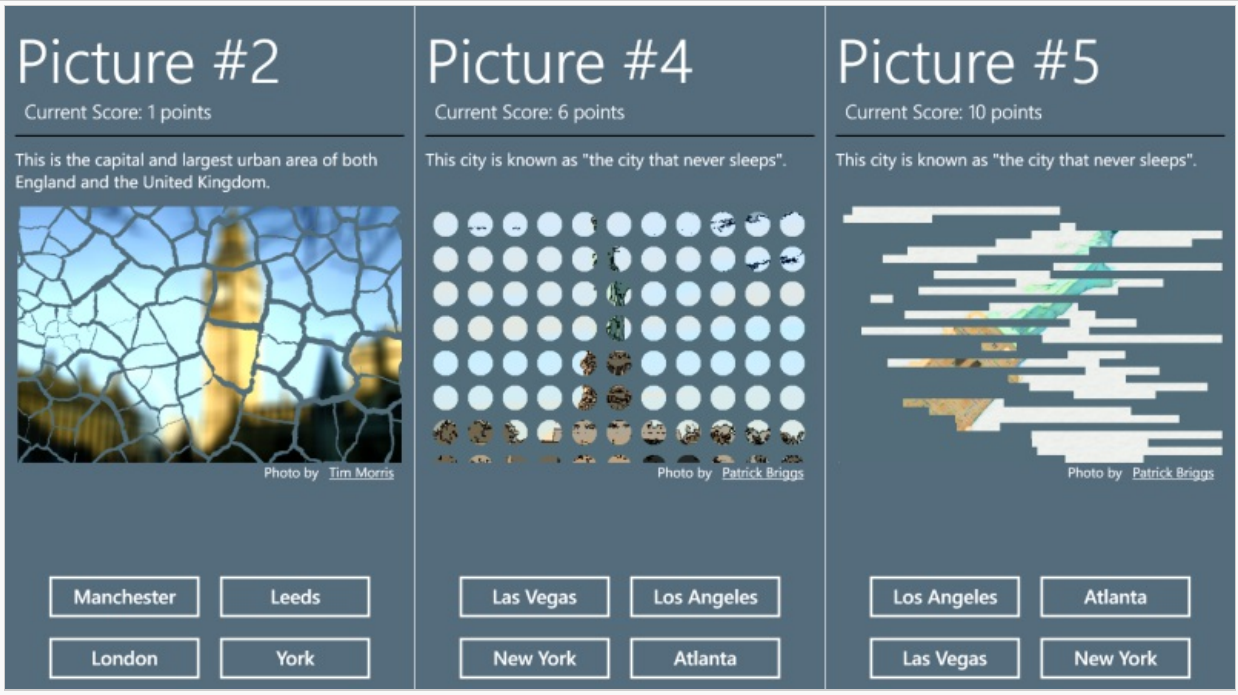
Picture Quiz has 3 alpha masks:

(C) Copyright Nokia 2014. All rights reserved.



Alpha masks used in the game

An alpha mask defines how the images will be combined. In this case, we will be combining our pictures with **maskBg.jpg**, which is just an image with the same color of the background of the game. The alpha mask sets the pattern. All the black areas in the alpha mask are replaced the the original picture, all the white areas are replaced with the **maskBg.jpg**. This way we will be removing bits from the original image. The final result looks like this:



Example of the Image Fusion filter

The `ApplyDifficultyToImage` method receives a `Stream` of the image to modify, the difficulty setting to use and the Image control to render the image. For every Question, a new `EditingSession` is created and disposed, after the picture is processed.

Wiring up the Quiz page

Now that we are able to process our pictures, all we need to do is add the game logic to the **Quiz.xaml** page. Here's the code:

(C) Copyright Nokia 2014. All rights reserved.

```
public partial class MainPage : PhoneApplicationPage
{
    private QuizViewModel _viewModel;
    private Random _rnd;
    private bool _hideHint = false;

    private ImageProcessor _imgProcessor;
    private ImageProcessor.Difficulty _difficulty = ImageProcessor.Difficulty.Easy;

    // Constructor
    public MainPage()
    {
        InitializeComponent();
        _imgProcessor = new ImageProcessor();
        _rnd = new Random();
        _viewModel = new QuizViewModel();
        _viewModel.PropertyChanged += _viewModel_PropertyChanged;
        this.DataContext = _viewModel;

        LoadQuestionsAndReset();
    }

    protected override void OnNavigatedTo(NavigationEventArgs e)
    {

        //Apply game settings
        if
(IsolatedStorageSettings.ApplicationSettings.Contains("HidePictureDescription"))
        {
            _hideHint =
(bool)IsolatedStorageSettings.ApplicationSettings["HidePictureDescription"];
            if (_hideHint)
                tbQuestion.Visibility = System.Windows.Visibility.Collapsed;
        }

        base.OnNavigatedTo(e);
    }

    private void _viewModel_PropertyChanged(object sender,
System.ComponentModel.PropertyChangedEventArgs e)
    {
        if (e.PropertyName == "CurrentQuestion")
        {
            LoadQuestion();
        }
    }

    private async void LoadQuestion()
    {
        //Randomize answers and populate the buttons
        string[] answers = new string[4];
        _viewModel.CurrentQuestion.Answers.CopyTo(answers, 0);
        answers[3] = _viewModel.CurrentQuestion.CorrectAnswer;

        answers = answers.OrderBy(x => _rnd.Next()).ToArray();
```


(C) Copyright Nokia 2014. All rights reserved.

```
        btnAnswer1.Content = answers[0];
        btnAnswer2.Content = answers[1];
        btnAnswer3.Content = answers[2];
        btnAnswer4.Content = answers[3];


        //Load the picture and apply some filters to obfuscate it
        Uri questionUri = new Uri(_viewModel.CurrentQuestion.Picture, UriKind.Relative);
        Stream questionImageStream = Application.GetResourceStream(questionUri).Stream;
        await _imgProcessor.ApplyDifficultyToImage(questionImageStream, _difficulty,
questionImage);
    }

    private void AnswerButton(object sender, RoutedEventArgs e)
    {
        var btn = sender as Button;
        if ((string)btn.Content == _viewModel.CurrentQuestion.CorrectAnswer)
        {
            // Points are incremented according to the current difficulty
            // Easy = 1, Moderate = 2, Hard = 3, and so on
            // Double points if playing without hints

            if (_hideHint)
                _viewModel.Score += (int)_difficulty * 2;
            else
                _viewModel.Score += (int)_difficulty;

            IncreaseDifficulty();
        }

        if (_viewModel.Round < 5) //five rounds per game
        {
            _viewModel.Round++;
            _viewModel.NextQuestion();
        }
        else
        {
            // Display message asking the user if he/she wants to repeat. If not, go
back to the menu
            CustomMessageBox messageBox = new CustomMessageBox()
            {
                Caption = "Game over!",
                Message = "Your score: " + _viewModel.Score + Environment.NewLine +
"Play again?",
                LeftButtonContent = "yes",
                RightButtonContent = "no"
            };

            messageBox.Dismissed += (s1, e1) =>
            {
                if (e1.Result == CustomMessageBoxResult.LeftButton)
                    LoadQuestionsAndReset();
                else
                    NavigationService.GoBack();
            };

            messageBox.Show();
```


(C) Copyright Nokia 2014. All rights reserved.

```
        }
    }

    private void LoadQuestionsAndReset()
    {
        _viewModel.Score = 0;
        _viewModel.Round = 1;
        _difficulty = ImageProcessor.Difficulty.Easy;
        _viewModel.LoadQuestions();
    }

    private void IncreaseDifficulty()
    {
        if (_difficulty < ImageProcessor.Difficulty.ReallyHard)
            _difficulty++;
    }
}
```

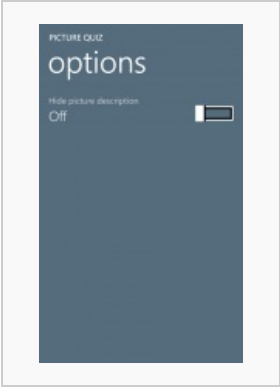
When a new question is loaded, the possible answers are randomized and the `ImageProcessor` class is called so the image can be obfuscated. If the user chooses the right answer, the difficulty will increase. Every game has 5 rounds and once every game is finished the user is asked if he/she wants to repeat. If so, the game state is reset and the game starts again.

Additional features

A few additional things were added to enhance the user experience. Some pages were added: a Menu page, an About page and an Options page, where you can disable the in-game text hint. When playing without text hints, points are doubled. Page transitions were also implemented using the Windows Phone Toolkit library.

Here are a few screen captures of the final application:

Menu Page



Options Page

Sample Question (Easy)

Sample Question (Hard)

Future work

The code is currently available at GitHub. In the future I may add some other features. This is a list of what I have in mind:

- Retrieving pictures using the Flickr API;
- Add multiple FilterGroups per difficulty tier, selected randomly.

Final Notes

In this article I tried to demonstrate that the Nokia Imaging SDK can not only be used to create photo editing applications but also any type of app that you can think of that involves some sort of image editing. This SDK is really powerful and easy to use. As you saw, you can completely transform an image with just a few lines of code. Thanks to it wonderful performance and really low memory consumption, you can target every phone range without performance issues.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.