Name of Experiment: Grayscale Preview	**Exp No:** WP11

Background: Student should have basic knowledge of C#.

Summary: This experiment is related to camera operation in advance manner. Here, user can saw the Grayscale Preview of image before capturing the image. This kind of feature is useful in learning the DIP subject.

Learning Objective: Student can understand the difference between GrayScale and ARGB images.

Target Platforms: This experiment is tested on Windows Phone Emulator and Windows Phone (Lumia 800).

Procedure:

Step1. Repeat steps [1-6] as in experiment no WP6.

Step2. Add three buttons on the UI screen and named them as Gray On and Gray Off and one as Capture.

Step3. Define the method PumpARGBframe in order to convert the ARGB images into GrayScale by simple mathematical conversion.[Refer the Source code]

Step4. Write down the event handler for the button GrayOn, call the new thread for getting the preview window for GrayScale image, create the writable bitmap image and display it on the screen.[Refer fig. no 3]

Step5.Write down the event handler for the button GrayOff, set the MainImage.Visibility=Visibility.Collapsed.

Setp6. Implement the camera capture image event for button Capture.[Refer fig no 4]

Step7. Also, implement the CaptureImageAvailable event in order to save the captured image into camera roll.[Refer Source code]

Step8. Save all the changes made to the experiment.

Step9. Press F5, to deploy the experiment on the windows Phone Emulator for debugging purpose.[Refer fig. no 2]

Source Code	Comments

MainPage.xaml
`<phone:PhoneApplicationPage`
`x:Class="WP8.MainPage"`
 	
`xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"`
`xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"`
`xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"`
`xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"`
`xmlns:d="http://schemas.microsoft.com/expression/blend/2008"`
`xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"`
`mc:Ignorable="d" d:DesignWidth="800" d:DesignHeight="480"`
`FontFamily="{StaticResource PhoneFontFamilyNormal}"`
`FontSize="{StaticResource PhoneFontSizeNormal}"`
`Foreground="{StaticResource PhoneForegroundBrush}"`
`SupportedOrientations="Landscape" Orientation="LandscapeLeft"`
`shell:SystemTray.IsVisible="False">`
 	
`<!--LayoutRoot is the root grid where all page content is placed-->`
`<Grid x:Name="LayoutRoot" Background="Transparent">`
`<Grid.ColumnDefinitions>`
`<ColumnDefinition Width="640" />`
`<ColumnDefinition Width="160"/>`
`</Grid.ColumnDefinitions>`
`<Rectangle Width="640" Height="480" HorizontalAlignment="Left" >`
`<Rectangle.Fill>`
`<VideoBrush x:Name="viewfinderBrush" />`
`</Rectangle.Fill>`
 	
`</Rectangle>`
`<Image x:Name="MainImage"`
`Width="320" Height="240"`
`HorizontalAlignment="Left" VerticalAlignment="Bottom"`
`Margin="16,0,0,16"`
`Stretch="Uniform"/>`
`<StackPanel Grid.Column="1" >`
`<Button`
`Content="Gray: ON"`
`Name="GrayscaleOnButton"`
`Click="GrayOn_Clicked" />`
`<Button`
`Content="Gray: OFF"`
`Name="GrayscaleOffButton"`
`Click="GrayOff_Clicked" />`
`<Button`
`Content="Capture"`
`Name="shutterButton"`
`Click="shutterButton_Click"/>`
`</StackPanel>`
 	
`<TextBlock Height="40" HorizontalAlignment="Left" Margin="8,428,0,0" Name="txtDebug" VerticalAlignment="Top" Width="626" FontSize="24" FontWeight="ExtraBold" />`
 	
`</Grid>`
	← Gray:On
	
	← Gray:OFF
	
	←Capture
	
	← TextBlock for Message Display.

```
</phone:PhoneApplicationPage>
```

MainPage.xaml.cs

```
using System.Windows;
using Microsoft.Phone.Controls;
using Microsoft.Devices;
using Microsoft.Xna.Framework.Media;
using System.Threading;
using System.Windows.Media.Imaging;
using System;
namespace WP8
{
    public partial class MainPage : PhoneApplicationPage
    {
        PhotoCamera mycam;
        public static ManualResetEvent pauseEvent = new
ManualResetEvent(true);
        private WriteableBitmap wb;
        private Thread ARGframes;
        private bool pumpArgframe;
        int photoCounter = 0;
        MediaLibrary libraray = new MediaLibrary();
        // Constructor
        public MainPage()
        {
            InitializeComponent();
        }

        protected override void
OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            //base.OnNavigatedTo(e);
            if
(PhotoCamera.IsCameraTypeSupported(CameraType.Primary) == true)
            {
                mycam = new
Microsoft.Devices.PhotoCamera(CameraType.Primary);
                mycam.Initialized += new
System.EventHandler<CameraOperationCompletedEventArgs>(mycam_Initial
ized);
                mycam.CaptureImageAvailable += new
EventHandler<ContentReadyEventArgs>(mycam_CaptureImageAvailable);

                viewfinderBrush.SetSource(mycam);
            }

            else {
                Deployment.Current.Dispatcher.BeginInvoke(delegate()
                {
                    txtDebug.Text = "Camera device is not
available";
                });
                GrayscaleOffButton.IsEnabled = false;
                GrayscaleOnButton.IsEnabled = false;
            }
        }

        public void mycam_Initialized(object sender,
CameraOperationCompletedEventArgs e) {
            if (e.Succeeded)
            {
```

←Thread ARGframes
←using System.Threading
←Microsoft.Xna.Framework.Media.

```
                Deployment.Current.Dispatcher.BeginInvoke(delegate()
                {
                    txtDebug.Text = "camera is initialised";
                });
            }
        }

        protected override void
OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e)
        {
            //base.OnNavigatedFrom(e);
            if (mycam != null) {
                mycam.Dispose();
                mycam.Initialized -= mycam_Initialized;
                mycam.CaptureImageAvailable -=
mycam_CaptureImageAvailable;
            }
        }

        void PumpARGBFrames()
        {
            // Create capture buffer.
            int[] ARGBPx = new
int[(int)mycam.PreviewResolution.Width *
(int)mycam.PreviewResolution.Height];

            try
            {
                PhotoCamera phCam = (PhotoCamera)mycam;

                while (pumpArgframe)
                {
                    pauseEvent.WaitOne();

                    // Copies the current viewfinder frame into a
buffer for further manipulation.
                    phCam.GetPreviewBufferArgb32(ARGBPx);

                    // Conversion to grayscale.
                    for (int i = 0; i < ARGBPx.Length; i++)
                    {
                        ARGBPx[i] = ColorToGray(ARGBPx[i]);
                    }

                    pauseEvent.Reset();

Deployment.Current.Dispatcher.BeginInvoke(delegate()
                    {
                        // Copy to WriteableBitmap.
                        ARGBPx.CopyTo(wb.Pixels, 0);
                        wb.Invalidate();

                        pauseEvent.Set();
                    });
                }

            }
            catch (Exception e)
            {
                this.Dispatcher.BeginInvoke(delegate()
                {
                    // Display error message.
```

← method for convering ARGB to GrayScale image.

```
                    txtDebug.Text = e.Message;
                });
            }
        }

        internal int ColorToGray(int color)
        {
            int gray = 0;

            int a = color >> 24;
            int r = (color & 0x00ff0000) >> 16;
            int g = (color & 0x0000ff00) >> 8;
            int b = (color & 0x000000ff);

            if ((r == g) && (g == b))
            {
                gray = color;
            }
            else
            {
                // Calculate for the illumination.
                // I =(int)(0.109375*R + 0.59375*G + 0.296875*B +
0.5)
                int i = (7 * r + 38 * g + 19 * b + 32) >> 6;

                gray = ((a & 0xFF) << 24) | ((i & 0xFF) << 16) | ((i
& 0xFF) << 8) | (i & 0xFF);
            }
            return gray;
        }
 private void GrayOn_Clicked(object sender, RoutedEventArgs e)
        {   MainImage.Visibility = Visibility.Visible;
            pumpArgframe = true;
            ARGframes = new System.Threading.Thread(PumpARGBFrames);

            wb = new
WriteableBitmap((int)mycam.PreviewResolution.Width,
(int)mycam.PreviewResolution.Height);
            this.MainImage.Source=wb;
            ARGframes.Start();
            this.Dispatcher.BeginInvoke(delegate()
            {
                txtDebug.Text = "ARGB to GrayScale ";
            });
}
    private void GrayOff_Clicked(object sender, RoutedEventArgs e)
        {
            MainImage.Visibility = Visibility.Collapsed;
            pumpArgframe = false;
            this.Dispatcher.BeginInvoke(delegate()
            {
                txtDebug.Text = "";
            });
        }

        private void shutterButton_Click(object sender,
RoutedEventArgs e)
        {
            if (mycam != null)
            { mycam.CaptureImage();
            }
                    }
```

← Gray:ON

←Gray:OFF

← Capture Button

```
public void mycam_CaptureImageAvailable(object
sender,ContentReadyEventArgs e)
        {
            photoCounter++;
            try
            {
                string filename = photoCounter + ".jpg";
                this.Dispatcher.BeginInvoke(delegate()
                {
                    txtDebug.Text = "Camera image is available";
                });
                libraray.SavePictureToCameraRoll(filename,
e.ImageStream);
                this.Dispatcher.BeginInvoke(delegate()
                {
                    txtDebug.Text = " Image is Saved";
                });
            }
            catch (Exception ex){
                txtDebug.Text = ex.Message;
            }
        }
    }
}
```

← Camera Image Available

Screenshots

Fig. no 1 UI design for Experiment WP11

Fig. no 2 Output display on the Windows Phone Emulator

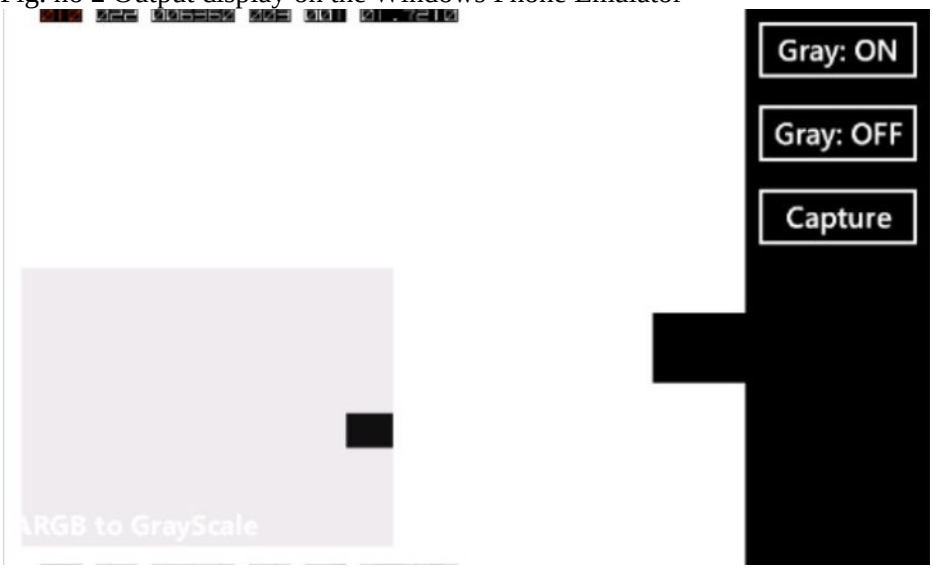

Fig. no 3 GrayOn button to get the Preview Window

Fig. no 4 Capture the Image and saved into camera roll.

Observations:

It is observed that GrayScale preview is an advance feature for Camera Operation. It is very useful in studying the DIP subject.