Use Camera with XNA

This article explains a method to access Camera with XNA.

Introduction

Under Windows Phone, Camera can be accessed only by silverlight. Windows Phone 7.5 Tango add possibility to share silverlight and XNA context in a same page. With small tricks, it's possible to use this capability to add a camera support in a XNA application.

Sharing context

In first you need a project which share context between XNA and silverlight. To do it generate a new project based on "Windows phone silverlight and XNA" template. This template add SharedGraphicsDeviceManager singleton with App.xaml

```
<Application
    x:Class="Camera_XNA.App"
    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:xna="clr-namespace:Microsoft.Xna.Framework;assembly=Microsoft.Xna.Framework.Interop">

    <!--Application Resources-->
    <Application.Resources>
    </Application.Resources>

    <Application.ApplicationLifetimeObjects>
        <!--Required object that handles lifetime events for the application-->
        <shell:PhoneApplicationService
            Launching="Application_Launching" Closing="Application_Closing"
            Activated="Application_Activated" Deactivated="Application_Deactivated"/>


    <!--The SharedGraphicsDeviceManager is used to render with the XNA Graphics APIs-->
    <xna:SharedGraphicsDeviceManager />
    </Application.ApplicationLifetimeObjects>

</Application>
```

This singleton is used to switch between XNA and silverlight Context and create XNA elements.

This template give two phonepages:

- MainPage.xaml : silverlight phonepage
- GamePage.xaml : XNA phonepage.

To use XNA, GamePage use three important steps :

1. Create a GameTimer to call OnUpdate and Ondraw callback.
2. Activate XNA context in OnNavigateTo function.
3. Deactivate XNA context in OnNavigateFrom function.

Add Camera access

Camera can be initialized only by a VideoBrush instance. In GamePage.xaml add an ui element which can use a VideoBrush. In the example will use a Canvas

```
<Canvas x:Name="cameraDisplayFake" Width="0" Height="0"></Canvas>
```

To create a Texture from camera we need :

1. `PhotoCamera` : silverlight Camera instance
2. `int []` : camera image buffer
3. Texture2D : XNA texture.

start camera

These objects will be created in a startCamera function

```
void startCamera()
 {
           //if camera instance exist, stop camera.
            if (Camera != null)
            {
                stopCamera();
            }

            //Create a PhotoCamera instance
            if (PhotoCamera.IsCameraTypeSupported(CameraType.Primary))
```


(C) Copyright Nokia 2013. All rights reserved.

```
            {
                Camera = new PhotoCamera(CameraType.Primary);
            }
            else if (PhotoCamera.IsCameraTypeSupported(CameraType.FrontFacing))
            {
                Camera = new PhotoCamera(CameraType.FrontFacing);
            }
            else
            {
                MessageBox.Show("Cannot find a camera on this device");
                return;
            }

           //you must wait camera initialization before create the buffer image and XNA texture
            Camera.Initialized += (a, b) =>
           {
                   //move to UI thread
                    Dispatcher.BeginInvoke(() =>
                        {
                           //if XNA texture exist dispose it and create a new texture with correct preview size
                            if (CameraTexture != null)
                                CameraTexture.Dispose();
                            CameraTexture = CameraTexture = new Texture2D(SharedGraphicsDeviceManager.Current.GraphicsDevice,
                                                                          (int)Camera.PreviewResolution.Width,
                                                                          (int)Camera.PreviewResolution.Height);
                            //Create image buffer with correct preview size.
                            CameraBuffer = new int[(int)Camera.PreviewResolution.Width * (int)Camera.PreviewResolution.Height];
                        }
                    );
                };

            //initialize camera with a video use by an UI element to initialize camera.
            var brush = new VideoBrush();
            brush.SetSource(Camera);
            cameraDisplayFake.Background = brush;
        }
```

`VideoBrush` will start camera initialization and when Camera is Initialized, we can finish objects creation and access preview buffer.

update texture

Update step use :

1. `PhotoCamera.GetPreviewBufferArgb32` : give current preview buffer.
2. `Texture2D.setData<int>()` : set pixels to the texture.

`PhotoCamera` and `Tetxure2D` don't have same color representation red and blue are swapped. So to set Texture pixels you must swap red and blue values of `photoCamera` pixels.

```
         void updateCameraTexture()
        {
            //if CameraBuffer != null, camera is initialized;
            if (Camera != null && CameraBuffer != null)
            {
                //get preview Image buffer
                Camera.GetPreviewBufferArgb32(CameraBuffer);

                //swap Red and Blue channel
                for (int i = 0; i < CameraBuffer.Length; ++i)
                {

                    CameraBuffer[i] = (int)((uint)CameraBuffer[i] & 0xFF00FF00)
                                      |
                                      (CameraBuffer[i] >> 16 & 0xFF)
                                      |
                                      (CameraBuffer[i] & 0xFF) << 16;
                }
                //unset Texture
                SharedGraphicsDeviceManager.Current.GraphicsDevice.Textures[0] = null;
                //update pixel values
                CameraTexture.SetData<int>(CameraBuffer);

            }
        }
```

Better position to call this function is in `OnDraw()` callback.

stop Camera

To stop camera we add `stopCamera()` function. This function disposes `PhotoCamera` instance to stop camera.

```
void stopCamera()
        {
            //dispose PhotoCamera
            if (Camera != null)
                Camera.Dispose();
            Camera = null;
            //remove Image buffer
            CameraBuffer = null;
            //remove VideoBrush used
            cameraDisplayFake.Background = null;
        }
```

Warning:

- Never try to save a `PhotoCamera` and `VideoBrush` instance.
- Always use `PhotoCamera` instance with only one `VideoBrush`.
- Always use `VideoBrush` with only one UI element.

If you don't do it, camera preview update will be doesn't work correctly.

(C) Copyright Nokia 2013. All rights reserved.

Links

- How to: Combine Silverlight and the XNA Framework in a Windows Phone Application
- Monster Cam : Application which use this method and deform camera input in real time.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.