.NET MAUI - reporting events to the GUI

In this page, we will send a notifiation of an event back to the .NET GUI.

The resulting project is as follows: ExcersisTapCallback.zip

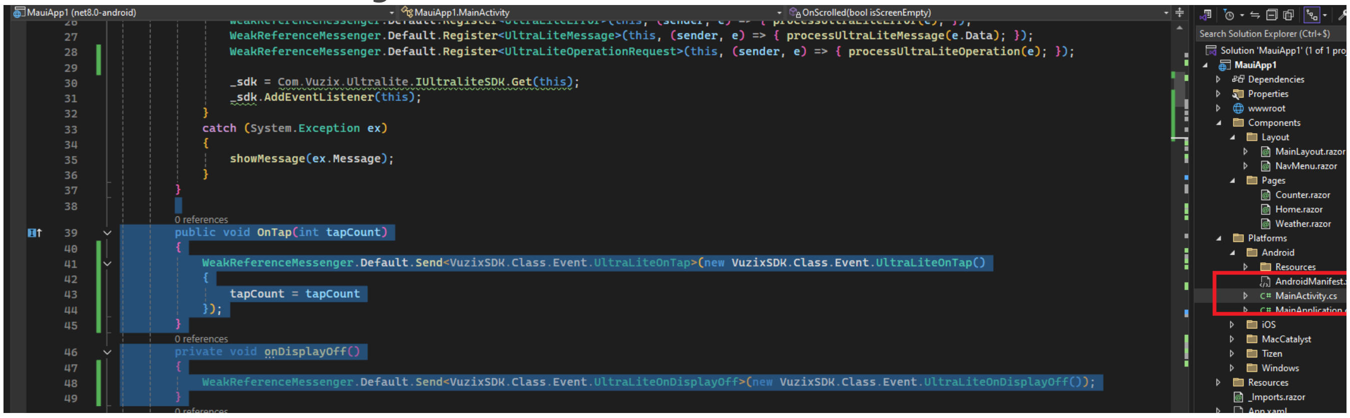
This way, we're completely disconnected with the Android stack and can focus on development in MAUI or .NET and only consider a few functions to interact with the glasses.

Note on code design: in a real world application, you would register a handler in a service in .NET - as their "Dependency Injection" is very good. But this code will be sufficient to demonstrate the interface with the glasses.

So let's go!

Well start off again with the last project .NET MAUI - Handling taps

1. Update the Nuget package, we require at least VuzixSDK 1.0.1
2. Change the event functions in the **MainActivity** to send the events via the **WeakReferenceMessenger**



```
            WeakReferenceMessenger.Default.Register<UltraLiteMessage>(this, (sender, e) => { processUltraLiteMessage(e.Data); });
            WeakReferenceMessenger.Default.Register<UltraLiteOperationRequest>(this, (sender, e) => { processUltraLiteOperation(e); });

            _sdk = Com.Vuzix.Ultralite.IUltraliteSDK.Get(this);
            _sdk.AddEventListener(this);
        }
        catch (System.Exception ex)
        {
            showMessage(ex.Message);
        }
    }

    public void OnTap(int tapCount)
    {
        WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnTap>(new VuzixSDK.Class.Event.UltraLiteOnTap()
        {
            tapCount = tapCount
        });
    }
    private void onDisplayOff()
    {
        WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayOff>(new VuzixSDK.Class.Event.UltraLiteOnDisplayOff());
    }
```

```
public void OnTap(int tapCount)
{
   WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnTap>(new
VuzixSDK.Class.Event.UltraLiteOnTap()
```

```
    {
        tapCount = tapCount
    });
}
private void onDisplayOff()
{
    WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayOff>(new
VuzixSDK.Class.Event.UltraLiteOnDisplayOff());
}
private void onDisplayOn()
{
    WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayOn>(new
VuzixSDK.Class.Event.UltraLiteOnDisplayOn());
}
private void onDisplayTimeout()
{
    WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayTimeout>(new
VuzixSDK.Class.Event.UltraLiteOnDisplayTimeout());
}
private void OnPowerButtonPress(bool turningOn)
{
    WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnPowerButtonPress>(new
VuzixSDK.Class.Event.UltraLiteOnPowerButtonPress()
    {
        turningOn = turningOn
    });
}
private void OnScrolled(bool isScreenEmpty)
{
    WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnScrolled>(new
VuzixSDK.Class.Event.UltraLiteOnScrolled()
    {
        isScreenEmpty = isScreenEmpty
    });
}
```

Your MainActivy.cs will look as follows:

```
using Android.App;
using Android.Content.PM;
using Android.OS;
using Android.Widget;
using CommunityToolkit.Mvvm.Messaging;
using VuzixSDK.Class;
using Com.Vuzix.Ultralite;
using Layout = Com.Vuzix.Ultralite.Layout;
using TextAlignment = Com.Vuzix.Ultralite.TextAlignment;
using VuzixSDK.Enum;
using Android.Graphics;
using System.Diagnostics.Tracing;
using static System.Net.Mime.MediaTypeNames;

namespace MauiApp1
{
    [Activity(Theme = "@style/Maui.SplashTheme", MainLauncher = true, ConfigurationChanges =
ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | ConfigChanges.ScreenLayout |
ConfigChanges.SmallestScreenSize | ConfigChanges.Density)]
    public class MainActivity : MauiAppCompatActivity, Com.Vuzix.Ultralite.IEventListener
    {
        IUltraliteSDK _sdk;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            try
            {
                WeakReferenceMessenger.Default.Register<UltraLiteError>(this, (sender, e) => {
processUltraLiteError(e); });
                WeakReferenceMessenger.Default.Register<UltraLiteMessage>(this, (sender, e) => {
processUltraLiteMessage(e.Data); });
                WeakReferenceMessenger.Default.Register<UltraLiteOperationRequest>(this, (sender, e) => {
processUltraLiteOperation(e); });

                _sdk = Com.Vuzix.Ultralite.IUltraliteSDK.Get(this);
                _sdk.AddEventListener(this);
            }
            catch (System.Exception ex)
            {
                showMessage(ex.Message);
            }
        }
        public void OnTap(int tapCount)
        {
            WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnTap>(new
VuzixSDK.Class.Event.UltraLiteOnTap()
            {
                tapCount = tapCount
            });
        }
        private void onDisplayOff()
        {
            WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayOff>(new
VuzixSDK.Class.Event.UltraLiteOnDisplayOff());
        }
        private void onDisplayOn()
        {
            WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayOn>(new
VuzixSDK.Class.Event.UltraLiteOnDisplayOn());
        }
        private void onDisplayTimeout()
        {
            WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnDisplayTimeout>(new
VuzixSDK.Class.Event.UltraLiteOnDisplayTimeout());
        }
        private void OnPowerButtonPress(bool turningOn)
        {
            WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnPowerButtonPress>(new
VuzixSDK.Class.Event.UltraLiteOnPowerButtonPress()
            {
                turningOn = turningOn
            });
        }
        private void OnScrolled(bool isScreenEmpty)
        {
            WeakReferenceMessenger.Default.Send<VuzixSDK.Class.Event.UltraLiteOnScrolled>(new
VuzixSDK.Class.Event.UltraLiteOnScrolled()
            {
                isScreenEmpty = isScreenEmpty
            });
        }
```

```
    public void clearScreen()
    {
      _sdk.Canvas.RemoveText(_lastTextId);
      _sdk.Canvas.RemoveAnimation(_lastAnimationId);
      _sdk.Canvas.RemoveImage(_lastImageId);
    }
    public void showMessage(string message)
        {
            MainThread.BeginInvokeOnMainThread(() =>
            {
                var toast = Toast.MakeText(this, message, ToastLength.Short);
                toast.Show();
            });
        }
        protected void processUltraLiteError(UltraLiteError error)
        {
            if (_sdk.IsConnected)
            {
                if (!_sdk.IsControlledByMe)
                {
                    _sdk.RequestControl();
                }
                if (_sdk.IsControlledByMe)
                {
                    string _title = $"[Error]{(error.Source != null ? " " + error.Source : "")}";
                    string _error = (error.Exception != null ? $"Exception : {error.Exception.Message}" : "Error
occured");
                    _sdk.SendNotification(_title, _error);
                }
            }
        }
        int _lastTextId = -1;
        int _lastImageId = -1;
        int _lastAnimationId = -1;

    protected void processUltraLiteMessage(String message)
    {
      if (_sdk.IsConnected)
      {
        if (!_sdk.IsControlledByMe)
```

```
                {
                    _sdk.RequestControl();
                }
                if (_sdk.IsControlledByMe)
                {
                    _sdk.SetLayout(Layout.Canvas, 0, true);
                    bool _messageSucceeded = false;
                    if (_lastTextId >= 0)
                    {
                        _messageSucceeded = _sdk.Canvas.UpdateText(_lastTextId, message);
                    }
                    else
                    {
                        _lastTextId = _sdk.Canvas.CreateText(message, Anchor.Center);
                        _messageSucceeded = (_lastTextId != -1);
                    }
                    if (!_messageSucceeded)
                    {
                        showMessage("Text failed");
                    }
                    _sdk.Canvas.Commit();
                    SystemClock.Sleep(1000);
                }
            }
        }
        protected void processUltraLiteOperation(UltraLiteOperationRequest Request)
        {
            if (_sdk.IsConnected)
            {
                if (!_sdk.IsControlledByMe)
                {
                    _sdk.RequestControl();
                }
                if (_sdk.IsControlledByMe)
                {
                    _sdk.SetLayout(Layout.Canvas, 0, true);
                    if (Request.Operation == eUltraLiteOperation.ShowImage && Request.ImageBitMap != null)
                    {
                        LVGLImage image = loadLVGLImage(Request.ImageBitMap);
                        bool _imageSucceeded = false;
                        if (_lastImageId >= 0)
                        {
□□□□□□□_imageSucceeded = _sdk.Canvas.UpdateImage(_lastImageId, image);
                        }
                        else
                        {
                            _lastAnimationId = _sdk.Canvas.CreateImage(image, Anchor.Center);
□□□□□□□_imageSucceeded = (_lastImageId != -1);
□□□□□□}
                        if(!_imageSucceeded) showMessage("Image failed");
                        _sdk.Canvas.Commit();
                    }
                    if (Request.Operation == eUltraLiteOperation.ShowAnimation && Request.AnimationBitMap != null)
                    {
                        LVGLImage[] image = loadLVGLImage(Request.AnimationBitMap);
                        int _animationDelay = 500;
                        if(_lastAnimationId >= 0)
                        {
                            _sdk.Canvas.RemoveAnimation(_lastAnimationId);
□□□□□□}
□□□□□□_lastAnimationId = _sdk.Canvas.CreateAnimation(image, Anchor.Center, _animationDelay);

□□□□□□if (_lastAnimationId == -1)
                        {
                            showMessage("Animation failed");
                        }
                        _sdk.Canvas.Commit();
                    }
                }


            }
            else
            {
                showMessage("SDK is not connected");
            }
        }
        private static Bitmap loadBitmap(byte[] bitmapbytes)
        {
            BitmapFactory.Options options = new BitmapFactory.Options();
            // https://proandroiddev.com/image-decoding-bitmaps-android-c039790ee07e
            options.InSampleSize = 2;

            //options.InTargetDensity = 640 * 2;
            //options.InTargetDensity = 480 * 8;
            //options.InScaled = true;
            options.InPreferredConfig = Bitmap.Config.Argb8888;
            /* options.InMutable = true;

            options.InSampleSize = 8;
□□□options.OutWidth = 600;
□□□options.OutHeight = 400;
□□□options.InScaled = scaled;*/
            Bitmap bmp = BitmapFactory.DecodeByteArray(bitmapbytes, 0, bitmapbytes.Length, options);

            return bmp;// resize(bmp, 640, 480);
        }
        private static LVGLImage[] loadLVGLImage(List<byte[]> images)
        {
            List<LVGLImage> _images = new List<LVGLImage>();
            foreach (var image in images)
            {
                _images.Add(LVGLImage.FromBitmap(loadBitmap(image), LVGLImage.CfIndexed1Bit));
            }
            return _images.ToArray();
        }
        private static LVGLImage loadLVGLImage(byte[] image)
        {
            //ColorObject[] _colors = { LVGLImage.IColorMapper.White, LVGLImage.IColorMapper.Mid };
            //LVGLImage _img = new LVGLImage(LVGLImage.CfIndexed1Bit, 480, 640, _colors, image);
            LVGLImage _img2 = LVGLImage.FromBitmap(loadBitmap(image), LVGLImage.CfIndexed1Bit);
            return _img2;
        }
    }
}
```

3. Go back to the MAUI section, **Home.razor** and add the following code. The OnInitialized() will be called by the framework when the page is ready. Then, we register a listener to a message of type **UltraLiteOnTap** and write some text in a string to show we received it

```
    string _eventText = "";
    protected override void OnInitialized()
    {
        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnTap>(this, (sender, e) => {
            _eventText = $"You have tapped {e.tapCount}.";
            StateHasChanged();
        });
    }
```

```
<h1>Hello, world!</h1>
@_eventText
@if (MyMessage != null && MyMessage.Length>0 )
{
<p>You wrote: @MyMessage</p>
}
```

We do the same for all the remaining events:

```
    protected override void OnInitialized()
    {
        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnTap>(this, (sender, e) =>
        {
            _eventText = $"You have tapped {e.tapCount}.";
            StateHasChanged();
        });

        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnDisplayOff>(this, (sender, e)
=>
        {
            _eventText = $"Display off event.";
            StateHasChanged();
        });
        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnDisplayOn>(this, (sender, e)
=>
        {
            _eventText = $"Display on event.";
            StateHasChanged();
        });
        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnDisplayTimeout>(this,
(sender, e) =>
        {
            _eventText = $"Display timeout event.";
            StateHasChanged();
        });
        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnPowerButtonPress>(this,
(sender, e) =>
        {
            _eventText = $"Power button press {(e.turningOn? "ON" : "OFF")}";
            StateHasChanged();
        });
        WeakReferenceMessenger.Default.Register<VuzixSDK.Class.Event.UltraLiteOnScrolled>(this, (sender, e)
=>
        {
            _eventText = $"Scrolled {(e.isScreenEmpty?"Empty screen":"Non empty screen")}.";
            StateHasChanged();
        });

    }
```

At this point, you should be set to build your GUI as imagined. And display them on the glasses with ease.

Revision #2
Created 8 October 2024 21:09:40 by Tim
Updated 8 October 2024 21:34:05 by Tim