MobileHMI
Creating Displays for Mobile Devices

APPLICATION NOTE

August 2014

Description: Guide to configuring MobileHMI displays that perform optimally on mobile devices.
General Requirement: Basic knowledge of GraphWorX64 and MobileHMI.

Introduction

MobileHMI is the next generation HMI/SCADA mobile application for industrial, manufacturing and building automation. MobileHMI is an application developed to allow access to operational information from anywhere and at any time. Leveraging the capabilities of the ICONICS platform technologies for HMI/SCADA and AnalytiX, MobileHMI will bring a new level of flexibility to any MobileHMI or SCADA installation. With the ability to launch GraphWorX64 displays, real-time and historical trends, navigate the Asset Catalog, monitor alarms and subscribe to push notifications for alarms, MobileHMI gives users control of enterprise scale operations from a wide variety of mobile devices.

While you can simply save your existing GraphWorX64 displays in a MobileHMI-friendly format, these displays can frequently be slow or cumbersome on mobile devices due to their relatively large size and the lack of a mouse pointer. This application note will guide you through creating GraphWorX64 displays specifically optimized for your mobile devices, giving tips for best performance and taking advantage of features specifically designed for phones and tablets.

To create a new display for mobile devices, go to **File → New Settings** and choose **MobileHMI** for the Display Type.

Considerations for Mobile Apps

Network Use

As opposed to a desktop, which is often physically connected via Ethernet to a fast network, mobile devices are almost exclusively wireless, and often have to deal with cellular connections that are both slow and may have predefined limits that could incur fees if they are exceeded.

You will want to optimize your network usage, sending only as much data as necessary to the client. See the next section, "Tips for Best Performance" for advice in this area.

Hardware

Mobile devices can be very powerful, but most of them still cannot compare with the power of a desktop. Apps you find in the various app stores perform well on mobile devices because they have been designed specifically with those devices in mind. We need to use a similar theory when designing displays for mobile devices.

You will want to design displays that are smaller and less complex than those used on a desktop. See the next section, "Tips for Best Performance" for advice in this area.

Different screen sizes, aspect ratios, and orientations

If you plan on supporting more than one type of device you may find that you have to design displays for many different resolutions and aspect ratios. A display designed for a Surface Pro 2 (16:9) may look stretched on an iPhone 4s (3:2), and the buttons will be smaller and harder to click for an iPhone user.

Plus, most modern devices can change their orientation, or aspect ratio between portrait and landscape.

See the section, "Supporting Varying Resolutions, Aspect Ratios, and Orientations" for tips on how to deal with different resolutions, aspect ratios, and orientations.

Touch interface

Most mobile devices will primarily have a touch interface, not a mouse. Fingers are not as accurate as mouse pointers, and your users may find that buttons and links that are easy to click on a desktop application are frustrating to click on a touch device.

See the section, "Designing for Touch" for suggestions in this area.

Tips for Best Performance

GraphWorX64 displays that were created for a desktop machine may not perform well on a tiny cell phone. Here are some tips to create displays that will perform best on mobile devices and use as little of your data plan as possible. In general, most of these tips come down to reducing the amount of data, both in terms of display size and in point updates.

Use the "Compressed" display type. File size will be greatly reduced, as will the amount of time and bandwidth it takes to download the display across a network. Files are cached uncompressed on the device, so there's no worry about having to decompress a page every time you go back to it.

Use the smallest possible PNG/JPG images. Don't import a huge image and stretch it down inside GraphWorX64, as this will

store or reference the entire image and scale it in runtime. This means both a large download and a performance hit to do the scaling on the fly.

Instead, use an external photo application to reduce the image size, or for embedded images, right-click on the image in GraphWorX64 and choose "Convert to image". Even though the object is already an image, this will convert it to an image of the smaller size.

For JPG files, use an external photo application to adjust the quality of the encoding to reduce the file size. Some JPG files can be made very tiny (in bytes) but still retain much of the quality of the original image.

Use as few objects as possible. Fewer objects means less graphic to load and render. Use objects intelligently - make every object count! If you don't need an object, don't add it. The simpler the better.

Use as simple objects as possible. Simple rectangles, circles, and polylines with few vertices will generally be easier on your devices than image files. Use solid colors without a lot of effects – things like gradients and shadows take more processing power to render than a solid fill.

Use a slower data rate. You will reduce your network (data plan) and server usage if your clients are getting data at a slower rate. This can be reduced with the ScanRate setting for the display.

Use as few points as possible on any one display. Rather than creating one display that has a large number of data points, create displays that summarize the data and create other displays to "drill down" to see the details. For instance, rather than displaying every point for a particular building on one display, create a display that summarizes the building by floor (with only one or two points per floor), allow the user to click on a floor to load a display that summarizes the rooms on the floor, and then allow the user to click a room to see the individual points for that room.

Test on the weakest device you expect to use. If your display works well on a weak device, it'll work even better on a more powerful one. If you develop everything for your highest performing device you may be disappointed at the performance you get from your weakest device.

> **NOTE:** Phones are not necessarily weaker than tablets. Consider the hardware, not the size of the device. You can easily have expensive phones that are much higher performing than cheaper tablets.

Supporting Varying Resolutions, Aspect Ratios, and Orientations

There are several options you can take advantage of when designing displays to be used on various resolutions, aspect ratios, and orientations.

Faceplate

You can use the faceplate feature of GraphWorX64 to estimate how a fixed-scale display will look on multiple types of devices. The faceplate can be chosen when the display is created and changed in configure mode.

To show, hide, or change the faceplate, go to the **MobileHMI Configuration** ribbon.

Scalable Displays (Canvas)

Setting a canvas display to be scalable will allow the display to shrink or expand for various-sized devices. To set a display to be scalable, either select **Scalable** in the Advanced section of the New Display dialog, or in a pre-existing display select the canvas, go to the Properties panel, and set **IsScalable** to true.

There are two ways a canvas display can scale. It can either preserve its aspect ratio (Uniform) or stretch to fit the available space (Fill). (If all of your devices have the same aspect ratio then this setting will not make any visible difference.) **Uniform** scaling will leave empty space (filled with the display background color) on the top and bottom or sides of a display that has a different aspect ratio than the original display. **Fill** scaling will stretch the display to fill all of the available space, but squares will become rectangles and circles will become ovals. These options can be set during display creation, or on the Properties of the canvas.

Grid Displays

When creating a display using File → New Settings you have the option of creating a **Grid** display instead of canvas. Grid displays are more complex and can take some time to learn, but they can provide a powerful scaling alternative. Grid displays allow you to designate certain areas of a display to scale and others to remain fixed, creating displays that can appear to fill the available space intelligently without stretching the individual elements – circles remain circles and squares remain squares.

Grid displays can be used to create one display that deals well with different aspect ratios, resolutions, and even orientations!

For more information on grid displays, see the application note entitled, "GraphWorX64 - Using Grid Panel for Resizable Displays".

Orientation Lock

The easiest way to deal with varying orientations is to simply pick one and have your displays fixed to that orientation. If you would like to lock your display to one orientation, select the canvas, go to the Properties panel, and set **OrientationLock** to the desired orientation.

However, fixing the display to one orientation may annoy users who prefer to use their devices in an orientation counter to the one you have picked. This will vary by user and by device - a Surface Pro user may prefer to use his device in landscape mode so he can use a keyboard, but an iPhone user may prefer to use his device in portrait mode because it is easier to hold that way.

If you would like to allow your devices to freely switch between portrait and landscape, consider grid displays (see previous section) or see some of the options below.

Layer Orientation Visibility

Each layer on a display has an **OrientationVisibility** property which can make it visible only in horizontal mode or vertical mode. This can allow you create two layers, one for portrait mode and one for landscape mode.

> **NOTE:** OrientationVisibility is an Advanced Mode option – you may need to go to the View ribbon and change Application Mode to Advanced to see it.

Consider the display below in both portrait and landscape mode. The square area with the house stays in the same place, while the two orange squares can be on layers that hide or show based on the orientation.

Figure 1 - Portrait versus Landscape

Be aware that the objects on these hidden layers will increase the size of your display, and thus may decrease performance. If you have any data points on these layers you may want to set **ReleaseDataWhenHidden** to True to reduce bandwidth usage unless you have the same set of points elsewhere on your display (the same point on a display twice will *not* double your data usage for that point).

Move Objects on Orientation Change

As an alternative to the layer method above, there is a local variable that represents the orientation of the display. This variable is called, **localsim::currentOrientation**, and you can find it on the Simulations tab of the tag browser. 1 is portrait mode and 2 is landscape.

Instead of hiding the orange areas based on the orientation, you can actually move them with a Location Dynamic. This method is less intuitive to set up, but more efficient for the display, as it doesn't require duplicate information.

You can even combine both of these methods, if desired – have some information on layers and some that moves.

Swap Canvas Dimensions on Orientation Change

The **SwapDimensionsOnOrientationChange** property of the canvas can be useful in displays that are not orientation-locked. When this option is set to true the canvas dimensions will be swapped before any scaling or stretching is done. Scalable displays with this option set to false may attempt to scale when the orientation is changed, leading to either very stretched displays or displays with a lot of unused space.

Scrolling

Another option is to use a fixed-scale display with SwapDimensionsOnOrientationChange set to false and allow the orientation change of the device to make different areas visible. The user can be allowed to scroll or pan to these areas as well.

Referring to Figure 1 above, with this technique the display dimensions would be square (for example, 1136x1136) and there would be four orange areas, two of which are off the screen at any given time.

Customized Displays per Device

You also have the option of creating displays specifically customized for a particular device or set of devices. If you are taking advantage of the MobileHMI AppHub you can configure an AppHub to direct specific device types to particular displays.

For example, if your set of client devices includes 16:9 tablets, 16:9 phones, and 4:3 phones you can create three copies of your

main display, one for each of these sets of devices. Each of these displays would take the aspect ratio and type of device into consideration – the phone displays might have bigger buttons and be locked in portrait mode, and the 4:3 display would have a slightly different layout than the 16:9 displays.

This option involves maintaining multiple copies of what is essentially the same display, possibly making it harder to make changes in the future, but it allows you to more fully control exactly what the display will look like on each device.

See the application note entitled, "MobileHMI - Using the Applications Hub 'App Hub'" for more information about the AppHub .

Designing for Touch

Use big buttons and controls. You will want to design your buttons and other controls to be much bigger than their desktop counterparts. In addition, smaller devices will need relatively bigger buttons for the best experience – a button that is well sized for a tablet user may be frustratingly small for a phone user when the display has been scaled to the smaller screen.

You may want to take advantage of Smart Tiles to create areas of your screen that are both informative and have a large "touchable" area. See the application notes entitled, "GENESIS64 - Smart Tiles for HMI SCADA Applications" and "GraphWorX64 - Smart Tile Technology" for more information on Smart Tiles.

Use safe zone protection. To avoid accidental touches on critical areas you can designate part of your display as a "safe zone" by creating an object (such as an image or shape) and adding a "Safe Touch Zone" pick dynamic to it.

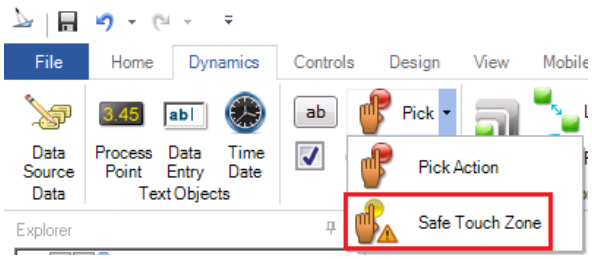

Figure 2 - Safe Touch Zone

Your buttons, process points, and other controls can be configured to only allow input when the safe zone is active. Edit the desired dynamic and set **SafeTouch** to true.

> **NOTE:** SafeTouch is an Advanced Mode option – you may need to go to the View ribbon and change Application Mode to Advanced to see it.

In runtime, a user must be touching a designated "Safe Touch Zone" object at the same time that he interacts with a dynamic designated as "SafeTouch". There is also a local variable, **localsim::safeTouchZoneActive**, you can reference to see whether the Safe Touch Zone is active. You can use this variable to give the user visual feedback that the Safe Touch Zone is active (unhide some text, or change the color of something). You can find this variable on the Simulations tab of the tag browser.