From Design to Developer

an informal outline of “how to’s” and best practices when exporting visual assets for theme

Define what is common

Once the design has settled on a common set of text sizes, colors, textures, and shapes we can form a coherent story on what the designers and developers can use to define a theme. Establishing these specifications ensures a consistency across the system and allows engineers to make informed decisions in the absence of a designer.

Text:

Color		**Size**	
Normal	**- Example**	Small	- Example
Highlight	**- Example**	Medium	- Example
Dissabled	**- Example**	Large	- Example

Color Palette:

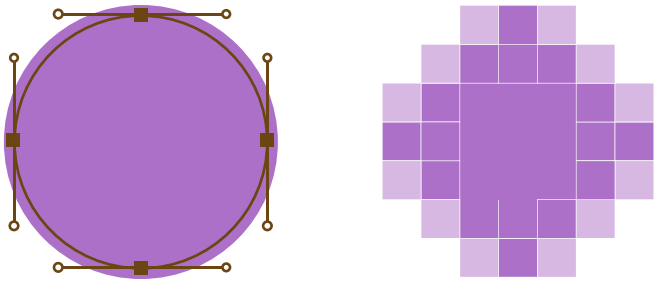
Visual touch states:

It's best to hard code as little as possible to keep things update-able. For this reason any property that can be commonly defined should be so defined in a document such as the theme.ini (look at page 6). Additional definitions could include common **heights, widths, margins, padding, or opacity**

Any element that has the potential to be more complex than a single color (with a border perhaps) needs to be created as a graphic to enable another theme creator to develop their own theme package. Even if in our theme the graphic looks the same from state to state or with other another graphic we should abstain from using graphics outside of their intended destination. Textured graphics ideally need to be as small as possible and capable of being tiled.

Note: the optimal way to generate graphics for a theme is in SVG format. SVGs (scalable vector graphics) are scalable, easily updated, and usually have cleaner edges because they are based on mathematical points rather than pixels. They can be time consuming to build, but once created can speed up a large part of maintaining and exporting for the theme.

MeeGo is not at this moment capable of handling SVG's in its theme engine, however it is a goal. Building your graphics as SVG compliant not only enables you to export for different screen densities, but will put your theme in a position for future updates and changes to both theme and code.

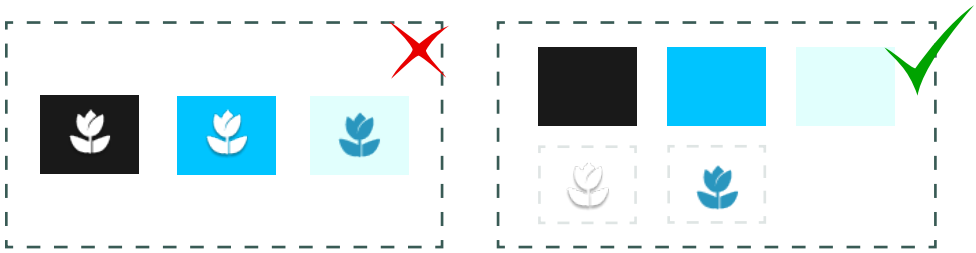


Please refer to James Ketrenos' work on scaling applications between device sizes and resolutions at: http://muxbox.jf.intel.com/wiki/index.php/~jketreno/scaling or at our joint presentation slides at: http://www.slideshare.net/Zybean/crossing-the-resolution-divide

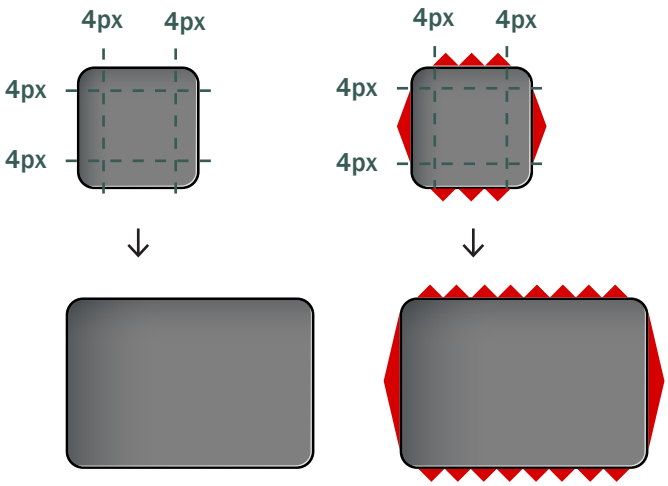
Separating your graphics

When an engineer is putting together a layout (in QML for our purposes), they prefer to write as little code as possible. Keep in mind that the UI you are creating assets for is likely going to be ported to a different size and resolution to run on a variety of devices. *Flexibility is important!* By following these guidelines it makes everyone's life a little easier:

I. **Separate icons and backgrounds**: if the background graphic, selected, and highlighted shape or colors are used in multiple places then only one graphic needs to be generated which can be called repeatedly. The same is true for icons. Doing this creates part of our consistency in a theme, reduces package sizes, and enable components to be built in the code that can be duplicated from app to app.



II. **Reduce your shapes and use borderImage**: should your shapes or assets contain rounded corners or other unique styling on the side, reduce it to the smallest size and make use of QML's abilities. The property borderImage allows the developer to specify "borders" on a png that will automatically keep the edges from distorting while the shape scales to fit the needs of the interface. The developers in fact prefer this method.



Note: the pixel specification of the border should be the lowest number possible to promote the best possible scaling

borderImage, depending on what it's set to, can either tile the middle parts or stretch them. This example is tiling horizontally and stretching vertically

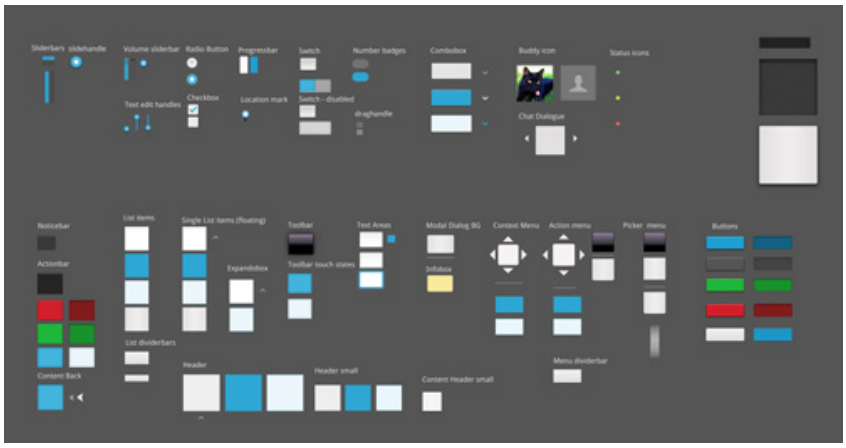
To specify your borders, write them in to a text file with a .sci extension and write the following items inside.

```
border.top:    4
border.left:   4
border.right:  4
border.bottom:4
source:        box-background.png
```

An SCI file needs to be made for each graphic. Name the SCI file the same name as the asset and place it in the same folder as its corresponding graphic. An SCI file will automatically be referenced by the MeeGo theme engine for it's respective graphic.

III. **Think hard, then export:** what is the best way to do this? If it's just one application you're working on, perhaps you can get away with the QML export script available for Photoshop or Gimp (see page 8). Perhaps some other export script is best. I do not recommend 'slicing' your graphics as you usually leads to a bloated graphics packages.

In terms of a whole system theme, a good thing may be to create a components document that has all the common assets reduced to the smallest parts to be directly exported from PSD to theme. If SVG becomes possible, then a single document with a parser in the theme engine would be the ideal.



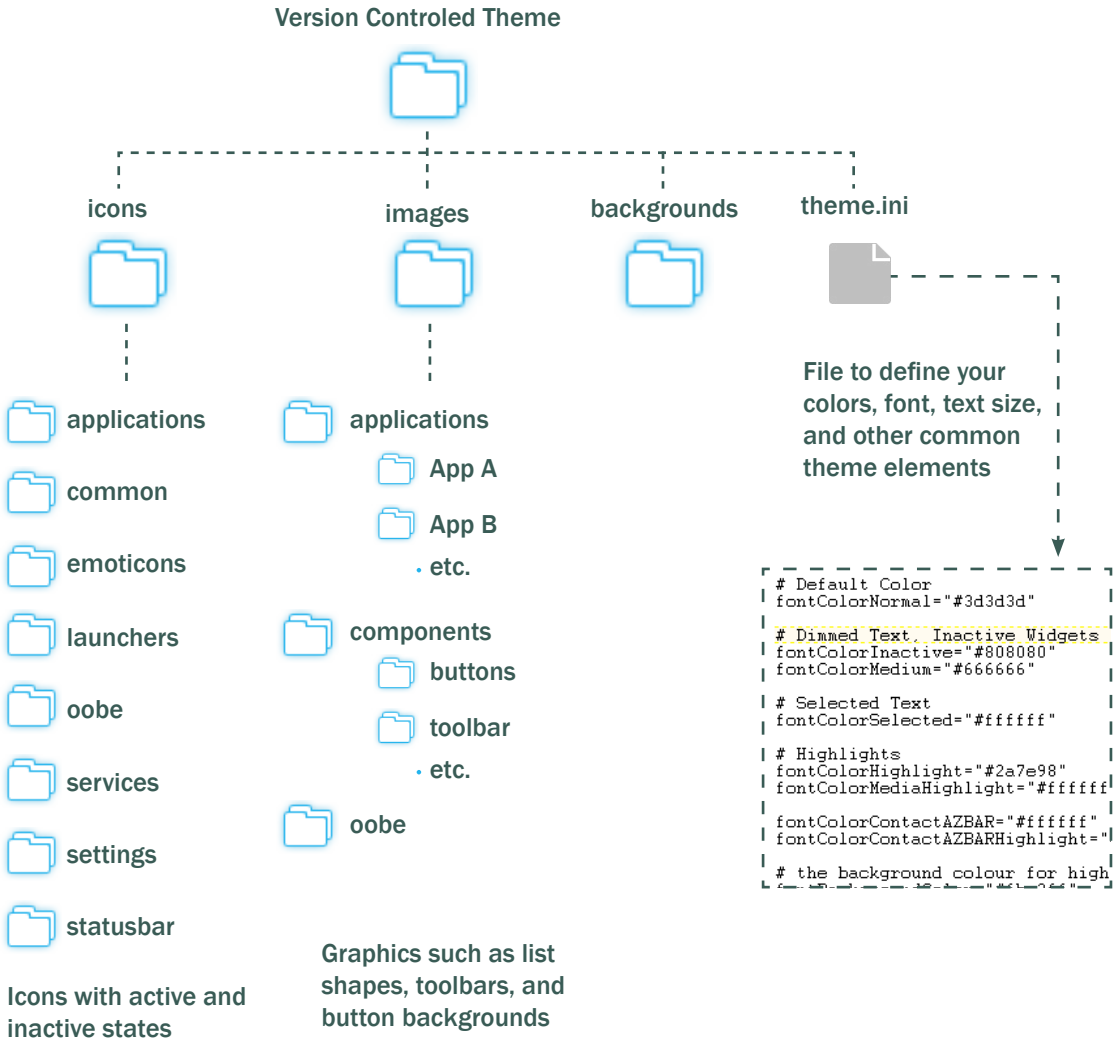
MeeGo componenet psd for theme: each element is broken down in to it's most basic parts

Asset management

I. **Designate a gatekeeper:** someone should be monitoring the theme to make sure everything is consistent, up-to-date, on par, and as complete as possible.

II. **Stay organized:** most graphics should end up in a common directory since they will be used all over the system. If there are some application specific graphics then they should go into their own folder.

Icons also need to have a directory of their own. It's easier to keep track of what already exists and how you are using your icons if you keep them in a dedicated location. Following a naming convention above will also help you to organize all of your graphics properly.



III. **Naming conventions:** a label may seem to make sense initially or make sense to the asset creator, but if it's meaning or its ability to be located easily is lost on other then the name is useless.
By first location your icons in an intelligible directory structure and then using a common naming convention, such as the one below, the intention of your assets will be more apparent to your coworkers and yourself two months down the line.

(directory) / [application] - [action] - [local position(toolbar, top, bottom, left, right, etc.) or variant (signal strength, battery level, what's being acted upon, etc.)] - [state(pressed, unpressed, selected, active, up, down, etc)] . [png, svg, jpg, etc.]

For example we would name the following graphic as such:

theme/icons/common/media-shuffle-active.png

theme/icons/applications/camera-capture-mode-auto-selected.png

theme/icons/settings/wifi-signal-strong.png

Name your asset based on its action rather than what it looks like and you need only include a local position or variant if it's beneficial.
When it comes to the states of your graphics pick one paradigm and stick with it. For example the MeeGo theme uses a normal, 'active,' and 'selected' designation for our graphics.

From Photoshop and Gimp → QML wizardry

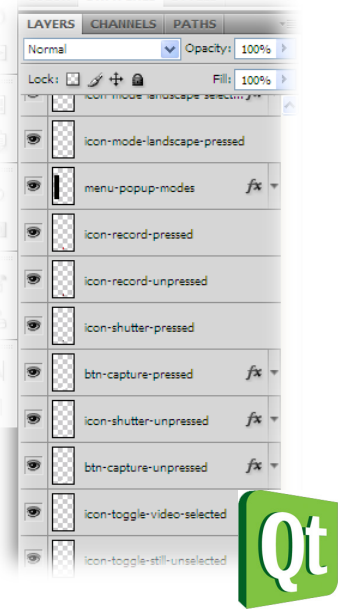
The QML export script works very much like the 'Export layers to files' script that Photoshop CS4 currently has. (I'm not aware if it is compatible with CS5 as well) It will trim and export the layers to files and create a source file describing the location of each graphic and its opacity if it applies. To get a clean export though there is a bit of prep (at least for Photoshop) that needs to occur before you press the 'Run' button. They are as follows:

1. **Save as new:** you don't want to do this to your original file!

2. **Rasterize and condense.** Make sure all shapes and layers are in the state you would want them to appear in code. This is where you would resize shapes that are rounded or have styles to the most compact state. Anything you're not sure of that can wait until later, can wait until later. You can always re-export if you have to.

3. **Name your layers** according to the convention mentioned above and be sure to use dashes **NOT** underscores or spaces. For some reason underscores interfere with the initial run of the source file in QT Creator. A developer can batch change later if it's really necessary.

4. **Remove all grouping.** It will cause an error in the exporting process and stop. You should be left with only layers.



For Cyrene's Peace of Mind

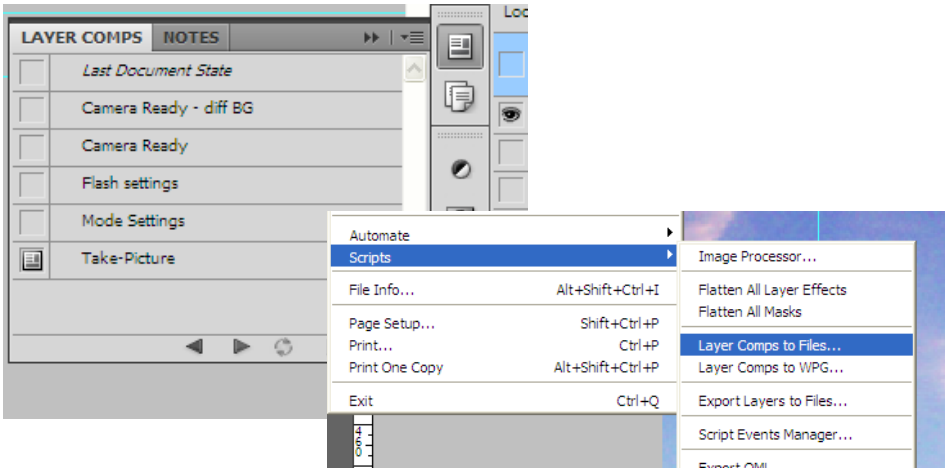
Cyrene would love to see these things happen:

I. Keeping all the icons in one file in vector format - preferably in an Illustrator .ai file. Finding and pulling icons out of a Photoshop file is all well and good until you need to change something or duplicate a shape. Having a master file helps me keep better tabs on what is and is not done, keeps things organized, and easier to manipulate than with the less capable vector functions of Photoshop.

II. Include down states in your Photoshop files for buttons or any other touchable object. I don't want to guess on what was intended.

III. Use Layer Comps in your Photoshop file(s). It's really easy and may reduce the amount of PSD's you generate. When it comes time to export full-screen mockups all you have to do is run a script, "Layer Comps to files," and the deed is done.

IV. Don't design your PSD file in such a way that it can't be broken apart easily. Try to build the pieces with the knowledge that they will be reduced to basic parts to be used by the theme. (If you're recreating the pieces in SVG format then this is not a worry)



If you require more information on any part of this please email me.
Cyrene Domogalla | cyrene.domogalla@intel.com