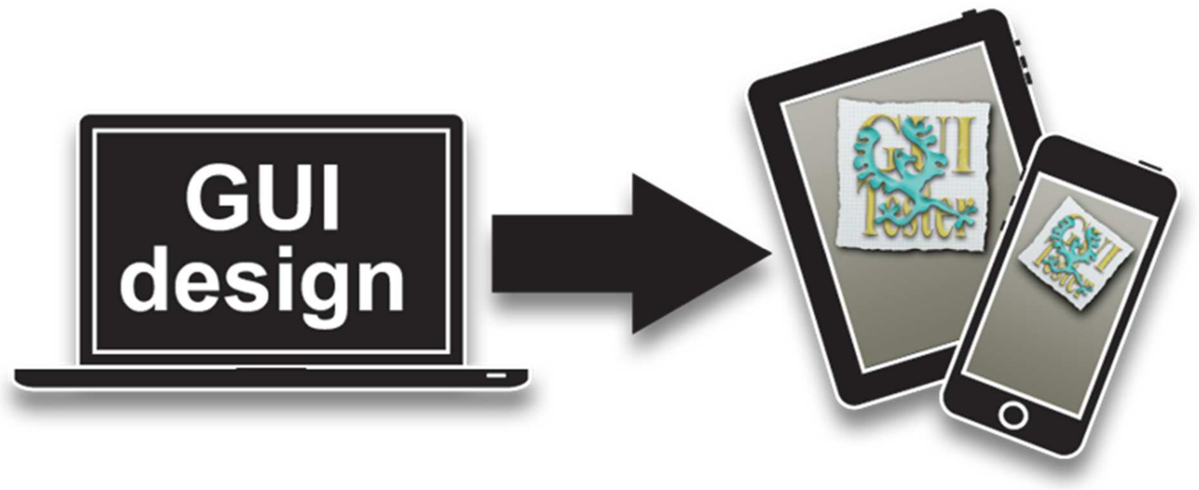
GUI Tester v1.0.0

Introduction

GUI Tester makes it possible to visualize ideas to iPhone and iPad devices in a fast and simple way that still can be close to the finished product. The result will be shown in its correct environment, on an actual iOS device, with the accurate look and feel. It's just up to the designer to choose the GUI's complexity from simple sketches to a more advanced GUI with buttons, navigation bars etc.

Design the graphics in your preferred environment. Create xml files to describe the GUI. Upload your data (images and xml files) to a reachable place somewhere on the www. Add the URL in GUI Tester, load the GUI and try it out as it was an actually app.



This document will describe all the steps to create your first GUI in detail.

Only a few features will be used in this first example.
Check out the full documentation for details about all other features.

Setup XML documentation:
http://www.sk-software.com/guitester/docs/gui_setup_xml.pdf
Scenes XML documentation:
http://www.sk-software.com/guitester/docs/gui_scenes_xml.pdf

For the latest news and more examples. Visit our homepage:
http://www.sk-software.com/guitester

Get started

Finally! Now we will create a simple GUI and test run it in GUI Tester.

Make sure (the obvious) to install GUI Tester on your iOS device. This first test will only support landscape mode on iPhone 3GS, 4 and 4S. More about different devices later.

Open your text editor, preferable one that reads xml in a decent matter.

Okay, we need two xml documents the first one is used to tell GUI Tester some basic information about the GUI. The GUI’s title, which devices it supports, splash images and a few more things. Let us call this one “first_gui.xml”.

Start with this content, explanation will follow.

```
<setup>

    <title>First GUI</title>

    <splash_image_time>2</splash_image_time>

    <iphone>

        <scenes_xml>iphone/first_gui_iphone.xml</scenes_xml>

        <portrait>
            <screen_3.5inch>

                <splash_image>
                    <url>iphone/images/portrait-splash.png</url>
                </splash_image>

            </screen_3.5inch>
        </portrait>

    </iphone>

</setup>
```

Tags used in the file “first_gui.xml” above.

<setup>
The root element of the XML file. Required.

<title>
The name of your GUI. Will be shown in GUI Tester. Required.

<splash_image_time>
A feature to simulate the time to load the app just to make it possible to show the splash screen correctly.

<iphone>
iPhone specific information. (Use <ipad> for a GUI that supports iPad)

<scenes_xml>
URL to the xml-file that describes the scenes in the GUI.

<portrait>
Content for the portrait mode is added inside this tag.
Use the tag <landscape> to add support for landscape mode.

<screen_3.5inch>
Content for devices with 3.5 inch displays (iPhone 3GS, 4 and 4S)
Is added inside this tag. Use the tag <screen_4.0inch> to add support for iPhone 5 displays.

<splash_image>
Splash image used for 3.5 inch displays and portrait mode.

<url>
URL to the image.

That’s all for this example. To add support for landscape mode and more devices see the section “Orientation and device support” below.

So now we have the first xml-file ready to use. Next step is to create the xml-file that will describe the actual look of the GUI. The tag <scenes_xml> in the setup xml is used to tell GUI Tester which file to load. You can see in the example above that we named it “first_gui_iphone.xml”

Create a file called “first_gui_iphone.xml” and add this content. Save it in a subfolder called “iphone”. Explanation will follow.

```
<scenes>
    <scene>
    <name>scene01</name>

    <objects>

        <portrait>
            <screen_3.5inch>

                <button>
                    <pos>34,100</pos>
                    <default_image>
                        <url>/iphone/images/iphone-scene02-default.png</url>
                    </default_image>
                    <pressed_image>
                        <url>/iphone/images/iphone-scene02-pressed.png</url>
                    </pressed_image>
                    <z_order>15</z_order>
                    <goto>
                        <destination>scene02</destination>
                        <transition>CurlUp</transition>
                    </goto>
                </button>

                <image>
                    <pos>0,0</pos>
                    <url>/iphone/images/iphone-main-bkg-portrait.png</url>
                    <z_order>10</z_order>
                </image>

            </screen_3.5inch>
        </portrait>

    </objects>

</scene>

<scene>

    <name>scene02</name>

    <objects>

        <portrait>
            <screen_3.5inch>

                <button>
                    <pos>34,100</pos>
                    <default_image>
                        <url>/iphone/images/iphone-scene01-default.png</url>
                    </default_image>
                    <pressed_image>
                        <url>/iphone/images/iphone-scene01-pressed.png</url>
                    </pressed_image>
                    <z_order>15</z_order>
                    <goto>
                        <destination>scene01</destination>
                        <transition>CurlDown</transition>
                    </goto>
                </button>

                <image>
                    <pos>0,0</pos>
                    <url>/iphone/images/iphone-main-bkg-portrait.png</url>
```

```
                        <z_order>10</z_order>
                    </image>

                </screen_3.5inch>
            </portrait>

        </objects>

    </scene>

</scenes>
```

Tags used in “first_gui_iphone.xml”.

<scenes>
The root element of the XML file. Required.

<scene>
At least one scene is required. In this test we use two scenes.

<name>
The name of the scene. It’s required for the scene to have a name.

<objects>
Here will we add all the objects for this scene.

<portrait>
Content for portrait mode is added inside this tag. For this example we only use portrait to keep it simple. Use <landscape> to add support for landscape mode.

<screen_3.5inch>
Content for devices with 3.5 inch displays (iPhone 3GS, 4 and 4S) is added inside this tag. Use the tag <screen_4.0inch> to add support for iPhone 5 displays.

<button>
Declares a button. In this test we use images to display the button. Read more about buttons in the full documentation.

<pos>
Sets the position of the button.

<default_image>
Declares the image used for the default state of the button.

<url>
URL to the image.

<pressed_image>
Declares the image used for the pressed state of the button.

<url>
URL to the image.

<goto>
Declares the action (which scene to go to) when the button is pressed.

<destination>
The name of the scene that we will transfer to.

<transition>

What kind of transition to be used. Read more about valid transition types in the full documentation.

<z_order>

Not always necessary but it’s easier to have the control than not. Read full documentation for more info.

<image>

Declares an image. In this example we uses one image as a background.

<pos>

Sets the position of the image.

<url>

The URL to the image.

That’s it. The first GUI is created and it contains the basic components just to show how it can be configured with two scenes and a button to navigate between them.

Now upload the two xml files and the needed images to a web server.
A public folder on a service like dropbox will also work.
Just make sure that the files are directly accessible from an URL.
The duplicated images with the “@2x” prefix are the retina version of the images.

Following files needs to be uploaded:

- first_gui.xml
- /iphone/first_gui_iphone.xml
- /iphone/images/portrait-splash.png
- /iphone/images/portrait-splash@2x.png
- /iphone/images/iphone-main-bkg-portrait.png
- /iphone/images/iphone-main-bkg-portrait@2x.png
- /iphone/images/iphone-scene01-default.png
- /iphone/images/iphone-scene01-default@2x.png
- /iphone/images/iphone-scene01-pressed.png
- /iphone/images/iphone-scene01-pressed.png
- /iphone/images/iphone-scene02-default@2x.png
- /iphone/images/iphone-scene02-pressed.png
- /iphone/images/iphone-scene02-pressed@2x.png

XML-files and images used in this example can be found here:

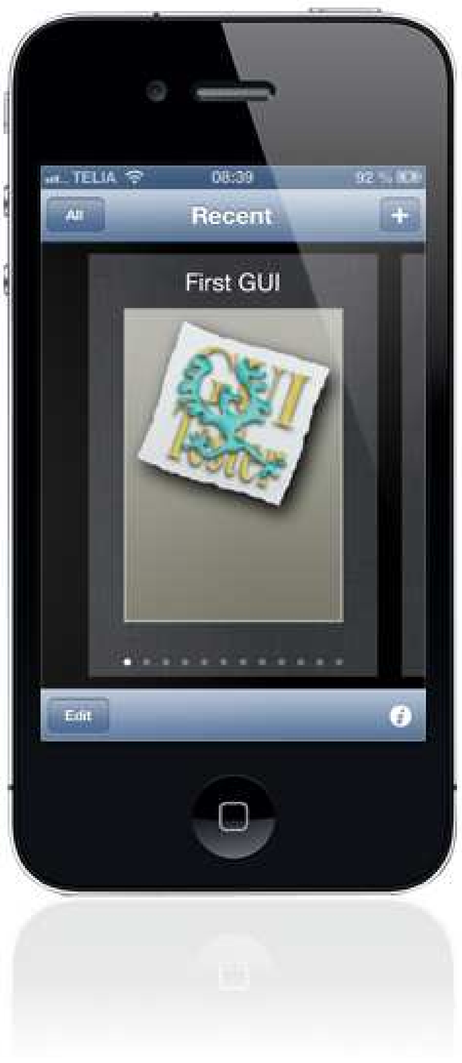
http://www.sk-software.com/guitester

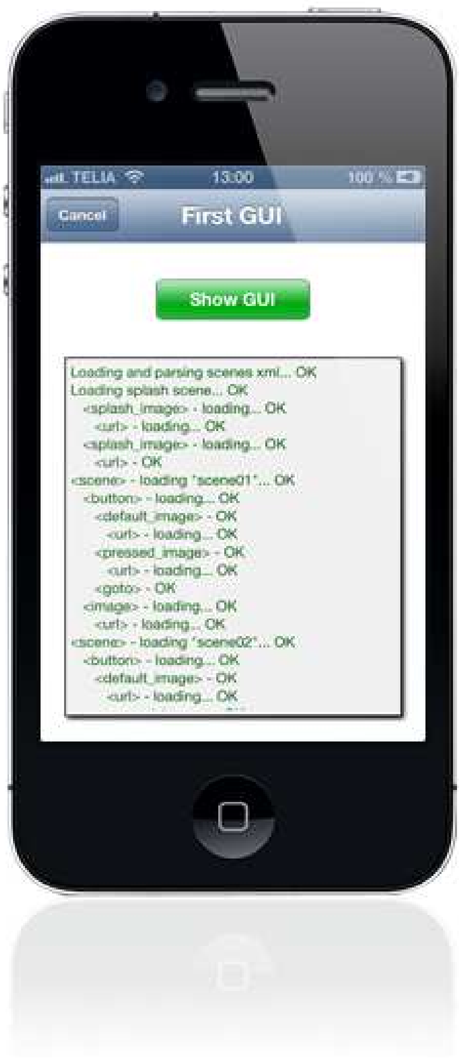
It’s finally time to test the GUI in GUI Tester.

Start GUI Tester on your iPhone.
Add the GUI by tapping the + button in upper right corner.
Enter the URL to the setup xml file, “first_gui.xml”.

Tap the “Save” button and wait for the setup file to be loaded. When finished you will return to the “Recent” view and a preview image of your GUI will be shown. Tap the preview image and wait for the GUI to load and run.

To return from the GUI tap and hold for 3 seconds.





```
Loading and parsing scenes xml... OK
Loading splash scene... OK
  <splash_image> - loading... OK
    <url> - loading... OK
  <splash_image> - loading... OK
    <url> - OK
<scene> - loading 'scene01'... OK
  <button> - loading... OK
    <default_image> - OK
      <url> - loading... OK
    <pressed_image> - OK
      <url> - loading... OK
    <goto> - OK
  <image> - loading... OK
    <url> - loading... OK
<scene> - loading 'scene02'... OK
  <button> - loading... OK
    <default_image> - OK
      <url> - loading... OK
```

Tips and tricks

There are some different methods how to create a GUI. The first and the simplest one is a GUI with only a background image for each scene.
To switch between scenes add <tap> tags over the areas on the background image that you want to trigger the switch (or even a big tap area over the entire scene if you just want to show some conceptual images.

A simple GUI like that is well suited for a fast glance of the application and its general idea.

The second approach are to create a GUI with more functionality, by using real iOS UI buttons, text fields, navigation bar etc. This is good when you want to test the general design, functionality and the flow in an app. It’s a way to give your design a kick start into the real thing without doing any programming.

GUI Tester supports many of the standard iOS UI elements. Labels, images, tool bars, tables, navigation bar etc. This makes it easy to create a design with a classic iPhone look and feel.

To make your design feel less static and more alive use the <animation> tag. Only a few frames can be needed to make a point of how the finished design is supposed to work and how it should look.

Another way to make the design more usable is to use the tags <page_scroll> and <image_scroll>. With these tags you can easily simulate scroll views. Make a high score table that you can scroll down or an image viewer where you can actually slide between images.

See the full documentation for a complete list of features.

Orientation and device support

The Getting Started example above only supports portrait mode and devices with 3.5 inch screens (iPhone 3GS, 4 and 4S).

To add support for more devices and orientations you need to add some tags.

The <portrait> tag is used to describe the scene when in portrait mode.
To describe the scene in landscape mode add the tag <landscape>.

```
<scene>
    <objects>

        <portrait>

<!-- All objects for the portrait mode inside this tag. -->

        </portrait>

        <landscape>

<!-- All objects for the landscape mode inside this tag. -->

        </landscape>

    </objects>
</scene>
```

Another alternative is to make a design that supports auto rotate.
To do this add all your objects directly inside the <objects> tag.

OBS! If you have any object with autorotation the <portrait> and <landscape> tags will be ignored. You can’t mix objects that auto rotates with the <portrait> or <landscape> tags.

```
<scene>
    <objects>

<!-- Objects inside this tag will auto rotate.
Use the <auto_resize> tag to setup how the object should behave on rotation. -->

        <image>
            <pos>0,0</pos>
            <url>yourimage.png</url>
            <auto_resize>width,height</auto_resize>
        </image>

    </objects>
</scene>
```

The <auto_resize> tag will define how the object is automatically resized on rotation. See full documentation for a complete list of valid values.

Next step is to add support for different devices.

GUI Tester supports GUI’s for both iPhone and iPad.
iPhone and iPad uses different xml files to describe their scenes.

You set this up in the setup.xml
Use the <iphone> tag to add support for iPhone and the <ipad> tag for iPad.

```
<setup>

    <iphone>

        <!-- Add tags for the iPhone interface here. -->
        <scenes_xml>iphone_scenes.xml</scenes_xml>

    </iphone>

    <ipad>

        <!-- Add tags for the iPad interface here. -->
        <scenes_xml>ipad_scenes.xml</scenes_xml>

    </ipad>

</scene>
```

The setup xml must have at least one of the <iphone> or <ipad> tag to be valid.

On iPhone we also need to add tags to support iPhones with the 4.0 inch screen (iPhone 5).
To do this we will use the tag <screen_4.0inch>. The tag <screen_3.5inch> is used for devices with 3.5 inch screens (iPhone 3GS, 4 and 4S).

```
<scene>
    <objects>

        <portrait>

            <screen_3.5inch>

<!-- All objects for 3.5 inch devices in portrait mode inside this tag. -->

            </screen_3.5inch>

            <screen_4.0inch>

<!-- All objects for 4.0 inch devices in portrait mode inside this tag. -->

            </screen_4.0inch>

        </portrait>

        <landscape>

            <screen_3.5inch>
```

```
<!-- All objects for 3.5 inch devices in landscape mode inside this tag. -->

            </screen_3.5inch>

            <screen_4.0inch>

<!-- All objects for 4.0 inch devices in landscape mode inside this tag. -->

            </screen_4.0inch>

        </landscape>

    </objects>
</scene>
```

As with the orientation tags we can also make a design that supports auto resizing for the different screen sizes.
To do this just add your objects directly inside the <portrait> and/or <landscape> tags.

```
<scene>
    <objects>

        <portrait>

<!-- All objects for both 3.5 and 4.0 inch devices in portrait mode inside this tag.
Use the <auto_resize> tag to setup how the object should behave when resizing. -->

        <image>
            <pos>0,0</pos>
            <url>yourimage.png</url>
            <auto_resize>width,height</auto_resize>
        </image>

        </portrait>


        <landscape>

<!-- All objects for both 3.5 and 4.0 inch devices in landscape mode inside this tag. -->

        </landscape>

    </objects>
</scene>
```

The <auto_resize> tag will also work for different screen sizes. Just in the same way as it works for different orientations.

As a final example we will do a design that supports both autorotation and different screen sizes. This is easily done by adding all your objects directly inside the <objects> tag.

```
<scene>
    <objects>
```

```
<!-- All objects directly inside this tag will make them both resize for different screen sizes and automatically
adjust on rotation. Use the <auto_resize> tag to setup how the object should behave when resizing. -->

        <image>
            <pos>0,0</pos>
            <url>yourimage.png</url>
            <auto_resize>width,height</auto_resize>
        </image>

    </objects>
</scene>
```