Creating a Skin MyGUI 3.0.0  
  
Part 1 - skin buttons (Button)  
  
Skin (Skin) is a description of how it will look widget (Widget). Skin does not have to be in one piece texture, and can be split into parts (BasisSkin). Each of these parts, you can assign additional parameters (Property) and alignment (Align). There may be several skin conditions (State), for example, usually a button differently when we look at it or just put things clicked the mouse. To the skin was visible, it is necessary to describe at least one state - «normal».

Creating a simple skin  
  
So, let the artist painted texture «MyButton.png» from which we have to make a button:



Try to describe a simple variant skin .  
  
It is assumed that you are familiar with xml format . Basic version xml resource file with a single resource .

<?xml version="1.0" encoding="UTF-8"?>

<MyGUI type="Resource" version="1.1">

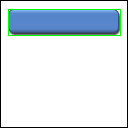
<!-- Our skin is the easiest option -->

<Resource type="ResourceSkin" name="MyButton">

</Resource>

</MyGUI>

So, as we see the resource itself is described in the node «Resource». Attribute «type» describes the type of resource , in this case «ResourceSkin», means that the resource is skinned . Attribute «name» indicates the resource name , the names of all the resources of all types must be unique. Named in the code we can access the resource , as it makes the widget instance . This skin will work and you can access it , but of course nothing will be seen , since we did not describe the visual part .  
  
To begin, select the texture in the editor and find out the size of the button and add this information to the resource.



<Resource type="ResourceSkin" name="MyButton" **size="113 27" texture="MyButton.png"**>

</Resource>

We pointed out the size of our skin - 113 in width and 27 in height. The size is specified in pixels, between the values ​​worth one space. We also specify the name of the texture in which our skin . But in order for the skin still appeared necessary to describe at least one part of one state.

<Resource type="ResourceSkin" name="MyButton" size="113 27" texture="MyButton.png">

<BasisSkin type="SubSkin" offset="0 0 113 27" align="Stretch">

<State name="normal" offset="8 9 113 27"/>

</BasisSkin>

</Resource>

In this specification, there are many new nodes with attributes : firstly appeared node «BasisSkin» describes the part of the widget and the child node to it «State» describing one condition of the part .  
  
In our simple example, the skin consists of one part , which extends over the entire skin . This part is of type «SubSkin» specified in the attribute «type». SubSkin is one of many types of parts for the widget. This type is essentially a portion of a texture that will stretch on this part . Attribute «offset» sets the position of the inside of the skin in the format «Left Top Width Height» (« top left width height "). dimensions in pixels , between the values ​​of the gap. Since our part stretched in all its skin then be in the position coordinates 0 0 and the size equal to the size of the skin . In attribute «align» specified behavior of the changing size of the skin . Meaning «Stretch» ​​indicates that the part will stretch along with skin .  
  
Some of the present description of a state of «State». State has the name specified in the attribute «name», by this name in the code , you can switch the state of the skin to change the visual display . In attribute «offset» specified offset within the texture. Offset is a pixel format «Left Top Width Height». It is important to understand that the shift in the texture , you can specify an arbitrary , it does not necessarily have to look up to the size of the skin and do not necessarily have the same position. In our case, the button in the texture itself is drawn offset 8 and 9 of the top left pixel , and the size we choose as the part of the texture to be displayed clearly , pixel by pixel .  
  
Before continuing, save the resource file named «MyButton.xml» and copy together with the texture «MyButton.png» in the resources folder MyGUI.  
  
In the code, load the skin and create a button with our skin:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 113, 27),

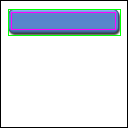
MyGUI::Align::Default,

"Main");

If you did everything right button to appear, but which have not yet pressed and has no text.



Adding text part  
  
To start, you need to define a border that will contain the text to text climbs up on the curb or button.



Get the size of Part 105 by 18 pixels and displacement inside the skin by 3 pixels on both the left and top. Now we add our part of the text.

<Resource type="ResourceSkin" name="MyButton" size="113 27" texture="MyButton.png">

<BasisSkin type="SubSkin" offset="0 0 113 27" align="Stretch">

<State name="normal" offset="8 9 113 27"/>

</BasisSkin>

**<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">**

**<State name="normal" colour="#FFFFFF"/>**

**</BasisSkin>**

</Resource>

As we can see , in some other type of N , namely «SimpleText». This type can display text. We pointed offset by 3 pixels to the left and the top 105 and the size of 18 pixels . I remind you of the offset within the specified size of the skin . Alignment «align» set «Stretch» ​​as part of the text , together with skin stretched .

In describing the state of «State», besides the name , there is another attribute, «colour». This attribute is responsible for the color of the text . In Type «colour» there are several formats. Meaning «# FFFFFF» means white.  
  
Now modify the code to create a button in order to see the text :

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 113, 27),

MyGUI::Align::Default,

"Main");

button->setFontName("Default");

button->setTextAlign(MyGUI::Align::Center);

button->setCaption("MyButton");

After the launch button to appear with the text:



The stretch of the  
  
If now our button stretch horizontally, we see an undesirable effect, namely, the distortion of the button:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, **313**, 27),

MyGUI::Align::Default,

"Main");

button->setFontName("Default");

button->setTextAlign(MyGUI::Align::Center);

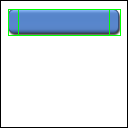
button->setCaption("MyButton");



To avoid this effect, we want to divide the skin apart and show them the necessary alignment. If our button will stretch horizontally, then we need to create 3 pieces for texture. Left-hand side will always be attached to the left border of the skin, the right side will be attached to the right border of the skin and the central part will stretch along with skin. For a more understandable description of the alignment, below is a table which shows the alignment values ​​for the skin consisting of 3 parts:

|  |  |  |
| --- | --- | --- |
| Left VStretch | HStretch VStretch (Stretch) | Right VStretch |

Our skin split into 3 parts:



Left-hand side turned width of 11 pixels, 12 pixels wide right and the central portion 90 pixels wide. Now add the 3 parts in the skin, instead of the previous one.

<Resource type="ResourceSkin" name="MyButton" size="113 27" texture="MyButton.png">

<BasisSkin type="SubSkin" offset="0 0 11 27" align="Left VStretch">

<State name="normal" offset="8 9 11 27"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 0 90 27" align="Stretch">

<State name="normal" offset="19 9 90 27"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 0 12 27" align="Right VStretch">

<State name="normal" offset="109 9 12 27"/>

</BasisSkin>

<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">

<State name="normal" colour="#FFFFFF"/>

</BasisSkin>

</Resource>

The first part is aligned with the left edge «Left VStretch» ​​and has a width of 11 pixels and starts with 0 as the offset is relative to the portion of the skin . Status of the 11 has the same width of the pixels but begins with 8, as the offset is relative to the state of the structure of .

The second part is the central and stretched along with skin , so she alignment specified «Stretch». Width of the pixel is 90 and starts at 11 , just where the left side ends . Condition also has a width of 90 pixels but begins with 19 , ends where the left side of the texture.

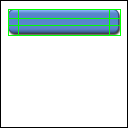
The third part is aligned to the right side of the skin «Right VStretch», has a width of 12 pixels and begins with 101 , where ends the central part (11 + 90). Condition also has a width of 12 pixels but starts with 109 as the offset is relative to the texture.  
  
Now when stretched horizontally our button is not distorted.



To button could be stretched in all directions, you need to have 9 parts. For a more understandable description of the alignment, below is a table which shows the alignment values ​​for the skin consisting of 9 parts:

|  |  |  |
| --- | --- | --- |
| Left Top | HStretch Top | Right Top |
| Left VStretch | HStretch VStretch (Stretch) | Right VStretch |
| Left Bottom | HStretch Bottom | Right Bottom |

We divide our skin into 9 parts:



The height of the first 3 parts of 10 pixels happened. The height of the lower part 3 was found to be 11 pixels. Height of the center was found to be 3 parts 6 pixels. You can re-describe 9 parts for the skin, and can be duplicated 3 times for our 3 part written in the previous skins and only correct values ​​and vertical alignment as horizontal values ​​have not changed. Here's what happened:

<Resource type="ResourceSkin" name="MyButton" size="113 27" texture="MyButton.png">

<BasisSkin type="SubSkin" offset="0 0 11 10" align="Left Top">

<State name="normal" offset="8 9 11 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 0 90 10" align="HStretch Top">

<State name="normal" offset="19 9 90 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 0 12 10" align="Right Top">

<State name="normal" offset="109 9 12 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="0 10 11 6" align="Left VStretch">

<State name="normal" offset="8 19 11 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 10 90 6" align="Stretch">

<State name="normal" offset="19 19 90 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 10 12 6" align="Right VStretch">

<State name="normal" offset="109 19 12 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="0 16 11 11" align="Left Bottom">

<State name="normal" offset="8 25 11 11"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 16 90 11" align="HStretch Bottom">

<State name="normal" offset="19 25 90 11"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 16 12 11" align="Right Bottom">

<State name="normal" offset="109 25 12 11"/>

</BasisSkin>

<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">

<State name="normal" colour="#FFFFFF"/>

</BasisSkin>

</Resource>

Now change the code to create a button so that it stretched vertically:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 313, **127**),

MyGUI::Align::Default,

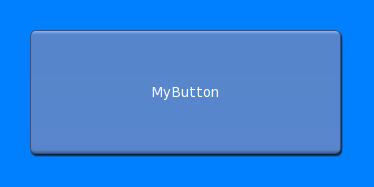
"Main");

button->setFontName("Default");

button->setTextAlign(MyGUI::Align::Center);

button->setCaption("MyButton");

And run the application. Here's a button you should see:

 Now our button can be stretched without distortion in all directions.  
  
  
skin condition  
  
Our button works, but when the mouse and clicking the mouse , it does not change its appearance . To ensure that it changed its view necessary to describe the state that uses a button:  
1. «Disabled» - button is disabled , the user does not have access to it.  
2 . «Normal» - normal state of the button .  
3 . «Highlighted» - button is highlighted , the user rolled the mouse button .  
4 . «Pushed» - button is pressed, the user clicked the mouse.  
  
Ask the artist to draw us in all of these key states 4 and arrange them in a single column for the convenience of description skin . Texture called «MyButton2.png» can immediately copy it to the resources folder MyGUI.



Change the name of the texture in the skin with «MyButton.png» on «MyButton2.png». The button should be painted gray, as now in the texture on this site is located state «disabled». To take first the very first part and duplicate it in status 4 times, changing their names. You should get the following description of the piece:

<BasisSkin type="SubSkin" offset="0 0 11 10" align="Left Top">

**<State name="disabled" offset="8 9 11 10"/>**

<State name="normal" offset="8 9 11 10"/>

**<State name="highlighted" offset="8 9 11 10"/>**

**<State name="pushed" offset="8 9 11 10"/>**

</BasisSkin>

Now we use a little trick. Distances buttons drawn in the texture with different states, the same for all parts, so we can just swap «Top» coordinate values ​​«offset» able parts. Distances from the state equal to the upper 29, 58 and 87 pixels. Now just let us add these values ​​to the coordinates «Top». For the first part of the description should have the following form:

<BasisSkin type="SubSkin" offset="0 0 11 10" align="Left Top">

<State name="disabled" offset="8 9 11 10"/>

<State name="normal" offset="8 **38** 11 10"/>

<State name="highlighted" offset="8 **67** 11 10"/>

<State name="pushed" offset="8 **96** 11 10"/>

</BasisSkin>

Now you can run the program and make sure that the left upper part correctly responds to the mouse as a pointing and clicking on:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 113, 27),

MyGUI::Align::Default,

"Main");

button->setFontName("Default");

button->setTextAlign(MyGUI::Align::Center);

button->setCaption("MyButton");







Now do the same manipulation with the rest of the skin, you should get the following option descriptions skin:

<Resource type="ResourceSkin" name="MyButton" size="113 27" texture="MyButton2.png">

<BasisSkin type="SubSkin" offset="0 0 11 10" align="Left Top">

<State name="disabled" offset="8 9 11 10"/>

<State name="normal" offset="8 **38** 11 10"/>

<State name="highlighted" offset="8 **67** 11 10"/>

<State name="pushed" offset="8 **96** 11 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 0 90 10" align="HStretch Top">

<State name="disabled" offset="19 9 90 10"/>

<State name="normal" offset="19 **38** 90 10"/>

<State name="highlighted" offset="19 **67** 90 10"/>

<State name="pushed" offset="19 **96** 90 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 0 12 10" align="Right Top">

<State name="disabled" offset="109 9 12 10"/>

<State name="normal" offset="109 **38** 12 10"/>

<State name="highlighted" offset="109 **67** 12 10"/>

<State name="pushed" offset="109 **96** 12 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="0 10 11 6" align="Left VStretch">

<State name="disabled" offset="8 19 11 6"/>

<State name="normal" offset="8 **48** 11 6"/>

<State name="highlighted" offset="8 **77** 11 6"/>

<State name="pushed" offset="8 **106** 11 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 10 90 6" align="Stretch">

<State name="disabled" offset="19 19 90 6"/>

<State name="normal" offset="19 **48** 90 6"/>

<State name="highlighted" offset="19 **77** 90 6"/>

<State name="pushed" offset="19 **106** 90 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 10 12 6" align="Right VStretch">

<State name="disabled" offset="109 19 12 6"/>

<State name="normal" offset="109 **48** 12 6"/>

<State name="highlighted" offset="109 **77** 12 6"/>

<State name="pushed" offset="109 **106** 12 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="0 16 11 11" align="Left Bottom">

<State name="disabled" offset="8 25 11 11"/>

<State name="normal" offset="8 **54** 11 11"/>

<State name="highlighted" offset="8 **83** 11 11"/>

<State name="pushed" offset="8 **113** 11 11"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 16 90 11" align="HStretch Bottom">

<State name="disabled" offset="19 25 90 11"/>

<State name="normal" offset="19 **54** 90 11"/>

<State name="highlighted" offset="19 **83** 90 11"/>

<State name="pushed" offset="19 **113** 90 11"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 16 12 11" align="Right Bottom">

<State name="disabled" offset="109 25 12 11"/>

<State name="normal" offset="109 **54** 12 11"/>

<State name="highlighted" offset="109 **83** 12 11"/>

<State name="pushed" offset="109 **113** 12 11"/>

</BasisSkin>

<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">

<State name="disabled" colour="#FFFFFF"/>

<State name="normal" colour="#FFFFFF"/>

<State name="highlighted" colour="#FFFFFF"/>

<State name="pushed" colour="#FFFFFF"/>

</BasisSkin>

</Resource>

You may notice that in each triplet parts values ​​«Top» coincide, since these parts are on the same horizontal line. This can be used to easily copy values ​​coordinates «Top». Now run the application and play with the button, you should get the following:







To see the status of «disabled» button to lock:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 113, 27),

MyGUI::Align::Default,

"Main");

button->setFontName("Default");

button->setTextAlign(MyGUI::Align::Center);

button->setCaption("MyButton");

**button->setEnabled(false);**

The button should look as follows:



Our button looks very attractive, but what else can we do to make it really alive? Let's change the color of the text depending on the state. Normally, the text color is not quite white. When you hover text color will be the most white. When you click the text color is gray. When blocking the text color is black. Here is an example of such a description of the text which is responsible for:

<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">

<State name="disabled" colour="**#000000**"/>

<State name="normal" colour="**#CCDDFF**"/>

<State name="highlighted" colour="**#FFFFFF**"/>

<State name="pushed" colour="**#606060**"/>

</BasisSkin>

And, accordingly, will look like this button in different states:









It is also possible to control the shift text down, it may need pressed for better visual display. Manage shift occurs with the attribute «shift». The value is either «true» or «false». Here is an example description of this shift:

<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">

<State name="disabled" colour="#000000" **shift="false"**/>

<State name="normal" colour="#CCDDFF" **shift="false"**/>

<State name="highlighted" colour="#FFFFFF" **shift="false"**/>

<State name="pushed" colour="#606060" **shift="true"**/>

</BasisSkin>

And the result when you:



Skin properties (Property)  
  
Now let's take a closer look at the code to create the button:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 113, 27),

MyGUI::Align::Default,

"Main");

**button->setFontName("Default");**

**button->setTextAlign(MyGUI::Align::Center);**

button->setCaption("MyButton");

In bold methods to specify the font and text alignment. Are these properties need to write each time creates a button? The answer - no! In the skins have the ability to specify some properties that will be used with this widget created skinned.  
Let's add these properties to the skin:

<Property key="FontName" value="Default"/>

<Property key="TextAlign" value="Center"/>

The property described in the node called «Property». The property name is described in the attribute «key» and the value of the property described in the attribute «value». Full description of the button:

<Resource type="ResourceSkin" name="MyButton" size="113 27" texture="MyButton2.png">

**<Property key="FontName" value="Default"/>**

**<Property key="TextAlign" value="Center"/>**

<BasisSkin type="SubSkin" offset="0 0 11 10" align="Left Top">

<State name="disabled" offset="8 9 11 10"/>

<State name="normal" offset="8 38 11 10"/>

<State name="highlighted" offset="8 67 11 10"/>

<State name="pushed" offset="8 96 11 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 0 90 10" align="HStretch Top">

<State name="disabled" offset="19 9 90 10"/>

<State name="normal" offset="19 38 90 10"/>

<State name="highlighted" offset="19 67 90 10"/>

<State name="pushed" offset="19 96 90 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 0 12 10" align="Right Top">

<State name="disabled" offset="109 9 12 10"/>

<State name="normal" offset="109 38 12 10"/>

<State name="highlighted" offset="109 67 12 10"/>

<State name="pushed" offset="109 96 12 10"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="0 10 11 6" align="Left VStretch">

<State name="disabled" offset="8 19 11 6"/>

<State name="normal" offset="8 48 11 6"/>

<State name="highlighted" offset="8 77 11 6"/>

<State name="pushed" offset="8 106 11 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 10 90 6" align="Stretch">

<State name="disabled" offset="19 19 90 6"/>

<State name="normal" offset="19 48 90 6"/>

<State name="highlighted" offset="19 77 90 6"/>

<State name="pushed" offset="19 106 90 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 10 12 6" align="Right VStretch">

<State name="disabled" offset="109 19 12 6"/>

<State name="normal" offset="109 48 12 6"/>

<State name="highlighted" offset="109 77 12 6"/>

<State name="pushed" offset="109 106 12 6"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="0 16 11 11" align="Left Bottom">

<State name="disabled" offset="8 25 11 11"/>

<State name="normal" offset="8 54 11 11"/>

<State name="highlighted" offset="8 83 11 11"/>

<State name="pushed" offset="8 113 11 11"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="11 16 90 11" align="HStretch Bottom">

<State name="disabled" offset="19 25 90 11"/>

<State name="normal" offset="19 54 90 11"/>

<State name="highlighted" offset="19 83 90 11"/>

<State name="pushed" offset="19 113 90 11"/>

</BasisSkin>

<BasisSkin type="SubSkin" offset="101 16 12 11" align="Right Bottom">

<State name="disabled" offset="109 25 12 11"/>

<State name="normal" offset="109 54 12 11"/>

<State name="highlighted" offset="109 83 12 11"/>

<State name="pushed" offset="109 113 12 11"/>

</BasisSkin>

<BasisSkin type="SimpleText" offset="3 3 105 18" align="Stretch">

<State name="disabled" colour="#000000" shift="false"/>

<State name="normal" colour="#CCDDFF" shift="false"/>

<State name="highlighted" colour="#FFFFFF" shift="false"/>

<State name="pushed" colour="#606060" shift="true"/>

</BasisSkin>

</Resource>

Now you can create a button as follows:

MyGUI::Gui::getInstance().load("MyButton.xml");

MyGUI::Button\* button =

MyGUI::Gui::getInstance().createWidget<MyGUI::Button>(

"MyButton",

MyGUI::IntCoord(30, 30, 113, 27),

MyGUI::Align::Default,

"Main");

button->setCaption("MyButton");

1. insert enlarged images which are sticks with dimensions derived from the dimensions are measured  
   2. Perhaps describe tiling