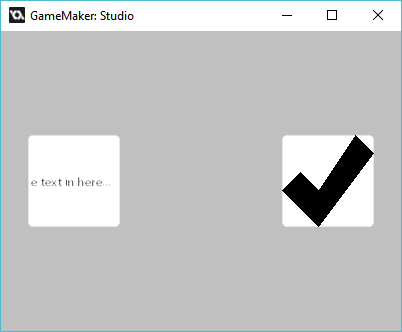
This tutorial is going to focus on a specific aspect of objects in uiz: Backgrounds. Backgrounds? No, not the type of backgrounds that fills your room like in the normal game maker, we are talking about “uiz object backgrounds”.

Sometimes you want your objects to look similar. Your button should look a little bit like your string input box, and that should look a little like the checkbox. In uiz, the way these objects are build is a bit the same for every object. Here is an example of an of a stringbox and a checkbox:

(see example 28)

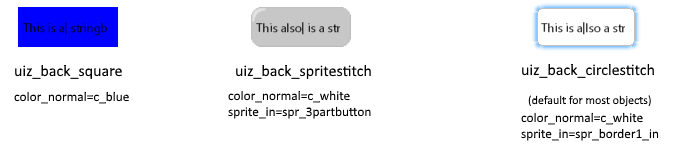
IMAGE 35:



You can see that they look somewhat the same. Although there is some text in the stringbox and a giant checkmark in the checkbox, the “backgrounds” are the same. If the text and the checkmark weren’t there, then the objects would look identical. This is what “background” means, and it has been made so you can easily customize one object, and then easily change that for other objects.

**So I do specify a background?** Well, easy. You just need to know which variables to edit for an object. To find out, open the manual and go to “Object background & stitching”. Here you can find all available background scripts. A background script can be just a normal game maker script that has some code in it that draws the background for the object. You can read about all the different backgroundscripts in the uiz manual. Here is an example of 3 different backgroundscripts:

IMAGE 36:



Now to set a different background for an object, you need to use the variable texturescript, which is most often uiz\_back\_circlestitch by default. Just set texturescript to another backgroundmode if you desire so. Like this:

EXAMPLE 29:

[CODE]

//initialize uiz

uiz\_init()

//create our stringbox object

stringbox=uiz\_c(obj\_uiZ\_stringbox)

//setup some variables

stringbox.posinframex=px;

stringbox.posvalx=25

stringbox.posinframey=uiz\_center;

stringbox.posvalwtype=dp;

stringbox.posvalhtype=dp;

stringbox.posvalw=1;

stringbox.posvalh=1;

//fix our stringbox object.

uiz\_fixgeneralpos(stringbox)

//create our checkbox object

checkbox=uiz\_c(obj\_uiZ\_checkbox)

//setup some variables

checkbox.posinframex=px;

checkbox.posvalx=375

checkbox.setpointx=uiz\_right;

checkbox.posinframey=uiz\_center;

checkbox.posvalwtype=dp;

checkbox.posvalhtype=dp;

checkbox.posvalw=1;

checkbox.posvalh=1;

//set out background:

checkbox.texturescript=uiz\_back\_square;

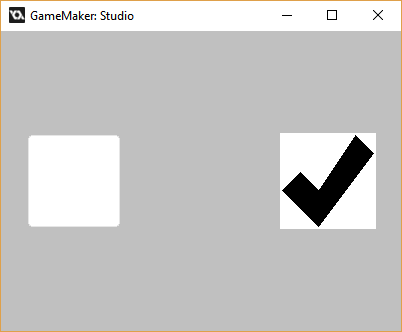
//fix our checkbox object.

uiz\_fixgeneralpos(checkbox)

[/CODE]

Which looks like:

IMAGE 37:



On the left you can see a stringbox with texturescript=uiz\_back\_circlestitch, and on the right a checkbox with texturescript=uiz\_back\_square.

Now, besides specifying a background, we can also customize it.

There are tree values every background has: a texturemode, a sprite, a color and a margin. Not all of these values have to be used for every backgroundscript. The uiz\_back\_square for example only uses only color. The variable names for the way a background should look is:

* sprite\_texturemode
* border\_texturemode
* sprite\_normal
* color\_normal
* margin\_normal
* sprite\_over
* color\_over
* margin\_over
* sprite\_in
* color\_in
* margin\_in
* sprite\_out
* color\_out
* margin\_out

What? That’s way more than the 4 mentioned before. Well, a background still takes 5 values (a texturemode, a bordertexturemode, a sprite, a color and a margin), but it switches which values it uses. It chooses this by the mouse. If the mouse is not over the object, than the background will use all the \_normal values. If the mouse hovers over the object, but doesn’t click it then the \_over values will be used. If the mouse hovers over the object, and clicks the object, then the \_in values are used. The \_out values can be ignored.

Here’s a table showing the what values are used:

|  |  |  |
| --- | --- | --- |
| Mouse not over object | Mouse hovers over object | Mouse clicks object |
| -texturescript  -sprite\_normal  -color\_normal  -margin\_normal  -sprite\_texturemode  -border\_texturemode | -texturescript  -sprite\_over  -color\_over  -margin\_over  -sprite\_texturemode  -border\_texturemode | -texturescript  -sprite\_in  -color\_in  -margin\_in  -sprite\_texturemode  -border\_texturemode |

This way if we hover the mouse over our object, our color can change for example. Then when we click again, our color can change again. Here is an example in which we change the color of the checkbox to blue when the mouse hovers over it, and to red when the mouse clicks it.

EXAMPLE 30:

[CODE]

//initialize uiz

uiz\_init()

//create our stringbox object

stringbox=uiz\_c(obj\_uiZ\_stringbox)

//setup some variables

stringbox.posinframex=px;

stringbox.posvalx=25

stringbox.posinframey=uiz\_center;

stringbox.posvalwtype=dp;

stringbox.posvalhtype=dp;

stringbox.posvalw=1;

stringbox.posvalh=1;

//fix our stringbox object.

uiz\_fixgeneralpos(stringbox)

//create our checkbox object

checkbox=uiz\_c(obj\_uiZ\_checkbox)

//setup some variables

checkbox.posinframex=px;

checkbox.posvalx=375

checkbox.setpointx=uiz\_right;

checkbox.posinframey=uiz\_center;

checkbox.posvalwtype=dp;

checkbox.posvalhtype=dp;

checkbox.posvalw=1;

checkbox.posvalh=1;

//set out background:

checkbox.texturescript=uiz\_back\_square;

checkbox.color\_normal=c\_white;

checkbox.color\_over=c\_blue;

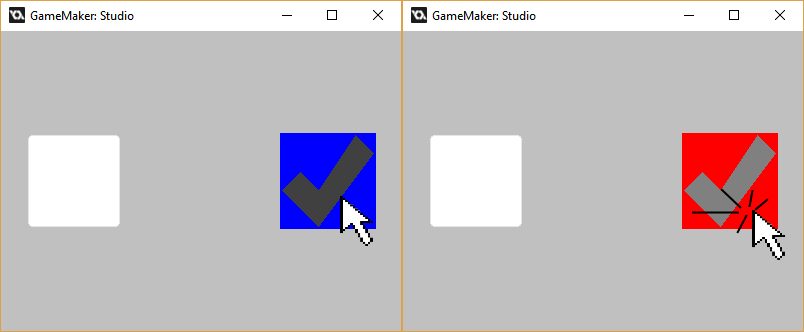
checkbox.color\_in=c\_red;

//fix our checkbox object.

uiz\_fixgeneralpos(checkbox)

[/CODE]

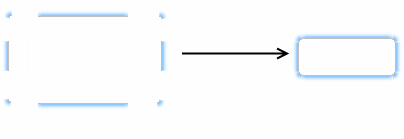
IMAGE 38:



The color part was the easy to understand part, but what about the others?

**Sprite and margins:** There is not one single way that every background scripts implements/handles margins and sprites. Some backgroundscripts will require a sprite with one image, while others would need 12 images or something. How these react to sprite and margins can be found in the manual in the page “Object background & stitching” Just note that some objects also use the margin variables inside their own code. For example, the stringbox will not draw any text outside it’s given margin.

**Spritestitching** is a term, and a way a backgroundscript can work. For example, a button might have seem like one single image, but it is in fact build up out of several sprites.  
IMAGE 39:

  
When these different sprites are being combined into one sprite that is called "stitching" in uiz.

**Texturemode:** Texturemode is another thing you need to understand. It influences the way sprites are stitched together. There are 3 texturemodes:

* uiz\_texturemode\_fill
* uiz\_texturemode\_tile
* uiz\_texturemode\_tilefit

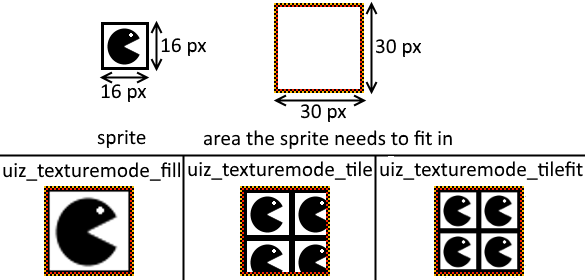
Let say we have a sprite of 16x16, and we want that sprite to fill an area of 30x30 pixels. We could stretch the sprite and “**fill**” out the area with one big sprite.

We could also choose to not stretch the sprite at all, meaning that for the width, we can fit one sprite in, and then still have 30-16=14 pixels left. We can just take 14 pixels in width of the sprite, and put it there. We are now only drawing this sprite partly. This approach is called “**tile**”.

Using uiz\_texturemode\_tile, we are cutting only 2 columns of pixels from the sprite. Two sprites almost fit, for this reason we can choose to cram 2 slightly resizes sprites into the space. Making each sprite 15 pixels wide, and thus filling the 30 pixels space without cutting any sprites off. This approach is called “**tilefit**”.

Here is a picture showing the different texturemodes:

IMAGE 40:



You might not need to fiddle around with all these values if you think the default background is ok, but considering you’re making a game, you might want to make a theme. So in that case, even which texturemode to use must be considered. Just set it for the object using “instance\_id.sprite\_texturemode=texturemode\_macro”.

**Whats the difference between sprite\_texturemode and border\_texturemode?** If you use circle stitch for example, your sprite exist out of two type of images, a lot of borders pieces, and one large center piece. You might want the border to be tilefit and the big middle to be fill.

**Values of -1.** A lot of times, when you don’t want to specify some values or just want to use default values, setting some variables to -1 will automatically handle it. This doesn’t work for every single variable, so it’s a good thing to read the manual article about backgrounds.

That concludes stuff for this tutorial. This tutorial didn’t get into every single variable into detail and had way less example codes, so I do hope you understand everything, and are able to make your own backgrounds for uiz objects. (making your own background scripts will be covered in another tutorial)