Windows! Finally. A pretty nice and usefull feature to use in uiz.

Creating windows exists out of multiple parts:

* Creating your window and defining it’s size.
* Setting which buttons you want (minimize, maximize, close)
* Customizing your window (backgrounds, colors)
* Getting the “mainframe” of a window
* Fixing your window
* Putting stuff in your window

Creating your window is fairly easy, you can use either **uiz\_window\_create(width,valuetype[px dp or fc],height,valuetype[px dp or fc])**, or use **uiz\_window\_create\_animation(width,valuetype[px dp or fc],height,valuetype[px dp or fc],xfunction,yfunction,alphafundtion,speed)**. But we’ll solely use the \_animation variant, since it looks way nicer.

Here comes customizing the window: First you want to set the 3 variables “button\_cross”, “button\_maximize” and “button\_minimize”. You might also want to set “resizable”

Note that you should edit the button\_ variables BEFORE you “fix” the window using uiz\_fixgeneralpos(). Else the variables will have no effect. If you want to set the button\_ values after you’ve fixed the window for the first time, then you will HAVE to call uiz\_window\_resetbuttons() about which you can read more in the manual. Just make sure you DO NOT call this script before fixing your window at least one time, else it will crash.

Further customizing your window can be done by changing the background. A window even has 2 backgrounds: One background covering the entire window, and another smaller one covering the top bar of the window. The big background can be accessed through the normal background variables on the window object. The small top bar background can be accessed by adding a “top\_” prefix and then use the normal variables on the window object. The top\_ background does not react to the mouse though.

To place objects inside the window, you'll first have to get a reference to the "**main window frame**". Normally, an windowholder object, returned by uiz\_window\_create, is the parent, of the top window bar, the edges and the cross, maximize, minimize buttons and the main window part. You can get the main frame, in which you will have to place your objects, by accesing the variable “**mainframe**” in your window object.

Now, with all knowledge combined, we can create a window like and put something in it like this:

EXAMPLE 40:

[CODE]

//initializing uiz

uiz\_init()

//creating window

window=uiz\_window\_create\_animation(1,dp,1,dp,uiz\_anticipate\_in,uiz\_anticipate\_out,uiz\_one,1)

//setting the buttons

window.button\_cross=false;

window.button\_maximize=true;

window.button\_minimize=true;

window.resizable=true;

//getting the mainframe

mainframe=window.mainframe;

//add object

grad=uiz\_c(obj\_uiZ\_gradientsquare)

//put it into the mainframe

uiz\_setparent(grad,mainframe)

grad.posinframex=uiz\_fill;

grad.posinframey=uiz\_snaptop;

grad.posvalhtype=fc;

grad.posvalh=0.5;

uiz\_fixgeneralpos(grad)

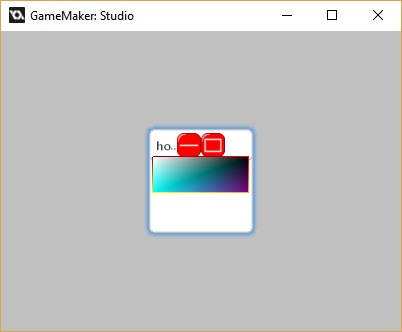
//fixing the window

uiz\_fixgeneralpos(window)

[/CODE]

Giving:

IMAGE 29:



Besides you being able to completely make your own windows, there are some standard “quick windows” to use.

Here is a list of them:

* Uiz\_popup\_ok
* Uiz\_popup\_yesno
* Uiz\_popup\_string

They are very easy to use, just look at an example:

EXAMPLE 41:

[CODE]

//init uiz

uiz\_init()

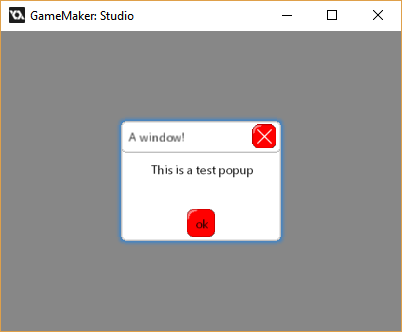
//create popup window

Window=uiz\_popup\_ok("This is a test popup","A window!",true)//we will need to store window for later usage

//no fixing or anything else is needed

[/CODE]

IMAGE 29:



Now the “blockbackground” argument on the window is kind of interesting. It allows you to make everything in the background unclickable, and forces the user to put all attention on your popup. Enabling this function will “gray out” everything behind the message. This argument is set to true in the example above.

**How to get if the ok button has been pressed?** Well, you have to check if the window still exists.

The yes/no window can be created in much the same style but how do I get the answer of this box? Well it’s a little big more complicated. First because game maker can only return a single value, the local objectvalues: “uizreturnyes”, “uizreturnno” and “uizreturnwindow” will be created after you run the script. Also why have a uizreturnwindow variable? Well, the function uiz\_popup\_yesno doesn’t always return a window instance id. Normally it should return a window instance id, but when a blockbackground is set to true, then it will return the instance id of a obj\_uiZ\_square which overlays the entire screen. (This is also true for the normal uiz\_popup\_ok, but it doesn’t really matter there)

Another way of getting the buttons is by accessing “yesbutton” and “nobutton” on whatever uiz\_popup\_yesno returns. This will work whether the script returns a window or a square.

Knowing all of this, the following code can be used to get the answer of a popup\_yesno, where r refers to a value previously gotten in example 42 :

[CODE]

if instance\_exists(r.yesbutton) and r.yesbutton.kmouseover=2 then{ //yes!

}else{

if instance\_exists(r.nobutton) and r.nobutton.kmouseover=2 then{ //no!

} }

[/CODE]

Don’t worry if you don’t understand every little bit of code in there but it is useful to know what kmouseover does, which you will learn in the next tutorial. This code though, does need to be in a step event constantly checked if you want to know the answer. The “//yes!” and “//no!” parts only fire for 1 step long.

Stringboxes also work about the same way, but the script for “getting” the value is:

[CODE]

if instance\_exists(r) and r.indestroy=1 then{ string=r.str;}

[/CODE]

I hope this tutorial was useful to you, next time we’ll cover “kmouseover” and I’ll teach you how to make scrollbars.