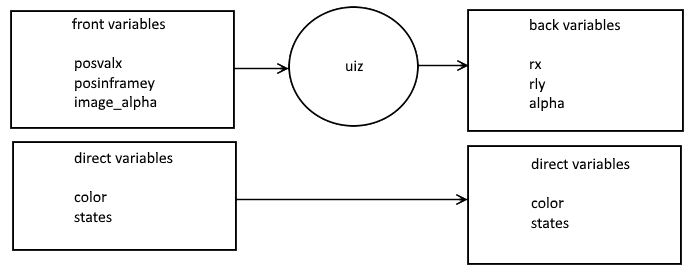
Hi, welcome back to the second tutorial in the series “Making your own uiz objects and scripts” Today, we’ll focus on variables.

There are practically 3 types of variables for uiz. Direct, front and back variables. Front variables are the ones the person making the ui uses. These are variables like “posinframex”, “posvalhtype”, “head\_alpha”, etc. The back variables is what you are going to use today. They are the variables that can be used inside objects, after they have been processed using uiz.

IMAGE 1.



Besides that, direct variables are variables that have direct influence over your object. They are variables that give color or states to your objects. They are most often not standard uiz variables, and object specific. The front variables need to be converted to back variables using fixing scripts like “uiz\_fixgeneralpos”.

Now, here are some back variables:

-rx: The left coordinate of the object

-ry: The top coordinate of the object

-rlx: The right coordinate of the object

-rly: The bottom coordinate of the object

-parent: What object is the parent. You probably don’t need to use this. But can be used to acces other variables in your ui tree.

-inframe: Whether your object is in is 0: outside the area in which it’s supposed to be. 1: Partly outside and partly inside the area in which it’s supposed to be. 2: Completely inside the area’s in which it’s supposed to be.

-alpha: Note that the alpha is automatically set within the uiz\_cntn() script.

-width: pixel value of the width

-height: pixel value of the height

-kmouseover: For this you need to have some code in your step event, as explained in last tutorial. This will be 0-mouse not over object. 1-mouse over object. 2-mouse over object and being pressed. 3-mouse over object and just pressed (only works for one tick). 4-mouse over object and just released (only works for one tick)

The one’s you’ll be needing the most are rx, ry, rlx, rly, width and height.

Let’s create an object that will draw a simple square. Look at last’s tutorial on how to setup things exactly but, make a create event like this:

[CODE]

///#define uiZ

//#sprite user

//#spritenum 0

//#objectversion 1.0

uiz\_objinit()

[/CODE]

Make a begin step event like this:

[CODE]

uiz\_beginstep()

[/CODE]

To your destroy, game end and room end events, this should be added:

[CODE]

uiz\_endobject()

[/CODE]

And a draw event like this:

[CODE]

If uiz\_cntn() then{

//put code here

uiz\_containend()

}

[/CODE]

Now we are going to edit the //put code here part.

Use the draw\_square function to draw a simple square where your object should be. For this, use the rx,ry,rlx,rly coordinates. Do it like this:

[CODE]

If uiz\_cntn() then{

//put code here

Draw\_square(rx,ry,rlx,rly,c\_white,alpha)

uiz\_containend()

}

[/CODE]

A few things though: We are using the draw\_square functions. This is because using the normal drawing functions will give problems when using shaders. Using the normal draw\_rectangle function will create artifacts in color, since the drawing color of an object is not properly passed to a shader. For any type of shape that has a single color and only changes it color using the draw\_set\_color functions, there will be artifacts. If you are getting artifacts and then drawing a primitive with color using draw\_primitive\_color might be a better option. If there are still problems contact me. However, since 1.4.1760 some of these problems have been fixed, and you’ll have to try and see if you get any problems. If you do get any problems, try upgrading your game maker. Refer to the manual for drawing functions.

Anyways create your object using some code like:

EXAMPLE 1:

[CODE]

//init

uiz\_init()

//create object

ownobject=uiz\_c(obj\_uiZ\_yourownobject)

ownobject.posinframex=uiz\_center;

ownobject.posinframey=uiz\_center;

ownobject.posvalwtype=fc;

ownobject.posvalhtype=fc;

ownobject.posvalw=0.5;

ownobject.posvalh=0.5;

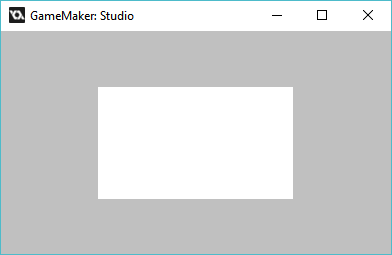
//fix

uiz\_fixgeneralpos(ownobject)

[/CODE]

Test our you object using by running your game and you should see:

IMAGE 2:



Easy! Now we can also add a direct variable called color that will give our square some color. If you look at last’s tutorial, we can take the list of different types of values we can use:

* v:value
* s:string
* b:Boolean
* c:color
* o:object
* p:sprite
* t:texturemode
* f:font
* z:sizetype
* a:animation
* i:texturescript

We are giving it a color, and thus we need to use the letter c. Defining our color should look like:

[CODE]

color=c\_white//c//color//

[/CODE]

Don’t forget to use the right amount of “//”, at the right places.

Now we edit our draw event to support our color like this:

[CODE]

If uiz\_cntn() then{

//put code here

Draw\_square(rx,ry,rlx,rly,color,alpha)

uiz\_containend()

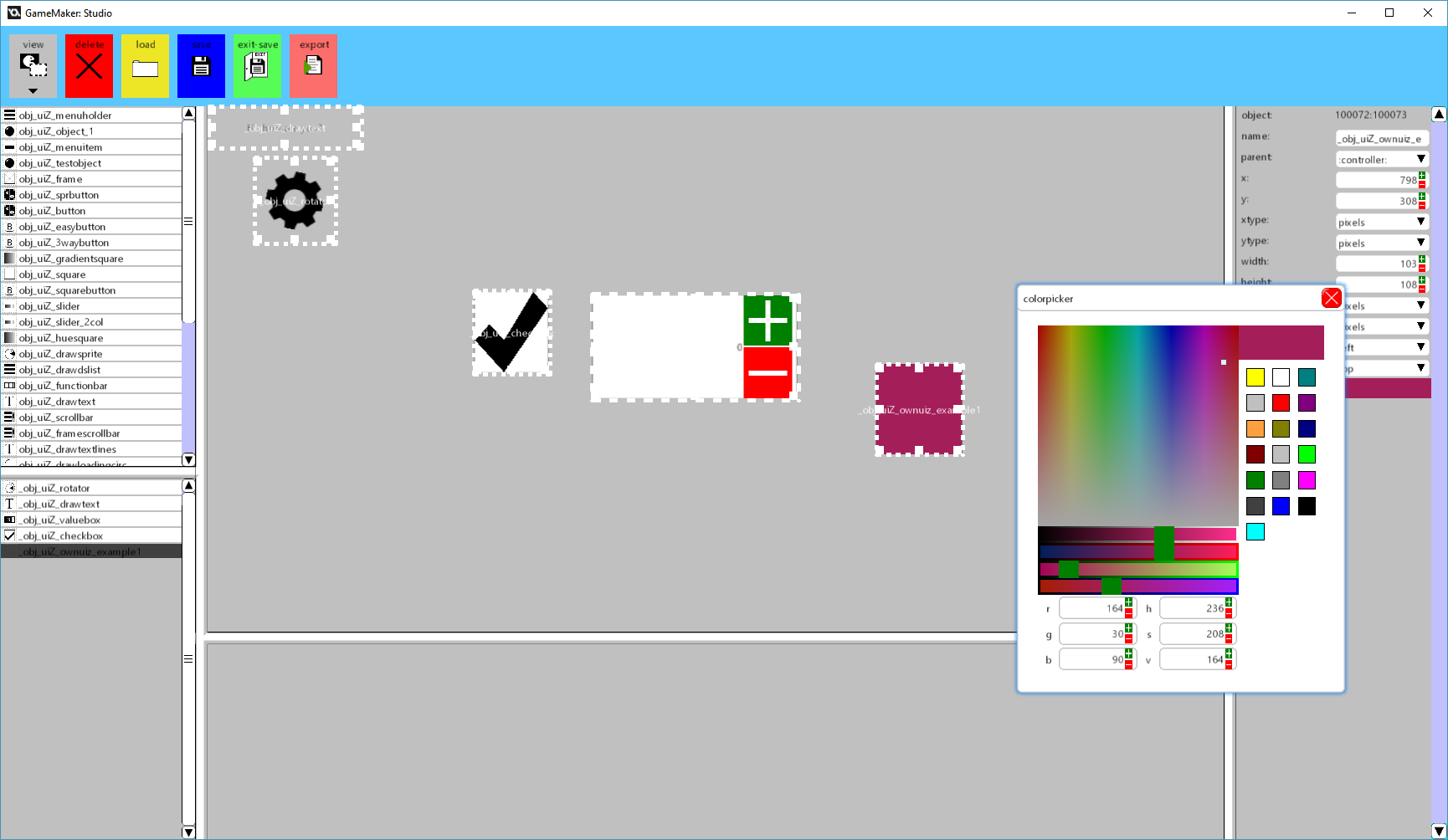
}

[/CODE]

And we are done. We can now set the color of our object using: “instance\_id.color=color\_we\_want\_to\_use”.

Now, let’s look at what all that trouble of defining values in comments was for. SAVE YOUR PROJECT and open up the designer. You should see it ask you to change your files. This is because we added a new object. Follow the process as explained in the tutorial “designer” and after you’ve done that we can use our object!

IMAGE 3:



As you can see, we can easily set a color for our new object.

Now that we know how variables work, we are going to look at how to implement some premade uiz scripts and stuff, mainly backgrounds.