This tutorial we’ll cover alpha & depth.

The first thing that you should know is that there are different types of alpha in uiz:

-There is alpha that will directly affect the object, and only the object itself.

-There is alpha that will affect the object, and all the children (and grandchildren) of the object.

The first one is quite straight forward. You can utilize this alpha by setting the variable image\_alpha. Image\_alpha is also a build in variable in game maker (you might have recognized it). Just like in game maker, alpha values need to be from 0 up to 1. A value of 0 means that the object is invisible, and a value of 1 (which is the default) means that the object is fully visible. In the example below we’ll make the gradientsquare object half visible, but leave the alpha of the square object. We can do this by setting image\_alpha to 0.5 for gradientsquare:

EXAMPLE 18:

[CODE]

//initialize uiz

uiz\_init()

//create our gradientsquare object

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

//our parent is the uiz controller object.

//setup some variables

gradient.posinframex=px;

gradient.posinframey=px;

gradient.posvalx=50;

gradient.posvaly=50;

gradient.posvalwtype=px;

gradient.posvalhtype=px;

gradient.posvalw=200;

gradient.posvalh=200;

gradient.image\_alpha=0.5;//set the alpha

//fix our square object.

uiz\_fixgeneralpos(gradient)

//create our square object

square=uiz\_c(obj\_uiZ\_square)

//set the parent

uiz\_setparent(square,gradient)

//setup some variables

square.posinframex=fc;

square.posinframey=fc;

square.posvalx=0.3;

square.posvaly=0.6;

square.posvalwtype=px;

square.posvalhtype=px;

square.posvalw=40;

square.posvalh=50;

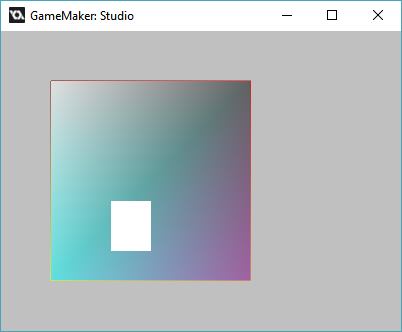
//fix our square object.

uiz\_fixgeneralpos(square)

[/CODE]

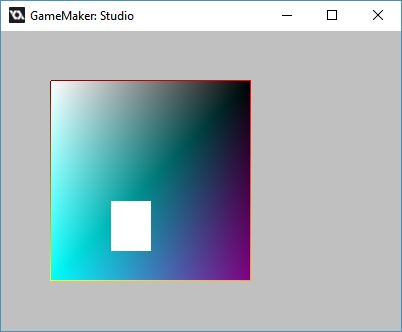
Giving:

IMAGE 23:



Compare that with an image\_alpha of 1:

IMAGE 11



Our second way of setting alpha is by using the “head\_alpha” value of an object. What this will do is set the alpha for that object, and then also apply that alpha to all the children (all the object that are below that object in the ui tree) We’ll set the head\_alpha of “gradientsquare” to 0.5:

EXAMPLE 18:

[CODE]

//initialize uiz

uiz\_init()

//create our gradientsquare object

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

//our parent is the uiz controller object.

//setup some variables

gradient.posinframex=px;

gradient.posinframey=px;

gradient.posvalx=50;

gradient.posvaly=50;

gradient.posvalwtype=px;

gradient.posvalhtype=px;

gradient.posvalw=200;

gradient.posvalh=200;

gradient.head\_alpha=0.5;//set the alpha

//fix our square object.

uiz\_fixgeneralpos(gradient)

//create our square object

square=uiz\_c(obj\_uiZ\_square)

//set the parent

uiz\_setparent(square,gradient)

//setup some variables

square.posinframex=fc;

square.posinframey=fc;

square.posvalx=0.3;

square.posvaly=0.6;

square.posvalwtype=px;

square.posvalhtype=px;

square.posvalw=40;

square.posvalh=50;

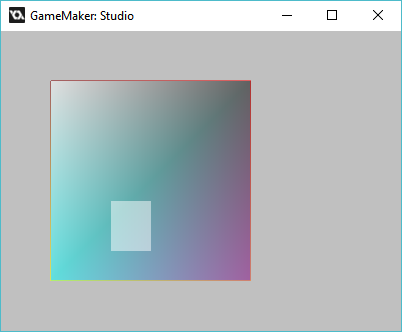
//fix our square object.

uiz\_fixgeneralpos(square)

[/CODE]

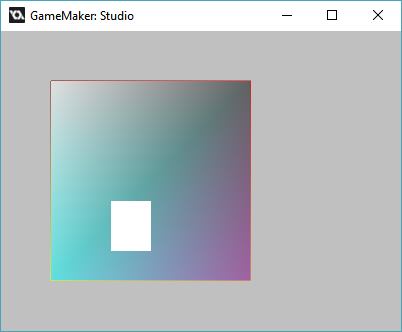
Which will give:

IMAGE 24:



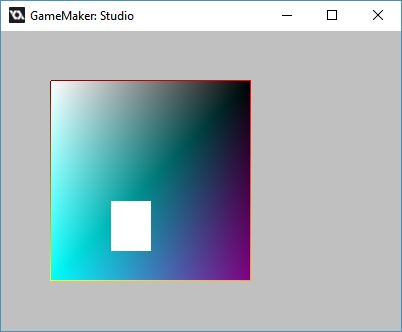
Compare that to the result we got in the last example (example 18):

IMAGE 23:



And compare it to the same objects with an alpha of 1:

IMAGE 8:



You can also make combination of these two values, when you do this the final alpha of your object is a combination of image\_alpha and head\_alpha. Practically image\_alpha is multiplied with head\_alpha. So if image\_alpha=0.5 and head\_alpha=0.5 then the alpha value being shown to the user will be an alpha value of 0.25. Here is an example:

EXAMPLE 20:

[CODE]

//initialize uiz

uiz\_init()

//create our gradientsquare object

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

//our parent is the uiz controller object.

//setup some variables

gradient.posinframex=px;

gradient.posinframey=px;

gradient.posvalx=50;

gradient.posvaly=50;

gradient.posvalwtype=px;

gradient.posvalhtype=px;

gradient.posvalw=200;

gradient.posvalh=200;

gradient.head\_alpha=0.5;//set the alpha

gradient.image\_alpha=0.5;//set the alpha

//fix our square object.

uiz\_fixgeneralpos(gradient)

//create our square object

square=uiz\_c(obj\_uiZ\_square)

//set the parent

uiz\_setparent(square,gradient)

//setup some variables

square.posinframex=fc;

square.posinframey=fc;

square.posvalx=0.3;

square.posvaly=0.6;

square.posvalwtype=px;

square.posvalhtype=px;

square.posvalw=40;

square.posvalh=50;

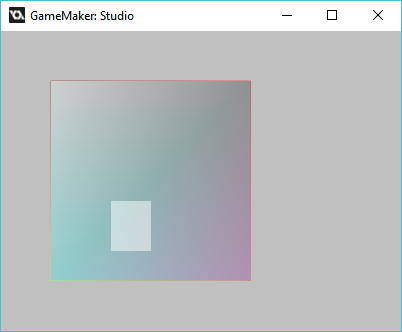
//fix our square object.

uiz\_fixgeneralpos(square)

[/CODE]

This will give the following:

IMAGE 25:



Compare to image 11,23 and 24.

**Depth** is also a concept in game maker itself. Uiz handles depth in much the same way. (it just uses some code around the depth code that is already there). Sometimes when you have two objects, you might need them to overlap, but you want a specific object to be drawn over all other objects. For example, a window needs to be drawn over/above all other non-window objects. We would way that the depth isn’t very “deep”. Objects that are deeper in the background will have a higher “depth”. When setting an objects depth in uiz, that depth will also apply to all children and grandchildren (anything below that object in the ui tree).

**How do we set depth?** You should NOT change the “depth” variable. This is handled by uiz code. What you should do is use the variable “adddepth” to do this (yes those are 3 “d”’s). The addepth variable is a inversed from the normal game maker value. Giving adddepth a high value (or a positive) will mean that the object will be more on the foreground. Givin adddepth a low value (or a negative) will make the object go to the background.

We’ll just make 2 object purposefully overlap each other like this:

EXAMPLE 21:

[CODE]

///Example 21:

//initialize uiz

uiz\_init()

//create our gradientsquare object

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

//our parent is the uiz controller object.

//setup some variables

gradient.posinframex=px;

gradient.posinframey=px;

gradient.posvalx=50;

gradient.posvaly=50;

gradient.posvalwtype=px;

gradient.posvalhtype=px;

gradient.posvalw=200;

gradient.posvalh=200;

//fix our square object.

uiz\_fixgeneralpos(gradient)

//create our square object

square=uiz\_c(obj\_uiZ\_square)

//set the parent

//our parent is the uiz controller object.

//setup some variables

square.posinframex=px;

square.posinframey=px;

square.posvalx=100;

square.posvaly=150;

square.posvalwtype=px;

square.posvalhtype=px;

square.posvalw=200;

square.posvalh=120;

//fix our square object.

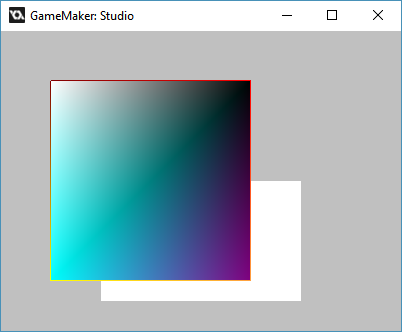
uiz\_fixgeneralpos(square)

[/CODE]

Giving:

IMAGE 26:

.



Now we want the white square to be in front of the gradientsquare. Let set the square adddepth to +5:

EXAMPLE 22:

[CODE]

//initialize uiz

uiz\_init()

//create our gradientsquare object

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

//our parent is the uiz controller object.

//setup some variables

gradient.posinframex=px;

gradient.posinframey=px;

gradient.posvalx=50;

gradient.posvaly=50;

gradient.posvalwtype=px;

gradient.posvalhtype=px;

gradient.posvalw=200;

gradient.posvalh=200;

//fix our square object.

uiz\_fixgeneralpos(gradient)

//create our square object

square=uiz\_c(obj\_uiZ\_square)

//set the parent

//our parent is the uiz controller object.

//setup some variables

square.posinframex=px;

square.posinframey=px;

square.posvalx=100;

square.posvaly=150;

square.posvalwtype=px;

square.posvalhtype=px;

square.posvalw=200;

square.posvalh=120;

square.adddepth=5;

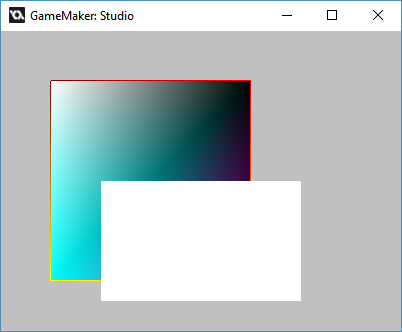
//fix our square object.

uiz\_fixgeneralpos(square)

[/CODE]

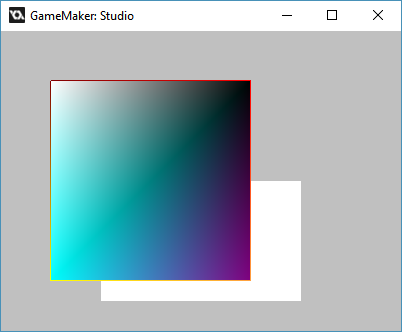
Which gives:

IMAGE 27:



Compare that to what we had before changing any depths:

IMAGE 26:



That was it for this tutorial! Hope you learned something again. Next time we’ll look more into how to customize objects themselves and how to use animations in uiz. Until then!