Tutorial 2:

Hi welcome back to the second beginners tutorial to uiz.

Today we’ll cover what “parents” are. It is quite important that you understand them properly since it is the way you will have to structure all of your ui.

Sometimes, we want to put one object into another. Why do we want to do this?

This might be because we have a window for example, with a few buttons inside of it. You want those buttons to be “connected” to the window.

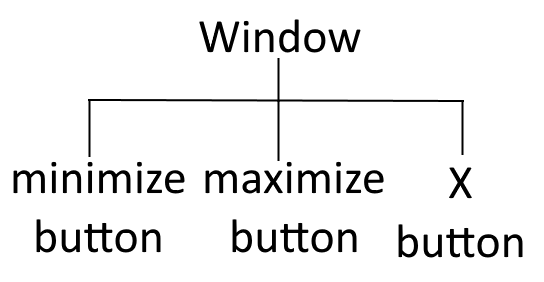
Another example might be is that you have some sort of square, in which you want to put some health bars, stamina bars, etc…. That square with the different bars could be snapped to the right side of your screen. When we change the size of our window, the square should move with it, but we also want our bars to move with it.

That’s where parenting comes in. It literally allows us to put one object inside another. The object which were putting in another object is called the “child” and the object we’re putting stuff in is called the “parent”. A child can also be another parent for another object.

What we ABSOLUTELY CANT do, is making a child parent of it’s own parent. This will cause a lot of issues. Bugs, loops, crashes, etc.

Parenting is like a “tree structure”, just like you would have your family tree. In the case of our window, we might have some text, a minimize and an X button. The tree structure would look like this:

IMAGE 5:



The things that parents do, is they dynamically influence their children’s data. For example, when the position of a parent moves 5 pixels to the left, the child will also move 5 pixels to the left. If the parent is put at 0.5 alpha, the child’s alpha will be halved. The parent’s size only has effect on it’s children in some cases, which we’ll cover later.

Another thing to note is that “positions” are added together. For example: a parent might be at position (100,150). The specified position of the child is (100,25). What will happen now is that the position of the parent stays (100,150) but the position of the child will add up and change to (200,175). If you would make the position of a child (0,0) it will always be placed inside the top-left corner of that parent.

It is very important that you understand how it works in order to properly use uiz.

Now let’s look at how we actually define our parents. Simple, we just call the uiz\_setparent function. It takes 2 arguments: The object we want to give/set a parent (our new “child”) and the object we want to put our object in (our new “parent”). So if we wanted put our “square” inside an “circle”, we would do this by calling uiz\_setparent(square,circle).

Now, remember the 4 important points? Creating, parenting,variables and fixing. Parenting is in here because it is gives the structure to your ui. You don’t always need to set a parent but ofter you do.

Quick technical detail here: If you didn’t set a parent for your object, then the default parent for your object is obj\_uiZ\_controller. Which is an object from which only one exists and should exist. This object is created by using uiz\_init().

Example:

Here we put a “square” inside a “gradientsquare”:

[CODE]

///Example 6:

//create event of a newly created object.

//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 gradientsquare 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=px;

square.posinframey=px;

square.posvalx=25;

square.posvaly=25;

square.posvalwtype=px;

square.posvalhtype=px;

square.posvalw=40;

square.posvalh=50;

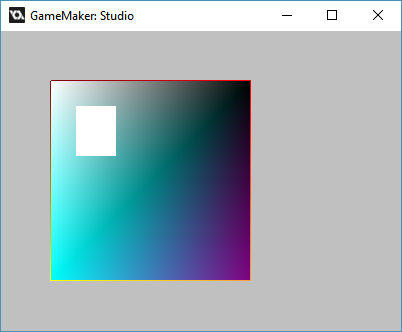
//fix our square object.

uiz\_fixgeneralpos(square)

[/CODE]

Which should look like:

IMAGE 6:



I hope this was somewhat clear. Later parenting will come back in advanced structures like grids, frameset etc…. Until next tutorial!