In the last few tutorials we’ve looked at how to customize objects themselves, now we’re going back to positioning and structuring. Putting objects in other objects is a way of making structures, but there are more ways to reach structure in your ui. These ways are called frame stuctures and they refer to when frames are being actively put in certain positions to create a neat ui. We can create grids for example. Making a grid is like making a table, one like this:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Here, every single cell in this table would represent a frame in uiz. (for frames, refer to tutorial 5)

We’re going to learn how to create such a grid(table) structure in uiz, then how to put things in the frames(cells) of it.

**Positioning complete structures.** There are some problems with setting custom positions and sizes for structures. Putting the structure inside a frame and then transforming and scaling that frame will work just fine. Just don’t touch any of the position variables for a structure(like posinframex/y, posvalw/htype, etc). By default posinframex/y is set to uiz\_fill.

**Creating a grid.** Creating a grid can be easily done using only 3 lines of code.

**Line 1: uiz\_grid\_create(gridw,gridh) :** In this function we create our grid and specify the number of frames it should have. Saying the grid would be 5x5 would mean that you would create 25 frames. This function also returns the instance id for your grid, which you will need.

**Line 2: uiz\_setgridframes(grid):** This is an important script. Not using this script will CRASH your game. It is an extra grid initialization script. Needs to be directly after using uiz\_grid\_create(). The grid argument should be the instance id of your grid.

**Line 3: uiz\_fixgeneralpos(instance id):** Our good old fixing script. You’ll need this else other weird things will happen.

Creating a grid can be done using this:

EXAMPLE 30

[CODE]

//initialize uiz

uiz\_init()

//create our grid

grid=uiz\_grid\_create(5,5)

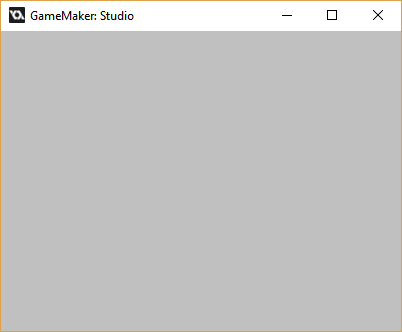
uiz\_setgridframes(grid)

uiz\_fixgeneralpos(grid)

[/CODE]

This will give the following result:

IMAGE 42:



Nothing? What is this? Well, remember that frames are invisible. The frames are there, there’s just nothing in them.

**Debugging frame stuctures:** If you want to test out your ui, setting mark=true on every frame is not a bad idea. What do I mean by this? Try it yourself. Search for obj\_uiZ\_frame inside your game maker resource list, open that object up and open the script in it’s create event. There should be a line like this:

[CODE]  
mark=false//b//mark

[/CODE]

Change the line to:

[CODE]  
mark=true//b//mark

[/CODE]

Now see what happends when we run:

IMAGE 41:



Now, we can see what our grid really looks like. Before we’ll dive in how to put objects in our grid, we’ll focus on changing the grid itself first. Every single cell in the grid has the same size, lets change that. You should know that we can’t specify the size of every frame. When we want to specify sizes, we’d have to change the size of that entire row or column. You’ll see that in a minute.

**Changing up row/col sizes.** The function uiz\_gridsize\_row(grid,col,size,sizetype{px.dp.xtra.fc}) and uiz\_gridsize\_col(grid, col,size,sizetype{px.dp.xtra.fc}) should be used for this. For this we need to specify a row/col inside our grid. See the example below:

EXAMPLE 3:

[CODE]

//initialize uiz

uiz\_init()

//create our grid

grid=uiz\_grid\_create(5,5)

uiz\_setgridframes(grid)

uiz\_gridsize\_col(grid,2,100,px)

uiz\_fixgeneralpos(grid)

[/CODE]

Showing:

IMAGE 43



Now the middle row is resized!

A few things to note:

* Uiz\_gridsize\_col comes after uiz\_setgridframes().
* If you look at the example you might notice we filled in 2, at the place in which we needed to specify the colomn which we use. Though in the picture we can see that the third frame has changed size. This is because the row/col arguments counts from 0 to the size you specified in uiz\_grid\_create minus 1. So in our case, 0 would refer to the most left colomn, while 4 would refer to the right most colomn. Giving it a col value of 5 will crash your game.
* Inside our grid we can use px values, dp values, fc values and xtra values. There are no other value types compatible with grids or other frame stuctures.

**The xtra data type.** Let say we have a grid of 3 rows where the first row is 50 pixels, the second row 0.33 fc, and the third is 1 xtra. What will happen is that the first and second row are set to their right size. Now lets say the grid’s parent is 300 pixels tall. The first row will be 50 pixels, the second row will be 100 pixels (300\*0.33=100) which leaves 250 pixels over. This is where the 1 xtra part comes in handy. The 1 xtra value will take in all space that is left for it. That is what the xtra data type does: It uses any space that is left. The above example would look like:

EXAMPLE 33:

[CODE]

//initialize uiz

uiz\_init()

//create our grid

grid=uiz\_grid\_create(5,3)

uiz\_setgridframes(grid)

//set our grids sizes

uiz\_gridsize\_row(grid,0,50,px)

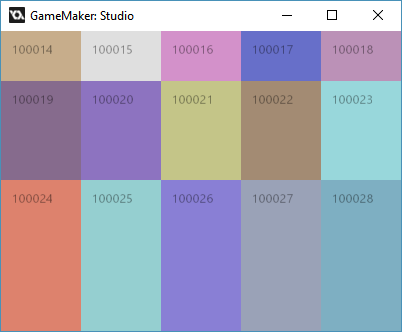
uiz\_gridsize\_row(grid,1,0.33,fc)

uiz\_gridsize\_row(grid,2,1,xtra)

uiz\_fixgeneralpos(grid)

[/CODE]

IMAGE 44:



What is neat about the xtra data type is that you can have two different rows being that value:

EXAMPLE 34:

[CODE]

//initialize uiz

uiz\_init()

//create our grid

grid=uiz\_grid\_create(5,4)

uiz\_setgridframes(grid)

//set our grids sizes

uiz\_gridsize\_row(grid,0,1,xtra)

uiz\_gridsize\_row(grid,1,50,px)

uiz\_gridsize\_row(grid,2,0.33,fc)

uiz\_gridsize\_row(grid,3,1,xtra)

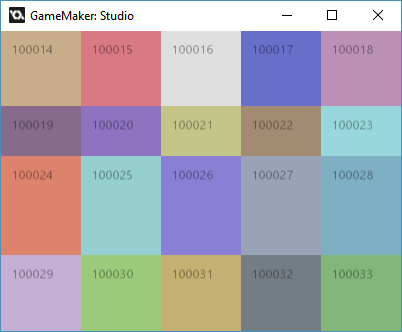
//fix the grid

uiz\_fixgeneralpos(grid)

[/CODE]

This will make:

IMAGE 45:



Both the upper frame and lower frame here are 1 xtra. If you compare image 45 to image 44 you can see that both these 1 xtra’s are half the size of the third row. (Or second if you start counting from 0). Also the first and last row’s of image 45 combined are the same height as the 1 xtra row on image 44.

Anther thing you can do with xtra is that you can still specify the size of something. Right now, we’ve only worked with values of 1 xtra, not 2 extra or 3 xtra. But you can do that. What will happen is that more space is allocated to the row that has 2 xtra, as opposed to the row which has 1 xtra, which is getting smaller. If you would have two 1 xtra’s, and make them both 2 xtra nothing will change. If you would make the top one 200 xtra and the bottom one 100 xtra, it would give the same effect as if the top one was 2 xtra and the bottom one 1 xtra.

*Technical detail:* The size in px of an xtra value is calculated by:

Space\_left\*(current\_xtra\_value/all\_xtra\_values\_added\_together).

Here is an example of varying xtras:

EXAMPLE 35:

[CODE]

//initialize uiz

uiz\_init()

//create our grid

grid=uiz\_grid\_create(5,4)

uiz\_setgridframes(grid)

//set our grids sizes

uiz\_gridsize\_row(grid,0,2,xtra)

uiz\_gridsize\_row(grid,1,50,px)

uiz\_gridsize\_row(grid,2,0.33,fc)

uiz\_gridsize\_row(grid,3,1,xtra)

//fix the grid

uiz\_fixgeneralpos(grid)

[/CODE]

IMAGE 46:



Compare that with image 45.

It’s always usefull for one value in your grid to be an xtra value, to make scaling better.

When working with scrollable frames or something, xtra values shouldn’t be used.

**Putting objects inside your grid**. This can be quite easily done. Remember that all these “squares” or “cells” are still frame objects, which means that we can get an instance id. We need an instance id if we want to do anything with the frames. You can get the instance id by using the function: uiz\_gridobject(grid,col,row). This function requires the grid’s instance id, and the column and row of what you want to set. Just know that the colomn and rows count from 0 so 0,0 is the top left frame. Here is an example:

EXAMPLE 36:

[CODE]

//initialize uiz

uiz\_init()

//create our grid

grid=uiz\_grid\_create(5,4)

uiz\_setgridframes(grid)

//set our grids sizes

uiz\_gridsize\_row(grid,0,2,xtra)

uiz\_gridsize\_row(grid,1,50,px)

uiz\_gridsize\_row(grid,2,0.33,fc)

uiz\_gridsize\_row(grid,3,1,xtra)

//fix the grid

uiz\_fixgeneralpos(grid)

//get a frame from the grid

frame=uiz\_gridobject(grid,2,2)//do not change anything about this frame

//add object to grid

rot=uiz\_c(obj\_uiZ\_rotator)

//put the rot inside a frame that we got from inside a grid

uiz\_setparent(rot,frame)

//make the objet fill the frame, and therefore fill the cell in the grid

rot.posinframex=uiz\_fill

rot.posinframey=uiz\_fill

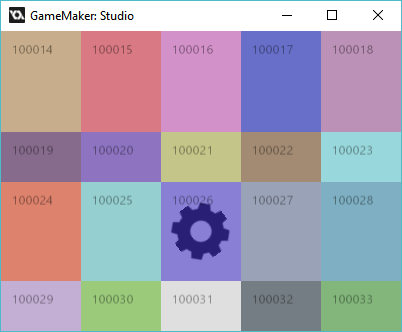
//fix the object

uiz\_fixgeneralpos(rot)

[/CODE]

Which looks like:

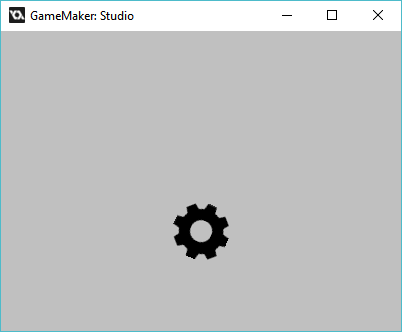
IMAGE 26:



Now we can remove the frames by setting mark to false again in the create event of the obj\_uiZ\_frame object. Just LEAVE the comment on the end, as it might seem like unimportant rubbish but it’s quite important. (you can learn about this in different tutorials).

Unmarking all the frames again will give you this:

IMAGE 27:



And voila you should now know how to use grids, congrats! But be aware, more structures are coming next tutorial.