In the last few tutorials we’ve looked at how to customize objects themselves, now we’re going back to positioning and structuring. We are going to be taking a look at advanced ui structures. This time: Grids.

**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 in the width and height.

**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(). Doesn’t need to be called twice.

**Line 3: uiz\_fixgeneralpos(instance id):** Our good old fixing script.

**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]

Don’t touch the comment, it might not seem important, but it is.

Now when with the debugging of frames on, look at the following code:

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 look like:

IMAGE 24:



Without the mark function on all the frames on, this all would be black.

**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.

A few things to note:

* Uiz\_gridsize\_col comes after uiz\_setgridframes().
* 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 column, 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 structures.

**The xtra data type:** The xtra data type will take any space that is left in the grid, and will use that up. Multiple rows/cols in the grid can have and sizetype of type “xtra”. You can change the value for size, and it will have effect. The hight the number of size, the more space it will get compared to other rows/cols who also have an xtra value.

*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 25:



**Putting objects inside your grid**. This can be quite easily done. You can get the instance id of a frame in a grid by using the function: uiz\_gridobject(grid,col,row). This function also counts the cols and row from 0.

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

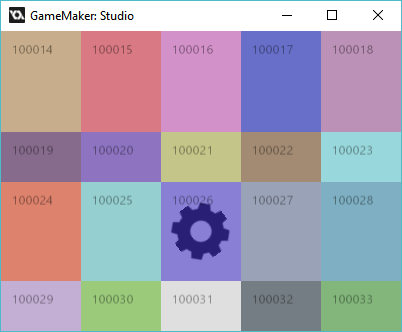
//fix the object

uiz\_fixgeneralpos(rot)

[/CODE]

Which looks like:

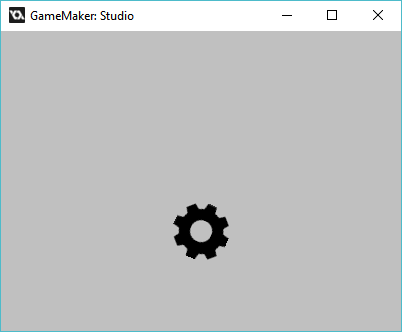
IMAGE 47:



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 48:



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