This tutorial is for the real pros who want the most out of their games. This tutorial focuses on how you can use uiz to the fullest extent. We will start with creating a new object.

There are a few script that need to run in certain events. We’ll start off with the easy ones:

* uiz\_beginstep() needs to run in the begin step event. This event ensures that the mouse is correctly checked and makes sure that uiz knows what object the mouse is over.
* Optionally you could choose to put “kmouseover=uiz\_mouse\_isonobject\_leftcheck(id)” in your normal step event, to get a variable that holds the current mouse data.
* Some scripts and a statement in the draw event. This ensures that your object cannot be drawn in places it isn’t supposed to be drawn
* A lot of stuff in your create event.
* The Destroy, Game End and Room End events should all have “uiz\_endobject()”. Else memory leaks will be created.

The begin step and normal step events should be easy to do for you, and we won’t be getting into those any further. The create event however is the trickiest of all.

Here is an example of the create event of “obj\_uiZ\_drawtext”:

[CODE]

///#define uiZ

//#sprite images

//#spritenum 9

//#objectversion 1.0

uiz\_objinit()

color=c\_black//c//text color//

text=""//s//text//

center=true//b//center//

font=-1//f//font//

[/CODE]

Why are all these comments necessary? Well, they are required for the uiz designer to work. The designer searches for certain strings in your objects. It’s somewhat sensitive and for the first few //# lines, now spaces can be added to the beginning, or other things to the end. These //# comments NEED to be placed on separate lines.

The first “///#define uiZ” HAS to be on the first line, this lets the uiz designer know that this is a uiz object.

The line “//#sprite images” shows the designer what sprite to use. This just means the little icon in the list, and the big icon when placing objects. YOU should NEVER set this line to “//#sprite images”, but instead to “//#sprite user” This way, the default uiz objects and your own object don’t get messed up in future updates. To add a sprite, edit “spr\_uiZ\_images\_user” inside a folder called designersprite, inside your sprites.

The line “//#spritenum X” is bound to //#sprite, as it defines what subimage to use from the sprite “spr\_uiZ\_image\_user”. Set x to the image index of your sprite. Set to 0 if you don’t feel like making a sprite.

“//#objectversion 1.0” Refers to what version of the object you are using. When you change this value, the uiz designer will notice it and will make a new “uiz shadow object” for that object. Generally the official uiz objects count in decimal values for bug fixes to the object, and whole numbers for new features. If you have made changes to your object and want to import those in the designer, you should change this number.

An additional option can be “//#exclude” Which won’t show up in the designer. If you set this, you can also choose to not include all comment markings normally needed.

After setting these values in the top lines of any script in the create event, the script “uiz\_objinit()” without any arguments needs to be called. This imports and sets up all important variables for uiz to work.

Setting variables is the trickiest part though. It needs to be done in the following fashion:

Variable\_name=Default\_value//one\_letter\_indicating\_what\_type\_of\_data\_this\_is//Some\_nice\_text\_for\_in\_the\_designer//

The “//” on this end is important. Also all variables need to be placed on a new line and variable declarations cannot be global or script specific and variable declarations can also not end with “;”. Not giving any comments after “Variable\_name=Default\_value” will not include your variable in the uiz designer. For types of data can be used:

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

These letters NEED to be SMALL, NON capital letters. Picking a letter not in this list, will result in the uiz designer acting weird, and your variable will not get included.

**Draw gui event:** The draw GUI event NEEDS the following code:

[CODE]

If uiz\_cntn() then{

//your own code here

uiz\_containend()

}

[/CODE]

This NEEDS to be done to ensure your object isn’t drawing outside your area. There are some problems though. Because of the way that uiz prevents objects getting outside their allowed areas, shaders and surfaces need to be used in a certain way. Also before you ask, yes that is an if-statement meaning that if your object is completely outside any areas in which it should be, it will not draw. Normally, if you have a lot of code in the draw event that needs to check certain things, that would be considered “bad programming practice”. If you do that here, things get even worse because you don’t exactly know when things are going to run or not. So try putting as much code as possible in the step event(s).

Anyways that wraps things up for this tutorial, you should know how to make your own object now, next time I’ll show you how to actually program them and what variables to use in that process.