Game Maker Tutorial 5: *1945 Part 3*

Scoring, lives, and damage

Finally, you will create a nice information panel that displays this data, along with the score.

1. Similarly to our invisible Controller Object ('controller_enemy') that we created to manage the time-release of enemy planes, we're going to create another Object that will draw and manage our game's information panel.

   First, you will need one big Sprite to serve as the art for the game's information panel. It has been provided with the other files for this tutorial and looks like this:

We will use this panel to show the Game Score, Damage State of the player's aircraft (in the black area at the left – try to envision it has a 'health meter'), plus the number of 'lives' (planes) the player has remaining.

first we need that panel defined as a Sprite in our game.

1 **Add, Add Sprite**, find this graphic and name it **spr_bottom**.

2 Deselect and **Precise collision checking** boxes, the former because we want to keep the color of its bottom-left pixel and the latter nothing collides with it and 'Precise collision checking' burn up a

3 Create an Object and name it **controller_life**. It *is* visible but do *not* assign a Sprite to it;

4 Draw Event to place this Sprite image on the screen instead. To make sure that this information panel resides above everything else being drawn, give this **controller_life** Object a **Depth** of '**–100000**.'

5 controller_life Object, add a **Draw Event**.
We want to draw that control panel (pictured above) as a Sprite image and place it correctly on the screen..
use the **draw** tab and right-click on the '**Draw a Sprite image**' icon to put it into the Actions box.

Drawing Ideas

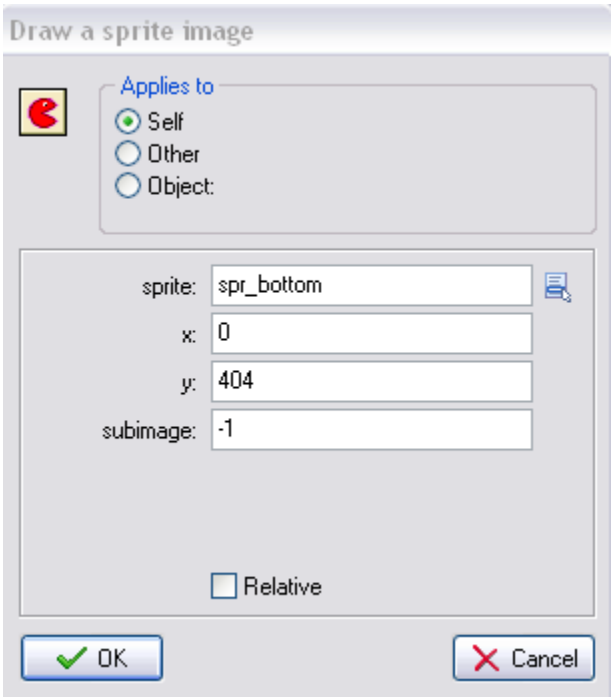
Normally in *Game Maker*, during each step an instance's Sprite is re-drawn at the correct location in the Room.

When you create a Draw Event and put Actions into it, however, this is no longer the case. Instead, these Draw Event Actions determine what is being drawn.

There is a whole collection of Actions available in *Game Maker* just for drawing. Most of them can be found in the **draw** tab.

Note that you can also use other Actions in a Draw Event, but drawing Actions only make sense within the context of a Draw Event. Within other types of Events, drawing-type Actions are basically ignored.

Now, the pop-up window that opens regarding this Action to draw a Sprite image is important, and filled out so it looks like this:



About "Subimage"

Using –1 for the value of the subimage means that the **current** subimage is drawn.

Since there is just one subimage in this Sprite, we don't really care about this feature, but if a Sprite consisted of multiple subimages, then you could indicate here which subimage you want to see.

.

This will place the correct Sprite at the bottom of the screen.

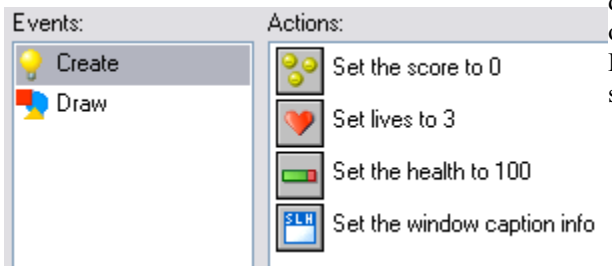
READ (x = how many pixels from the left edge of the screen we want this Sprite; which is **0** because we want it to line up with the left edge. y = how many pixels from the top of the screen we want it. Since this is a 640 x 480 size game screen, and this panel is 76 pixels tall at its top edge, it needs to be placed **404** pixels down from the top.)

Next, we need to add a **Create Event** with some specific Actions to establish a few housekeeping details.

a) The first Action of the controller_life's **Create Event** is set the game score to 0.
On the **score** tab, '**Set the score**,' new score = '**0**' (the default).

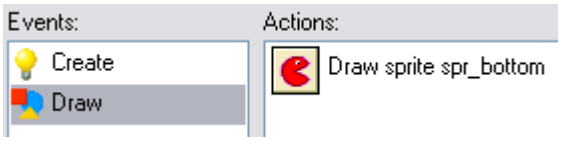
b) Second, set the number of lives to 3. Look for the heart icon
'**Set the number of lives**' and drag it into the Actions box;
set the new lives value in the pop-up window to '**3**'.

c) Third, set the health to 100.
Drag the '**Set the health**' icon into the Actions box
set the health value to '100.'



d) Finally, since we'll be showing the game's score, health, and lives on this information panel,
Drag the '**Set the window caption info**' icon over to the Actions
show score,' 'show lives,' and 'show health' = '**don't show**.'

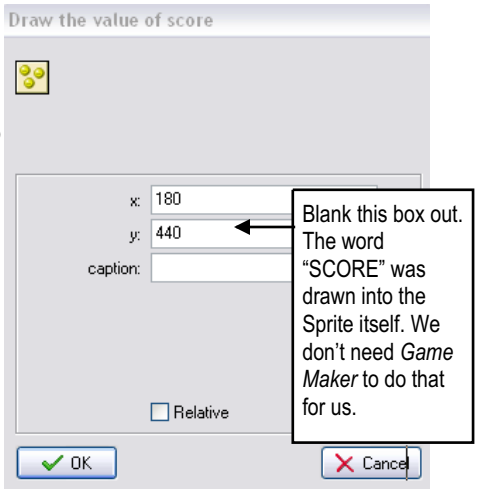
Now it's time to draw the score on the information panel.
TAKE NOTES ON HOW TO DO THIS!!!



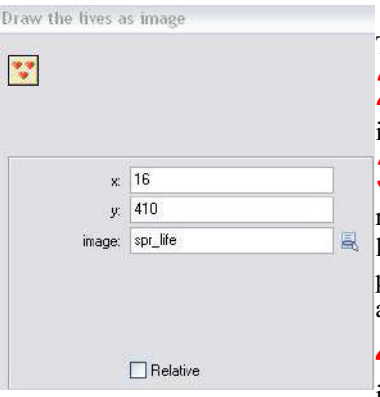
The steps:

1. Go back to the **Draw Event** that you've already created – the one where it's drawing that big Sprite at the bottom of the screen.

   The default color is black, and our background is dark, so we need to use a color that provides better contrast.



1 select tab **draw**, '**Set the color**' and choose yellow from the pop-up menu. *Now the drawing color for the score will be yellow.*



To draw the score itself on the information panel:

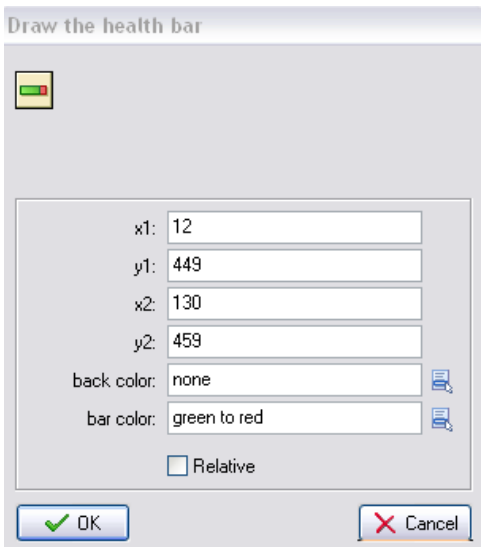
2 select tab **score**, '**Draw the value of score**' (the square-gold icon on the top-right), and in the

3 pop-up window set the values to: x = '**180**;' y = '**440**.' This means that *Game Maker* will draw the value of the score at a location on the screen that is 180 pixels from the left edge and 440 pixels from the top edge; exactly where we want it within the artwork of our information panel Sprite.

4 Change the default **caption** area to an empty box; the caption "SCORE" is drawn into the Sprite itself,

5 make a new Sprite called **spr_life** and select the image 'life' from the clip art folder

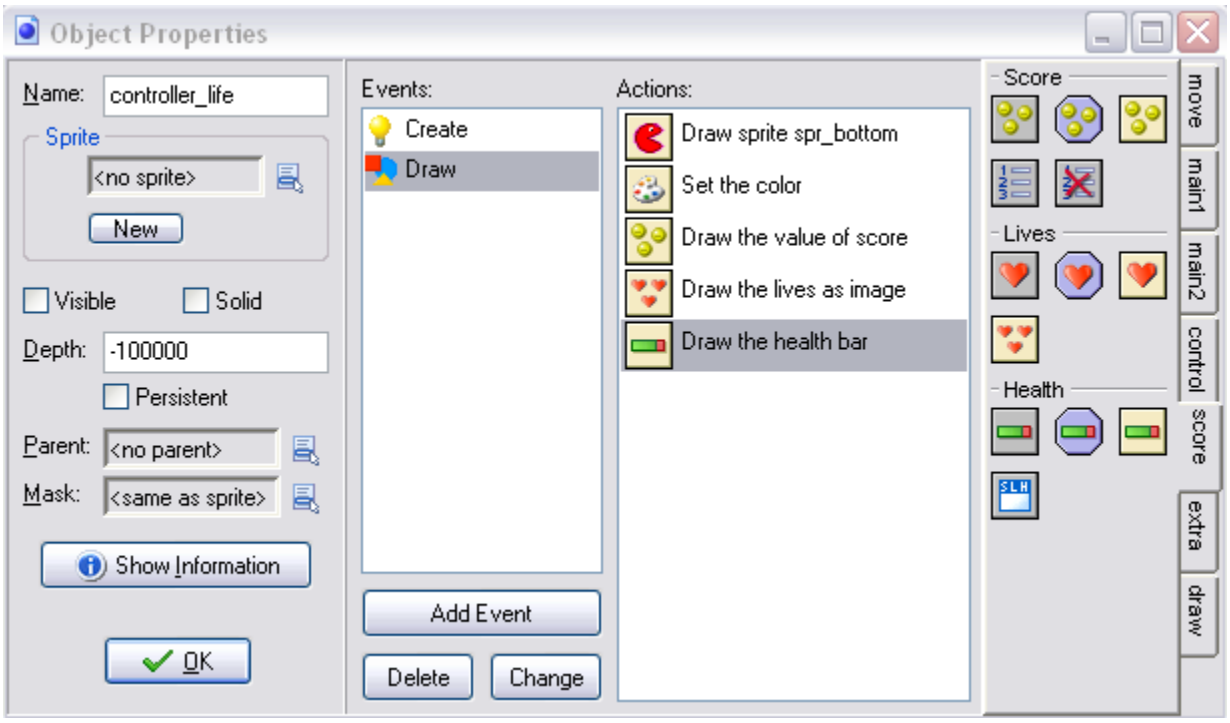
Now that we have a tiny Sprite image standing by, lets put one on the screen for every life the player has available. Back in the Object Property window for the controller_life Draw Event, on the **score** tab put the '**Draw the lives as image**' icon (with the three hearts) into the Actions box. In the pop-up window, set x = '**16**' (placing it near the left edge); y = '**410**;' and use the image **spr_life** that you just made.



All right, two down (score and number of lives) and one to go (the health meter). Not surprisingly, drawing the health meter is another similar type of action. Drag the '**Draw the health bar**' icon over to the Actions box. In order to have *Game Maker* drawn it in the proper space on the information panel Sprite, the pop-up window asks for the coordinates of the rectangle's four corners. They are: x1 = '**12**;' y1 = '**449**;' x2 = '**138**;' and y2 = '**459**.' You don't need a **back color**, and the **bar color** would look fine in the standard green-to-red

(although you can easily experiment by selecting a different bar color from the pull-down menu). This is shown in the illustration to the right.

That means the total **Draw Event** for controller_life thus far looks like this:



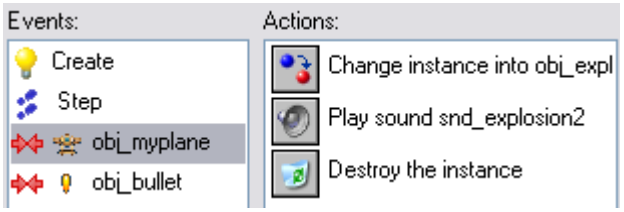
We're not out of the woods yet, however. We still need to actually *check* to find out what the health and lives are so that they can be drawn properly!

Programmer Logic

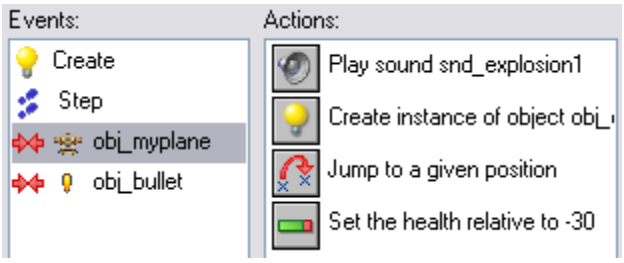
You're now seeing, block-by-block, the logic that programmers use to create games. Computers are very literal, and so programmers have to create very detailed, task-oriented instructions (by way of writing "code") telling the computer *exactly* what to do an in *exactly* what order. Using *Game Maker* is an easy way to explore the world of writing computer code for games.

2. But before we can do that, if we're going to have a heath meter for obj_myplane so that it can sustain more damage, we need to change the Collision Event with obj_enemy1 so that obj_myplane is not automatically destroyed.

Bef ore



Aft er



Open up the panel for **obj_enemey1**. Select the existing **Collision Event** with **obj_myplane** and let's clean up the Actions for that Event. Start by selecting the 'Change instance into obj_explosion2…' and **delete it**. Do the same for the '**Destroy the instance**' Action. *Those Actions are 'out.'*

Now, you've got the big, game-ending explosion happening, but you don't want that for just a single 'hit' to the main plane. So, open up the 'Play sound **snd_explosion2**' Action and change it to '**snd_explosion1**' (the smaller ka-boom).

When the main plane gets hit, you should still see an explosion, but not the big, game-ending one. So, go to the **main1** tab and '**Create an instance of an Object**,' and use Object: **obj_explosion1**. Leave the x and y coordinates at '**0**' and check the box next to the word '**Relative**' at the bottom. That is, now you'll see the explosion centered on wherever obj_enemy1 collides with the main plane.

Next, when this enemy airplane crashes into the player's plane, the enemy plane needs to regenerate. We use this Collision Event and

add that Action '**Jump to a given position**,' on '**Self**,' x = '**random(room_width)**,' y = '**–16**')

Just a Reminder

When that enemy plane regenerates and "Jumps to a given position," it is affecting its 'Self.' It will move to a random column (the 'x = random(room_width)' variable), and a row that is 16 pixels *above* the top of the screen ('y = –16'), which is just enough to place that entire Sprite off the top edge.

If you wanted to buy the player a little more time after a collision before that enemy fighter instance returns, set "y" to a much larger negative number (making the regenerated enemy plane 'fly' back down to the top of the screen over time) or plug in some extra random amount of pixels by making the y variable "random(-180)-65" or something like that.

Finally, this crash no longer destroys the player's plane, but now merely inflicts damage upon it. Let's have it do 30 points of damage (out of 100 per 'life').

Tab to **score**, '**Set the health**,' value = '**–30**,' and check the '**Relative**' box.
That means this Collision Event will trigger this Action that reduces the current health of the player's aircraft by 30 points.

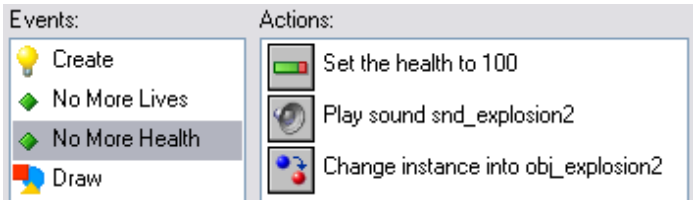
2.

Open the **controller_life** Ojbect. To check to see if the health has gone below 0, there is an Event in the "Other" category.
Press **Add Event,** and select **Other: No more health**. The Actions for this No More Health Event start on the **score** Tab;

select '**Set the health**' to '**100**.' *That gives the new plane 100 fresh hit points.*

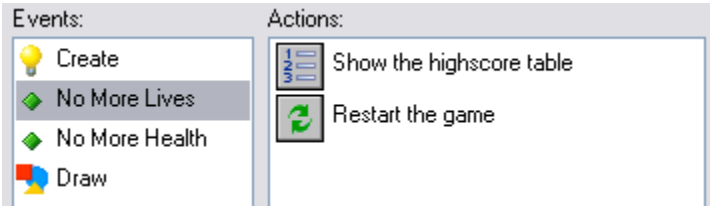
Next, play the big "ka-boom" sound effect. (Tab to **main1**, '**Play a sound**,' '**snd_explosion2**.')

We hear the big "ka-boom," and now we need to see it. That is, we need to change 'obj_myplane' into the big explosion Object. Do this by on the same tab page by selecting '**Change the instance**,' apply it to '**Object: obj_myplane**.' Change it into '**obj_explosion2**' and set "perform events" to '**not**.' *Beautiful...* It should look like this:



Of course, without damaging the main plane being a game-ending event anymore, we need to define how the game actually *ends*. While you're still in **controller_life**, add an **Other: No more lives** Action. The game-ending Actions that should occur when the number of lives reaches zero (or less) are: Tab **'score**,' '**Show the highscore table.**' In the pop-up, select background: '**back_water**;' border: '**show**;' new color: **red**; other color: **black**; and a nice serif font. *Feel free to experiment here.*

To finish this task, go to the '**main2**' tab and select '**Restart the game**' and you're all finished. The No More Lives Event will look like this:



Note that you don't have to create a new instance of the main plane Object (obj_myplane) because that happens automatically when the big "ka-boom" (obj_explosion2) plays its Animation End Event. If you double-click on **obj_explosion2,** you'll see that there is already an action in its Animation End Event that creates a new instance of obj_myplane, along with other necessary events, as shown below: