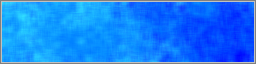
Game Maker—Creating a Menu  
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Part One--Basics

There are many ways to create simple menus in Game Maker. However, having a sprite for each object, and having separate objects for each button can take up space and increase the loading time.

A menu can be compacted to have only one object, and one sprite, which will be created in this example.

First, make a sprite that contains one button in the main menu. I will use this sprite:



Customize the transparency, precise collision checking, smooth edges, and preload texture to whatever you need (keep the bounding box on Automatic, however). If you want, you can change the origin. It doesn’t make much of a difference for now.

Now make an object, and name it obj\_menu\_choice. Assign the sprite to it, and put one in the room.

You now have a useless rectangle thing! The result can be seen in exhibit1.gm6.

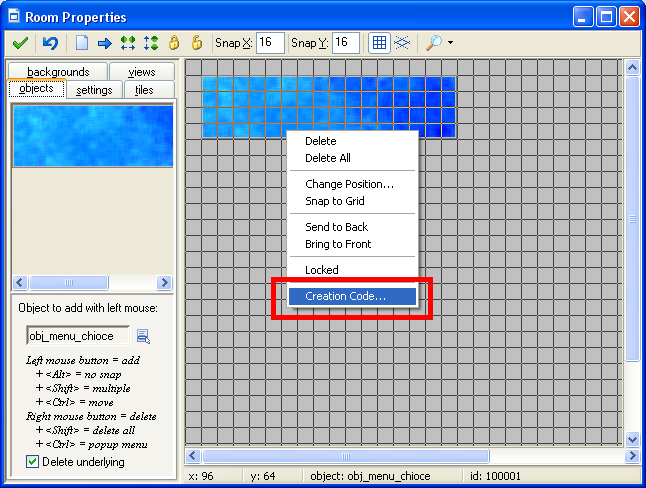
Part Two—Functions

The only thing separating one instance from another of the same object is the instance creation code. This will be used to give each button text and define what code to do when clicked. This is how it’ll be done in this tutorial…

In each instance creation code of the button, we specify two variables:

1. Text—tells the button what should be written on it (‘Play’, ‘Exit’, ‘Help’, etc.)
2. Code—tells the button what it should do when it is clicked on.

Let’s make the button that we have right now the ‘Play’ button. This button will bring the user to the room where they actually play the game. To get to the instance creation code of any instance, open the room properties of the room that contains the instance, hold Ctrl and right-click on the instance, and click “Creation Code…”



In the creation code of the button, type the following code:

text="Play";

code="room\_goto(rm\_play);”;

This just sets two variables for the instance. These will be used when creating the events and actions for obj\_menu\_choice.

Click the mark on the Instance Creation Code and room. Make a room called rm\_play (this is the “playing field” for your game). You can design your game in it later.

Open the Object Properties window for the object obj\_menu\_choice. When it Is clicked, then you want to execute the code provided previously (room\_goto(rm\_play)). In the Mouse Left Pressed Event, insert the code:

execute\_string(code);

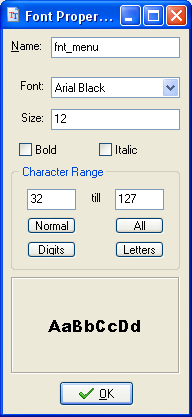
This actually executes the string in the variable code as if it was normal code instead of a string.

Now mark out of everything and play the game. The result is in exhibit2.gm6 (I also changed the background color of rm\_play so it doesn’t look like the button just disappears).

Part Three—Rollover Text

Most rollover effects use three sprites for when the button is inactive, when the mouse cursor hovers over it, and when the mouse cursor is clicking it. However, although this is perfectly possible in Game Maker, there is a trick that makes the button still have a decent rollover effect without using two more images—this is to change the text color that is written inside of the button instead of changing the actual sprite.

First we need three colors that will be used for each case. I will use c\_black, c\_aqua, and c\_white for each of the cases respectively. Then we need a font. Create a font and name it fnt\_menu. Specify whichever settings you want. I’m using these settings:



In the Draw event, we specify the basic font, alignment, and alpha values for the text:

draw\_sprite(sprite\_index,image\_index,x,y);  
draw\_set\_font(fnt\_menu);  
draw\_set\_alpha(1);  
draw\_set\_halign(fa\_center);  
draw\_set\_valign(fa\_center);

Note the draw\_sprite() function on the top. This is to prevent the sprite of the button from disappearing (anything in the Draw event prevents the sprite from being drawn).

So now we need a code that checks to see if the mouse is over it or clicking on it to change the color correctly. This will take some coding practice. Let’s make four variables that define the left, top, right, and bottom of the button.

To save memory, make them temporary variables. Temporary variables are deleted when the action is finished. The variables I will use are left, right, top, and bottom.

This is the code:

**var** left,right,top,bottom;  
 left=x-sprite\_xoffset;  
 right=x-sprite\_xoffset+sprite\_width;  
 top=y-sprite\_yoffset;  
 bottom=y-sprite\_yoffset+sprite\_height;

It’s not as murderous as it looks. sprite\_xoffset is actually the offset of the origin’s x position (or where the x position in the origin is with respect to the left side of the sprite). So x-sprite\_xoffset is the left side of the sprite. If you add the width of the sprite (sprite\_width) to the left side of the sprite, then you get the right side of the sprite. The same logic is used with the top and bottom variables. The **var** line is just defining the variables as temporary ones.

Now we need to detect if the mouse is over the button or not. We want to see if the cursor is between left, right, top, and bottom.

if(left<mouse\_x **and** mouse\_x<right **and** top<mouse\_y **and** mouse\_y<bottom)

{

*//code*

}  
else{draw\_set\_color(c\_black);}

This statement checks to see if the mouse position is hovering over the sprite, and if it is, then do some code. If not, then that means the mouse isn’t over the button, and we set the color to black.

Now we need to see if the mouse is clicking on the button or not.

if(left<mouse\_x **and** mouse\_x<right **and** top<mouse\_y **and** mouse\_y<bottom)

{

**if**(mouse\_check\_button(mb\_left))

**{**

draw\_set\_color(c\_white);

**}**

**else**

**{**

draw\_set\_color(c\_aqua);

**}**

}  
else{draw\_set\_color(c\_black);}

This means that if the left mouse button is down, then the menu button is being pressed, and to change the color to white. Otherwise, it means the mouse is hovering over it, so change the color to aqua.

Finally, we draw the text. The text alignment is centered, so we want to draw the text in the center of the button.

draw\_text(mean(left,right),mean(top,bottom),text);

This tells the program to draw the text in the middle of the sprite (the average of left and right sides, and the average of top and bottom sides).  out of the windows and play the game.

There is a problem with this, however. You never see the white text (the “clicked” color) because it immediately moves to the other room if you do click. Therefore, we have to change the  Left Pressed event to  Left Released. Go to the obj\_menu\_choice object and change it accordingly.

If done correctly, it now has a rollover effect! Yay!

The result is stored in exhibit3.gm6.

Part Four—Conclusion

To create more menu buttons, you can now simply place them in the room and define their text and code values. If you want, you can also add some cool background effects. I have created a semi-complete main menu as an example. It can be seen in exhibit4.gm6.

Thanks for reading, and happy programming!

**Tutorial Written by Snail Productions  
© Copyright Snail Productions**