Process Runner

by Timothy Sassone

General Information

Process Runner is a Game Maker 7 extension created to allow the easy creation and deletion of process. Processes are pieces of code which are executed every step. In effect, Process Runner allows you to add code to any objects step event. Actually, any event can be used, but whatever event you choose, you are stuck with using for all objects. This may be changed in a later version, but for the moment I would advise using the step or draw event.

Updates

v1.10 – Fixed a bug that was preventing any use of the extension due to a mis-spelling.
V1.20 – Fixed helpline for process_init function. Added support for local processes.
V1.30 – Fixed another fatal error. Added encryption to tighten security hole.

Functions

process_init();
Requires no functions. Must be called before any other process_ functions will work.

process_step();
Calls all processes code. Put this into the event in which you want all the code executed in, I advise the draw event so that it is executed as though in the step event, but allows the use of drawing function.

process_create(name, code);
Creates a new process with the given name and code. The code argument must be a string.

process_create_ext(name, code, object, initial state);
An extended version of process_create. Using this you can set what object the code is executed by (in process_create the code is executed by the object that created the process) and weather the process should start paused (0) or not (1).

process_delete(name);
Deletes the given process. Obviously, the process must exist for this to work.

process_start(name);
Starts a paused process.

process_stop(name);
Stops a running process. Stop processes won't run.

process_toggle(name);
Stops a running process and starts a paused one.

process_get_code(name);
Returns the code of the given process.

process_get_obj(name);
Returns the object of the given process.

process_get_state(name);
Returns the state of the given process.

process_local_init();
Requires no functions. Must be called before any other process_local_ functions will work.

process_local_step();
Calls all local processes code. Put this into the event in which you want all the code executed in, I advise the draw event so that it is executed as though in the step event, but allows the use of drawing function.

process_local_create(name, code);
Creates a new local process with the given name and code. The code argument must be a string.

process_local_create_ext(name, code, object, initial state);
An extended version of process_local_create. Using this you can set what object the code is executed by and weather the process should start paused (0) or not (1).

process_local_delete(name);
Deletes the given local process. Obviously, the local process must exist for this to work.

process_local_start(name);

Starts a paused local process.

process_local_stop(name);
Stops a running local process. Stop local processes won't run.

process_local_toggle(name);
Stops a running local process and starts a local paused one.

process_local_get_code(name);
Returns the code of the given local process.

process_local_get_obj(name);
Returns the object of the given local process.

process_local_get_state(name);
Returns the state of the given local process.

explode_string(array, sep, data);
Splits a string into an array by the given seperator. Credit to GMLscripts.com

implode_string(sep, array, size);
Puts a string back together after it's split by explode_string. Credit to GMLscripts.com

string_encrypt(string, key);
Encrypts *string* using the password *key*. Credit to xDanielx.

string_decrypt(string, key);
Decrypts *string* using the password *key*. Credit to xDanielx.

Note: The implode_string, explode_string, string_decrypt and string_encrypt are not mine. The scripts said to credit GMLscripts.com and xDanielx respectively, so I require that you do the same.

Tutorial

Open Game Maker, start a new game and install the Process Runner extension file. Create an object and call it obj_controller. Add a "Piece of Code" action in the Creation event. Enter the following code:

```
process_init();
process_create("DrawFPS", "draw_text(0,0,string(fps))")
```

The first line calls the process_init function, which must be called before any other

process_* functions are used. The second is only slightly more complex. It adds a process which draws the current frame rate in the corner of the screen. However, if you run the game now nothing will be drawn. For the processes to work you must have the process_step function somewhere. Add a draw event and add a piece of code that simply says:

```
process_step();
```

This causes all process codes to be executed. It can be put in any event, but remember, it will simulate executing the code in that event of the other objects. I prefer to have it in the draw event so you can use the draw functions.

This is all the necessary code to use the processes, however, it does not touch on the other functions. These other functions are not as simple, but allow much more power and flexibility. The first of the new function, and probably one of the most important, is process_create_ext. It resembles the other *_ext functions (built into Game Maker, such as draw_sprite_ext) in that it simply adds more functionality to the original process_create function. In this case, it allows you to set what object the code should be executed by, and the starting state for the process. The state of a process is simple, it is running, or it is stopped. When a process is stopped it's code is not executed when process_step is called. Add a new object called obj_moving and give it a sprite. Now, back in the obj_controller object, add this to the creation event code:

```
process_create("MoveRight", "x+=4", obj_moving, 1)
```

This will add a process which adds 4 to x, effectively moving the object four pixels right. It will be executed by obj_moving and starts on/running. (1=on, anything else=off) Add a similar process called MoveLeft that starts stopped. Now, if you test the game (after placing the obj_moving object, I assume you remembered to do that) the object will move right and never stop. In the step event of the obj_moving object add this code:

```
if x>room_width {
process_stop("MoveRight")
process_start("MoveLeft") }
```

This is quite simple. When the object hits the far right side of the room it stops the move right process and starts the move left process.

Now, this isn't all the commands, and is obviously not a good efficient use of the extension, however, I hope it has helped you to understand the basics.

PLEASE CREDIT TIMOTHY SASSONE AND GMLSCRIPTS.COM!!!