Awesome 2d RTS Engine:

How-To Guide

By Joseph (Rudeguss – Forum username)

Contents

1. Creating Units  
 1.1 Basics  
 1.2 Peons  
 1.3 Editing Stats  
 1.4 Attack Types

2. Creating Projectiles  
 2.1 Basics

3. Creating Spells  
 3.1 Basics  
 3.2 Types of Spells  
 3.3 Adding Spells to Units

4. Creating Resources  
 4.1 Basics  
 4.2 Resource Types

5. Creating Buildings  
 5.1 Basics  
 5.2 Adding Buildable Units

6. Creating Buttons  
 6.1 Basics  
 6.2 Spells  
 6.3 Units

7. Creating Enemy Units

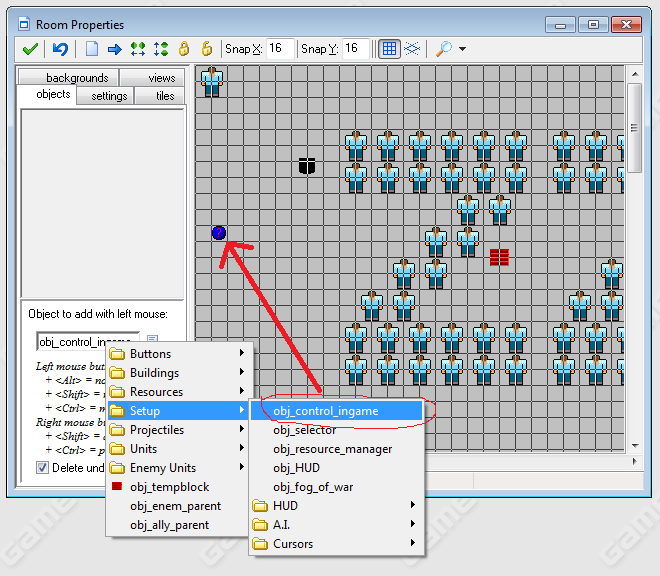
8. Important Global Variables

9. Multiplayer  
 9.1 Basics  
 9.2 Useful Variables  
 9.3 Units  
 9.4 Buildings  
 9.5 Maps  
 9.6 Spells  
 9.6 Other

Important Notes

IMPORTANT: As of newer versions (14.4+ or something) you should not be duplicating things from ‘Template’ folders, just the objects outside of them.

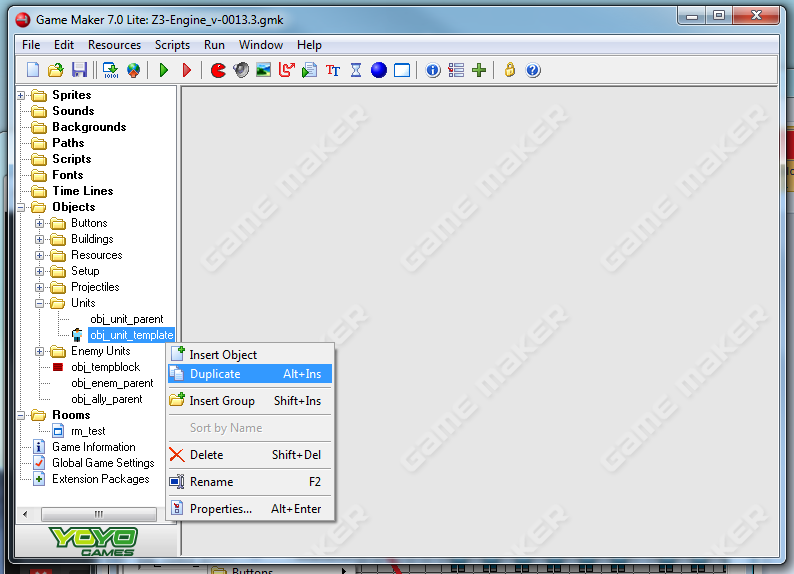
When creating the room for the games, there is one object that is ALWAYS required to be there; the obj\_control\_ingame object. Other than that, you can put whatever units or building in the room that you want. (Make sure that there is only one of these, and position of the object does not matter. Just make sure you can find it easily)



1. Creating Units

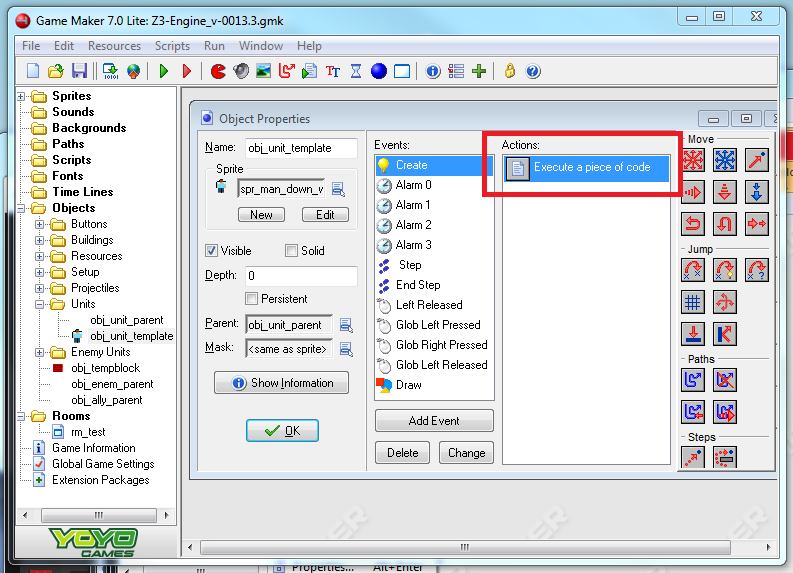
**1.1 Basics**

To create a new unit you just simply have to duplicate the obj\_unit\_template, which is found in the Units folder. Once this is done you can name the unit any name you desire.



**Changing the Name of you Unit, and assigning it your own Sprites**

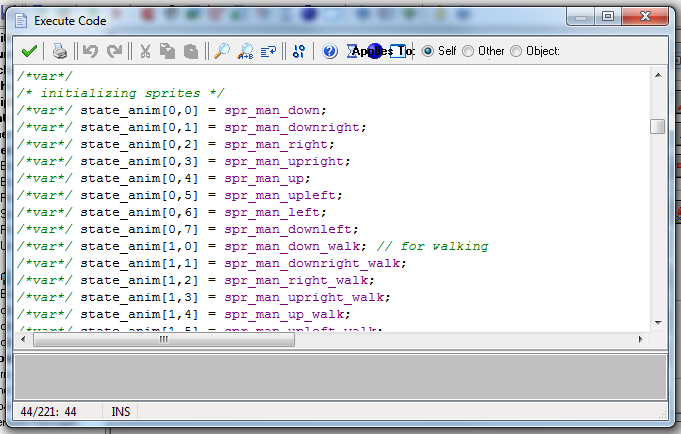
Now double click the Unit that you just made and click on the create event of the object. Everything that you will need to modify should be in this event. You shouldn’t need to touch anything else in this object, unless what you want to do is a little more complex.



To change the name of your unit you just need to edit the unit\_name variable



To change the sprites for your unit, you need to edit the state\_anim[#,#] variable, which is a two-dimensional array. This holds the long list of sprites that is used for all the different animations that the unit has (walking, attacking, gathering resources, constructing buildings and casting animation). The direction of the sprites is important – you should replace the sprites, with sprites of the same direction eg. When replacing my spr\_man\_down, make sure the sprite you’re replacing it with is also facing down.



**1.2 Peons**

Peons are the units that can gather resources and construct buildings. If you want your unit to be a Peon, find the ispeon variable and set it to true. Otherwise, you can set it to false, but it will not be able to gather resources or construct buildings. I should also note that the ability to construct buildings is NOT actually linked with the ispeon variable, but rather the iscratorer variable. The variable ispeon is for gathering resources and the iscratorer variable is for constructing buildings.

1.3 Editing Stats

There are several sections that hold the stats of your unit in the create event. These areas are: Setup, Stats, Peon Attributes, and Attacking. There are comments in these area describing the effects of the stats so that should help you.

1.4 Attack Types

There are 2 attack Types: close and ranged. Theirs is actually only one difference between these two; one produces a projectile object while the other one doesn’t. The attack\_range variable controls the attack range of the unit. You can actually set the unit to have an attack\_type of 0 (close range), if you want your unit to be ranged, but not create a projectile.

2. Creating Projectiles

2.1 Basics

To create a new projectile you just simply have to duplicate the obj\_projectiles\_template, which is found in the Projectiles folder. Once this is done you can name the projectile any name you desire. Projectiles are used for ranged attacks of your units but can also be used for magic projectiles, such as fireballs. Everything that you will need to modify should be in this event. You shouldn’t need to touch anything else in this object, unless what you want to do is a little more complex.

3. Spells

3.1 Basics

Spells are a little more complicated compared to units and projectiles. Most of the information of spells, like what they do, and the range, is actually all kept in a button object. This button object is then applied to a unit, and the unit can cast that spell. To create a spell you should duplicate the obj\_button\_spell1\_template object, which is found the folder Buttons>Spells. This is a button object, and like everything else, everything you need to edit is found in the create event.

3.2 Types of Spells

There are 4 different spell casting types. This affects HOW the spell is cast. The four different spell types are: Projectile (0), Target (1), Free Target (2), and Activate/Toggle (3). Projectile summons a projectile in the direction of the mouse. Target is used when you want your spell to target a particular unit, whether it be ally or enemy. Free Target aims at a particular spot on the ground. Activate casts the spell on the spot, also useful for toggle-able skills. The important variable in the spell object is the spell\_code variable which is a string. This string executes when the unit has successfully casted the spell. This is where you should put your code (for what the spell does), and make sure it is as a string.

3.3 Adding Spells to Units

This is relatively simple. Go to the create event of the unit that you want to add the spell to. Near the bottom there is a long list of variable like hud\_gen\_s1. These variables hold the buttons/skills/spells that the unit can perform. There are 18 of these, and they all have a different position on the HUD. You simply have to place your spell in one of these variables, and the spell will pop up on your unit when you select them.

4. Creating Resources

4.1 Basics

To create a new resource you should duplicate obj\_resource\_template object, which can be found in the Resources folder. Everything that you need to edit should be found in the create event.

4.2 Resource Types

There are two different resource types. In Warcraft terms, there are Trees and Goldmines. This only effects how the resources are visually collected.

5. Creating Buildings

5.1 Basics

To create a new building, duplicate the obj\_building\_template object, which can be found in the Buildings folder. Everything that needs to be modified can be found in the create event of the object. The Building object has three states: pre-built, in-construction, and fully built.

5.2 Adding Buildable Units

Adding buildable units to the building is very similar as to adding spells to a unit. You should duplicate the obj\_unit1\_template (it is a button), which can be found in the Buttons>Build Queue folder. Fill in the details of the unit that you want the button to make. Then add this button to the building, just like you would add a spell to a unit.

6. Creating Buttons

6.1 Basics

To create a generic button you should duplicate the obj\_button\_template object, which can be found in the Buttons folder. These buttons can be used for spells, buildings, unit orders and units. There are more specific buttons within the Buttons folder which you can duplicate, instead of the generic template, if the button does the particular function that you want it to do. Similarly to everything else, everything that you need to edit will be in the create event, EXCEPT that you will also need to add your own code in the global mouse left press event, which will dictate what will happen when you press the button.

8. Important Global Variables

obj\_HUD.target – This hold the instance id of the unit or building that is selected, when it is the only one selected. It is useful for codeing the casting of spells, as you probably want to know WHICH unit wants to cast a spell.

9. Multiplayer

**9.1 Basics**

For most multiplayer related things, you can find them in the ‘Multiplayer’ folder. Most objects that are used for single-player are also compatible with multiplayer. The major things that are made specific for multiplayer are units, buildings and spells. The important objects for multiplayer are: obj\_client, obj\_server, and scr\_msgHandler. Scr\_msgHandler is used for interpreting messages sent by client and server. Each game application involded in a multiplayer game will have an obj\_client. Only the host of the multiplayer game will have an obj\_server. All client objects only communicate directly with the server object, they do not communicate with each other (clients cannot directly send messages to each other, must go through server first).

Multiplayer uses faucetnet v1.4.2 extention, which means it requires gamemaker pro (minimum GM 8.0)

To communicate with the server (from the client) you have to write you message in a very particular way. Your message must start with a header containing a unique value:

write\_ushort(global.ServerSocket,headerid) //must be of type ushort  
 write\_int(blah,blah) // you can send whatever other information you want (any type)  
 // you do NOT need to do ‘socket\_send()’ (it is done for you)

You also need to add an entry to ‘scr\_msgHandler’ to read the contents of the messages, which uses that ‘headerid’ to identify the message sent. You should ‘read()’ the data in the same order that it was ‘write()’.

To start a multiplayer game, just transition to the ‘rm\_multi\_lobby’ room, or if you want to create your own room, just make sure the ‘obj\_init\_multiplayer’ is in there (that object just sets up some initial multiplayer variables)

**9.2 Useful Variables**

Variables only found in obj\_server (HOST):  
global.PlayerCount – number of players in the game  
global.player\_socket[X] – the socket id of the player with id X  
global.player\_name[X] – the name of the player with id X  
global.player\_race[X] – the race of the player with id X  
global.player\_team[X] – the team of the player with id X  
global.player\_color[X] – the color of the player with id X  
global.ServerState – the number of the state of the server  
A list of color information  
A list of race information  
A list of maps

Variables only found in obj\_server:  
global.PlayerId – Id of the client, consistent with the value on the server  
global.p\_race[X] - races of all other players (includes self)   
global.p\_team[X] - which teams all other players are on (includes self)  
global.p\_col[X] - colors of all other players (includes self)  
global.p\_name[X] – names of all other players (includes self)  
A list of unit and building aliases

**9.3 Units**

(NOTE: make sure you never duplicate anything from ‘Template’ folders, even for single player. This includes buildings and buttons)

Creating units for multiplayer is fairly tedious, and thus very error prone, so be careful. To create a unit for multi-player you can either duplicate obj\_unit\_multi\_shared (in Multiplayer > Units > Base) to create a new unit, or you can duplicate a unit you already made for single player and just move it into the ‘Base’ folder. Make sure the parent for this unit you just made is obj\_unit\_template\_multi (in Multiplayer > Units > Base > Template), this is very important.

Next step, go to ‘Player0’ folder (in Mulitplayer > Units) and duplicate one of the unit objects (any should do). Go into the create event of the duplicated unit and change the variable ‘baseObject’ to the unit you created in the previous step. Double check that the parent is something like obj\_p0\_unit\_template. Now do this for all the other ‘Player#’ folders, always checking that it has the correct corresponding parent (obj\_p#\_unit\_template)

Next step, go to obj\_client and go to the create event. You need to add an alias for it near the bottom. Be careful, there are separate ones for units and buildings. It should look something the following:

global.obj\_unit\_type[some\_unique\_number,0] = obj\_p0\_unit\_you\_just\_made  
global.obj\_unit\_type[some\_unique\_number,1] = obj\_p1\_unit\_you\_just\_made  
global.obj\_unit\_type[some\_unique\_number,2] = obj\_p2\_unit\_you\_just\_made  
etc…  
(make sure your unique number is less than 60000, or if you understand, it must fit in an unsigned short)

If you want to spawn a unit in multiplayer, do NOT use instance\_create(blah,blah,blah). You have to send a message to the server. Take a look at scr\_msgHandler, specifically msg id’s in the 800’s, or you can take a look at obj\_startpos0 (in Multiplayer > StartPos) for an example of what the message you send should look like.

**9.4 Buildings**

Creating buildings for multiplayer is the exact same process as creating a unit (as above) except everything is in the ‘Buildings’ folder (in Multiplayer)

**9.5 Maps**

When creating rooms for multiplayer games there are several requirements. Make sure that views are enabled and view0 is working. The creation code for the room should contain the following code:

instance\_create(-100,-100,obj\_control\_ingame);  
instance\_create(-100,-100,obj\_client);  
if (global.isHost) { instance\_create(-100,-100,obj\_server); }

Also if you want your map to be selectable from the lobby, you should add it to the list found in the ‘create event’ (near the bottom) of obj\_server (in Multiplayer).

**9.6 Spells**

Creating spells for multiplayer is very difficult as there is no ‘template’ and you have to write everything yourself and create custom messages to communicate with the server (you also have to do the synchronization between the clients yourself). The main idea behind this when a spell projectile is created client side, it sends a message to create that spell so it shows up correctly on the other clients. scr\_spellHndlr is used to receive the information about what spell to create, what is should look like and how it should behave. Taking a look at scr\_spellHndlr, the fireball spell, the stun spell, and msg==600 in scr\_msgHandler will help you understand how you can go about creating spells for multiplayer. (Note: only the fireball and stun spell work for multiplayer. I could not be bothered to do the other spells)

**9.6 Other**

If I didn’t mention a certain topic in multiplayer, like resources, then they will most likely be fully compatible for single-player and multi-player without have to change anything at all.