**Converting an objective map to Deathmatch, tutorial**

This tutorial describes how to script away all traces of the objectives in an Objective map to make it a [DeathMatch](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch) map.

Why would you need a tutorial on this? Well: If you can use the same \*.bsp for both game modes, It will be a lot simpler to keep the map updated. Because you only need to recompile ONE \*.map file to create TWO new versions when correcting an error or if you just need to redesign something in your map ( I've had compile times of up to 3 days, so you see why I like this idea ).

This tutorial is based on how I converted the map [dm/vemork\_factory\_small\_obj](http://gronnevik.se/rjukan/index.php?n=Main.MapDownload" \l "vemork_factory_small_obj) into the map [dm/vemork\_factory\_small\_dm](http://gronnevik.se/rjukan/index.php?n=Main.MapDownload" \l "vemork_factory_small_dm). The tutorial continues upon the [multiplayer bomb objective](http://gronnevik.se/rjukan/index.php?n=Main.ObjectiveExplosion) tutorial and the [multiplayer steal objective tutorial](http://gronnevik.se/rjukan/index.php?n=Main.ObjectiveTheft), but can be applied to any objective map that has [DeathMatch](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch) spawn points in it.

This is what you do to an Objective map to make it a [DeathMatch](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch) map.

**On this page...**

1. [Rename and move the map file](http://gronnevik.se/rjukan/index.php?n=Main.ConvertingObjToDm" \l "toc1)
2. [Write a new .scr file](http://gronnevik.se/rjukan/index.php?n=Main.ConvertingObjToDm" \l "toc2)
3. [Update references to the new names](http://gronnevik.se/rjukan/index.php?n=Main.ConvertingObjToDm" \l "toc3)

So lets start from the top:

**Rename and move the map file**

It would be really nice if you could ship a map with one \*bsp file and two separate \*.scr files for the Objective and [DeathMatch](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch) "modes". This, however, is not as easy as it might sound. You can get it to work if you only have one \*.scr present at any time, but as soon as they both are present, problems begin. The wrong versions of files start to load ( like wrong loading screens and stuff like that ). Beacause of this a rename of the \*.bsp file is recommended.

I copied **vemork\_factory\_small** from the **obj** directory to the **dm** directory and renamed it to **vemork\_factory\_small\_dm**.

**Write a new .scr file**

The difference between an objective map and a [DeathMatch](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch) map is that there are no bombs an other objectives present ( well, OK, there may be bombs... but not in the "Blow the objective" way... ). The good news is that most objectives are made up of a number of named objects. Thus they can be easily accessed via the \*.scr file and scripted in or out of existence.

- So to keep the main method ( script, subroutine, procedure or what ever you call it... the thing that starts with "main:" and ends with "end" ) clean we put the objective removal in a separate method called hide\_objectives and call it from the main method like this:

thread hide\_objectives

- Insert the line above before the **level waittill spawn** command ( And remember ther is no level waittill roundstart in a [DeathMatch](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch), as spawn = roundstart ).

- OK; lets do the actual magic in the hide\_objectives method... and dont be alarmed by the code mass, its basically the same code repeated four times. And besides: Im here to explain it to you in the // code comments.

Insert this ( or rather something like it, using the targetnames that you used in your map ) somwhere beneth the main method:

hide\_objectives:

// Start to remove all the explodersmashed brushes

// (the brushes that represents the objective

// after is has been blown up)

// Set how many explodersmashed brushes there are...

if ($explodersmashed == NULL) {

level.explodersmasheds = 0

} else {

level.explodersmasheds = $explodersmashed.size

}

// Insert all the explodersmashed brushes ( if any )

// into an array and loop the array

// while deleting all the objects in it

if (level.explodersmasheds > 0) {

level.explodersmashed = exec global/makearray.scr $explodersmashed

for (local.i=1;local.i<level.explodersmasheds+1;local.i++) {

level.explodersmashed[local.i] delete

}

} // ...all explodersmashed brushes should be gone now.

// Start to remove all the exploderchunk brushes

// (the brushes that represents the pieces of the

// objective getting thrown out of the explosion)

// Set how many exploderchunk brushes there are...

if ($exploderchunk == NULL) {

level.exploderchunks = 0

} else {

level.exploderchunks = $exploderchunk.size

}

// Insert all the exploderchunk brushes ( if any )

// into an array and loop the array

// while deleting all the objects in it

if (level.exploderchunks > 0) {

level.exploderchunks = $exploderchunk.size

level.exploderchunk = exec global/makearray.scr $exploderchunk

for (local.i=1;local.i<level.exploderchunks+1;local.i++) {

level.exploderchunk[local.i] delete

println ("Deleted Exploder Chunk " + local.i)

}

} // ...all exploderchunk brushes should be gone now.

// Start to remove all the exploderfire brushes

// (the brushes that represents fire

// and brimstone fx of the explosion)

// Set how many exploderfire brushes there are...

if ($exploderfire == NULL) {

level.exploderfires = 0

} else {

level.exploderfires = $exploderfire.size

}

// Insert all the exploderfire brushes ( if any )

// into an array and loop the array

// while deleting all the objects in it

if (level.exploderfires > 0) {

level.exploderfires = $exploderfire.size

level.exploderfire = exec global/makearray.scr $exploderfire

for (local.i=1;local.i<level.exploderfires+1;local.i++) {

level.exploderfire[local.i] delete

println ("Deleted Exploder Fire " + local.i)

}

} // ...all exploderfire brushes should be gone now.

// Start to remove all the explodertrigger brushes

// (the brushes that is used to activate / de-activatethe object)

// Set how many explodertrigger brushes there are...

if ($explodertrigger == NULL) {

level.explodertriggers = 0

} else {

level.explodertriggers = $explodertrigger.size

}

// Insert all the explodertrigger brushes ( if any )

// into an array and loop the array

// while deleting all the objects in it

if (level.explodertriggers > 0) {

level.explodertriggers = $explodertrigger.size

level.explodertrigger = exec global/makearray.scr $explodertrigger

for (local.i=1;local.i<level.explodertriggers+1;local.i++) {

level.explodertrigger[local.i] delete

println ("Deleted Exploder Trigger " + local.i)

}

} // ...all explodertrigger brushes should be gone now.

// HEADS UP! This code differs from the 3 code paragraphs above!

// Start to make all the exploder brushes solid

// (the brushes that represents the objective

// before is has been blown up )

// Set how many exploder brushes there are...

if ($exploder == NULL) {

level.exploders = 0

} else {

level.exploders = $exploder.size

}

// Insert all the exploder brushes ( if any )

// into an array and loop the array while

// making all the objects in it solid.

if (level.exploders > 0) {

level.exploders = $exploder.size

level.exploder = exec global/makearray.scr $exploder

for (local.i=1;local.i<level.exploders+1;local.i++) {

level.exploder[local.i] solid

level.exploder[local.i] thread global/exploder.scr::solidify

println ("Solidified Exploder " + local.i)

}

} // ...all exploder brushes should be solid now.

// Delete the model that represents the bomb.

$distiller\_bomb delete

// Remove the document objectives and their triggers

// ( see the the multiplayer steal objective tutorial )

$wall\_document\_trigger.target delete

$wall\_document\_trigger delete

$desk\_document\_trigger.target delete

$desk\_document\_trigger delete

end // End of the hide\_objectives method

Thats the scripting... you probably need to change the last 5 lines to reflect the targetname's you gave your objectives, but the rest of the code should wor "as is".

If you want to show the expoded obective istead of the intact one in your Death Match map; just exchange the behaviours of the exploder solidifying with the removal of the explodersmashed brushes ( to remove the exploder brushes and solidify the explodersmashed brushes ).

**Update references to the new names**

Now that you changed the name of the map file, you will need to change the references to it as well:

1. Name the new \*.scr file to fit the new map name.
2. Rename the \*.urc file to fit the new map name, and update the first

line in it to fit the new map name.

No need to change any \*.shaders or \*.tik files that I can think of... well maby if you are using an "ubersound workaround" you need to update the \*.tik file to suport your map or map type ( last entry in an "aliascache" line ).

You may want to change the loading picture or text, but you dont need to.

**That's all you need to know to create a** [**DeathMatch**](http://gronnevik.se/rjukan/index.php?n=Main.DeathMatch) **out of your Objective map.**