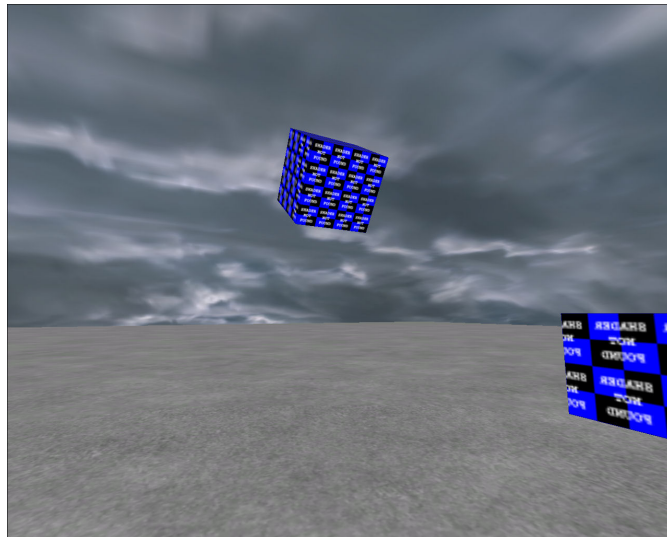


Tutorial Mapscripting

by Qualmi

22. Mai 2009



Shortened Version. This Tutorial will teach you the very basics of W:ET Mapscripting. It is a beginners guide, quick to read, and comes along with example maps. You will learn how to start a script, how to handle Mapevents and much more.

1 How to read this tutorial

Very easy. The first part will teach you all you must know to fully understand the second part. Its mainly theoretically and will explain some things, but not too much. In the second part of this tutorial you will then study some example maps i have prepared for you. Dont worry if you dont understand everything at first. You can always come back to read some things again. Also take a look into the scripts. I have put some comments there.

2 Theory

2.1 First Script

Code:

```
game_manager
{
    spawn
    {
        wait 200
        wm_allied_respawntime 5
        wm_axis_respawntime 5
        wm_set_round_timelimit 10
        wm_setwinner 0
    }
}
```

Copy and Paste this into a .txt-file and name it [NameOfYourMap.script](#). Save your file and extract it into [etmain/maps](#). Now before you start your map you should insert a [script_multiplayer-entity](#) and set the following values:

key: [scriptname](#)
value: [game_manager](#)

Save, compile and start your map. You should recognize by looking at the spawntimes that your script has been successfully started.

2.2 Basic structure of a script

Rough sketch

```
game_manager
{
    spawn
    {
        wait 500
        //...commands
    }

    //you can reach this block by typing trigger game_manager pointlabc
    //from every location of the script
    trigger pointlabc
    {
        //...commands
    }

    trigger point2
    {
        //...commands
    }
}

Entityblock1
{
    spawn
    {
        wait 500
        //...commands
    }

    death
    {
        //...commands
    }
}

.
.
.
.
.

EntityblockN
{
    spawn
```

```
{
    wait 500
    // ... commands
}

death
{
    // ... commands
}
}
```

2.2.1 The Trigger-Command

This command allows jump instructions. The command in general looks like [trigger Routine Subroutine](#). Once your script executes such a command, it jumps to the specified block, executes it, and then jumps back to its origin point.

2.2.2 Entity-Blocks

An entity-block is a block of your script, which belongs to an entity. By setting the value [scriptname](#) to an entity you have specified the name of the block this entity will call within your script when something happens to it (f.i. if a bridge gets destroyed or a gate has been breached). Of course you have to insert such a block into your script.

2.2.3 Event-Blocks

These are predefined subroutines of an entity-block. If f.i. there has been destroyed a bridge, then the script will search for the related entity-block and will call for an [event-block](#) in there. Examples for event-blocks are `spawn{...}` (Mapstart), `death{...}` (Entity gets destroyed), etc...

2.2.4 Targetname

By setting the value [targetname](#) you can identify an entity with a name and maybe later do some things from script to it like for example setting it invisible or make it move into certain directions.

2.3 Accum-Register

An accum-register serves different tasks in a map. They [store numbers](#), nothing else. F.i. in CTF Maps you can count the amount of flags which have been taken and then execute the end of the map if one team has reached a certain amount of captures. Also in the map oasis an accum-register serves to count the amount of destroyed guns. You can find accums in almost every script. They are very important.

2.3.1 A few important commands

There are more, but these are sufficient for the beginning.

```
accum n set a //allocation of value a
accum n inc a //increases the value with a
accum n inc -a //decreases the value with a
accum n set a //sets the value to a
accum n random a //allocation a, random number between 0 and a-1
accum n bitset y //sets bit y of n to 1
accum n bitreset y //sets bit y of n to 0
accum n abort_if_not_equal a //aborts the subroutine if n is not equal to a
accum n abort_if_equal a //aborts the subroutine if n is equal to a
accum n abort_if_less_than a //aborts the subroutine if n is less than a
accum n abort_if_greater_than a //aborts the subroutine if n greater than a
accum n trigger_if_equal a <routine> <subroutine> //triggers a block if n = a
praccum n //prints the value of accum n in console
```

2.3.2 A few restrictions on accums

There are a few things to take care. You can in total choose 10 [local accums](#) (each block) and 8 [globals](#) . The difference between local and global is their validity. A local accum is only valid in the block it has been declared. Example: in block a there exists one accum 1 which has a value of 5. In block b there also exists one accum 1 which has the value of 3. Those 2 accums are only valid in their specific blocks, they are local. Globals are valid in every block. You can declare them with [globalaccum n set a](#) .

3 Examplemaps

You can now study the example maps i have made. Its best to start with example map 1 and head over to the last one. This way you will learn the easiest way. I have listed the files for you here. Have fun.

Mapfile: 01_activate01.map

Scriptfile: 01_activate01.script

Target: push the button

What to learn: activate-event

Mapfile: 02_activate02.map

Scriptfile: 02_activate02.script

Target: push the button

What to learn: scriptname, targetname, alertentity-command

Mapfile: 03_mover01.map

Scriptfile: 03_mover01.script

Target: push the button, shoot the script_mover

What to learn: pain-event, death-event

Mapfile: 04_mover02.map

Scriptfile: 04_mover02.script

Target: push the button

What to learn: moving things in your map, adjusting speed

Mapfile: 05_mover03.map

Scriptfile: 05_mover03.script

Target: push the button

What to learn: moving things around certain axis

Mapfile: 06_mover04.map

Scriptfile: 06_mover04.script

Target: push the button

What to learn: accums and control_structures, setstate-command

Mapfile: 07_destructible_objective.map

Scriptfile: 07_destructible_objective.script

Target: destroy the wall

What to learn: func_explosive, constructible_class-command, how to end the game, trigger-command

Mapfile: 08_constructible_objective.map

Scriptfile: 08_constructible_objective.script

Target: construct and destroy your objective

What to learn: func_constructible