

STRONGHOLD

EDITING REFERENCE

BY TORMENTOR667

STRONGHOLD

EDITING REFERENCE

CONTENT

| | | |
|-----|-------------------------------------|----|
| 1.0 | Introduction | 2 |
| 2.1 | Preparing your machine | 3 |
| 2.2 | Setting up your map editor | 4 |
| 3.0 | The Action Script (ACS) | 5 |
| 4.0 | Your first Stronghold map | 6 |
| 4.1 | Basic settings | 6 |
| 4.2 | Configuring the Prison | 6 |
| 4.4 | Setting up the spawn zones | 7 |
| 4.5 | Setting up the monsters | 8 |
| 4.6 | Setting up the messages | 8 |
| 4.7 | Setting up the items | 9 |
| 4.8 | Setting up the spawn spots | 10 |
| 4.9 | Defining custom events (script 100) | 11 |
| 5.0 | The other gamemodes | 12 |
| 5.1 | Creating a Core map | 12 |
| 5.2 | Creating a Goal map | 12 |
| 5.3 | Creating a Limit map | 13 |
| 5.4 | Creating a Milestone map | 13 |
| 5.5 | Creating an Overmind map | 14 |
| 6.1 | Final Words | 15 |

STRONGHOLD

EDITING REFERENCE

INTRODUCTION

Welcome to the official editing reference for your very own **Stronghold** map. If you are reading this you most probably have already taken a look at the game and made yourself familiar with the possibilities and modes you can choose from which makes the whole thing a bit easier for you. If you haven't, don't worry, you will get a detailed look inside the meat here anyway.

So what is it all about in general? **Stronghold** is meant to change the way Doom is played. Instead of hunting down hellspawn in possessed bases and hell itself it's all about you preventing hordes of demons to overrun your UAC outpost.



It can be compared to a mixture of **Skulltag's Invasion** gamemode and the very popular **Tower Defense** games, just with a little twist. Monsters will attack in waves trying to reach a specific point or destroying a specific object in the map. Each wave you will get supplies from the mission control spawned to the map and with their help you have to defend these points and objects from the incoming hordes.

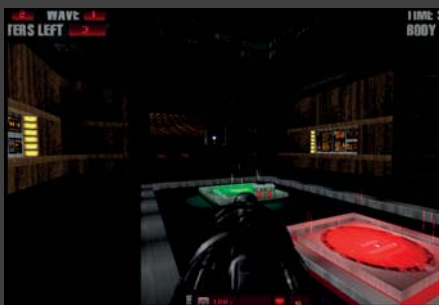
From a mapping aspect, **Stronghold - On The Edge Of Chaos** comes with a huge arsenal of weapons and items to battle the wide range of different monsters. Additionally it's packed with a common resource of textures, special effects, ambient sounds and decoration resources you can choose from. And if that's not enough, you still can add your very own content easily to your maps or addon packs.



This simply means: **The only limitation is your imagination!**

But what makes mapping for **Stronghold** so special and brilliant either way? Quite easy to tell: It's simple and powerful! The complexity of the script allows for unlimited possibilities but still is easy to understand, to access and to work with. All you need is a good idea and some basic ACS knowledge and you are ready to go.

With this editing reference, you will get an inside-look into the different aspects of setting up a **Stronghold** map, about the different invasion components (e.g. spawnzones/-spots, patrol routes, wave settings, reinforcements, supplies and events), the gamemode and even how you can create your own hub-based addon.



But enough now, let's start with the real work and fun!



Skulltag

Did you already know that Stronghold is completely **Skulltag*** multiplayer compatible?
(*98b at least needed)

STRONGHOLD

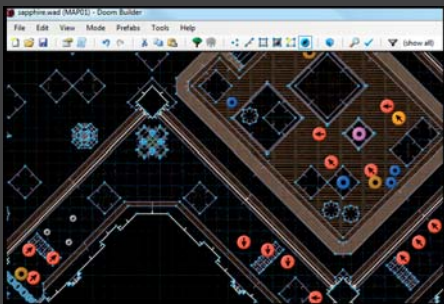
EDITING REFERENCE

PREPARING YOUR MACHINE

If you are an active member in the Doom community, you will most likely already have the latest versions of the most common utilities and source-ports out there. But in case you aren't sure, just read on.

The Map Editor

It doesn't actually matter which map editor you are using considering that a Stronghold map is also just a simple "ZDoom Format" map. Though regarding the sheer amount of extra resources and filetypes, **DoomBuilder 2** proved to be the perfect choice for working with Stronghold. Sure, there are countless other map editors that can be used as well (like WadAuthor or SLADE) but DB2 proved to be the right choice while working on Stronghold in the past. It is capable of loading all types of resources (custom DECORATE objects, TX_ marker textures and all that other evil stuff, new ZDoom



versions come with directly from a PK3 or a PK3-like directory structured resource). So, this is why this editing tutorial is mainly based and focused on DB2.

The Action Script Compiler

Most map editors already come with a shipped version of ACC (the compiler for your Stronghold ACS) but unfortunately it's not always the most frequent one. If you encounter errors while trying to compile your script, make sure to update to the latest ACC and ist .acs files before investigating your own script.

The Source Port

Considering the fact that you want to create your own maps for Stronghold you most likely have already played the



game and therefore have the latest version of the sourceport that's needed for **Stronghold**. Though it's never a bad idea to be up to date, so updating to the latest version of GZDoom isn't a bad idea as well.

GZDoom CFG for DB2

Doombuilder regularly doesn't come with a GZDoom editing config file. Either use "ZDoom (Doom in Hexen format)" then or use the one that comes with this editing reference and put it into your DB2 "/configurations/" directory.

So, if you have prepared these things, you are ready to go and start with setting up the resources and the map editor.

Summary

So, just to make sure we have everything that's needed

- » Doom Builder 2
- » GZDoom .CFG
- » Latest ACC (script compiler)
- » Latest version

Got it? Great, then let's go for the next few steps.



DoomBuilder 2

Download the latest version of DB2 from www.doombuilder.com



No GZDoom?

Stronghold supports also mapping for Skulltag or GZDoom/ZDoom in software mode!



ACC Compiler

Download the latest version of ACC (mingw) and the latest .acs files from svn.drdteam.org



GZDoom CFG

If you do not have a GZDoom DB2 editing .cfg, use the one that's included here.



GZDoom

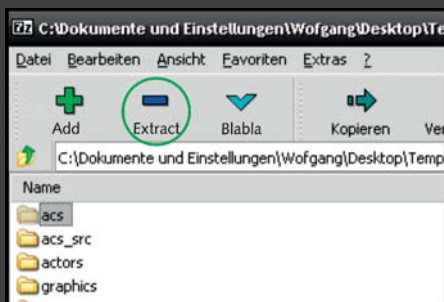
Download the latest version of GZDoom from grafzahl.drdteam.org For SVN builds, you can also visit svn.drdteam.org/gzdoom.

STRONGHOLD

EDITING REFERENCE

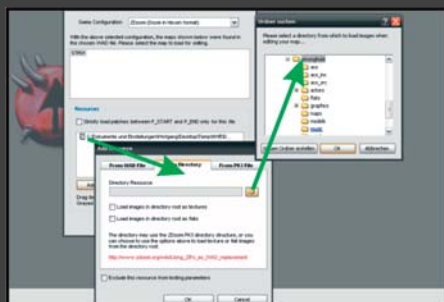
SETTING UP YOUR MAP EDITOR

First we - if not already done - get yourself a copy of **Stronghold**. After unzipping that one, you will find yourself with a **.txt** file and a **.pk3** file. Open the PK3 with a ZIP program of your choice (e.g. WinZIP, 7ZIP or WinRAR) and extract its contents into a new editing resource folder, for example create a "stronghold_res" subdirectory somewhere.



Extracting the PK3

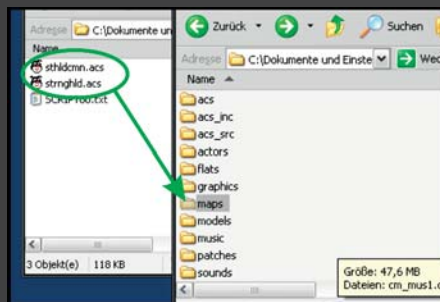
As you may have already noticed while waiting for the download to be finished, **Stronghold** is about 150mb large. Everytime you load resources from the PK3 makes DoomBuilder 2 unzip the complete PK3 - every single time! To avoid this, we extract the content from the Stronghold PK3 one single time and then we use



DoomBuilder's "From directory" - feature to load the resources directly from the root directory that we just created. This way you can save a lot of time as there is no more need to let DB2 extract the PK3 content each time you load your map with these resources.

Placing the ACS libraries

Stronghold is working with so called ACS libraries. You do not compile the whole 100kb script into each map, you only put the "settings" into the script part of your map. The "game" itself is stored in external precompiled acs files ("/acs/" directory in the pk3). Though for compiling the "settings" script you need the uncompiled "game" scripts so ACC understands what to do. Therefore, you have to copy the two acs files from "/acs_src/" into the folder where your new Stronghold map is saved, otherwise your script won't compile.



Summary

Quite a lot of stuff to take care of. But to keep things easy, here is a little checklist for you:

- » Extract the .PK3
- » Load resources "from directory" in DB2
- » Copy "/acs_src/" content to the same folder as your map is saved in

If these have been taken care of, you should be able to load your map easy and fast with all the necessary and additional resources Stronghold comes with as well as work on your map scripts and compile them.



PK3 vs. DIR

The Stronghold pk3 takes much longer to load in DB2 than an extracted directory.



ACS lib location

Place all .acs files from "/acs_src/" where your map is saved.



Script Libraries

You must not change the script libraries if you want to make your maps compatible with Stronghold.

STRONGHOLD

EDITING REFERENCE

THE ACTION SCRIPT

Before you start mapping now, there is one thing you need to know and that little thing is about the whole script issue.

Stronghold comes with script libraries. The whole gamemode and behaviour has been stored in precompiled acs files that simply get loaded when a Stronghold map starts. This way, the main core is placed on one spot and gets only loaded into the maps instead of heaving all the thousands of lines in each single map.

The positive aspect about this is, that the mapper only has to set a few variables, all the functions are stored somewhere else. Due to that, things are getting much easier.

Does this limit your imagination? Actually not, you still have the ability to change the script libraries themselves - the source files are there - and enhance the overall game the way you like it. The only problem is that this will also change the stock maps when loading your new work with **Stronghold**, but sometimes also this can be intended (e.g. gameplay modifications).

```

1 #include "common.acs"
2 #import "strengthd.acs"
3
4 script 100 open
5 {
6     Waves = 10;
7     HateSpicenter = 996;
8
9     EventScript = 101; // similar to old script 667, but a bit different
10    // called when new wave x is announced message disappears, as items
11    // for wave x spawn.
12    // First argument given to script is the integer x
13
14    // Prison stuff
15    PrisonSpot = 678; // There should be multiple teleport spots with this tid
16    // (Not a series of tids as in older versions)
17    // Set to 0 (or don't set it at all as it is the default) if map doesn't
18    PrisonExit = 699;
19
20    PrisonCamStart = 691; // This is the tid of the first camera for the jail.
21    // These will be set as the cameras for the CT_1 through CT_6 textures.
22
23    PrisonTeleporterInTid = 99;
24
25    PrisonTeleporterTrustAngle = 0; // Angle the teleporter trusts when it is.
26    PrisonExitTrustAngle = 128; // Angle prison exit trusts when there are pl
27
28    // ...
29    // ...
30    // ...
31    // ...
32    // ...
33    // ...
34    // ...
35    // ...
36    // ...
37    // ...
38    // ...
39    // ...
40    // ...
41    // ...
42    // ...
43    // ...
44    // ...
45    // ...
46    // ...
47    // ...
48    // ...
49    // ...
50    // ...
51    // ...
52    // ...
53    // ...
54    // ...
55    // ...
56    // ...
57    // ...
58    // ...
59    // ...
60    // ...
61    // ...
62    // ...
63    // ...
64    // ...
65    // ...
66    // ...
67    // ...
68    // ...
69    // ...
70    // ...
71    // ...
72    // ...
73    // ...
74    // ...
75    // ...
76    // ...
77    // ...
78    // ...
79    // ...
80    // ...
81    // ...
82    // ...
83    // ...
84    // ...
85    // ...
86    // ...
87    // ...
88    // ...
89    // ...
90    // ...
91    // ...
92    // ...
93    // ...
94    // ...
95    // ...
96    // ...
97    // ...
98    // ...
99    // ...
100   // ...

```

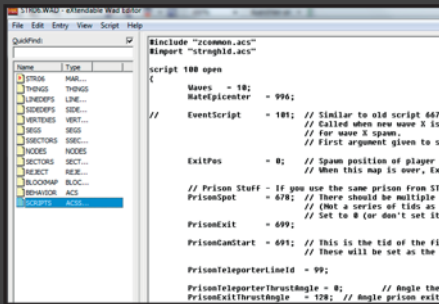
Writing the script from scratch?

Considering the huge size of the library and all the different possibilities you have to create your own game for **Stronghold**, writing a script from scratch is absolutely no fun. To make things easier for you, we have already provided you basic scripts (actually rips from Stronghold maps) that you can simply copy and paste directly into your DoomBuilder script editor. It makes life

much easier, as you do not have to look for all the variables and commands on your own, they are already here and only need to be altered.

There are 5 different gamemodes in Stronghold and so you will find 5 different script text files in the **/gamemode_scripts** directory which comes with this editing reference.

Though you can still write the **Stronghold** script from scratch or simply get a script from one of the other stock maps in the pk3. All of them are also as uncompiled SCRIPT lump in the map's wad.



The good thing about using our script templates or ripping scripts from stock **Stronghold** is that all of our mapper's have been very assiduous. Every little line and setting has comments attached which document the complete script. This makes creating your own Stronghold map even easier.

Though, we do want responsible mappers and therefore you will get an inside view of all the settings available. If you are impatient, feel free to start mapping now and read on.

» **Copy and paste script templates into your map for an easy and quick start.**



Script Templates

For each Stronghold gamemode there is already a finished script template for your map in the `/gamemode_scripts` directory.



Documentation

All the Stronghold map scripts are very well commented, working with these is easy as hell.

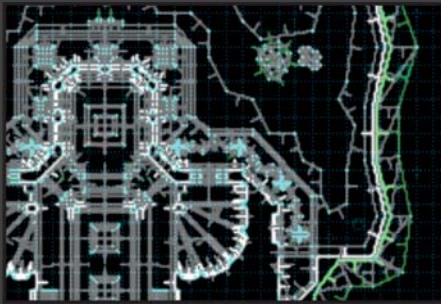
STRONGHOLD

EDITING REFERENCE

YOUR FIRST STRONGHOLD MAP

Time to start. According to the map script of STR02 or actually the script template „deadline_script.txt“ that you can find in the directory „/gamemode_scripts“, we will take a look at each function, each setting and each line to get a feeling for how the script works and what is actually possible.

To make things easier and to save you some time (as creating a whole map from scratch now might take a while), simply open up **STR02.WAD** from Stronghold, load the Pk3 as base resources and let us take a closer look.



Open up the script dialogue. The 1st line should be clear, the 2nd line is actually the most important one:

```
#import „strnghld.acs“
```

This imports the uncompiled gamescript of Stronghold (this is actually what has already been described on page 4 - 5), without this line, the map settings simply won't compile as all the functions and stuff are missing. Simply leave it as is.

Script 100 - The EventScript

Now we are proceeding to **script 100**, the one that contains all the settings for your waves, monsters, spawnspots and stuff. This is where the meat begins.

```
Waves = 10;  
HateEpicenter = 996;
```

The first line explains by itself, this is where you set the amount of waves.

After these waves have been beaten, the map simply ends. The second line defines the **HateEpicenter**, this is simply a referencepoint for the monster actors. Add this TID to a mapspot out of the map boundaries near the final waypoint/goal of the monsters.

```
EventScript = 101;
```

The event script refers to the script number that gets executed everytime a wave is about to start. We will take a closer look at this scrip later, just keep in mind that this setting is here.

The Prison Settings

In the next lines from 15 to 29 all the important settings for the prison will be made. For a little reminder: The Prison is the place where players get teleported in a multiplayer game when they run out of lifes. There they can spectate the remaining players on camera texture walls, join the map again through a gate or simply leave the map and go back to the intermission map through the exit.

```
PrisonSpot = 678;
```

There should be multiple Teleport destinations placed within your prison with this TID, these are the spots where players without lifes get moved to.

```
PrisonExit = 699;
```

This is supposed to be the Teleport destination on the battleground for players who join the action again (e.g. if other players use their life items to free comrades from the prison). Simply place a teleport destination with this TID near the player starts.

```
PrisonCamStart = 691;
```

This is the TID of the first camera for the jail. The next 5 TIDs are also reserved for camera things placed on the map. Make also sure to place camera



Prison Copy'n'Paste

As the prison is only a gameplay element, clever people simply copy and paste the whole structure.



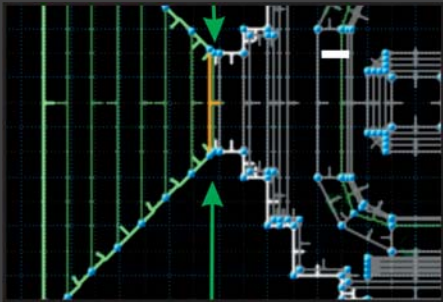
Deadline

Do not forget to assign script 255 to your deadline with the action ACS_ExecuteAlways for monsters.

textures into the prison, these will be automatically set to the textures CT_1 through CT_6.

```
PrisonTeleporterLineId = 99;
```

For this line, you need to take a closer look at the corresponding Prison line in STR02 as it needs to have several



important specifications to work. The initial version has to be

- » Impassable
- » Translucent, Additive, 255
- » MidTex „TELEPTY1“, red forcefield
- » Space behind the wall

The mechanics of this portal is always the same: As soon as a player uses one of his extra lives, the wall gets passable, the midtex changes into "TELEPTY1" (that's the green forcefield) and the line itself gets a new action that teleports one player back to the battlefield. Simply take a closer look at how the STR02 prison has been designed to save some time.

```
PrisonTele...TrustAngle = 0;  
PrisonExitThrustAngle = 128;
```

These two lines simply define the directions where player get thrust. The first line also refers to the Teleport back to the battlefield (in case it's deactivated) and the second one refers to the exit line in the prison, that leads back to the INTERMAP in case there are still players fighting (this refers to the line executing script 252 btw).

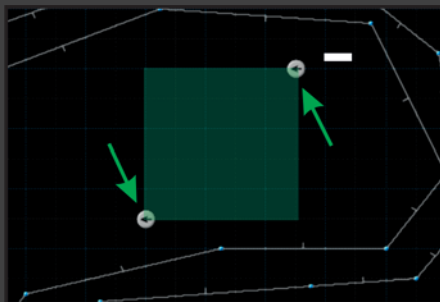
```
TeleporterOnLight = 99;  
TeleporterOffLight = 98;
```

These two lines are responsible for the dynamic light things near the forcefield that leads back to the battlefield. In the stock maps, the first line defines the green light, the second line defines the red light.

After this, you are done with the whole prison thing. I have to admit: I designed one single prison for my Stronghold maps and later copied and pasted the structures and things over and over again, as it is simply too much work to create these structures from scratch. This is also my main suggestion to save some time. After the copy and paste, you still can adjust the room to your liking but fortunately all the setup is already done.

Setting up the SpawnZones

Creating the boundaries for your zones is easy as hell, all you need to have are two map spots, one at the lower left corner, the other one on the upper right corner of your „spawning cube“.



The script will automatically calculate the area in between (even recognizing the z-heights!) and use this area as spawning zone.

```
AddZone (30, 31, 4);
```

With this line in your script, the zone gets added. The first number is the TID of the mapspot on the lower left, the second number is the TID of the mapspot on the upper right, and the



ClearZones();

Just in case you want to remove zones, simply use „ClearZones();“ and only readd the ones you want to have ingame - on the fly.

third number is the TID of the first patrol node. From this first node, you can continue the monsters route until they have reached the **Deadline**.

Setting up the monsters

No fun without monsters, time to set up the amount and type per wave.



Doing this is easy as hell and there are almost no limits for you.

```
WaveEnemy(1, „DoomImp“, 5);
```

This single line tells the script to launch 5 Imps in wave 1 towards the player's deadline. That's it! To add more types of enemies in the same wave, simply copy and paste this line and change the monster name (it's always the DECORATE name), change the amount of monsters (3rd column) and you are done. The first number is always the wave number, the second is the monster name and the third is the amount of monsters on Skill 3 (multipliers change this number for the other skill settings). Just another example:

```
WaveEnemy(2, „Demon“, 10);  
WaveEnemy(2, „DoomImp“, 5);  
WaveEnemy(3, „Spectre“, -5);
```

These 3 lines launch 10 Demons and 5 Imps in wave 2 and 5 Spectres in wave 3. You might now ask why is there a minus in front of the 5. This is just an additional feature that can be used for boss-fights. It simply deactivates the skill-scaler for this line. It means that no matter if you play on skill 1 or 5, there

will be always exactly 5 spectres on the battlefield.

At the very bottom of the **WaveEnemy()**; settings in **STR02**, you will find a line looking like this:

```
WaveEnemy(-1, „Arachno“, 1);
```

This line defines a **bonus wave**. Wavesettings (also for Ammo, Items, Messages, etc.) with a „-“ Prefix won't be called regularly but they can be called anytime through other scripts by using the following command:

```
CallWave (-1);
```

Imagine the player finding a secret plasmagun. As soon as it is collected, an additional bonus wave will be launched with a few more monsters. No reward without some blood.

Setting up the messages



The battlefield is always a huge chaos. It's sometimes very hard for a player to keep track of everything. Giving hints and information through the radio messages is always a great help and one of the useful features in Stronghold. Setting these up is also easy as hell.

```
RadioName[0] = RADIO_RANDOM;
```

...is one of the lines, you don't have to change, the real important stuff lies in the next few lines, so here we go:



Patrol Nodes

A very good tutorial about patrol routes can be found on ZDoom.org just in case you aren't familiar with that.



Bonus Wave

You can set up **bonus waves** by tagging wave settings ids with a „-“ as prefix. Call these bonus waves with „**CallWave (-X)**“



New content

You can add new content like monsters, items and powerups and spawn these through the script.



Static amount

To make sure that there is only a **specific** amount of monsters being spawned - no matter the chosen skill - use a „-“ in front of the number in your **WaveEnemy()**; setting.

STRONGHOLD

EDITING REFERENCE

```
WaveMessage(2,0,"Text");
```

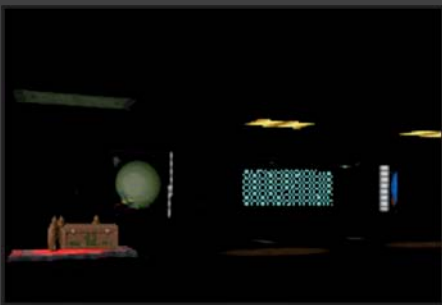
It's actually the same procedure as it was for the WaveEnemy(); setup. The first number defines the wave number (so in this case, the text gets displayed **before** wave 2 starts), the second number isn't used and the third column is the message itself. To make the text fit into the message box, make sure to use „\n“ commands for breaks. If you want to display more than one message, simply add another line below, just as it has been done for STR02. The messages will be displayed after each other.

This way you can set up several messages for each wave. If you want to set also a special victory message after all waves have been beaten, simply use the WaveMessageEnd(); command, just make sure it has the amount of waves +1 as WaveID

```
WaveMessageEnd(11,0,"Txt");
```

In this case, the amount of waves is 10, so the end message is tagged with 11.

Setting up the items



Time for setting up the amount of **powerups**, **weapons** and **health** - the three types of simple items in Stronghold.

```
WavePowerup(2, „Souls.“, 1);  
WaveMedic(2, „Medikit“, 1);  
WaveWeapon(2, „Shotgun“, 1);
```

Yes, you are right, it's easy as hell. These

three lines are responsible for the type and amount of specific stuff spawned at specific mapspots.

The first column is always the wave id, the items get spawned **before** the wave starts. The second column defines the type, here you also need to type the DECORATE name. The third column is responsible for the amount of items being spawned.

As you might figure out yourself, WavePowerup(); spawns things on the **powerup** spawn pads, WaveMedic(); spawns on the **health** spawn pads and WaveWeapon(); spawns on the **weapon** spawn pads and so on. You can indeed spawn a weapon on a powerup spawn pad, if it makes sense.

The supply of Ammo

First of all, ammo is a bit different from all the other spawn pads, mostly because it also gets scaled depending on the chosen skill settings - just as the amount of monsters is. Keep that in mind while defining your numbers.

```
WaveAmmo(2, BULLETS, 50);
```

Defining ammo per wave works almost the same as defining other wave settings. The first number is the wave id (ammo gets spawned **before** the wave starts), the second column is the ammotype, the third number is the actual amount of ammo that would get spawned on „Hurt me plenty“ with one player playing. The different ammotypes are as follows:

- » BULLETS
- » SHELLS
- » ROCKETS
- » CELLS
- » GAS
- » MINES

Other ammotypes are not supported. The system automatically calculates the amount of small and big ammo units



Multiple messages

For multiple messages per wave, simply add more WaveMessage(); lines. Messages will be displayed after each other.



Skill Scalers

The skill scalers only work for the amount of monsters and the amount of ammo. The number of other items gets never changed.

STRONGHOLD

EDITING REFERENCE

and spawns the necessary item count to reach the amount of needed ammo. That's fun!

Setting up the spawn spots

Last but not least, all the supply needs to be spawned somewhere. Time for setting up the spawnpads.



Stronghold has already several resources to indicate visually, where a spawnpad lies and what it actually spawns (flats and particle generators). Though, you are not forced to use this. The only thing you need to do is adding a few lines to your script:

```
AddMedicSpot(100);
AddAmmoSpot(101);
AddPowerupSpot(102);
AddWeaponSpot(103);
```

These are pretty self-explanatory, each one defines a mapspot as spawning spot for a item type.

You can also add more spawn spots of one type to your map, just make sure they do have different TIDs. The system will automatically roll through all spots of the same type and allocate the items.

```
AddAmmoSpot(101);
AddAmmoSpot(107);
```

Now onto the final two important lines for the game and gamemode:

```
ACS_ExecuteWait
(S_DEADLINE_GAME,0,0);
ACS_Execute
(S_MISSION_COMPLETE, 0, 0);
```

The first line indicates the gamemode (more on that later), the second line starts the ending sequence for each map as soon as the player has beaten each wave. These are necessary to make the game work!

Defining custom Events

Though, let's take a look at the script 101 that we have mentioned at the very beginning (page 6) and that gets implemented through this command

```
EventScript = 101;
```

Events are actions that can be placed between waves. You are able to add actions that surprise the player and totally change the setting.

STRO2 as an example uses this script to open a third spawning area for monster closer to the base after wave 7 at the beginning of wave 8.

```
script 101 (int wave)
{
    switch(wave)
    {
        case 8:
            Floor_Lower(...);
            AddZone(34,35,1);
            break;
    }
}
```

With this knowledge, the possibilities are endless. Imagine earthquakes with new spawning zones, moving zones from one area of the map to another, opening doors with new areas, changing weather, changing light... everything is possible. And so, you can even proceed on your map within several waves, changing the complete setting with just a few script lines.

And that's it actually, with this few pages of tutorial and information, you should be able to create your very own first **Stronghold Deadline** map. Time for the next gamemodes then.



Multiple Spawn Pads

You can add multiple spawn spots of the same type to your map, just make sure that they all have different TIDs.



Events

With the EventScript(); you can easily define actions that happen at the end or beginning of certain waves.



Spawnables

Complete lists of all spawnable monsters, items and weapons that already come with Stronghold can be found as text files in the „/object_lists/“ directory.

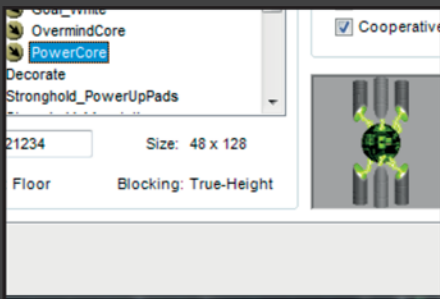
STRONGHOLD

EDITING REFERENCE

THE OTHER GAMEMODES

Creating a Core map

After learning how to create a Deadline map, the next and very easy step is your very own first **Core** map.



First of all, you need to place a **PowerCore** to your map (DoomEdNum 21234).

At the very top of the core script that you can also find as text file in the „/gamemode_scripts/“ directory, you will see this line again:

```
HateEpicenter = 996;
```

This time, it doesn't refer to a mapspot, it refers directly to the **PowerCore** itself.

Everything else needs to set up just the same way as it has been done for the **Deadline** script. The only difference can be found near the bottom:

```
ACS_ExecuteWait  
    (S_CORE_GAME,0,0);
```

This tells the Stronghold main script to display the Core health counter and end the game as soon as the core gets destroyed - therefore the only difference compared to the Deadline gamemode. And yes, that's all.

To give your imagination a little idea, you can also use other objects to defend, you are not forced to use our **PowerCore**. Also invisible dummy items in front of doors or moving NPCs could be used.

Creating a Goal map

This type of defense can be considered as „Multi-Core Defense“, instead of one single object, the monsters hunt down several actors just until all of them are destroyed. It's absolutely up to you, which kind of actors you want to use, everything's possible from npcs over cores to babes ;)

```
HateEpicenter = 996;
```

The script differences are very little, once again, the HateEpicenter is the TID for the objects that the monsters should hunt down.

```
GoalScript = 102;
```

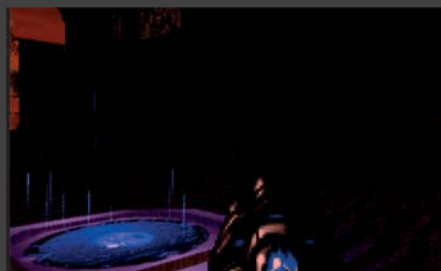
Next to the EventScript, that launches a specific script or action each time a wave is finished, you will also find a GoalScript this time, which launches a script or action each time a Goal gets destroyed.

For example you could change the monster route or the spawnzone, each time a goal gets destroyed so the monsters automatically aim for the next goal.

Other than this new line, the new script 102 and the obligatory gamemode line

```
ACS_EexecuteWait  
    (S_GOAL_GAME,0,0);
```

there is absolutely no difference in setting up a Goal defence gamemode. You just have a few more possibilities in terms of gameplay.



HateEpicenter();

Core and Goal gamemodes are very similar to Deadline gamemodes concerning the setup, the difference is that the HateEpicenter(); uses (a) specific actor/s

STRONGHOLD

EDITING REFERENCE

Creating a Limit map

What's the difference between a **Deadline** map and a **Limit** map?



Exactly 2 lines of code. The first one is the obligatory line that tells the main system which gamemode to play with.

```
ACS_ExecuteWait  
(S_LIMIT_GAME,0,0);
```

The second one is responsible for setting the amount of monsters that may pass the deadline until the game is over.

```
LimitMax = 100;
```

This simply states that the Limit is set to 100 but it can be anything else. Other than that, make also sure that the **LimitLines** the monastery need to reach have the (repeatable & walkover) special **226:ACS_ExecuteAlways** set to the script number **249** (actually different from Deadline games which use script number **255** for this).

Creating a Milestone map

After three gamemodes that are brazenly easy to implement and work with, it's time for something tricky. Milestone mode, here we come.

For the gamemode indicator, the standard line can be found again at the bottom part of the script after the wave setup:

```
ACS_ExecuteWait  
(S_MILESTONE_GAME,0,0);
```

The second difference compared to the **Deadline** gamemode is the MileScript

```
MileScript = 102;
```

We already know this kind of script, it's very similar to the **GoalScript()**; just in this case it gets executed everytime a Milestone is reached. Though, the mechanic is a bit different in this case. The whole idea of this gamemode is to



prevent demons reaching the milestone. But the game isn't over as soon as a milestone is breached, it just gets harder as monsters start spawning from the lost milestones then, much closer to the final deadline than before.

You can set these milestones in a direct row but it is also possible to set different locations and routes after each milestone has been breached.

In the script you will find these corresponding types of lines.

```
AddZoneMile(7,8,13,0);  
AddZoneMile(9,10,14,1);  
AddZoneMile(11,12,17,2);  
AddZoneMile(23,24,19,3);
```

The first number is the lower left mapspot of the spawning zone, the second number is the upper right mapspot of the spawning zone, the third number is the first node of the patrol route and the fourth number is the actual Milestone number where 0 is the one the monsters spawn at startup, 1 the one as soon as the first milestone is breached, and so on. In this case, the



LimitMax();

To define a different limit for Limit gamemode maps (default is 100), simply use **LimitMax=number**; to use a different value.



LimitLines

Don't forget to assign script 249 to the limit lines (**ACS_ExecuteAlways**). This is **different** from DeadLine games using script 255!

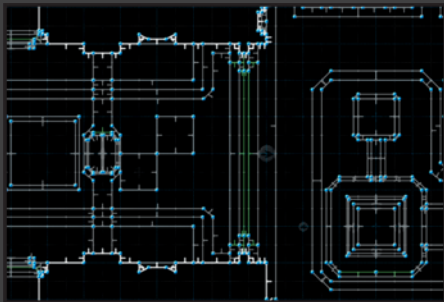


Milestones & Zones

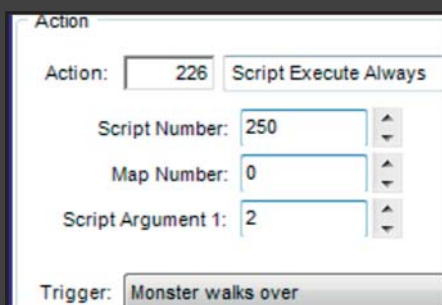
As soon as a milestone is breached, monster start spawning from a different spawnzone.

final milestone is also the deadline, but that goes without saying.

After the script has been prepared with the necessary additions, it's time to set up the milestone lines in the map.



The procedure is almost the same compared to **Deadline** and **Limit** gamemodes. The Milestonelines in question need to execute the script number **250** and use a number appropriate to their position as first script argument, e.g. the first Milestone



needs to have **1** as script argument, the second **2**, the third **3**. Last but not least, you also need to place a **Deadline** at the very end of your route, executing the script number **255**, just as in an **Deadline** game. So, in the end, we should have

» **3 Milestone lines**
(executing 250 with arg1 set to their number in the row)
» **1 Deadline**
(executing 255)

as well as the script additions that we already have done so far.

Last but not least, we still have the **MileScript()**, don't we? Setting this up works just the way the **EventScript()** and the **GoalScript()** work. Here is an example

```
script 102 (int mile)
{
    switch(mile)
    {
        case 1:
            door_close(25,40);
            break;
        case 2:
            door_close(21, 40);
            break;
        case 3:
            floor_lower(23, 100);
            break;
    }
}
```

This script closes the door tagged 25 as soon as the first milestone is reached, closes the door tagged 21 as soon as the second is reached and lowers the floor tagged 23 as soon as the third one is reached. Easy, isn't it?

Also take a look at the corresponding text file in the „/gamemode_scripts/“ directory.



With this knowledge you should be able to create your very own milestone maps. Don't forget: The maximum amount of milestones is actually three. Though... as always, with some tricky hacking, nothing is impossible, so in the end it's up to you I guess.

On to the final gamemode...



Milestone Maximum

You can't have more than three milestones plus one deadline.



Milestone & Deadline

Milestone gamemodes also use the regular Deadline, the milestone lines in the map execute script 250, the deadline executes 255.



Milestone Number

Each Milestone line gets defined by using the first script argument. For the first milestone it's 1, for the second it's 2, for the third it's 3.

STRONGHOLD

EDITING REFERENCE

Creating an Overmind map

The most fun and most impressive gamemode is definitely the **Overmind** one. On the other hand, it's unfortunately very hard to balance. Nevermind, what is it actually?

The mechanics of the Overmind gamemode is quite simple: The goal of the player(s) is to destroy the Overmind before he fulfills his mission - destroying (a) certain actor(s). In **Stronghold**, the Overmind had to destroy powercores, but it's up to you what you choose.



In the meantime, monsters are regularly spawned, following their routes. Each time the Overmind has lost 10% of his health, the next wave of enemies is starting and you will get another set of supplies, just as it is usual in other gamemodes.

First of all, in the very top part of **script 100** in the „**overmind_script.txt**“ you will find several new lines.

```
BossMonster = 18;
```

This is your Overmind. There is no need to use the predefined one from Stronghold, just make sure, it's powerful, huge and damn annoying.

```
BossTarget = 2;
```

The BossTarget is - how the name implies - the target or goal of the Overmind. This is exactly the thing he is going to attack. And for you, this means, as soon as all of these targets

tagged with this TID are destroyed, you have lost the game.

```
BossHealth = 30000;
```

With this command you can set a certain amount of health to the Overmind. The more, the better, the more challenging. But also keep in mind: The amount of health gets scaled depending on the amount of players and the skill setting so don't exaggerate this.

```
Thing_Hate(BossMonster,  
           BossTarget, 6);
```

This line was meant to be additional, it simply makes the Overmind ignore the player. It's pretty useful if you want the Overmind to stop losing time looking for the players and instead heading for his goals.

```
Thing_SetGoal  
(BossMonster, 3, 0, 1);
```

Same goes to this line. In case your map is more complex and the Overmind won't find his path to his goals on his own, make sure that he follows a patrol route. The second column/number (3) in this case is the first patrol route.

At the very end you will find the - yes, you guess it - obligatory gamemode script line

```
ACS_ExecuteWait  
(S_OVERMIND, 0, 0);
```

and that's all, at least in terms of the script. Concerning the map itself, put in the Overmind goals, put in the Overmind himself, add the regular spawnzones and routes and - if necessary - add a route for the Overmind, and you are ready to go.



Different Overmind

It's possible to create your own monster for the Overmind.



Waves?

The Overmind gamemode still has its regular „Wave“ setup. It's just that everytime the Overmind has lost 10% of his health, the next wave gets launched.

STRONGHOLD

EDITING REFERENCE

FINAL WORDS

So, after almost 16 pages of help, ideas, information and stuff, you finally made it and reached the last page of this editing Reference.

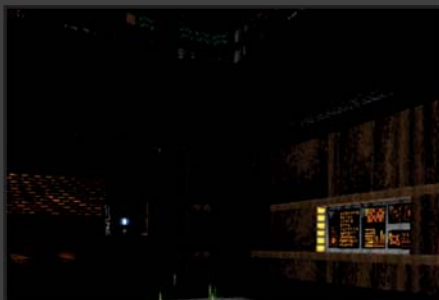
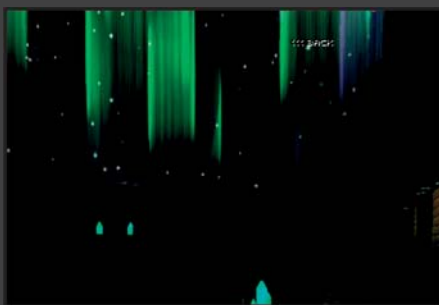
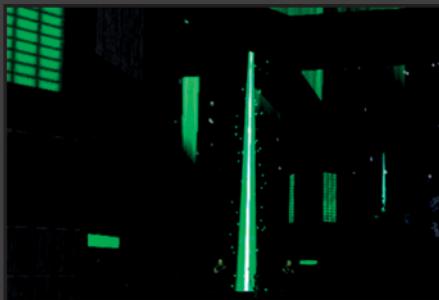
Hopefully, no questions are left to be answered and with a certain „learning by doing“ attitude you should be able to create standard **Stronghold** maps by your own without any big problems. In the end, it's quite easy and very entertaining as soon as you know about its twists.

Though, if you still encounter problems and if you still have questions, simply let us know. You can simply contact us in the official **Stronghold** forums over at DRDTeam.org but also on the Realm667.com forums.

In addition to that, we are eager to see the first custom maps for Stronghold, the first custom content, based on this brilliant script resources as we are very curious for what other people come up with.

Be creative, be productive,
we are looking forward to the results!

Cheers,
Tormmentor667



Editing Reference

Created by Tormmentor667 for
Realm667.com and DRDTeam.org