Dungeon Master™ WALLSET PACK

ASSET PACKAGE

– DM ITEMS, WALLSET –

FOR

LEGEND OF GRIMROCK

– USAGE MANUAL –

VERSION 3.0

MARCH 25, 2014

BY
RALF HINRICHSEN 'GERMANNY'

NOW WITH ALMOST ALL DM ITEMS!

Inquisitor and Diamond Edge

Ven Bomb

Ful Bomb

INTRODUCTION

This Dungeon Master wallset pack - short '***dmcsb_pack***' – is a modification resource for people who want to build a *Legend of Grimrock* custom dungeon.
The goal is to help recreating the old game Dungeon Master or its sequel Chaos strikes back with the Legend of Grimrock editor, but it can be used from anyone for any other idea.
At this time, the set is most complete! To rebuild a full DM, monster assets and massive Lua scripting are necessary. This set has most of the items and almost all dungeon assets you need included now.

I give most of my spare time for about five months to create the set. This was a big load of work!
Please notice that there was not enough time to script all objects well, alter for your needs.

LEGAL NOTES*

As far as i know, no assets, textures or models from other sources are used by me. Some script ideas may come from others, if i know of it, i will refer to them in the credits.
Don´t distribute or modify the graphical or model elements of this set under a new name or without my permission. The included scripts – if so – may used freely.
Please be respectful and reference me in your modification´s credits, if you use this pack in your custom dungeon.
**See last page for contact address and credits.*

WHAT DO YOU GET?

You get a wallset with all necessary assets to build a DM-dungeon with, there is no essential need to use other assets, all LoG original wall-assets have a dmcsb-replacement.

The difference to original LoG wallset is that it can be **pillar-less,** as Dungeon Master was.
There are two sorts of pillar assets included, a stone and a wood-pillar version.
Wall, ceiling and floor elements are seamless tiled.
DM item graphics and definitions are included.
There are special assets included, such as broken doors, broken walls, floor decorations, a fake wall, a secret wall, breakable blockages, all DM like buttons and more.

No Monsters! I wanted to build them, but there are some issues in blender export.

ANYTHING ELSE TO DO FOR YOU?

Item and wall object descriptions/stats are working examples, and are often not perfect scripted yet.
So you may modify the objects script entitys for your personal need!
Some assets need to be placed manually in editor, and ***due to seamless tiling you have to look after the orientation of manually placed floor/ceiling tiles*!**
For proper orientation and other tips, see next pages.

If you want to help with scripts or other additions, please contact me!*
**See last page for contact adress and credits.*

HAPPY MODDING AND HAVE FUN WITH IT.

PACKAGE STRUCTURE

All *dmcsb_pack* objects are in the folder **dmcsb_pack** within the mod_assets folder.
The folder structure within is similar to that from *Almost Human´s* original.
You don´t have to update the standard mod_assets scripts with the new entrys!

This pack includes a short example dungeon. The Dmcsb pack folder is embedded within.

INSTALLATION

Unpack the package **dmcsb_pack_ver1xx.7z** to your home.
(The Pack may have another name than mentioned above!)

If you want to check the example dungeon:
Copy the folder **dungeon_master_reloaded** out of the unpacked source with all content to

'*Your_Documents_folder*'\Almost Human\Legend of Grimrock\Dungeons

Start the game, choose Dungeon Editor and select **dungeon_master_reloaded.**
Ready to test!

If you want to use the pack into your dungeon:
Copy the folder **dmcsb_pack** out of the unpacked source, it is placed in the folder **dungeon_master_reloaded,** into the `mod_assets` folder from your mod:

'*Y_D_f*'\Almost Human\Legend of Grimrock\Dungeons\yourmod\mod_assets

Open the script **init.lua** from *your mods scripts* folder..

\yourmod\mod_assets\scripts\init.lua

..with a text editor and insert the line:

```
import "mod_assets/dmcsb_pack/scripts/dmcsb_pack_init.lua"
```

Start over the Grimrock Editor, choose *yourmod*, the new assets should appear in the list!
All dmcsb custom asset names start with **dm_** so you´ll find them easy. That´s it.

There are some wallsets defined:

Pillar-less wallset:	**dmcsb_wallset**
Pillar-less wallset and flat-ceiling:	**dmcsb_pl_flatceil**
With stone-pillars:	**dmcsb_pillars**
With stone-pillars and flat-ceiling:	**dmcsb_flatceil***
With wood-pillars:	**dmcsb_wpillars**
Mossy walls with wood-pillars:	**dmcsb_mwpillars**

**Note that the flat ceiling has no variations, but it works*

Please don´t use my other asset pack 'Germannys Dungeon Master and Custom Assets ' from Grimrock Nexus, all items from it are included and may have new names.

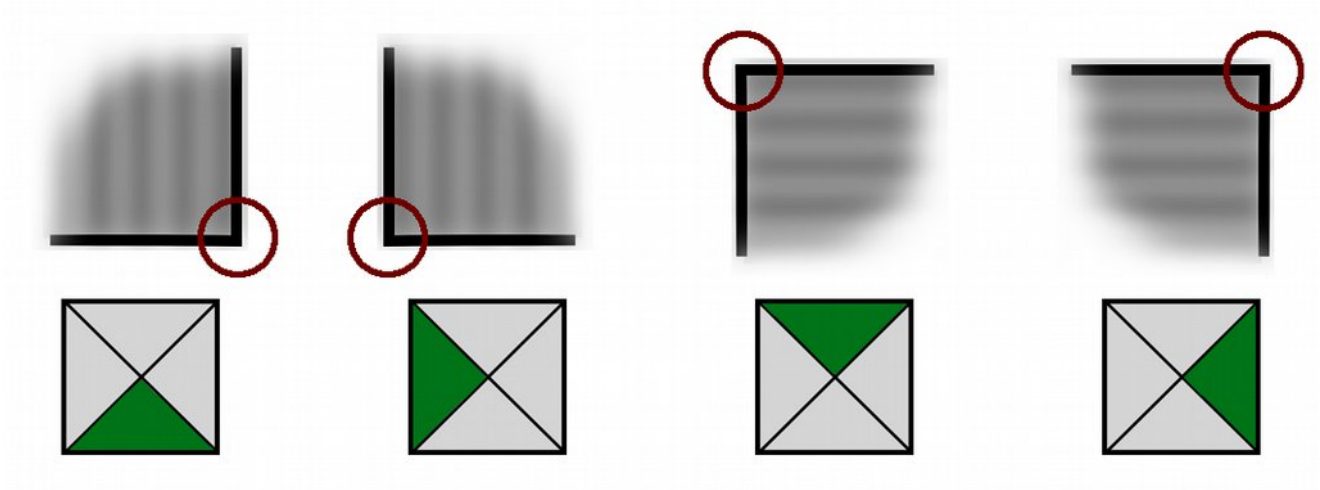
TIP SECTION

Special object orientations

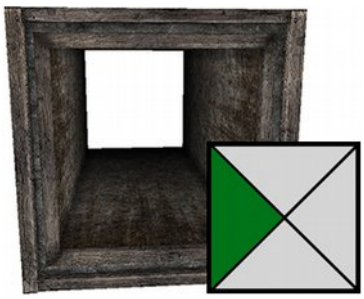
This pack use seamless tiled textures for floor, ceiling and walls.
Most is placed automatically correct, but some objects need extra investigation:

There is a special Pillar object to cover bad seams at the wall edges, place it as folllows:
(you don´t have to use it, optional)

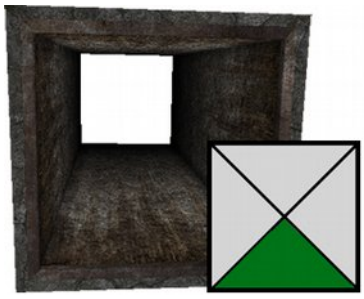
dm_pillar_walledge



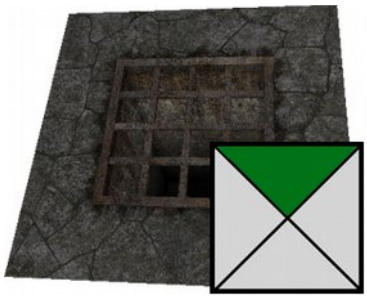
dm_ceiling_shaft



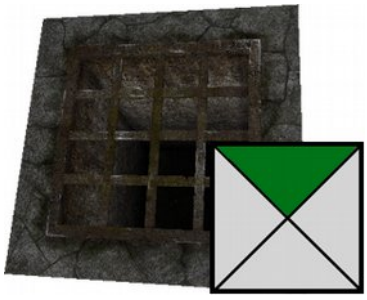
dm_floor_pit



dm_floor_drainage

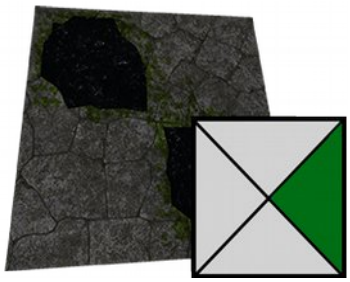


dm_floor_drain_big

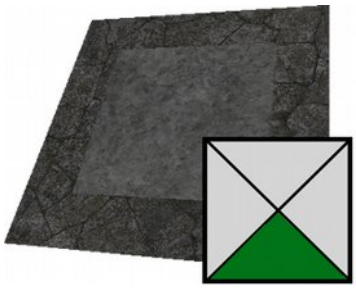


dm_ceiling_breakin

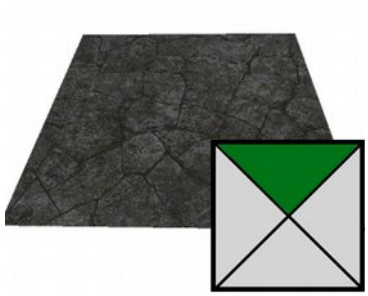
dm_floor_puddle



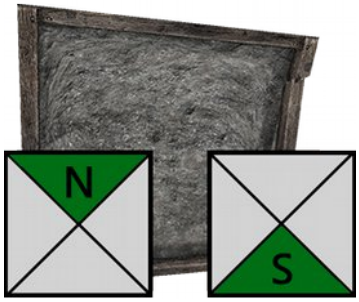
dm_pressplate_stone



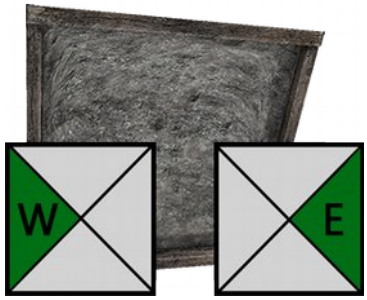
dm_pressure_plate



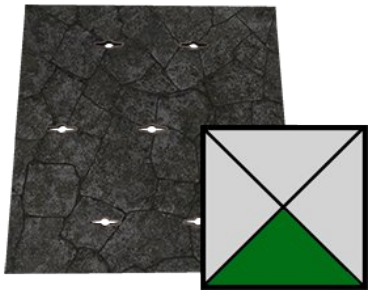
dm_ceiling_ns_cavein



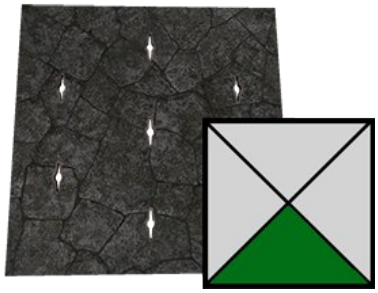
dm_ceiling_we_cavein



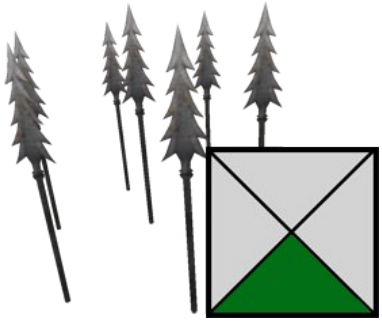
dm_floor_spiketrap



dm_floor_spiketrap_we



dm_spikes_floortrap and **dm_spikes_floortrap_we**



Placing a fake wall

Name: **dm_wall_illusion**

This wall-object is defined as torchholder.
To place the illusionary wall sucessful, let the block solid where it should be, no floor.
Then place the object at the desired side of the block.
Now change the solid block to floor – voila!

Placing destructible walls

Names: **dm_wall_breakable**
dm_wall_breakable_front
Names broken ones:
dm_wall_broken
dm_wall_broken_front

The first two objects are the destructible versions, one to place at tile center, the other at the front sides of a tile. The last two are the already broken walls that can placed as decorations, too.

Breakable walls can only destroyed by physical attack, no magic! Change this behaviour into *object.lua*, if you want.

The breakable walls are blockages and blocking the whole tile, orientate the front version so that the wall is directly in front of the direction the party come from.

OnDie() the breakable wall spawns a helper object **checker_broken** under the walls floor tile. You can use its helper object to check if the wall is broken.

Use destructible doors

Names: **dm_fakedoor_portk**
dm_fakedoor_wood
Names broken ones:
dm_fakedoor_portk_broken
dm_fakedoor_wood_broken
Names functional doors:
dm_door_portcullis_simple
dm_door_wood

There are three ways to use fake doors.

- the broken model only as decoration
- use the fake-doors as destroyable door only
- use the destroyable in combination with a real door

Example of using a fakedoor in combination with a real door

- place the blockage **dm_fakedoor_portk** at your desired location
- place a **dm_cross_lock** somewhere at a wall, set opened by **dm_crosskey**
- Create a counter, name it **fakDoor_counter**
- Create a script entity in the level, name it **irondoorChange**
  paste the script below into the script entity:

irondoorChange

```
-- script "irondoorChange"

function fakdoorRealIr(self)

local countNam = fakDoor_counter
local realDoorId = "dm_door_portk_sp1"
local fakDoorNm = "dm_fakedoor_portk"
local keyNam = "dm_crosskey"

-- Change the values below to your fake door location
for i in entitiesAt(2,7,19) do
          if i.name == "doorchecker_iron" then
           countNam:setValue(1)
          end
end

local doorXcnt = countNam:getValue()

     if doorXcnt == 0 then
           for j in entitiesAt(2,7,19) do
           if j.name == fakDoorNm then
           j:destroy()
           end
     end
     spawn("dm_door_portk", 2, 7, 19, 2, realDoorId)
           dm_door_portk_sp1:open()

     elseif doorXcnt == 1 then
           setMouseItem(spawn(keyNam))
           self:setOpenedBy("")


     else return false
     end

end
```

- connect the **dm_cross_lock** to the script entity **doorChange**

If a **dm_crosskey** is inserted into the lock and the fakedoor is not destroyed, the fakedoor will be replaced by a real door and then opened by script.
If the fakedoor is destroyed, it spawns *onDie()* a helper object **doorchecker_iron** below the floor. The script checks if it exists. If exists, a new key spawns, the party did not loose the key while trying to insert into the lock.
For the **dm_fakedoor_wood** the helper object is named **doorchecker_wood** .

These doors are designed for a pillar-less dungeon, there are other variants without pillars to use with pillared (wood or stone) dungeons! See listing.

dm_wall_lever

I´ve made a new lever that pops out the wall. Bad thing is, the click-point is a bit below, but i found no way to move it.
If you use the **dm_wall_lever**, position **dm_wall_leverwall** at the same place!

Special wall decorations

To cover transitions between wall and floor, i create dirt decoration objects, that can be placed at walls. They are only for beauty reasons and not necessary to use.

dm_wall_001dirt

dm_wall_plant

dm_wall_rocks

dm_wall_grass

dm_wall_mushroom

dm_wall_bigrocks

dm_wall_wshroom

It is a good idea to set them after you finished a level.

Blockages

There are some breakable blockage assets included.

dm_bigmushrooms

dm_bigmushrooms
If destroyed, some consumable mushroom slices are left to pick up.

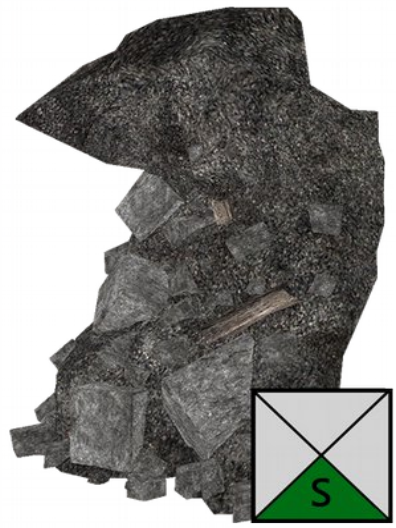
dm_bigmushrooms_broken

dm_timberstack
If destroyed, some pieces of logwood items can be picked.

dm_mushroom_slice

dm_spore_pods
If destroyed, a few pickable pod-dust items spawn.
! Be careful, the broken model covers a huge part of the floor, you may loose dropped items!
You can place **dm_deco_poddirt** under it - for better looking.

dm_cavein_breakable
A cavein asset similar to Grimrock original, but this one is breakable, you can dig your way through. When destroyed, some boulders and rocks are left on the floor.
The asset is scripted in that way you can only use item **'dm_pickaxe'** to damage it. Non-breakable asset already defined.
You need to place ceiling assets **dm_ceiling_ns_cavein** or **dm_ceiling_we_cavein** above, depends on the direction of cavein placement. (north-south, west-east)



dm_coffin_wood_01
Wooden coffin for floor placement, two breakable versions: Filled with remains and empty.
For viewing pleasure, place **dm_floordeco_sand** under the coffin.
A twin-coffin version is defined, too.

dm_coffin_recess

Wooden coffin for placement into the **dm_wall_recess** wall asset, two breakable versions: Filled with remains and empty. To make it destroyable at a wall, some tricks were necessary.

To place it functional, see picture ***dm_coffin_recess placement***.
After the coffin is destroyed, an invisible door asset is spawned at this position to prevent party to step behind the wall recess.

dm_coffin_recess placement

SPIKE TRAP PLACEMENT

I´ve created a simple floor spike-trap. There are two versions, one for north-south and one for east-west placement.

Two assets for each:
Floor replacement:
dm_floor_spiketrap
dm_floor_spiketrap_we
Spikes as pressure plates:
dm_spikes_floortrap
dm_spikes_floortrap_we

*I defined sound effects for the trap itself and for the party, they yell all if they are trapped^^*

Place the trap at your desired location. Then you need some scripts placed in your dungeon.
At first floor:
Script name: **TrapSetSnd**

```
-- Set connectors to all levers in dungeon

for m = 1, getMaxLevels() do
      for f in allEntities(m) do
         if  f.class == "PressurePlate" then
            f:addConnector('any', 'TrapSetSnd', 'playPplSnd')
         end
         end
end

-- Function to play custom sounds
function playPplSnd(self)

-- Set variables
local PpllLevl = self.level
local PplNam = self.name
local PplPosX = self.x
local PplPosY = self.y
```

```
-- play sounds for each state
if self:isDown() and PplNam == 'dm_spikes_floortrap' then
            playSoundAt("dm_spikeblade_up", PplLevl, PplPosX, PplPosY)

elseif self:isDown() and PplNam == 'dm_spikes_floortrap_we' then
            playSoundAt("dm_spikeblade_up", PplLevl, PplPosX, PplPosY)

elseif self:isUp() and PplNam == 'dm_spikes_floortrap' then
            playSoundAt("dm_spikeblade_down", PplLevl, PplPosX, PplPosY)

elseif self:isUp() and PplNam == 'dm_spikes_floortrap_we' then
            playSoundAt("dm_spikeblade_down", PplLevl, PplPosX, PplPosY)

end
end
```

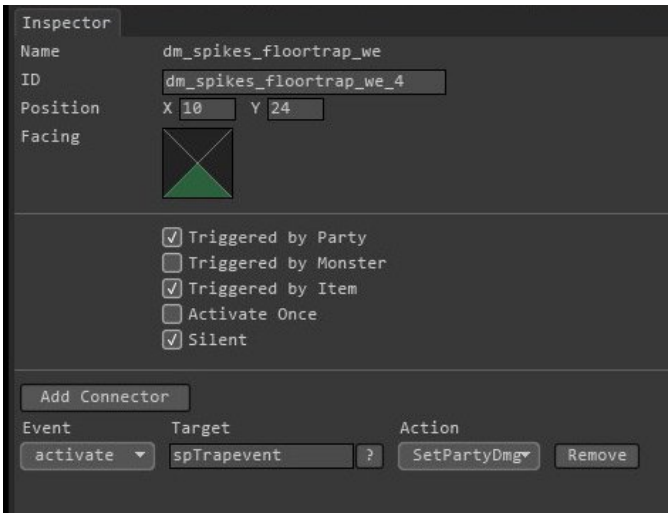
Next script is to damage the party and let them cry, place it near the trap(s) and connect it with **dm_spikes_floortrap** pressure plate. You can connect as many traps as you want to this script.

Script name: ***spTrapevent***

```
function SetPartyDmg(self)
if self:isDown() then
damageTile(self.level, self.x, self.y, self.facing, 6, "physical", 5)
playSoundAt("dm_party_pain", self.level, self.x, self.y)
else return false
end
end
```

Change the red value for damage amount!

dm_spikes_floortrap settings:



Important is to set it to '**silent**', because the sound for it is custom defined in first script here!

Try the example dungeon to see how it is done!

WHEEL LEVER, CONDUITS AND FIXING A GAS LEAK!

A new lever - **dm_lever_wheel** - is included in the pack. Can be placed as single lever on a wall or on the special assets for the **dm_wall_recess**, the **dm_deco_conduit_mid** or **dm_deco_conduit_mid_leak**, the last one is the version with leak that is used below.

To use **dm_lever_wheel** as an object you have to use at least one script to replace the sound effects:

leverSndSet

See description of leak setup below.
To get a better picture, test it in the example dungeon.

As example, a simple door open/close script:
Script name: **wheelDoorSw**

```
function openDoorFlip(self)

lvStateA = self:getLeverState()

if lvStateA == 'deactivated' then
  the_ID_of_your_door:open()
elseif lvStateA == 'activated' then
  the_ID_of_your_door:close()
end
end
```

Connect the **dm_lever_wheel** with this script, set connector:
Event: '**any**', Target: '**wheelDoorSw**', Action: '**openDoorFlip**'.
The Initial State of the lever is: '**Activated**'

Fixing a gas leak to operate the wheel lever!

To setup this action chain, more scripts and assets are necessary, was a hard task to figure this out^^

Assets used here:
dm_lever_wheel
- the lever
dm_wall_recess
- wall asset
dm_deco_conduit_mid_leak
- the tube placed in the recess behind lever
dm_conduit_leak_socket
- scripted socket placed at leak position
dm_textsign_conduit
- a textsign placed below the wheel (walltext)
dm_patch_conduit
- decoration; spawns while fixing over the leak
dm_conduit_fx_emitter
- burning gas stream emitter

dm_conduit_fx_light
- lightsource placed at gas stream
dm_deco_conduit_right and **dm_deco_conduit_left**
- decoration tubes; not needed
dm_metal_patch
- item needed for leak repair
dm_toolbox
- item needed for leak repair

dm_metal_patch

dm_toolbox

The idea:
- Party finds a tube with handwheel valve *(Example: operate an invisible engine via gas steam)*
- The valve is required to trigger next events *(Examples: open a door, dry a flooded level entrance..)*
- But the tube has a leakage; gas streams out and burns if the wheel valve is open *(activated)*
- Party must fix the leakage with a metal patch and tools *(they have to find the items of course)*
- For fixing, the handwheel valve has to be deactivated and metal_patch must attached to the tube.
- One click with the toolbox on the tube leakage while metal_patch is inserted will fix the tube.
- If handwheel is again activated, the burning gas steam & sound is gone and event is triggered.

The Scripts:

First Script is used to replace lever sounds; placed in dungeon at top level.

Script Name: **leverSndSet**

```
----begin ------------------------------------------------
-- Set connectors to all levers in dungeon

for l = 1, getMaxLevels() do
      for e in allEntities(l) do
         if  e.class == "Lever" then
            e:addConnector('any', 'leverSndSet', 'playLvSnd')
         end
         end
end

-- The function to play custom sounds
function playLvSnd(dmDungLev)

-- define and set variables
local lvState = dmDungLev:getLeverState()
local lvLevl = dmDungLev.level
local lvNam = dmDungLev.name
local lvFac = dmDungLev.facing
local lvPosX = dmDungLev.x
local lvPosY = dmDungLev.y

-- play sounds for each state
   if lvNam == 'dm_lever_wheel' and lvState == 'activated' then
      playSoundAt("leverwheel_open", lvLevl, lvPosX, lvPosY)

   elseif lvNam == 'dm_lever_wheel' and lvState == 'deactivated' then
      playSoundAt("leverwheel_close", lvLevl, lvPosX, lvPosY)

   end
end
```

You can add more sound replacements for other levers here.

Second Script: Place in dungeon for lever actions, add connector from **`dm_lever_wheel`**.
Initial state: **deactivated,** Action on connector: **any**

Script Name: **`steamChx`**

```
----begin ------------------------------------------------

    function togglActivity(self)

    local triggerCif = counter_2:getValue()
    local lvState = self:getLeverState()
    local lvLevl = self.level
    local lvNam = self.name
    local lvFac = self.facing
    local lvPosX = self.x
    local lvPosY = self.y
    local fxEmit = "dm_conduit_fx_emitter"
    local fxLight = "dm_conduit_fx_light"
    local fxEmId = "cond_fx_emit_1"
    local fxLiId = "cond_fx_light_1"
    local timerNam = timer_1

       if lvState == 'activated' and triggerCif == 0 then
          timerNam:activate()
             spawn(fxEmit, lvLevl, lvPosX, lvPosY, lvFac, fxEmId)
                spawn(fxLight, lvLevl, lvPosX, lvPosY, lvFac, fxLiId)
       elseif lvState == 'activated' and triggerCif > 0 then
          -- begin place further actions here!

          if dm_door_steamop:isClosed() then
             dm_door_steamop:open()
          end

          -- end place further actions here!
       elseif lvState == 'deactivated' then
             for i in entitiesAt(lvLevl,lvPosX,lvPosY) do
                 if i.id == fxEmId or i.id == fxLiId then
                   i:destroy()
                end
             end

          if timerNam:isActivated() then
             timerNam:deactivate()
          end

       end
    end
    ----end ------------------------------------------------
```

Third Script: Is found into the socket object definition **`dm_conduit_leak_socket`** inside dmcsb-pack ***objects.lua.*** You need this socket script in conjunction with these other, dungeon-placed script-entities.
It is required to place the socket!

Fourth Script: The sound script for looped non-loop sound; the burning gas stream.
Connected to '**timer_1**' , its settings:
Timer interval: **`1.356`**, Initial state: **`stopped`**
The timer interval is exactly the play length of the sound file!

Script Name: **`timedLoopSnd`**

```
----begin ------------------------------------------------
    function loopedSound(self)
     playSoundAt('dm_steamburn',dm_lever_wheel_1.level, dm_lever_wheel_1.x, dm_lever_wheel_1.y)
    end
    ----end ------------------------------------------------
```

For every new leak-fix setup, you have to set up a new function here for another lever!

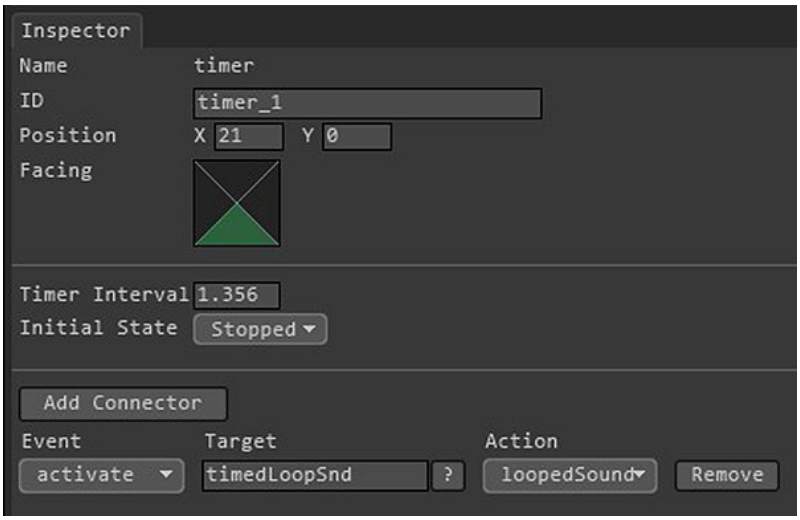
Replace the lever ID at red marked positions.
The loop-sound-timer:

Name: **`timer_1`**

Used to play a non-looped sound until the timer stops.
This has to be because looped sounds can not be destroyed!

Add connector to script
`timedLoopSnd`

timer interval: **`1.356`**
initial state: **`stopped`**



Please check the example dungeon to see how it works!

SHOOTING WALL TRAP SETUP

This is a simple trap, a wall with holes. You can set it up to shoot two arrows to a defined direction. I have pre-defined correct coordinates for the **`dm_wall_shooter`** asset, it will be easy for you to setup.

Used assets:
`dm_wall_shooter` (wall replacement)
`arrow` (the standard LoG arrow)
`dm_pressure_plate_hidden`
You can use the standard hidden plate as well..^^

Script Entity: **`arrowShotWall`**
You can set up four shooters here, each towards different directions.

```
-- shoots two arrows towards direction

function shootArrowWest()
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 3, 12, 0, 0, -1.5, 0.8266, 0.9051, 10, Nil, true)
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 3, 12, 0, 0, -1.5, 0.8266, -0.9051, 10, Nil, true)
end

function shootArrowEast()
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 1, 12, 0, 0, 1.5, 0.8266, 0.9051, 10, Nil, false)
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 1, 12, 0, 0, 1.5, 0.8266, -0.9051, 10, Nil, false)
end

function shootArrowNorth()
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 0, 12, 0, 0, 0.9051, 0.8266, 1.5, 10, Nil, false)
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 0, 12, 0, 0, -0.9051, 0.8266, 1.5, 10, Nil, false)
end

function shootArrowSouth()
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 2, 12, 0, 0, 0.9051, 0.8266, -1.5, 10, party, false)
shootProjectile("arrow", trapID.level, trapID.x, trapID.y, 2, 12, 0, 0, -0.9051, 0.8266, -1.5, 10, party, false)
end
```

Some **shootProjectile** Values that must/can be altered:
trapID is the **dm_wall_shooter** ID from the placed wall asset to use.
Blue value: Projectile speed
Green value: Attack power
The last two values define:
ignoreEntity – did not damage given entity, example= party
fragile – true= arrows are destroyed, false= arrows can picked up

Connect the pressure plate with the script, choose the script for direction, activate by party, ready!

WOOD PILLARS AND ADDITIONAL DECORATIONS

The wood pillars should be the new main pillars for this set. There are three versions now:
dm_pillar_wood_01 – standard
dm_pillar_wood_moss – with some moss on it
dm_pillar_wood_moss_full – moss over all, used in wallset
dmcsb_mwpillars

Decorations for wood-pillars

Five pillar decorations are available now:

- **dm_wpillar_chain**
- **dm_wpillar_deco_nailchain**
- **dm_wpillar_deco_ivy**
- **dm_spiderweb_wpillar_01**
- **dm_woodpillar_lantern**

dm_woodpillar_lantern, setup process

Place the lantern asset as needed, it is not defined as lightsource. You can place its lightsource objects manually or by script. In this way you are able to lighten or douse lanterns.
Use a light source at a pillar is a bit complicated, because of setting the light correct. Get an idea from lightsource placement images below.

Side effect of split model and light is a shadow that included the lantern model.

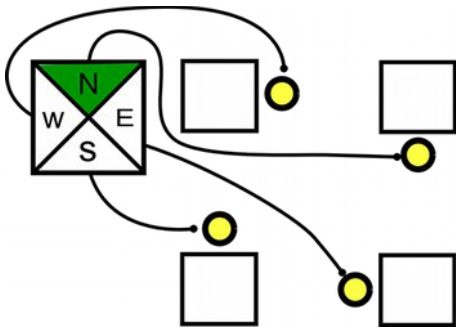
There are two light sources for the different positions around a pillar:

- **dm_light_lantern_ns**
- **dm_light_lantern_we**

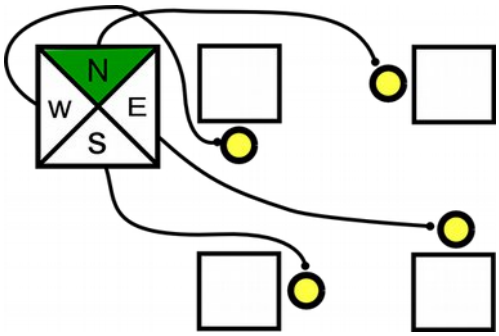
Placement is floor, not pillar.

Schematics:

dm_light_lantern_ns



dm_light_lantern_we



WALL OBJECTS

These are different decorations such as locks, torchholder, switches, button-walls or other (non)functional stuff. These are only some of them, much more included!

dm_wall_ivy_left

dm_winged_lock

dm_mirror_lock

dm_sconce

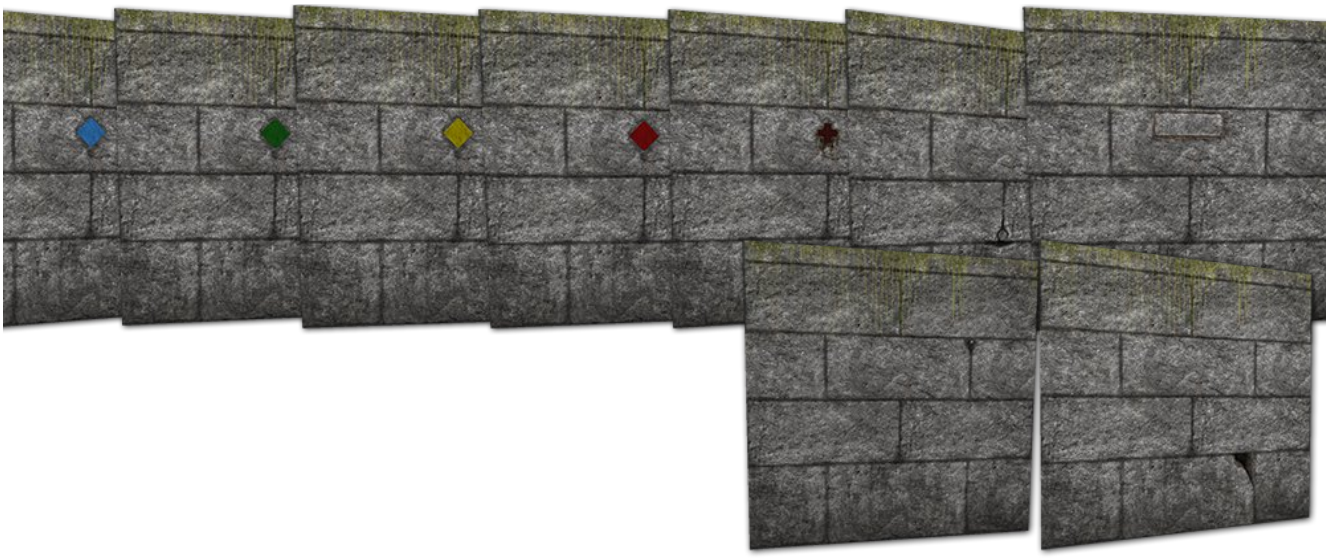
dm_deco_woodplanks

dm_wall_ring

dm_wall_ivy_right

BUTTON WALLS

Nine different Button-walls are available



DOORS

13 different door models are available. Not all are defined for woodpillars, but you can define new variants with other frames.

FOUNTAINS AND WATER

The fountain wall asset is functional, you can fill empty **flask** and the **dm_waterskin** with water here.

An example to let water flow into a dried wall fountain:

- Place a **dm_wall_fountain_dry** asset.
- Place a wall button or lever anywhere.
- Place a script entity near button, name it **repairFount**
- Connect the script to the button, check '*activate once*'

Script Entity: **repairFount**

```
function newFount()
  fountReplace = spawn("dm_wall_fountain", 1, 8, 29, 3)
    dm_wall_fountain_dry_ID:destroy()
      playSoundAt("dm_waterflowin", 1, 8, 29, 3)
end
```

Change red values for your dm_wall_fountain_dry_ID and location.

You can hear the water flow into the fountain; there is a item refill sound, too

ITEMS

Most of the original DM Items are recreated by me. Scripted values or abilities are not balanced or equal to the original Dungeon Master game. Please edit for your needs!

The Icon Atlas from V3.0:

The **dm_toolbox** item is defined as mortar, you can create new objects with it.
I defined two recipes for torches. They can build from **dm_bone_arm**, **dm_woodstick** and **dm_shredded_rags.**

dm_toolbox

CREDITS

I would like to thank ***Almost Human*** for Legend of Grimrock plus Editor, this is so much fun!
http://www.grimrock.net/

A big thankyou goes to Dr John Wordsworth for the Grimrock model toolkit!
http://www.johnwordsworth.com/grimrock-model-toolkit/

Many thanks to the community and anyone who released stuff!
http://www.grimrock.net/forum/

This pack uses the great portraits from Torsten von Nerdbot, hail him^^
http://angrymeteor.com/category/free-stuff/

A few images i used to create textures come from http://www.cgtextures.com/
The mushrooms, for example.

I have exchanged my water fountain script with the better version from
Pandafox, many thanks!

SOFTWARE USED

Blender 3D	*Thanks to the Blender foundation, you guys are great!*
Adobe Photoshop CS2	*Simply the best, but it cost too much (i am reg. user).*
Nvidia dds toolkit	*Plugin for Photoshop to maintain dds images*
LibreOffice	*The best free Office package out there (dump M$ office^^)*
TotalCommander	*Without Totalcmd life would be complicated ;)*
GMT	*Grimrock Model Toolkit – simply great, nothing goes without it*

CONTACT

If you want to contact me or place me in your credits, use this:

Ralf Hinrichsen 'germanny'
mail: webmaster@germanja.de

That´s it for now, i hope the set will help you in create a DM dungeon!
I want to add new stuff here that appears in the future.

Have fun and a good time,
Germanny