Dead Cells Mods documentation

Document purpose

This document explains in detail how the game data workflow works, how to manipulate its data, create scripts for it and upload mods to the Steam workshop. It's targeted to people who want to create mods and it explains everything from scratch, step by step. We'll make the difference between data modifications and scripting. If you already played with the pak and cdb there is probably a lot in here that you already know.

General presentation

There is a major difference between the flow for altering data and creating some scripts. The process of creating scripts for a mod is more straight forward but requires more technical skills.

Brief presentation of tech and production flows to modify data

The game uses a big file to store its data: the PAK. If you browse your game files, you can see it as "res.pak" at the root of the installation directory of Dead Cells. It embeds all files used by the game.

To modify any data in the game you need to be able to open and extract every file from this pak. It's one of the purposes of PAKTool.

With mods we introduced the possibility, for what we call a "mod pak", to override data present in the original pak. This mod pak will be loaded during runtime and each file it contains will override the original ones. The only exception to this "simple" flow is the CDB file, for more detailed information report to the CDBTool.

So, to create a mod, you have to extract files from the res.pak, modify the extracted files and then rebuild a mod pak. This mod pak will be a created with the PAKTool and contain only files you modified from the original res.pak (to be as light as possible).

To run the tools, you need to install (or already have installed the .NET Framework version 4.5.2 or above. You can find it on Microsoft site here : https://www.microsoft.com/fr-fr/download/details.aspx?id=42642

Brief presentation of scripting flow

The game now supports scripting for creating level structures, defining level info and mob roster in a level. The flow is very simple, when a mod is activated, it checks for the presence of a directory `./Script/Struct/` in the mod's directory and tries to load scripts present in it. It's compatible with some res.pak modificationts (you can change the res.pak and have scripts in the same mod).

The types of files

res.pak

As stated before, the res.pak is a container for all game files. It's in the installation root directory. It can be expanded using the PAKTool.

data.cdb

Data.cdb is a file containing a lot of different things concerning both gameplay and cosmetics. It's a big JSON file and can be edited with CastleDB. It contains rooms layouts, weapon parameters, fogs color etc...

.atlas & .png

In Dead Cells, all textures are PNGs and most of them are atlases. An atlas is multiple PNGs merged in one big PNG. In Dead Cells there are (few) standalone PNGs and .png/.atlas pairs. .atlas files contain information on where and how big a given image is stored in the .png. The AtlasTool is here to expand

atlases in several “original files” to allow you to modify them and then rebuild the atlas from expanded files.

.hx

They are script code files. They are loaded and interpreted by the game during different steps like: creating the structure of a level, setting level parameters, generating mob roster, spawning mobs etc.

How mods will behave in the game

The game supports multiple mods activated at a same time. In some cases, it will consider that two mods are incompatible with each other and prevent from activating the second one. In that case a message will warn the user on why the mod cannot be activated.

Two mods will be considered incompatible if one (or more) of this proposition is true:

- Both mods contain scripts
- Both mods modify the same file (except for the CDB)
- Both mods modify the same line of the CDB

A mod can only be activated when creating a new save or restarting a run from prison start.

For now, with each update (even minor ones) mods will be deactivated and current saves using mods will be reset (only the run, not meta items). This is to prevent the game from crashing and/or corrupt the saves.

What you can do with mods (current state)

This list is non-exhaustive

- Change some atlases (or part of atlases) (see list of unalterable atlases)
- Add floor junks
- Modify parameters in the CDB (except for table “Truelle”)
- Create script to define a customized level structure, world graph, mob roster

What you cannot do with mods (current state)

All this may change with future releases and the list is non-exhaustive.

For now you cannot modify some atlases:

- atlas/ui.atlas
- atlas/fxCommon.atlas
- atlas/ui.atlas
- atlas/fxCommon.atlas
- atlas/fxEnemy.atlas
- atlas/fxWeapon.atlas
- atlas/fxDisplace.atlas
- atlas/beheaded.atlas
- atlas/gameElements.atlas
- atlas/lore.atlas

For caching reasons, we cannot reload those during runtime and modifying them will result in a no-op

You cannot modify the table “Truelle” in data.cdb. It will be a no-op too.

Some parameters in the CDB may have no effect when changed, but there is no exhaustive list at this point.

Removing lines of separators in the CDB may cause the game to crash

Removing entries from atlases may cause the game to crash or graphic bugs (depending on which atlase is concerned)

Generally removing files or part of files will cause the game to crash at one point or another.

Language should be added or altered via the existing process: https://steamcommunity.com/games/588650/announcements/detail/1253537422578152950

Tools list

Except for castleDB, the tools presented are command line tools.

They accept - and / at the start of arguments. Argument names are NOT case sensitive.

Order of arguments is not important (as long as you respect the order -argument <parameter>):

Example:

```
-EXPAND -OUTDIR "C:\Tests\ExpandedAtlas\kingsGuardian\" -ATLAS
"C:\Tests\Expanded\atlas\kingsGuardian.atlas"
```

is equivalent to

```
-ATLAS "C:\Tests\Expanded\atlas\kingsGuardian.atlas" -OUTDIR "C:\Tests\ExpandedAtlas\kingsGuardian\" -
EXPAND
```

To run the tools you need to install (or already have installed) the .NET Framework version 4.5.2 or above. You can find it on Microsoft site here: https://www.microsoft.com/fr-fr/download/details.aspx?id=42642.

PAKTool

What is it for?

The pak tool is used to manipulate the .pak files. With it you can:

- Unpack all file data it contains into a directory, so you can visualize them and alter them
- Repack a full directory tree into a pak
- Create a diff pak against a reference pak containing only the files you changed and the lines of the CDB you altered. (creates a mod pak)

How do I use it?

```
-?: Display this help

-Expand -outdir <output directory> -refpak <input pak path> [-s]: Expands a given PAK to a file tree

-Collapse -indir <input directory> -outpak <output pak path> [-s]: Collapse a given file tree to a pak

-CreateDiffPak -refpak <input pak path> -indir <input directory> -outPak <output pak path> [-s]: Create
a pak from a directory with only what has changed or been added from the ref pak (typically for mods)
```

arguments:

```
-s/-silent: Do not display message error (deactivated by default)
```

AtlasTool

Be aware that Atlas tool is a 64bits executable and will not run on 32bits systems. Also, commands ExpandAll and CollapseAll may consume a few GB of RAM (up to 3.5GB)

What is it for?

This tool is used to:

- create an atlas from a directory tree containing PNGs
- create a directory tree with all PNGs from an atlas

How do I use it?

```
-?: Display this help

-Expand -outdir <output directory> -Atlas <input atlas path> [-s]: Expands a given Atlas to a file tree

-ExpandAll -indir <input atlases directory> -outdir <output directory> [-s]: Expands every atlas found
in indir into outdir

-Collapse -indir <input directory> -Atlas <output atlas path> [-s][-ascii]: Collapse a given file tree
to an atlas

-CollapseAll -indir <input directories> -outdir <output atlases path> [-s][-ascii]: Collapse every
directory in the input directory into atlases
```

arguments:

```
-s/-silent: Do not display message error (deactivated by default)

-ascii: Export atlases as ascii (binary by default)
```

Example:

```
AtlasTool -EXPAND -OUTDIR "C:\Tests\ExpandedAtlas\kingsGuardian\" -ATLAS
"C:\Tests\Expanded\atlas\kingsGuardian.atlas"
```

CDBTool

What is it for?

It's used to manipulate the CDB. You can:

- Create a directory tree where every table is a directory and every line in this table is a json file in the directory. It will also create files for the structure and property of the table.
- Create a CDB from a directory tree where every table is a directory and every line in this table is a json file in the tree.
- Create a directory tree with only altered and added lines/tables from a reference CDB and a "expanded" CDB directory tree.

Unless you want to edit the CDB with notepad line by line (which could be a valid option) or just want to browse it more easily than in CastleDB. You will probably won't need it.

How do I use it?

```
-EXPAND -OUTDIR C:\Tests\ExpandedCDB -REFCDB C:\Tests\PAKTool\Expanded\data.cdb"

-CREATEDIFFCDB "-INDIR" "C:\Tests\ExpandedCDB" -OUTDIR "C:\Tests\Diff" -REFCDB
"<SteamInstallPath>\steamapps\common\Dead Cells\res.pak"

-COLLAPSE -INDIR " C:\Tests\ExpandedCDB" -OUTCDB "C:\Tests\Expanded\data.cdb"
```

ScriptTool

What is it for?

It's used to create a bootstrap script file to start using script in a mod. It also includes a lot of comments and direction on how to use different structures and functions.

If you launch the tool without arguments, it can also be used to visualize and help create script code you can copy and paste with an UI but it's very basic at this moment.

How do I use it?

```
-CommandLine -NewFile -OutScript <output file name> [-s] : create a script file with default function and basic doc inside.
```

arguments:

```
-s/-silent: Do not display message error (deactivated by default)
```

Example:

```
ScriptTool.exe -commandline -newfile -outscript "C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340\Scripts\Struct\test.hx"
```

Be careful that it will overwrite any existing file without warning.

RoomEditor

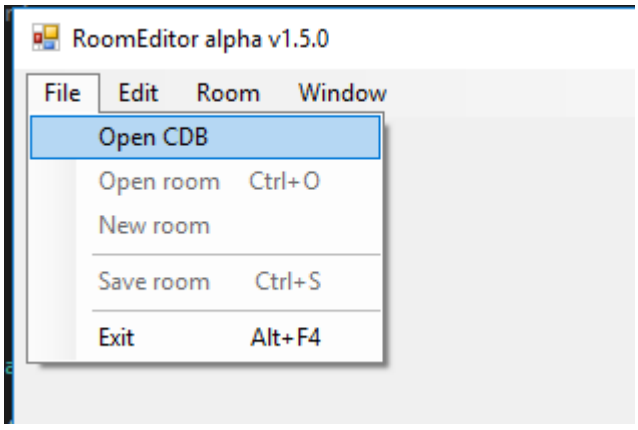
What is it for ?

It's used to create new rooms and edit existing ones. It's supposed to replace the CastleDB embedded room editor that can be difficult to use and have some performance problems in some cases.

How do I use it ?

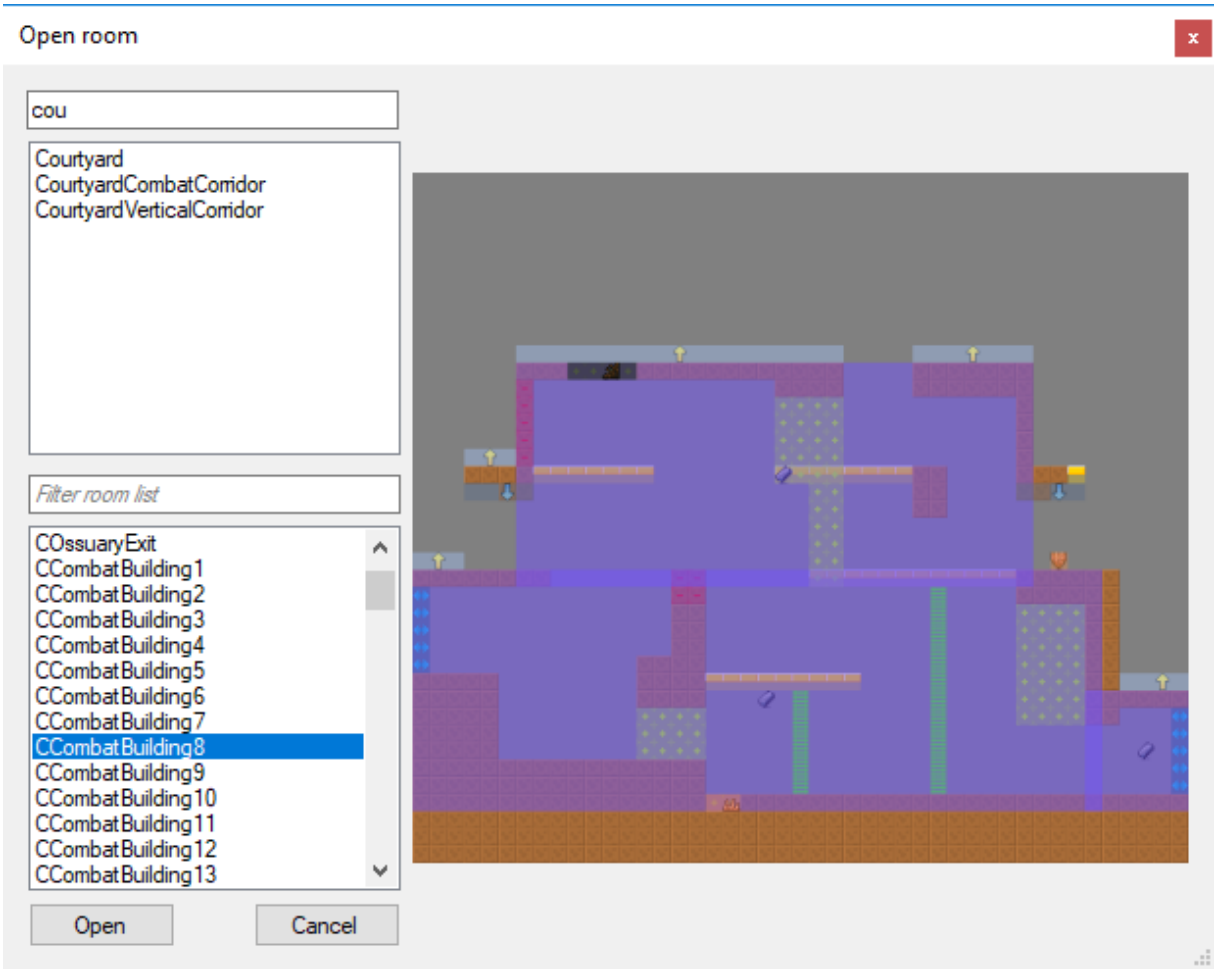
Starting

The first thing to do is to open the CDB that contains the rooms you want to edit :

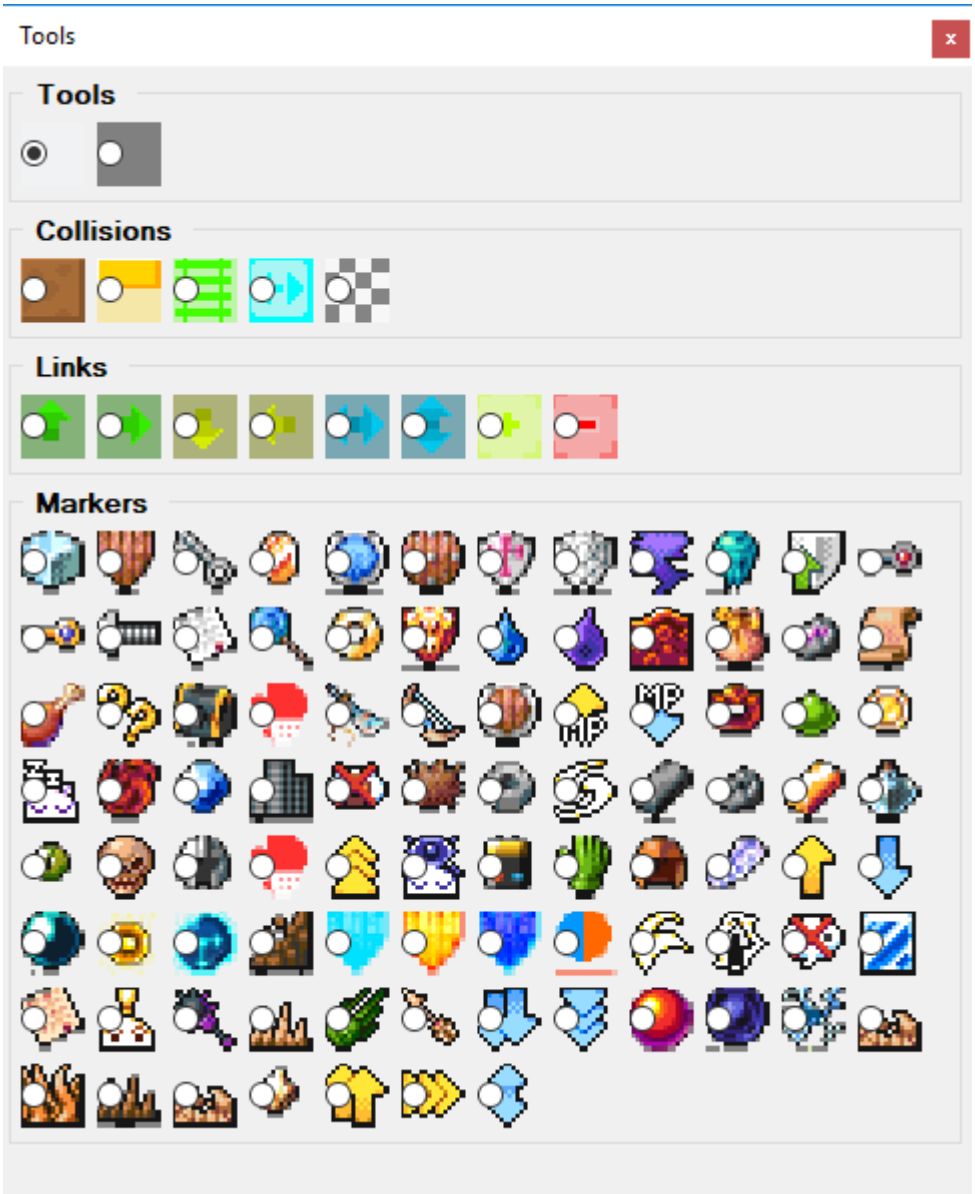


Opening a room

When you do so, it automatically opens the “Open Room” dialog for you to choose the room you want to edit. If you want to create a new one, simply cancel et and create a new room. The first field is to filter the first list containing the room groups. The second field is to filter the room list that are in the filtered (or selected) group in the first list. When you select a room, a preview appears on the right. If you double click on a room or select one and click on the OK button, the room is opened in the main window.



Tools form



It's the form where you can select your current tool. By default, it's the selection tool that is active. All tools are grouped by layer type : Collisions, Links, Markers. General tools (selection/eraser) are in their one group (Tools).

It's showed by default, if closed it can be brought back by pressing F5 on the main window or by selecting window -> Tools

All tools can be used by using rectangles by clicking, holding and dragging the mouse to place, select or erase a whole rectangle.

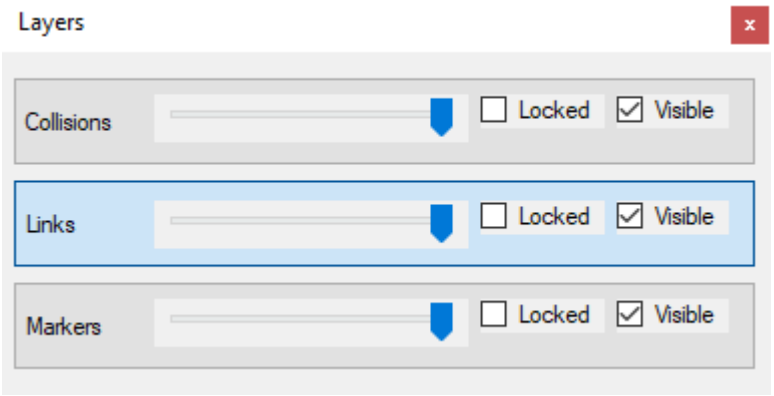
Selection and eraser are only active on the current layer that you can change in the layer form.

A tool tip appears on each tool to show their name when hovering a tool button.

You can also use a picking tool and bucket tool : By holding MAJ key, you can use the bucket for the current tool selected (if the bucket can be applied to the current tool). It will fill all empty tiles with the current tools (Applies for collisions and links tools). By holding Ctrl key, you can pick the tool used to

place the tile you are one. If multiple tiles are overlap each other, clicking multiple times will cycle through the different overlapping tiles.

Layers form



The layer form is where you can select the active layer. By default, it's the collision layer that is active.

It can be shown by pressing F7 on the main window or by selecting window -> Layers.

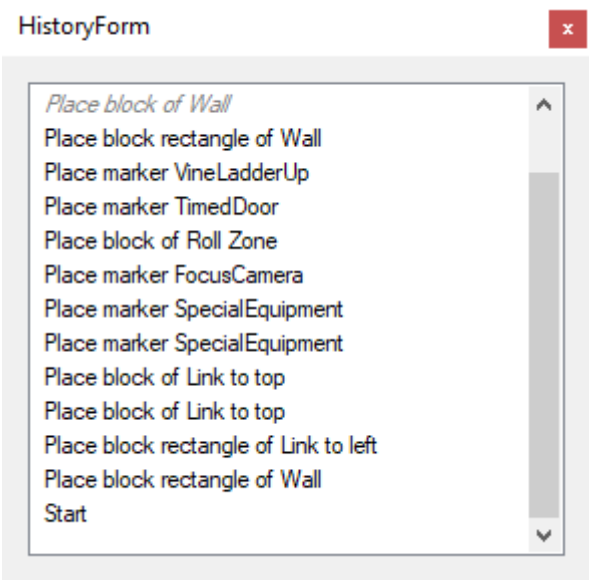
The current layer can change automatically if :

- you select a tile that is empty on the current layer but not on another layer
- you select a tool that is meant to be used on another layer

Else you'll have to select manually the layer you want to work on.

Each layer's alpha can be changed between 0% and 100% with the slider (0 on the left, 100 on the right). You can also lock a layer, meaning no change will occur on it (no selection, block placement, suppression…). You can also completely hide a layer with the "Visible" checkbox. Be careful that a invisible layer is not automatically locked and that you can potentially modify an invisible layer.

History form



The history form is where you can check and navigate through all action you did since the room edition beginning.

It can be shown by pressing F8 on the main window or by selecting window -> history.

If you undo something, the undone action will appear in light gray and italic.

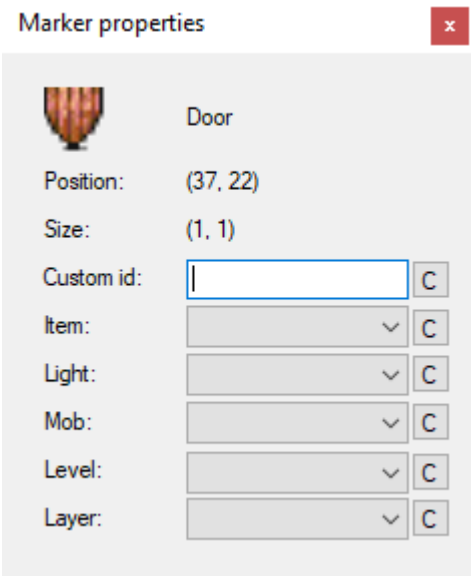
You can navigate through the history by double clicking on an entry: All action from bottom to the double-clicked item will be redone if necessary, all action above the double-clicked item will be undone if necessary. Be aware that all undone items will be lost as soon as you do a new action.

Preview form

It's the form where you can have a preview of the whole room whatever the size of it and the current level of zoom you are using. A black rectangle will materialize the area you are currently viewing in the main form.

It can be shown by pressing F6 on the main window or by selecting window -> Preview

Marker properties form



It's the form where you can edit the selected marker.

It will automatically appear when you select a marker. It can also be shown by pressing F9 on the main window or by selecting window -> Marker Properties.

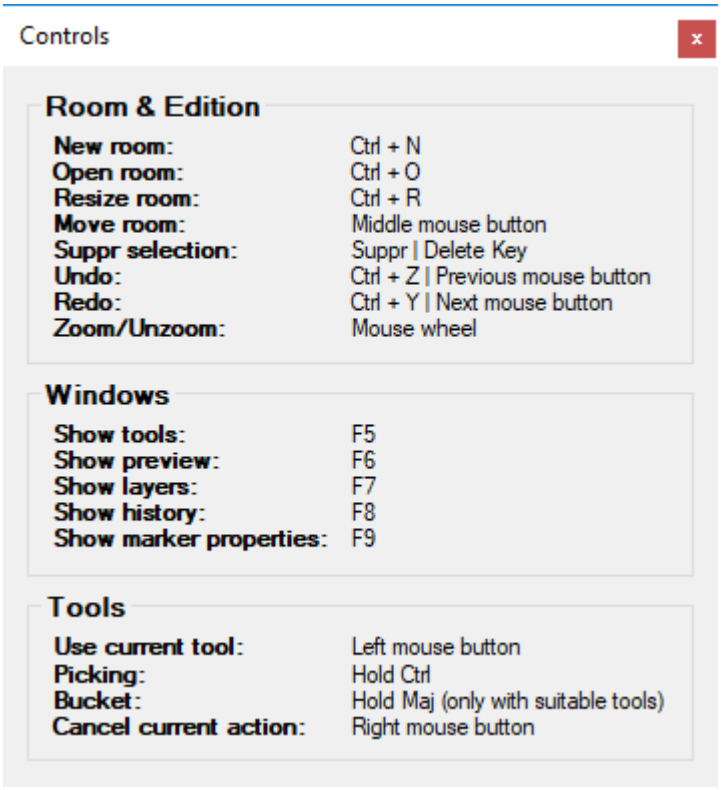
If you have no selection or a multiple selection, the properties will no longer be displayed and a message will say why (No selection, multiple selection…).

The custom id is freely editable, other properties must be selected in a list of predefined values. You can reset each field by clicking on the "C" button on the same line.

Main form

It's the form where you can edit the room. You can visualize your current zoom level and reset it with the controls in the top right corner of it.

Here is the list of available controls on the main form :

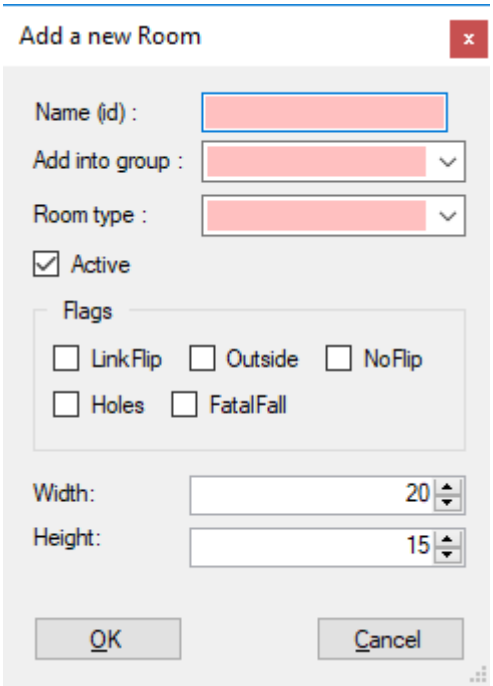


(| characters means “or”)

You can show this reminder anytime by selecting window -> Controls on the main form.

Create a new room

You can create a new room by selecting File -> New Room in the main form. When you do the following form appears :



The name id is free, but it should be unique, the background will be red if the name is not a valid one.

The group and room type are to pick from a predefined list. You can either select it or start typing and suggestions will appear. The backgroupd will be red if they are not valid.

Choose the flags of the room and the Width/Height, click OK and you can edit your newly created room as explained above.

Remember to save the room !

CastleDB

What is it for?

Edit the cdb file and edit rooms

Where can I find it?

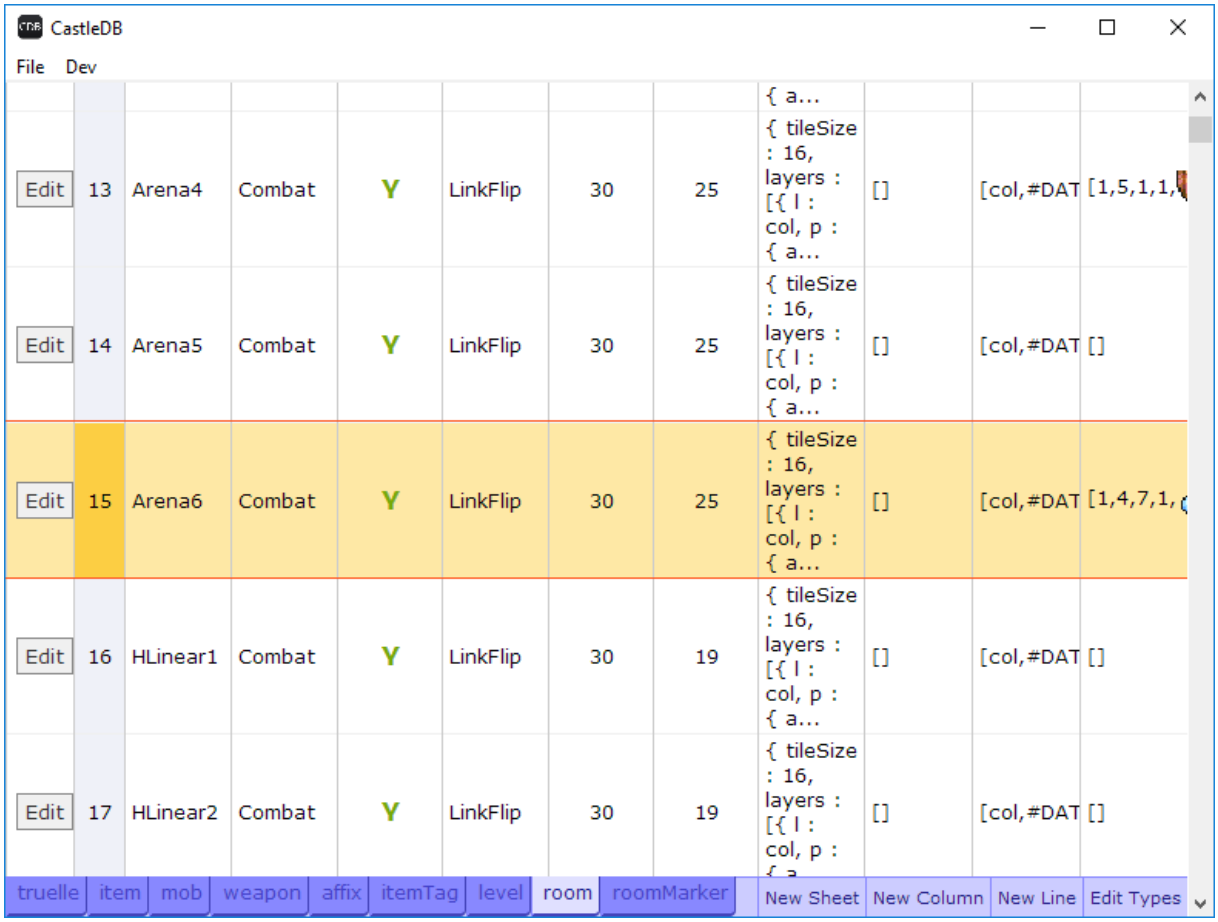
http://castledb.org/

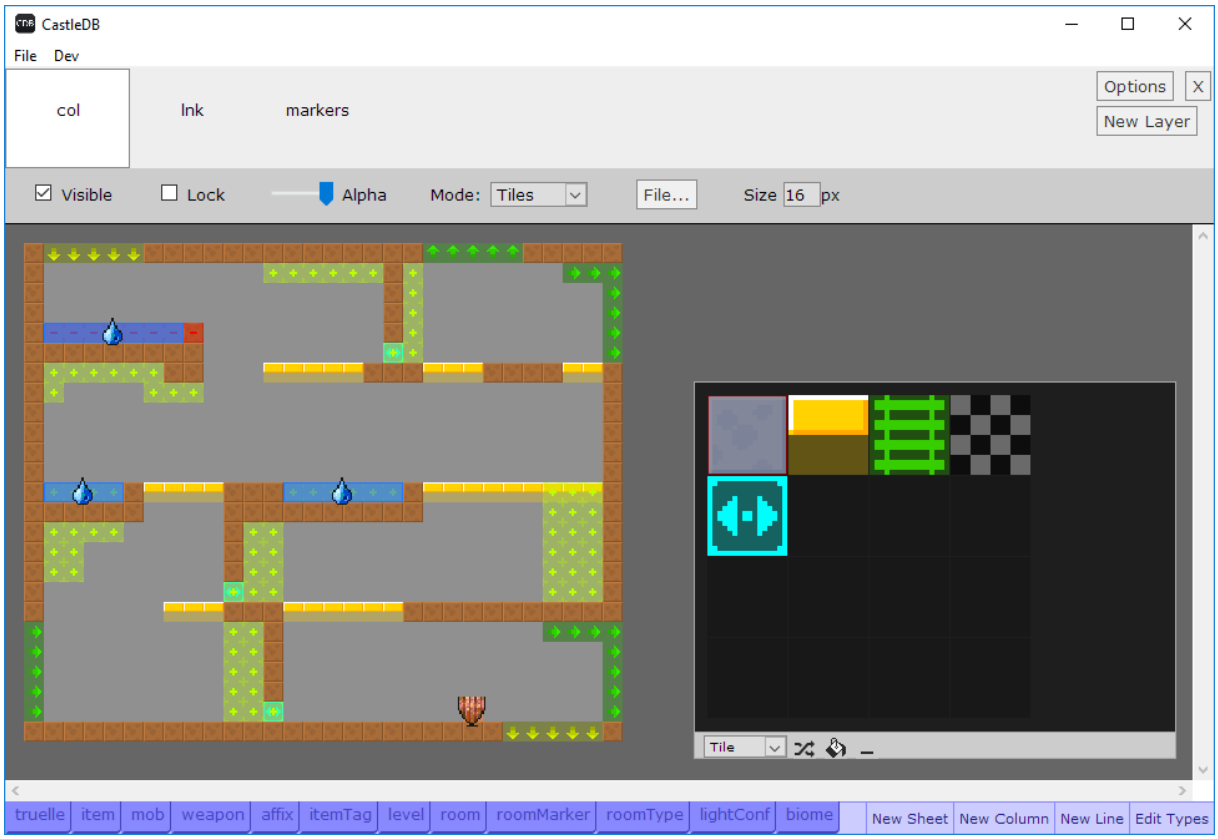
How do I use it?

1) Download the .zip at http://castledb.org/
2) Replace the castle.js in the castled you unzipped by the one found in `<Steam installation Dir>\steamapps\common\Dead Cells\ModTools\CastleDB (C:\Program Files (x86)\Steam\steamapps\common\Dead Cells\ModTools\CastleDB by default)`

You can edit /add rooms using the embedded editor :

From version 1.5.0 you can now use the room editor

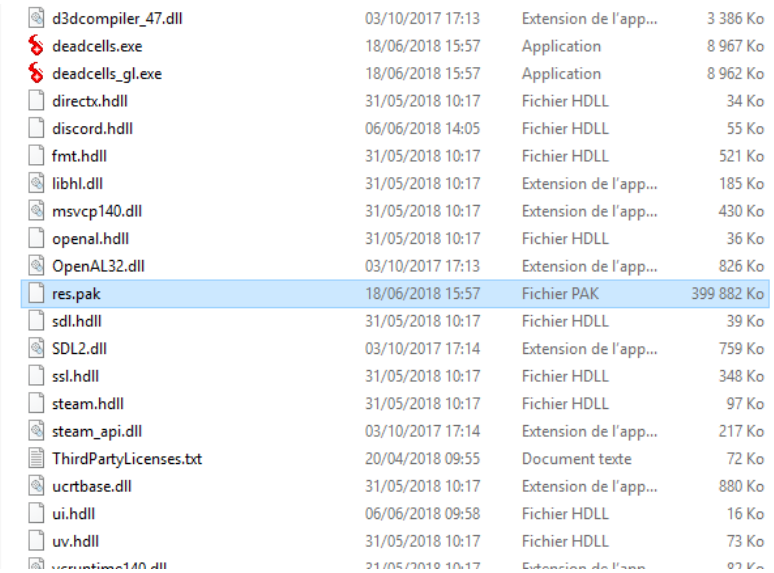




Full Process

Unpakking data

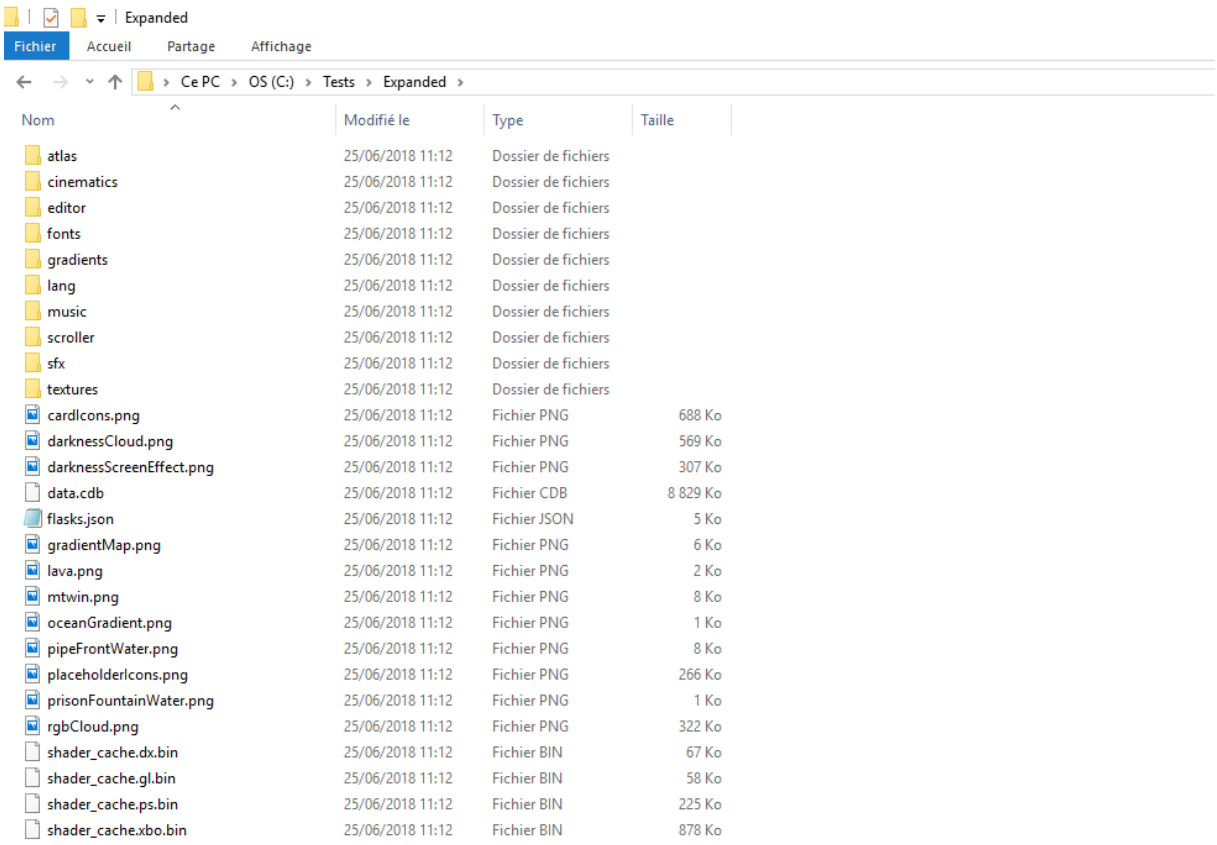
First, we have to locate the original res.pak used by the game. You can find it in the installation directory root: <Steam installation directory>\steamapps\common\Dead Cells (C:\Program Files (x86)\Steam\steamapps\common\Dead Cells by default).



Open a new command line window. Move current directory to where your PALTool.exe is located. Extract the res.pak you found previously in an empty directory (you must have writing rights in this directory) by typing the following:

```
PAKTool -Expand -OutDir C:\Tests\Expanded -RefPak "C:\Program Files (x86)\Steam\steamapps\common\Dead Cells\res.pak"
```

Once executed, we have our directory filled with the data files contained in the res.pak :



Modifying data

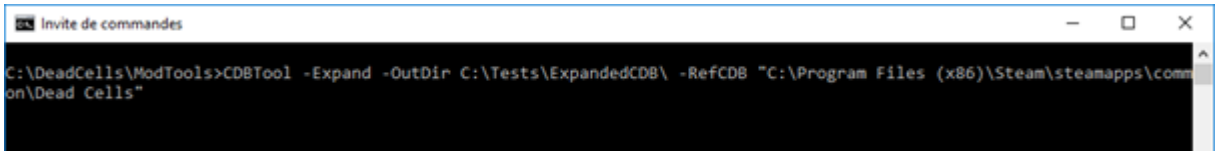
The CDB

You can now see the CDB file, called data.cdb at the root of the expanded directory. You can modify it either by breaking it down with the CDBTool or with CastleDB.

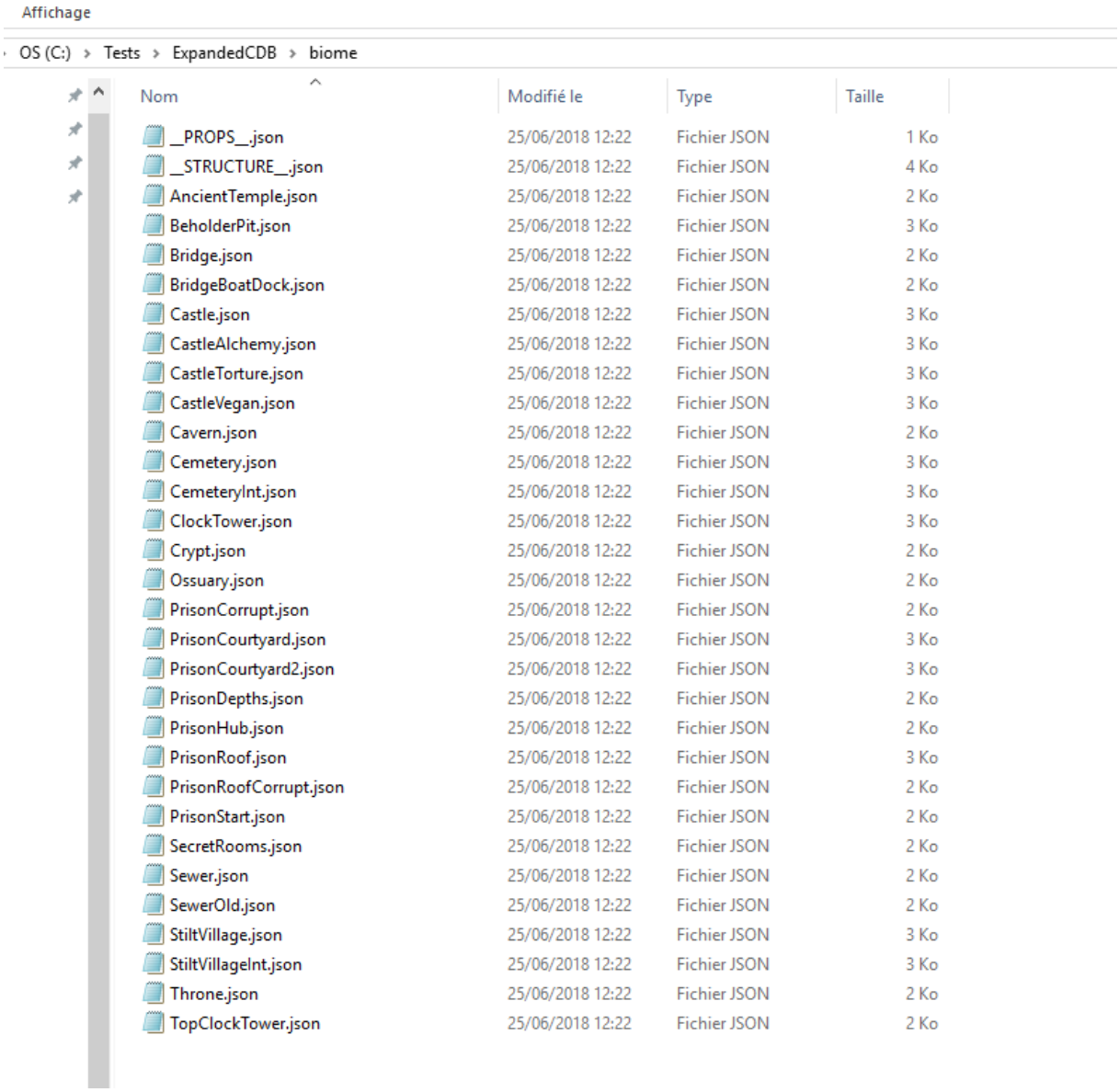
Breaking down the CDB

You Can break down the CDB into a directory tree containing json files for each line with the CDBTool. Here, we break down the one we extracted previously from the res.pak.

```
CDBTool -Expand -OutDir C:\Tests\ExpandedCDB\ -RefCDB " C:\Tests\Expanded\data.cdb"
```



After that a directory tree is created where each table in the data.cdb is a directory and each line in each table is a json file. Also special files __PROPS__.json and __STRUCTURE__.json are created for each table. It's strongly advised NOT to modify them as you may break the game or render your mod incompatible with the game/other mods very easily by doing so.

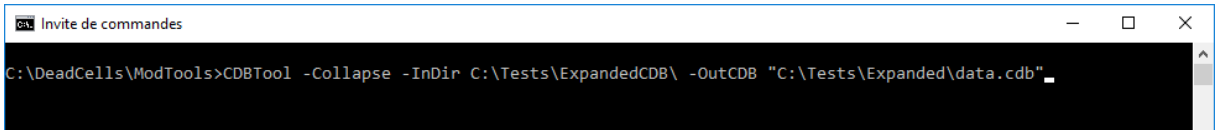


Rebuilding the CDB

After altering or adding files to this directory tree, you'll have to rebuild the CDB and put it back with the other files you are editing before creating the final mod.

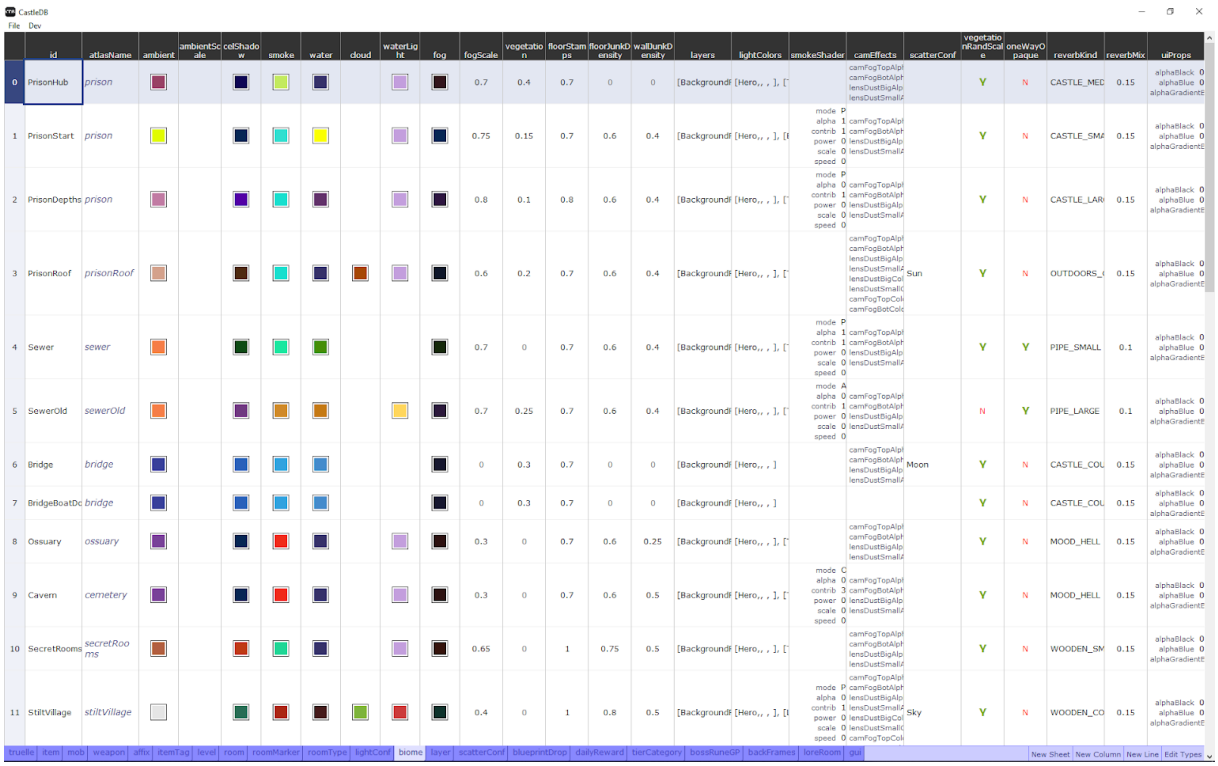
To do so you have to call CDBTool with some other arguments:

```
CDBTool -Collapse -InDir C:\Tests\ExpandedCDB\ -OutCDB "C:\Tests\Expanded\data.cdb"
```



Editing the CDB with CastleDB (solution not working atm)

You can edit in place your data.cdb using CastleDB.



Atlases

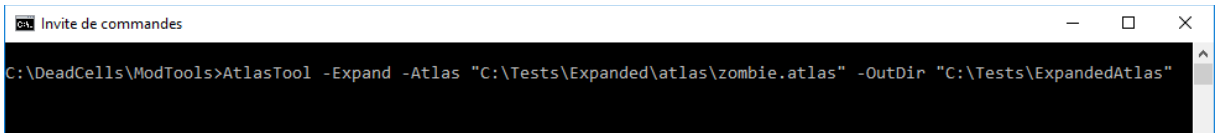
As we saw before, most of Dead Cells graphic data are packed in atlases. To modify them you can either modify directly PNGs or use the AtlasTool to break down atlases and build them back. The first solution surely demands less manipulation but can be tricky if you want, for example, change only one animation on a mob as everything is shuffled.

When breaking down an atlas into separate files, there is the advantage that every file created has a meaningful name, allowing you to identify easily what you are looking, and also grouping animation frames as they have the same name with a sequential number.

Let's work with an example: the base zombie, let's say you want to change its walk anim. All you have to do is:

Break down the atlas:

```
AtlasTool -Expand -Atlas "C:\Tests\Expanded\atlas\zombie.atlas" -OutDir "C:\Tests\ExpandedAtlas"
```



You can now easily identify frames of the walk animation:

For each frame there is actually 2 files: one representing the “diffuse” including “purple” zones which are interpreted as “glowing” in the game shader, one “normal map” used to give some “bumps” to models.

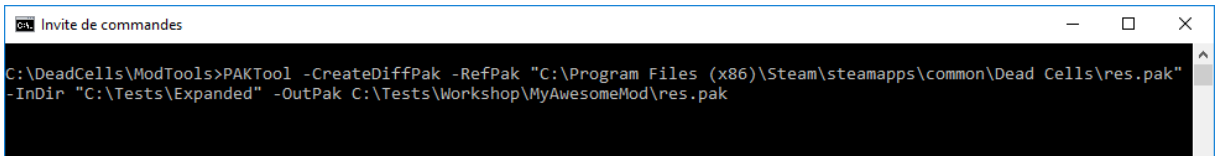
Be careful, you can add or alter files, it will impact the looks and timing of animations, but some animations (like attacks) rely on a given time. Also, you can see all files are ending with -=-xx-=- where xx is a number. Please do not remove or change them, they are addition to allow the atlastool to breakdown animation when every frame is named the same way.

Repakking modified data

Once you did every modification you want to, it’s time to build a mod res.pak so you can test it inside the game and eventually upload it to steam.

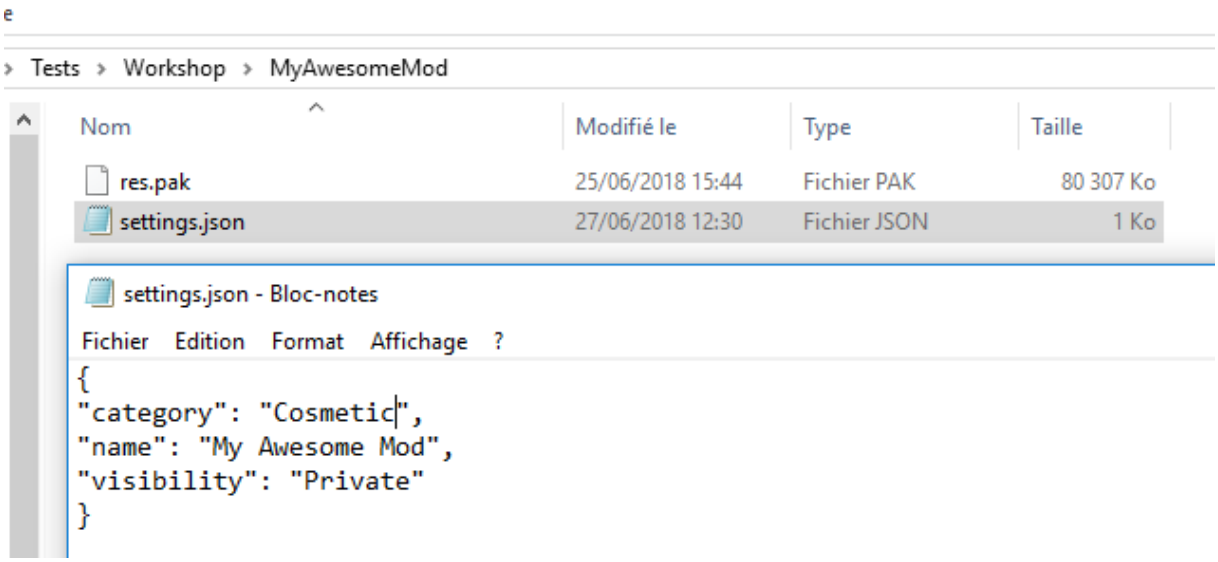
To do so you have to use the PAKTool again and create a diff pak from the original pak :

```
PAKTool -CreateDiffPak -RefPak "C:\Program Files (x86)\Steam\steamapps\common\Dead Cells\res.pak" -
InDir "C:\Tests\Expanded" -OutPak "C:\Program Files
(x86)\Steam\steamapps\workshop\content\588650\1405810340\res.pak"
```



Before you do that, don't forget to rebuild your data.cdb from your working directory if you work with a broken down cdb.

Now create a json file next to the res.pak created by the last step called settings.json and containing the following fields :



You can also add a preview.jpg or .png (max 1024x1024) to use as preview/thumbnail of your mod (like it's done for languages)

Be careful, the name cannot be changed once you uploaded the mod. The visibility is optional (visible to all by default) and can take the values Private, Public, Friends (case is not important) which correspond to "visible only to you", "visible to all" and "visible to you and your friends".

The available categories for now are "Gameplay", "Cosmetic", "Test".

At this moment there is no restriction to the category name other than it should not be "Language" as it is reserved for localization mods.

Creating some scripts

We'll use the dummy mod to create our first script. It will only create a very simple level with an entrance, a combat room and an exit, with a mob roster taken from an existing level and parameters from the same existing level.

You will then be able to download some more advanced scripting tutorials via mods.

1) Install the dummy mod (points 1 and 2).
2) Execute the script tool with the following arguments in a command line window :
   `ScriptTool.exe -commandline -newfile -outscript "C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340\Scripts\Struct\test.hx"`
3) Rename the test.hx in main.hx in
   `C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340\Scripts\Struct\`

This step can be avoided if you export a new file directly as main.hx, but beware that when doing so, if a modified main.hx already exists, it will be replaced without warning.

4) (optional) Launch the game and create a new run, you can see it uses a default struct only composed of one random entrance, a random combat room and a random exit that leads back to the start of the level.
5) You can now open the main.hx with your favorite text editor and see for yourself the code (and comments) used for this.

You will find some technical documentation on the script API in the `Scripts/TechnicalDocumentation` directory inside your ModTools directory (present in the install directory of Dead Cells). Or on https://dead-cells.com/mods

You can also find some samples on https://github.com/motion-twin/deadcells_mods_samples

Testing, creating and uploading the mod on Steam

Testing

You can test your mod locally on your machine without having to upload it to the internet each time. To do this:

1. Go to https://steamcommunity.com/sharedfiles/filedetails/?id=1405810340 and download the dummyMod

2. Navigate to <Steam installation Directory> \steamapps\workshop\content\588650\1405810340 (by default C:\Program Files (x86)\Steam\steamapps\workshop\content\588650\1405810340)

3. Replace the res.pak in the directory by your own mod pak you created

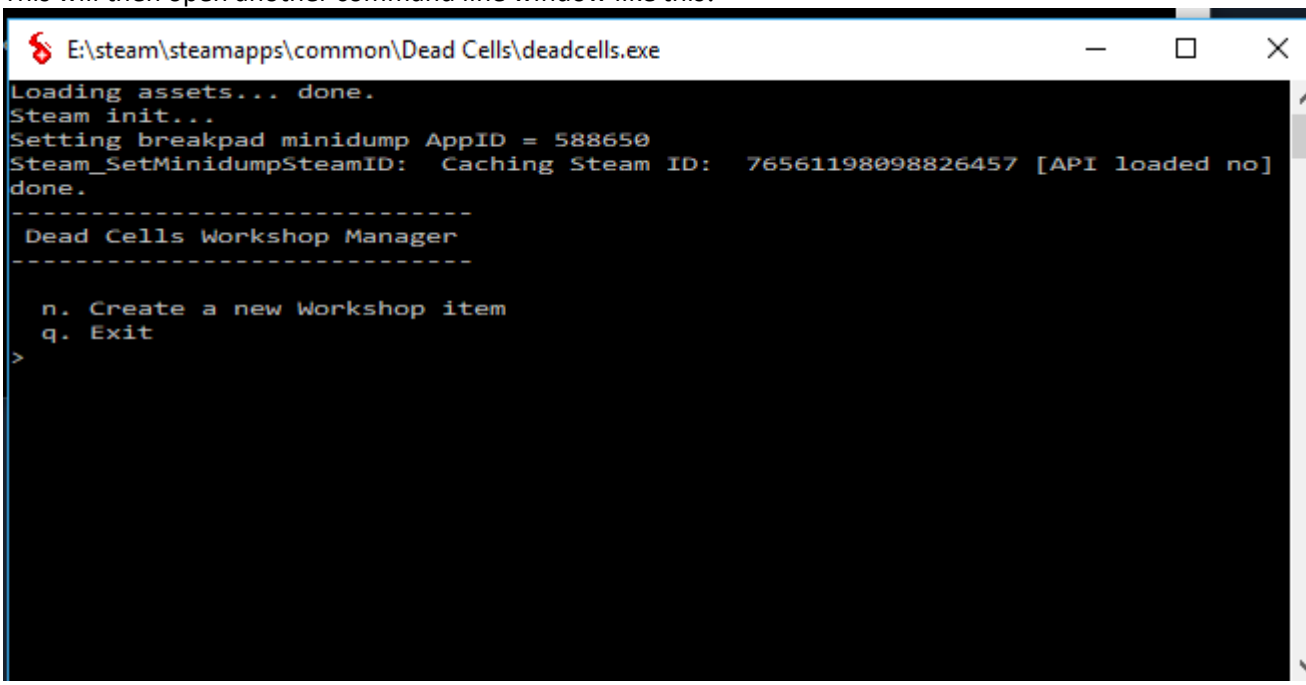
4. Launch the game and start a new run, activate "Dummy Mod" and you can test your own mod

Create and Upload:

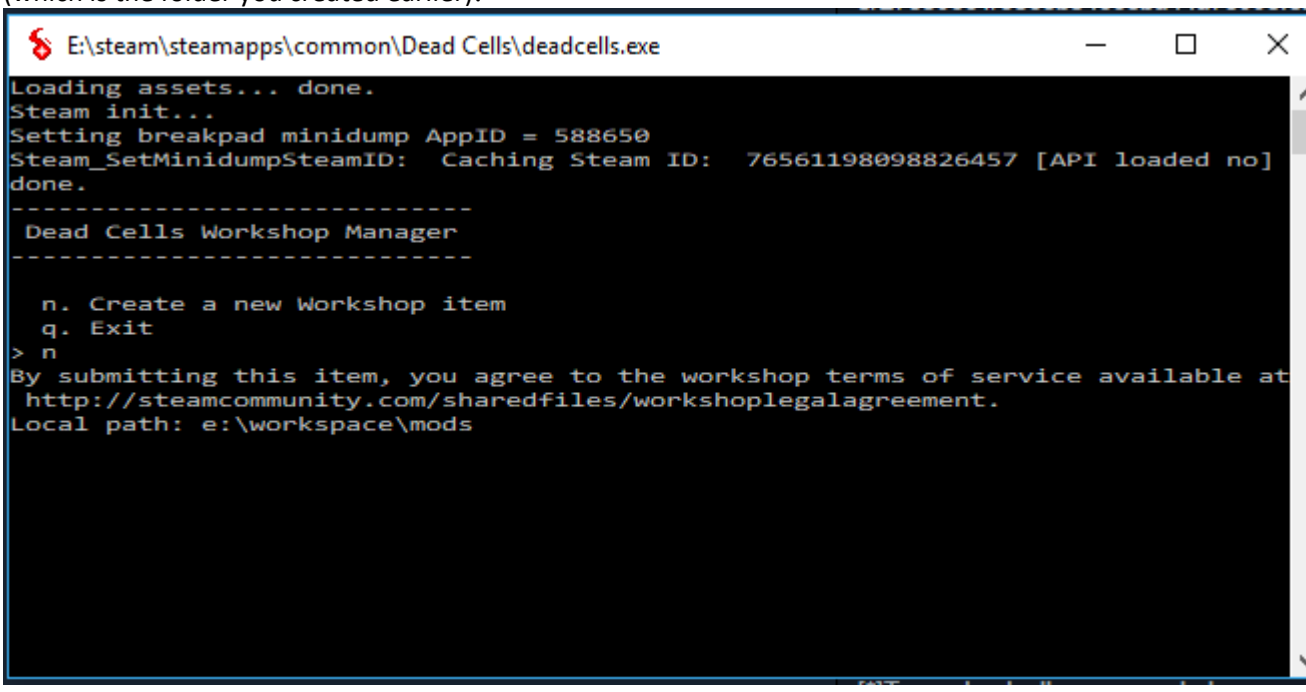
To upload your new mod to steam workshop and put it online you need to use the command line tool again.

1. Open the command line.
2. Navigate to the dead cells folder here: <DRIVE>:\steam\steamapps\common\Dead Cells
3. Type: deadcells.exe --workshop
4. Click on ok on the steam popup (and remember it might be hiding behind another window).

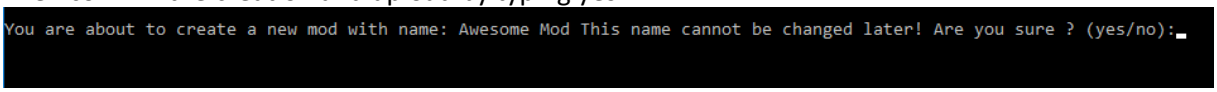
5. This will then open another command line window like this:



6. Choose “create a new workshop” item by typing 'n' and inputting the path to your mod (which is the folder you created earlier):



7. Then confirm the creation and upload by typing yes.



8. This will upload the file to your steam workshop and allow you to edit the workshop item through the steam interface.