The Witcher 3: Wild Hunt - ModKit Quick Guide
Sample Mod Creation Walkthrough

This document walks you through the process of creating four sample modifications for The Witcher 3: Wild Hunt.

1. Getting Started

1.1 ModKit Introduction

ModKit is a set of command line tools that allow you to uncook and unpack the game, export certain assets from it into modifiable formats, import them back into the form used by the game and then cook and pack everything so those modifications can be applied to the game. The main hub of the ModKit is wcc_lite.exe, which executes different operations depending on the arguments provided. The easiest way to use wcc_lite is through the Windows command prompt or batch files.

An overview of ModKit functionality can be found in the "ModKit - Quick Guide" document.

1.2 Install ModKit

To start creating mods for The Witcher 3: Wild Hunt you must first install ModKit. To do so, run the setup.exe file and follow the instructions. ModKit's main tool is called wcc_lite.exe and is placed under `[installation_path]\Witcher 3 Mod Tools\bin\x64\`. We will use it during the modding process for the various operations described in this document.

1.3 Uncook Game

To uncook the game, launch wcc_lite with the appropriate arguments:

```
uncook -indir=<game_path>\content -outdir=<dirpath>\Uncooked -imgfmt=tga
```

Where game_path is the location where The Witcher 3: Wild Hunt is installed and dirpath is where the uncooked game will be placed.

An example batch file might look like this:

```
call wcc_lite uncook -indir=F:\Witcher3\content -outdir=F:\Uncooked\ -imgfmt=tga
```

2. “Witcher The Slav” Mod (modDresik)

2.1 Introduction

This mod shows how to create a modification by changing and replacing the textures of Geralt’s starting outfit.

2.2 Export Assets

To export the assets required for this mod, you need to run wcc_lite in export mode; to do so, use the arguments shown below:

```
wcc_lite export -depot=<dirpath>\Uncooked\ -file=<filepath> -out=<filepath>
```

where:

- depot - directory into which game has been uncooked (see “1.3 Uncook Game”)
- file - the relative path to the file that will be exported
- out - where the file will be exported

NOTE: wcc_lite needs to be launched in export mode for each file separately.

To modify Geralt’s starting look, I had to export the right textures:

- shirt - characters\models\geralt\armor\armor_shirt\t_01_mg__shirt_d01.xbm
- trousers - characters\models\geralt\armor\armor__viper\l_01_mg__viper_d01.xbm
- boots - characters\models\geralt\armor\armor__viper\s_01_mg__viper_d01.xbm

I used a batch file for that:

```
::make directory for exported assets, if output directory didn’t exist wcc would fail
mkdir F:\Mods\Dresik\Raw

::export shirt texture (tip: ‘t’ at beginning of file name stands for torso part)
call wcc_lite export -depot=F:\Uncooked\
-file=characters\models\geralt\armor\armor_shirt\t_01_mg__shirt_d01.xbm
-out=F:\Mods\Dresik\Raw\t_01_mg__shirt_d01.tga

::export trousers texture (tip: ‘l’ at beginning of file name stands for legs part)
call wcc_lite export -depot=F:\Uncooked\
-file=characters\models\geralt\armor\armor__viper\l_01_mg__viper_d01.xbm
-out=F:\Mods\Dresik\Raw\l_01_mg__viper_d01.tga

::export boots texture (tip: ‘s’ at beginning of file name stands for shoes part)
call wcc_lite export -depot=F:\Uncooked\
-file=characters\models\geralt\armor\armor__viper\s_01_mg__viper_d01.xbm
-out=F:\Mods\Dresik\Raw\s_01_mg__viper_d01.tga
```

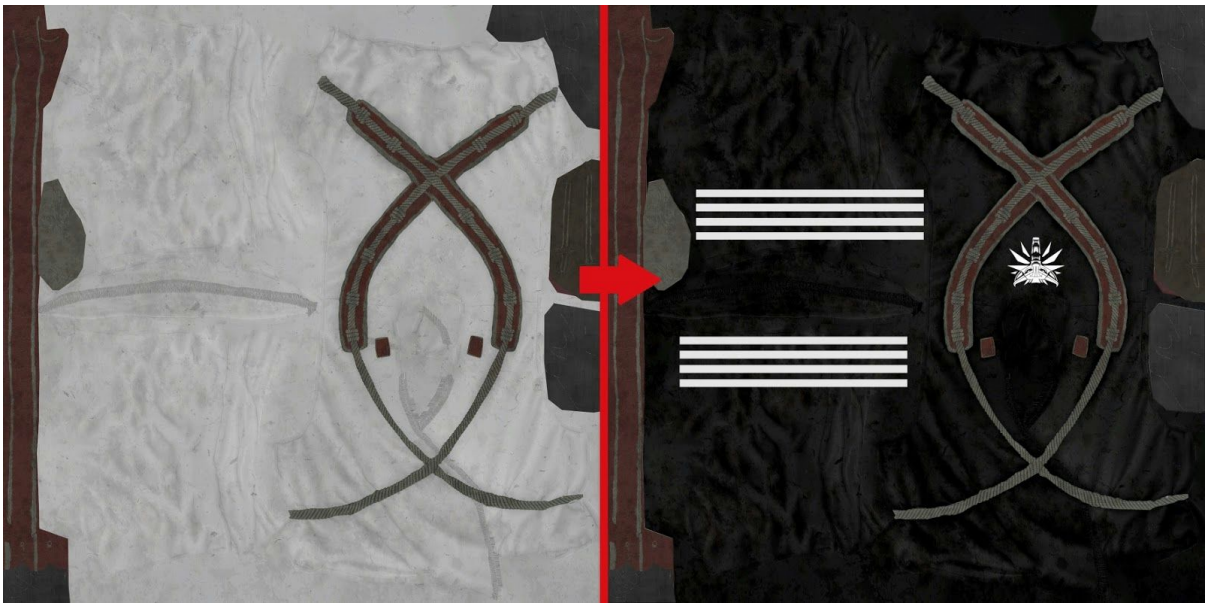
That gave me three .tga files that I could edit.

2.3 Modify Assets

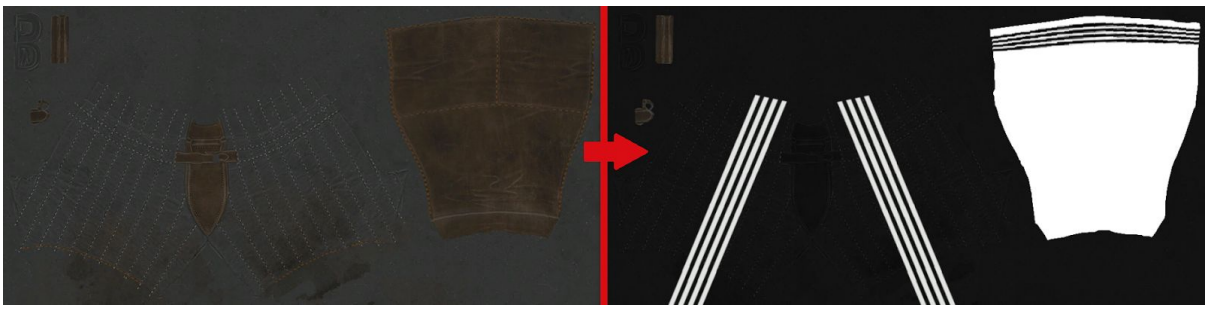
Next step after exporting the assets is to edit them. For textures, you can use any graphical editor that supports the exported formats.. For example Photoshop, GIMP, MS Paint, etc. (I used GIMP 2.8.14).

The results of my editing looked like this:

- Shirt:



- Trousers:



- Boots:

2.4 Import Modified Assets

After modifying, we need to import the assets back to the formats used by the game (in the case of textures, it's .xbm). This process is very similar to exporting and uses wcc_lite in import mode:

```
import -depot=<dirpath>\Uncooked\ -file=<filepath> -out=<filepath>
```

where:

- depot - directory where the uncooked game is (see "1.3 Uncook Game")
- file - path to the file that will be imported
- out - where the file will be imported

NOTE: Files in the output destination should be in their original directory structure (i.e. as in the uncooked game)

Batch file that I used:

```
::Import shirt texture
call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Dresik\Moded\t_01_mg__shirt_d01.tga
-out=F:\Mods\Dresik\Uncooked\characters\models\geralt\armor\armor_shirt\t_01_mg__shi
rt_d01.xbm

::Import trousers texture
call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Dresik\Moded\l_01_mg__viper_d01.tga
-out=F:\Mods\Dresik\Uncooked\characters\models\geralt\armor\armor__viper\l_01_mg__vi
per_d01.xbm
```

```
::Import boots texture
call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Dresik\Moded\s_01_mg__viper_d01.tga
-out=F:\Mods\Dresik\Uncooked\characters\models\geralt\armor\armor__viper\s_01_mg__vi
per_d01.xbm
```

2.5 Cook and Pack the Mod

Now that the assets are in the proper format, they need to be re-cooked, packed and have their shader.cache and metadata.store files generated. To do so, we'll use wcc_lite in four different modes (more about them in the "ModKit - Quick Guide" document)

I did it all in one batch file:

```
::Cook mod
call wcc_lite cook -platform=pc -mod=F:\Mods\Dresik\Uncooked\
-basedir=F:\Mods\Dresik\Uncooked -outdir=F:\Mods\Dresik\Cooked\

::Build texture.cache for mod
call wcc_lite buildcache textures -basedir=F:\Mods\Dresik\Uncooked\ -platform=pc
-db=F:\Mods\Dresik\Cooked\cook.db
-out=F:\Mods\Dresik\Packed\modDresik\content\texture.cache

::Pack mod
call wcc_lite pack -dir=F:\Mods\Dresik\Cooked\
-outdir=F:\Mods\Dresik\Packed\modDresik\content\

::Generate metadata.store for mod
call wcc_lite metadatastore -path=F:\Mods\Dresik\Packed\modDresik\content\
```

2.6 Add Mod to the Game

The last thing is to add the generated mod to the game. To do this, I just copied all the files from the Packed directory (modDresik folder) into the `<game_dir>\mods\` directory.

NOTE: The name of the folder with the mod must start with the word "mod" and can't contain any spaces.

The easiest way to check if everything works is to start a new game or load a saved game where Geralt has the items we modified in his inventory. Geralt's outfit should look somewhat different than usual:

NOTE: Since Geralt's starting outfit has him wearing trousers and boots from the Hunting set, this mod will change the look of those items throughout the game.

3. “Fabulous Roach” mod (modUnicorn)

3.1 Introduction

This example shows how to create a mod by modifying and replacing Roach’s textures and meshes.

3.2 Export Assets

To export the assets required for this mod, you need to run wcc_lite in export mode (more in: 1.3 Uncook game)

To modify Roach’s basic look, I had to export her:

- model - characters\models\animals\horse\draft\model\b_01_hd__brown_rideable.w2mesh
- textures
    - characters\models\animals\horse\draft\model\b_01_hd__dirt_d02.xbm
    - items\horse_items\saddles\model\s_01_hd__common_d01.xbm
    - items\horse_items\saddles\model\p_01_hd__common_d01.xbm

I used a batch file for that:

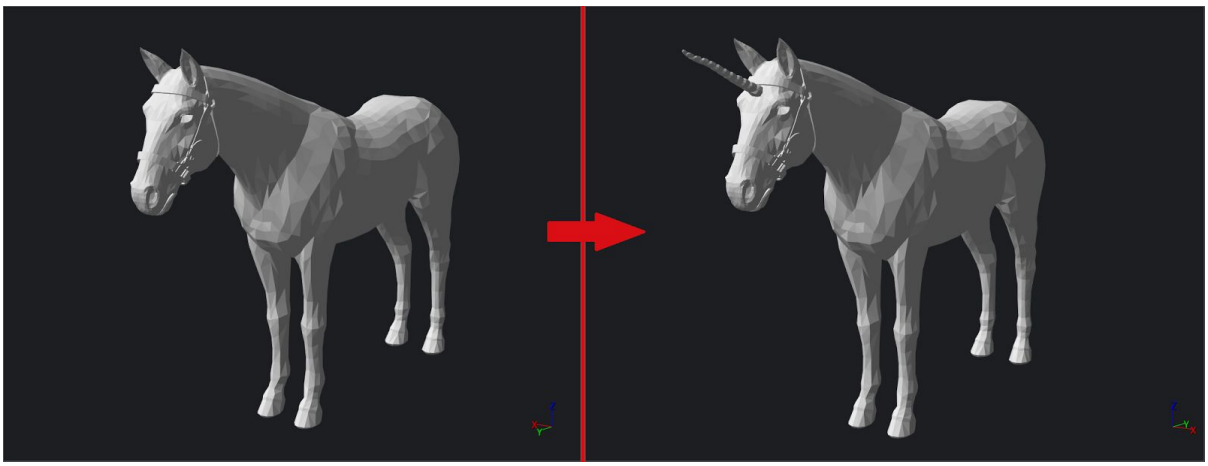
```
::make directory for exported assets, if output directory didn’t exist wcc would fail
mkdir F:\Mods\Unicorn\Raw

::export horse model
call wcc_lite export -depot=F:\Uncooked\
-file=characters\models\animals\horse\draft\model\b_01_hd__brown_rideable.w2mesh
-out=F:\Mods\Unicorn\Raw\b_01_hd__brown_rideable.fbx

::export horse texture
call wcc_lite export -depot=F:\Uncooked\
-file=characters\models\animals\horse\draft\model\b_01_hd__dirt_d02.xbm
-out=F:\Mods\Unicorn\Raw\b_01_hd__dirt_d02.tga

::export saddle textures
call wcc_lite export -depot=F:\Uncooked\
-file=items\horse_items\saddles\model\s_01_hd__common_d01.xbm
-out=F:\Mods\Unicorn\Raw\s_01_hd__common_d01.tga

call wcc_lite export -depot=F:\Uncooked\
-file=items\horse_items\saddles\model\p_01_hd__common_d01.xbm
-out=F:\Mods\Unicorn\Raw\p_01_hd__common_d01.tga
```

That gave me .fbx and three .tga files that I could edit.
3.3 Modify Assets

The next step after exporting assets is to edit them.
For textures, you can use any graphical editor that supports the exported formats. For example: Photoshop, GIMP, MS Paint, etc. (I used GIMP 2.8.14).
For meshes, you can use any 3D editor that can edit .fbx files. For example Maya, Blender, 3D Max, MS Visual Studio, etc.

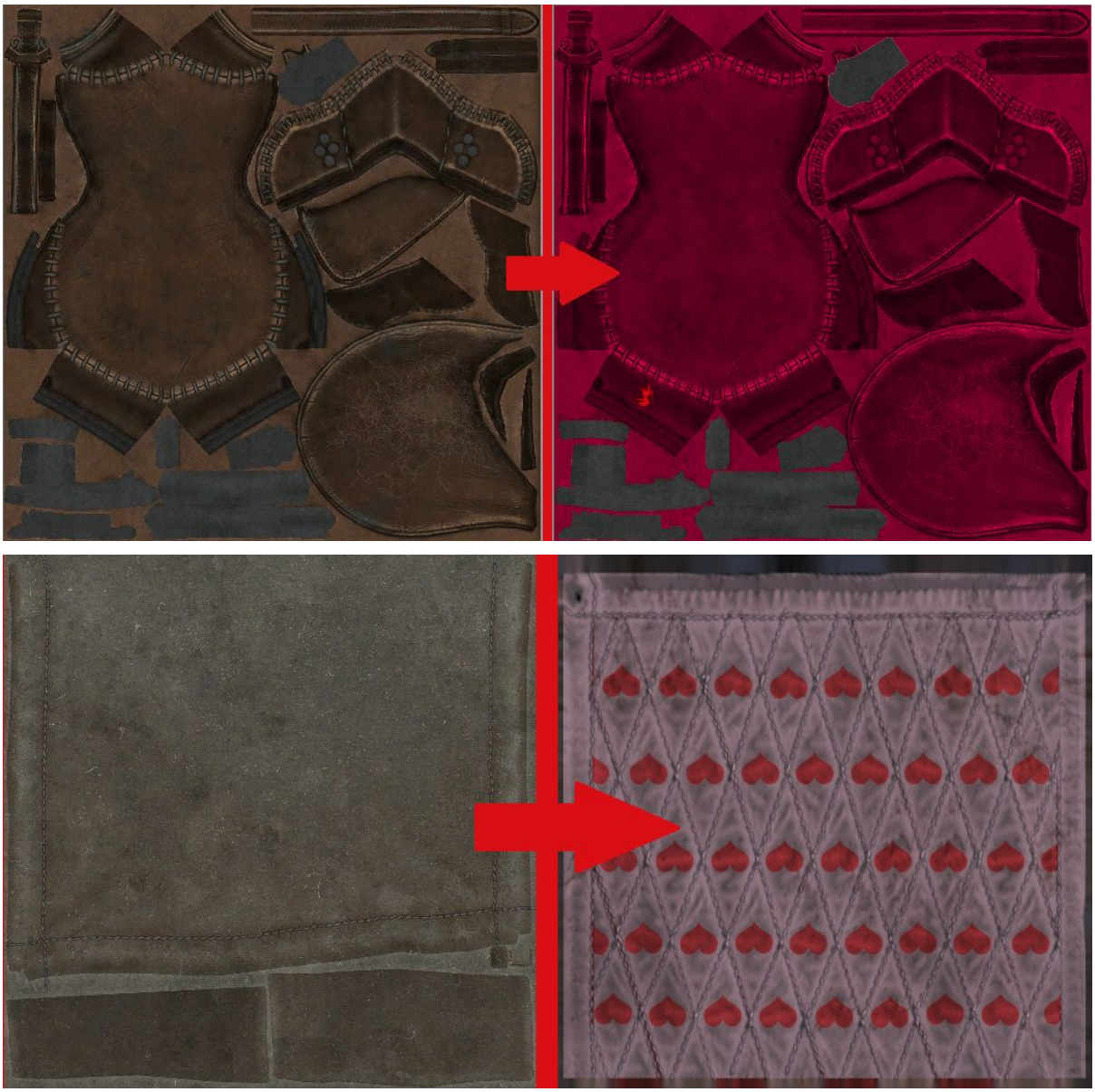
The results of my modifications looked like this:

- Model:



NOTE: The exported .fbx contains three horse models; these are different LOD meshes (Level Of Details, meshes used when a player is further away from horse) and all three should be edited individually.

- Textures:



3.4 Import Modified Assets

After modifying, we need to import assets back to the formats used by the game (in case of textures, that's .xbm and for meshes it's .w2mesh). This process is very similar to exporting and uses wcc_lite in import mode (more on that in: 2.4 Import Modified Assets).

The batch file that I used:

```
::import horse model
call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Unicorn\Moded\b_01_hd__brown_rideable.fbx
-out=F:\Mods\Unicorn\Uncooked\characters\models\animals\horse\draft\model\b_01_hd__b
rown_rideable.w2mesh

::import horse texture
call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Unicorn\Moded\b_01_hd__dirt_d02.png
-out=F:\Mods\Unicorn\Uncooked\characters\models\animals\horse\draft\model\b_01_hd__d
irt_d02.xbm

::import saddle textures
call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Unicorn\Moded\s_01_hd__common_d01.png
-out=F:\Mods\Unicorn\Uncooked\items\horse_items\saddles\model\s_01_hd__common_d01.xb
m

call wcc_lite import -depot=F:\Uncooked\
-file=F:\Mods\Unicorn\Moded\p_01_hd__common_d01.png
-out=F:\Mods\Unicorn\Uncooked\items\horse_items\saddles\model\p_01_hd__common_d01.xb
m
```

3.5 Cook and Pack Mod

Once the assets are in the proper format, they need to be re-cooked, packed and have their shader.cache and metadata.store files generated. To do so, we'll use wcc_lite in four different modes (more about them in "ModKit - Quick Guide" document)

NOTE: If the mod didn't modify any textures, generating the texture.cache file can be skipped.

I did it with a batch file:

```
::Cook mod
call wcc_lite cook -platform=pc -mod=F:\Mods\Unicorn\Uncooked\
-basedir=F:\Mods\Unicorn\Uncooked -outdir=F:\Mods\Unicorn\Cooked\

::Build texture.cache for mod
call wcc_lite buildcache textures -basedir=F:\Mods\Unicorn\Uncooked\ -platform=pc
-db=F:\Mods\Unicorn\Cooked\cook.db
-out=F:\Mods\Unicorn\Packed\modUnicorn\content\texture.cache

::Pack mod
call wcc_lite pack -dir=F:\Mods\Unicorn\Cooked\
-outdir=F:\Mods\Unicorn\Packed\modUnicorn\content\

::Generate metadata.store for mod
```

```
call wcc_lite metadatastore -path=F:\Mods\Unicorn\Packed\modUnicorn\content\
```

3.6 Adding the mod to the game

The last step is to add the generated mod to the game. To do this, I just copied all the files from the Packed directory (modUnicorn folder) into the `<game_dir>\mods\` directory.

NOTE: The Mod folder's name must start with the word "mod" and can't contain any spaces.

To check if everything works, just launch The Witcher 3: Wild Hunt, load a saved game or start a new one and summon your trusted mount. Roach should look different than usual:

NOTE: Since Roach uses the same mesh and texture as some other horses in the game, this mod will affect them as well

4. “Petard sWitcher” Mod (modBombs)

4.1 Introduction

This mod shows how to create a modification by modifying game scripts.

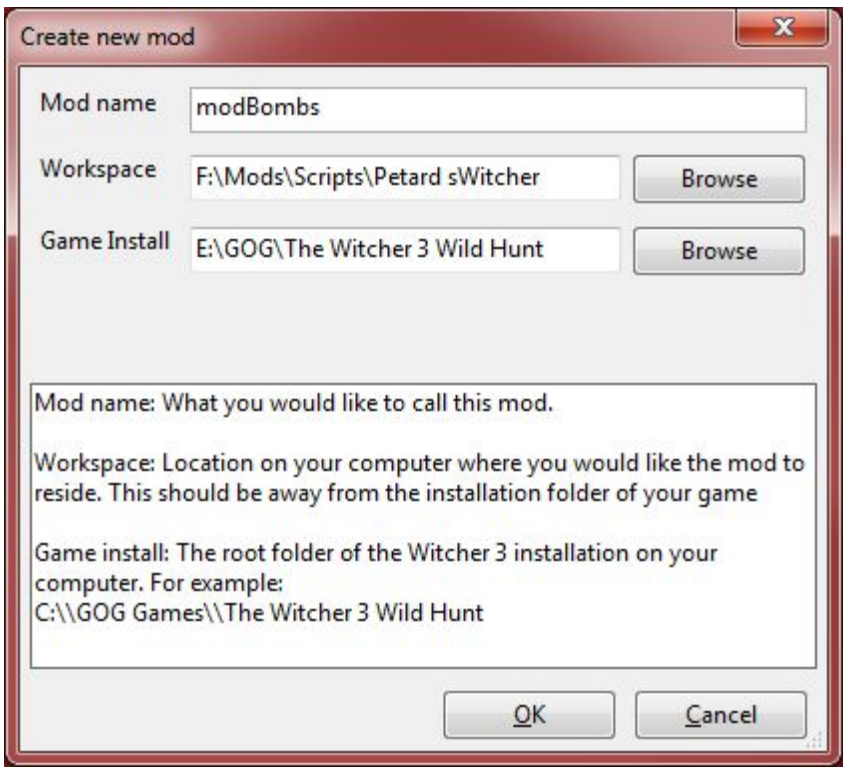
4.2 Preparing Scripts Mod

Game scripts can be found in:

```
<game_directory>\content\content0\scripts\
```

To create a script mod, the best solution is to use Script Studio, which is provided with the Mod Tool (the .exe is in `*\Witcher 3 Mod Tools\bin\x64\`).

Launch it, select *File* > *Create new mod*, fill in the pop-up window with the appropriate info and press OK.



Script Studio will copy all the scripts and create a mod solution file in the provided workspace location.

You can also copy scripts manually, as long as you remember to maintain their original directory structure.

4.3 Modify Scripts

The easiest way to modify Witcher Script (.ws files) is to use the above-mentioned Script Studio, but of course you can use any text editor of your choice if you wish (e.g.: MS Notepad, Notepad++, Sublime Text, vi, Visual Studio, etc.).

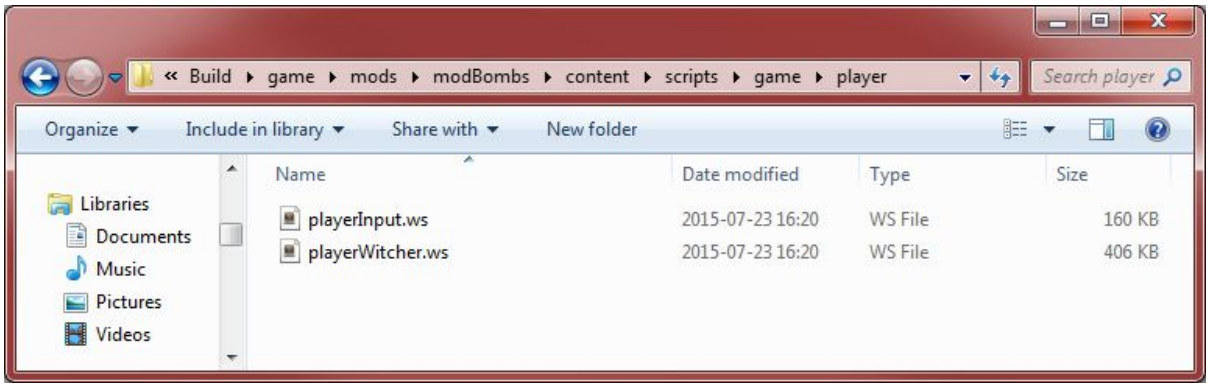
4.4 Add Scripts to Game as a Mod

If modifications were made using Script Studio, just go to *File* > *Install Mod* and Script Studio will copy all the necessary files to the proper destinations.

If you used another editor, copy the modified scripts (in their folders structure) to:

<game_directory>\mods\mod_name\content\scripts

In my case, the structure looks like this:



After launching the game (doesn't matter which deployment method was used), a splash screen showing the script compilation process appears:

If compilation finishes successfully, the game should launch (if it fails, a window with compilation errors appears).

5. Custom Equipment Sets Mod (modEqSets) - Detailed Guidelines About Script Modification

The following section will introduce you to the process of modifying scripts. We will implement a custom equipment sets mechanism - the player will be able to define his own equipped items presets and switch between them using hotkeys bound to the Select Specific Sign action (by default, the number keys 3-7).

5.1 Preparing the Workspace

(see "4.2 Preparing Scripts Mod")

List of required files:

game\player\playerWitcher.ws
game\player\playerInput.ws
game\setManager.ws - this file is new - create it in the "Local" group in solution tree in Script Studio

All files are attached as sample mod "modEqSets" - please refer to them while reading this walkthrough.

5.2 Setting up the *CSetManager* class with Support Structure *ItemsSet*

The following class is responsible for storing presets of items registered by the user, registering new presets and returning previously-saved presets.

```
public function registerSet( slot : int, eqList : array<SItemUniqueId> )
```

This function takes as parameters an int that indicates the slot number where the preset will be saved, and an array of *SItemUniqueId* - ID numbers of items that we want to store.
In the function we iterate over an array of IDs, adding them into a new array object, wrapping the array with an *ItemsSet* struct and finally inserting the struct into the *setsRegistered* array.

NOTE: first we need to Erase the existing set form given slot, as the Insert method doesn't replace the existing item.

```
public function restoreSet( slot : int ) : array<SItemUniqueId>
```

This function restores the set from *setsRegistered* and returns it in the form *array<SItemUniqueId>*.

5.3 Adding *CSetManager* to playerWitcher.ws

We need to add *CSetManager* to *playerWitcher*, as it needs to have access to the Inventory system. To do so, we add *public property mSetManager.* Before we can use it, it needs to first be initialized - we create a new *CSetManager* object and call the *Initialize()* method at the end of the *OnSpawned* event - this will be called after playerWitcher is spawned.

The final part of the playerWitcher.ws modification is to add the functions that will handle registering and restoring sets for agiven slot number.

`public function registerSet(slot : int)`
This method gets list of all equipped items with *GetEquippedItems()* and sends it to *CSetManager*.

`public function restoreSetFromSlot( slot : int )`
This function takes an array of *SItemUniqueId* from *CSetManager*, which represents the stored set, unequips items from all slots and equips items from the set.

5.4 The Final Step - Hooking the System into playerInput.ws

I decided to hook the system into the *OnSelectSign* event, as that way we can use existing key mappings and we don't need to modify the .ini or .xml configs to set up new input methods. The following modification will disable the default action for the number keys 3-7 and replace it with the new system.
First we add a bool var - it will determine if the player is in Set Register mode.
Next, we replace the existing functions with new ones - *restoreSetFromSlot* and *registerSet*, and use theGame.witcherLog.AddMessage() to print a message into the on-screen log.

5.5 The Scripts Are Now Ready

Follow the instructions from "4.4 Add Scripts to Game as a Mod" to add them to the game as a mod.

NOTE: Because "modBombs" and "modEqSets" modify the same .ws files they can't be used at the same time. By default the game will use the mod whose name comes first in alphabetical order - in this case, modBombs. Mod order can be set using priorities in the mods.settings file (more on that in the "ModKit Quick Guide" document, paragraph 11). Since the code changes introduced in these two samples don't directly interfere with each other, they can be merged together and added to the game as one new, separate mod.