# CastleMiner Z: OpenClassic

## Guide to Addons:

Tag Formatting Explained
Tag Types Explained
Tag Keyword List with Definitions
Adding Music, Models, and Texture Packs
Specifics of the Structure Editor

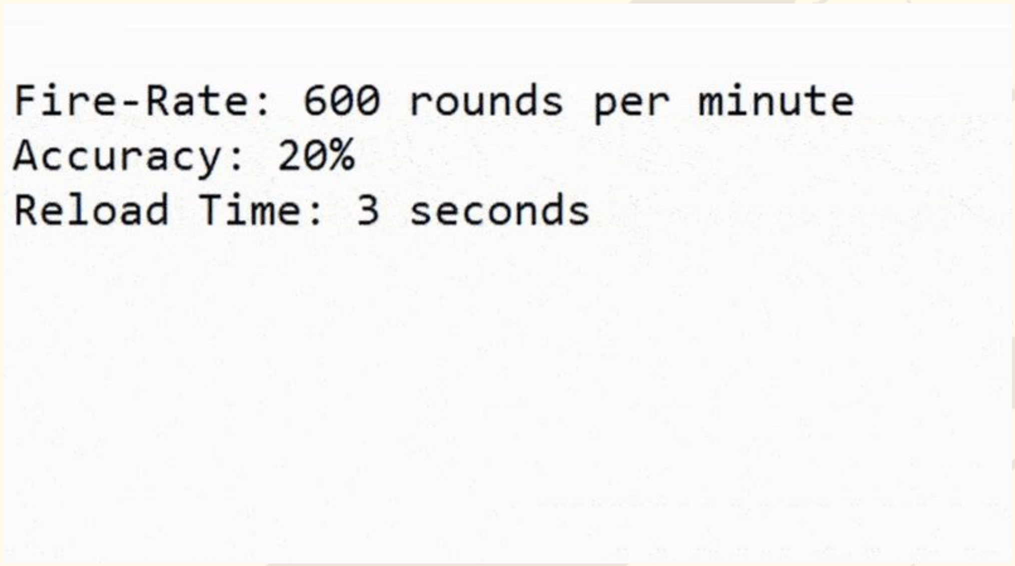
## Page Reference:

## Getting Started - What are TAGS?

If you've never programmed or made mods before, you may be scared or intimidated - but DON'T WORRY! OpenClassic requires **NO PROGRAMMING EXPERIENCE** to make addons, it's extremely simple! All we need to understand is the Tag format.

**What are Tags?**

Think of a tag like an editable template. A tag is basically a file for a specific game object that contains several easily editable values for all that object's traits. For example: let's say we have a gun tag. Editing the tag will show you a simple list of all the gun's traits such as fire-rate, accuracy, reload time, mag size, etc. To edit the tag there would be no programming needed, just find the value you want (it'll be clearly labelled) and change what it says!



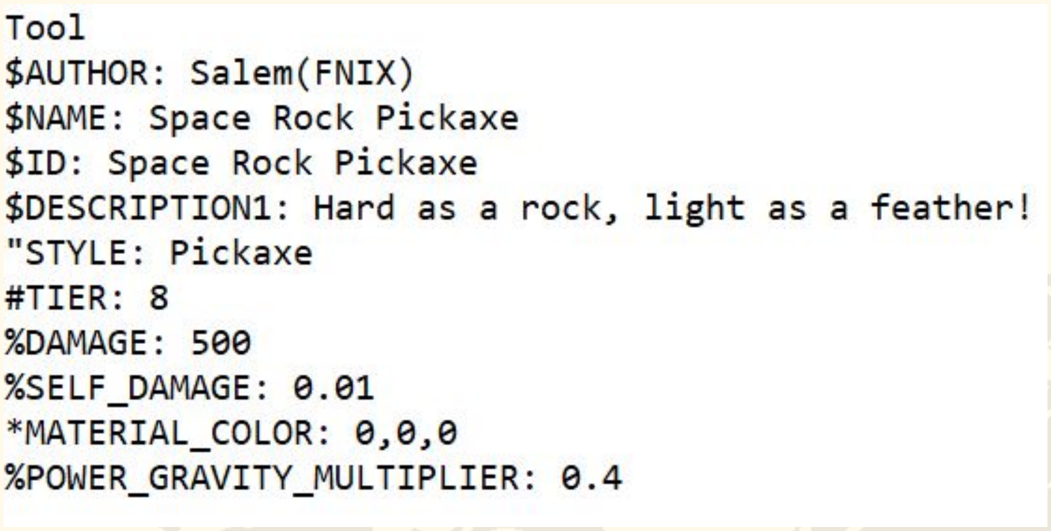
```
Fire-Rate: 600 rounds per minute
Accuracy: 20%
Reload Time: 3 seconds
```

Not animating? Click here!

**That's it, seriously!!**

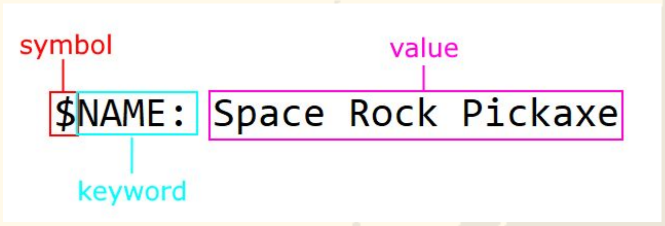
Now head to the next page to see how this is applied in OpenClassic.

## Getting Started - What are TAGS?



```
Tool
$AUTHOR: Salem(FNIX)
$NAME: Space Rock Pickaxe
$ID: Space Rock Pickaxe
$DESCRIPTION1: Hard as a rock, light as a feather!
"STYLE: Pickaxe
#TIER: 8
%DAMAGE: 500
%SELF_DAMAGE: 0.01
*MATERIAL_COLOR: 0,0,0
%POWER_GRAVITY_MULTIPLIER: 0.4
```

Above is an actual OpenClassic tag for a custom pickaxe. It's very similar to the example on the last page but as you can see there are a few additions. Let's go over each element and what they're called!



- **Keyword**: The name of the object's trait.
- **Value**: What we are setting the trait to be.
- **Symbol**: What kind of value the game expects for this keyword.

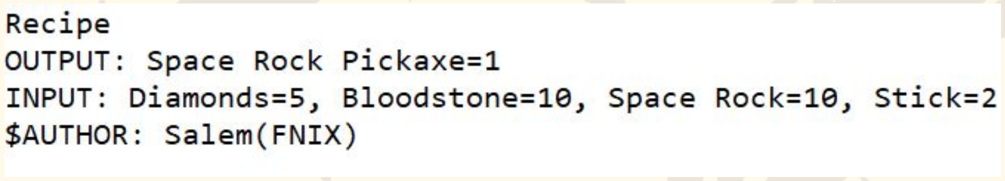
There are a couple different symbols, they're the only thing you'll have to memorize to make addons. Every keyword has a specific symbol attached to it that tells you what kind of value it expects. Some keywords can use multiple symbols.

- **$** means Text Value (such as "Hello!")
- **#** means Number Value (such as "53")
- **%** means Decimal Value (such as "5.3") OR Number Value
- **=** means True/False Value (such as "False")
- ***** means Color Value (such as "30, 25, 9" OR "Bloodstone")
- **"** means Quote Value, more on this later.

## Getting Started - What are TAGS?

Quote Keywords expect unique values from lists specific to that keyword. For example: the "*ANIMATION* keyword only takes values such as *Pistol*, *Assault Rifle*, *Rifle*, *SMG*, and the rest of the gun names only.

Keywords that do not use any symbol expect a unique value format exclusive to that keyword and tag. For example: Recipe tags use the unique keywords *INPUT* and *OUTPUT*. They don't have any symbol because they expect formats that the available symbols don't cover.



```
Recipe
OUTPUT: Space Rock Pickaxe=1
INPUT: Diamonds=5, Bloodstone=10, Space Rock=10, Stick=2
$AUTHOR: Salem(FNIX)
```

You'll get the hang of it once you start experimenting and creating your own tags.

**Speaking of, how does one create tags?**
OpenClassic tags are written as plain text files, so anyone can make them without any extra programs needed. You can create or edit them with any text editor, as long as you save it as a .CLAG file (not .TXT).

You start by creating a new text file and then declaring the Tag Type (what type of object the tag is, such as *Firearm* or *Recipe*) first. After you write the type, you can begin writing the keywords in any order you please. You can refer to a list of available keywords per Tag Type and what each keyword changes later in this guide. Some types use the same keywords, some have keywords unique to them, You do not need to include every available keyword for each tag. If you do not include one, the game will just use a default value.

Confused? The video tutorial goes into much more detail on tags if you need it.

# Quick Tips

**Launch Arguments**

There are two launch arguments you should know about when creating addons for OpenClassic. You can set launch arguments by opening CastleMiner Z’s properties on Steam, and adding them to the “LAUNCH OPTIONS” textbox.

- +quickload
    - The “+quickload” launch argument is extremely helpful in speeding up the testing process for your addons. With quickload enabled, you will be loaded into a Creative world the moment you launch the game, skipping all those extra clicks, and no world progress will save.
- +editor
    - The “+editor” launch argument is needed to access the Structure Editor, which is used to make builds for the “Structure” Tag Type. You can learn more about structures later in this guide.
- +noaddons
    - This launch argument will launch the game with all your addons disabled regardless of your folder configuration.

**Other Tips**

You can press F3 in Creative mode to view your coordinates, and other player information such as speed. You can also press F4 to open the Spawn Menu, and spawn enemies to test your weapons on.

In the crafting menu, you can right click to take you to the very bottom. You can also press a letter on your keyboard to take you to the first item in the category that starts with that letter.

If you’re familiar with *comments* in other programming languages, you can actually write them in tags very easily. OpenClassic will simply ignore any lines that don’t start with a usable keyword.

# Shared Keywords

Most keywords are unique to their tag, but there are a few that are shared and can be used the same on different tag types. *Quote Value* keywords will not be listed here because their expected values differ based on the type despite being the same keyword.

**Universal Keywords**

- $NAME
  - Straightforward, this sets the name of the addon and if its an object, the name of the object. If not set, the game will use a default value. For recipes, the default is the name of the OUTPUT item.
- $AUTHOR
  - This sets the displayed author name in the ingame Addons menu. If not set, there will be no author listed.

**Shared between Types: Tool, Item, Firearm, Block**

- $ID
  - The ID of the item, used by recipes and game mechanics. This is usually the exact same value as $NAME, but in the case your item has the same name as another, you can give it a different ID to be referenced instead. The default value is the $NAME.
- $DESCRIPTION1
  - The first line of the description in the crafting menu. If not set, it will be empty for most items. Tools will use their vanilla descriptions by default.
- $DESCRIPTION2
  - The second line of the description in the crafting menu. If not set, it will be empty for most items. Tools will use their vanilla descriptions by default.

**Shared between Types: Tool, Item, Firearm**

- %ICON_SIZE
  - The size of the inventory icon (if the model is custom).
  - Defaults to 1
- ICON_POSITION
  - The coordinates of the inventory icon (if the model is custom).
  - Defaults to 0, 0, 0
  - Expected values look like this: *3, 15, 0*

**Shared between Types: Tool, Firearm**

- %DAMAGE
  - The damage the item does.
- %SELF_DAMAGE
  - The damage the item takes each use (durability)

## Shared Keywords (Powers)

Power keywords are basically special abilities items and blocks can give players and enemies. For blocks, powers are active for anything touching them. For items, it depends on the *#POWER_MODE* value.

**Shared between Types: Tool, Item, Firearm, Block, Modifier**

- %POWER_SPEED_MULTIPLIER
- %POWER_RECIEVED_DAMAGE_MULTIPLIER
- %POWER_REGENERATION_MULTIPLIER
- %POWER_JUMP_MULTIPLIER
- %POWER_SENSITIVITY_MULTIPLIER
- %POWER_INACCURACY_MULTIPLIER
- %POWER_MINING_SPEED_MULTIPLIER
- %POWER_GRAVITY_MULTIPLIER
- %POWER_DAMAGE_PER_SECOND
- %POWER_SELF_DAMAGE_MULTIPLIER

**Shared between Types: Tool, Item, Firearm**

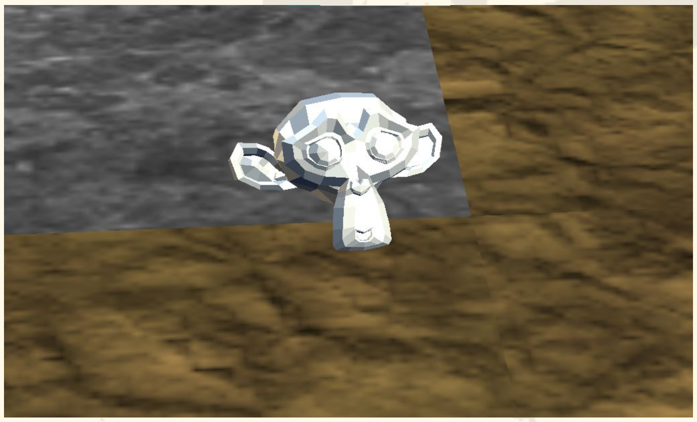
- “POWER_MODE
    - *Constant*
        - Powers are active as long as the item is in your inventory.
    - *Constant Equipped*
        - Powers are active as long as the item is equipped.
    - *Manual Equipped*
        - Powers are active if the item is equipped and the Power key is held.
    - *Manual Powerup*
        - Powers are active if the item is in the top left of your inventory and the Power key is held.
    - *Constant Powerup*
        - Powers are active as long as the item is in the top right of your inventory.
    - (Defaults to *Constant Equipped*)
- %POWER_SELF_DAMAGE_PER_SECOND
- %POWER_ZOMBIE_DAMAGE_MULTIPLIER
- %POWER_SKELETON_DAMAGE_MULTIPLIER
- %POWER_ALIEN_DAMAGE_MULTIPLIER
- %POWER_DEMON_DAMAGE_MULTIPLIER
- %POWER_FIRE_DRAGON_DAMAGE_MULTIPLIER
- %POWER_FOREST_DRAGON_DAMAGE_MULTIPLIER
- %POWER_SAND_DRAGON_DAMAGE_MULTIPLIER
- %POWER_ICE_DRAGON_DAMAGE_MULTIPLIER
- %POWER_UNDEAD_DRAGON_DAMAGE_MULTIPLIER

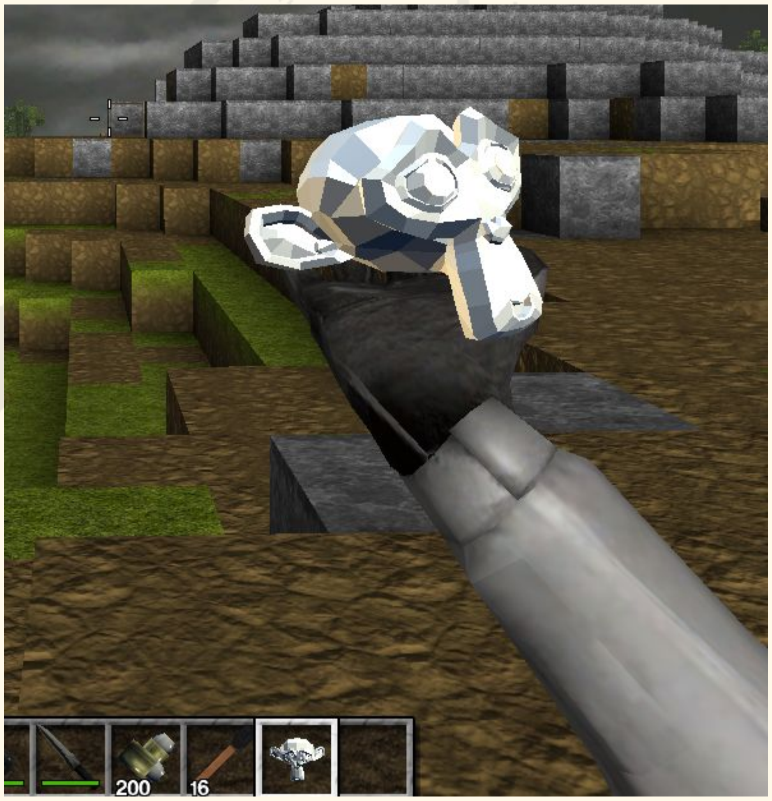
# Exclusive Keywords to: Firearm

- =AUTOMATIC
- %BULLETS_PER_SECOND
- #CLIP_SIZE
- #ROUNDS_PER_RELOAD
- %SECONDS_TO_RELOAD
- %RECOIL
- %INACCURACY
- %ZOOM_AMOUNT
    - The amount the gun zooms in while aiming. If this keyword isn't included, the gun will not be able to aim. Since custom animations are not supported, and aiming doesn't actually improve accuracy, I suggest excluding this keyword unless your gun is scoped.
- =SCOPED
- "PROJECTILE_TYPE
    - *Normal*
    - *Shotgun*
    - *Laser*
    - *Laser Shotgun*
    - *Rocket*
- *PROJECTILE_COLOR
- =LASER_DESTROYS
    - If it uses laser projectiles, this sets whether they break blocks (defaults to True).
- =LASER_BOUNCES
    - If it uses laser projectiles, this sets whether they some blocks can deflect it (defaults to True).
- $AMMO_NAME
    - Value can be an item name or ID.
- %BULLET_LIFETIME
    - How many seconds the bullet will exist for. This can be used to limit range. Defaults to 200 (Assault Rifle value).
- %BULLET_SPEED
    - Defaults to 100 (Assault Rifle value).
- =HIDE_CROSSHAIR
    - Hides the crosshair of the firearm (defaults to False).
- "SHOOT_SFX
    - Allowed values are all listed on Page 21: "Sound Quote List".
- "RELOAD_SFX
    - Allowed values are all listed on Page 21: "Sound Quote List".
- $"MODEL
    - Value can be the path to a custom model (without file extension) or the quote values below to reuse a vanilla model.
    - *Pistol*
    - *SMG*
    - *Assault Rifle*
    - *Rifle*
    - *Shotgun*
    - *LMG*
    - *Laser Pistol*
    - *Laser SMG*
    - *Laser Assault Rifle*
    - *Laser Rifle*
    - *Laser Shotgun*
    - *RPG*
- "ANIMATION
    - Value will set which vanilla animations the item will use. All quotes for $"MODEL are supported, alongside the below extras.
    - *Generic*
    - *Tool*
    - *Block*
    - *Empty*
- *MODEL_COLOR1
    - Sets the COLOR1, defined by the model.
- *MODEL_COLOR2
    - Sets the COLOR2, defined by the model.

# Exclusive Keywords to: Item

- #STACK_SIZE
- $”MODEL
    - Value can be the path to a custom model (without file extension) or the quote values below to reuse a vanilla model.
    - *Ore*
    - *Powder*
    - *Gem*
    - *Bars*
    - *Ammo*
    - *Rockets*
- *MODEL_COLOR1
    - Sets the COLOR1, defined by the model.
- *MODEL_COLOR2
    - Sets the COLOR2, defined by the model.
- =AMMO
    - Sets whether the item shows up in the “Ammo” crafting category. This does nothing else - if you want your item to actually be used as ammunition that must be set in the firearm tag.





# Exclusive Keywords to: Tool

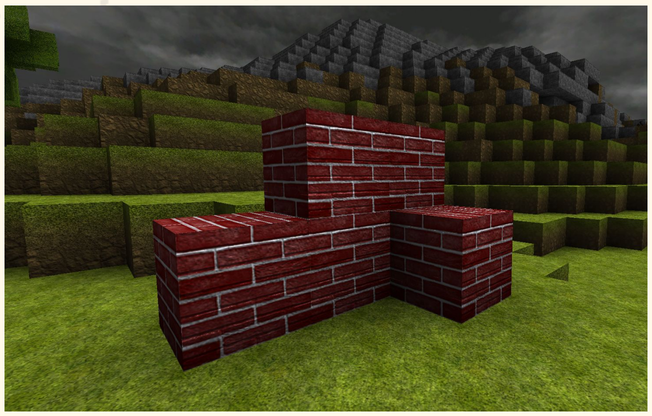
- "STYLE
  - This will set what type of tool this is. Supported values below.
  - *Pickaxe*
  - *Axe*
  - *Spade*
  - *Knife*
- #TIER
  - 0 = Wood
  - 1 = Stone
  - 2 = Copper
  - 3 = Iron
  - 4 = Gold
  - 5 = Diamond
  - 6 = Bloodstone
  - Values higher are not vanilla but will continue to get faster and stronger.
  - Do not set this keyword if you want it to be equal to bare hands.
- %SECONDS_TO_ATTACK
  - Currently only used by knives.
  - Defaults to 0.5 (the vanilla knife speed)
- *MATERIAL_COLOR
- $MODEL
  - Value should be the path to a custom model file (without file extension).

# Exclusive Keywords to: Block

- $"TEXTURE1
  - Value can be the path to a custom texture (without file extension) or it can be the name of any existing block to reuse their texture. The texture must be a 128x128 .PNG file. The same applies to the rest of the Block TEXTURE keywords below.
- $"TEXTURE2
- $"TEXTURE3
- $"TEXTURE4
- $"TEXTURE5
- $"TEXTURE6
- $MINE_SOUND
  - Allowed values are all listed on Page 21: "Sound Quote List".
- $DROP_ITEM
  - The name or ID of the item this block drops.
- #DROP_MINIMUM_AMOUNT
  - Defaults to 1.
- #DROP_MAXIMUM_AMOUNT
  - Defaults to 1.
- #ZOMBIE_RESISTANCE
  - Up to what tier of zombie its resistant to.
- #DRAGON_RESISTANCE
  - Up to what tier of dragon its resistant to.
- #FIREBALL_DAMAGE_TRANSMISSION
  - How much fireball damage passes it.
- #EXPLOSIVE_RESISTANCE
  - Up to what tier of explosive its resistant to.
- "TOOL_TO_MINE
  - *Hands*
  - *Pickaxe*
  - *Axe*
  - *Spade*
- #TIER_TO_MINE
  - Do not set this keyword if you want it to be mineable with bare hands.
- %SECONDS_TO_MINE
  - How long it takes to mine with the minimum tier required.
- SECONDS_TO_MINE_PER_ITEM
  - Sets a custom mine time for specific items. Not needed. Automatic values will be generated otherwise.
  - Expected values look like this: *Diamond Pickaxe=3, Gold Pickaxe=5, Iron Pickaxe=8*

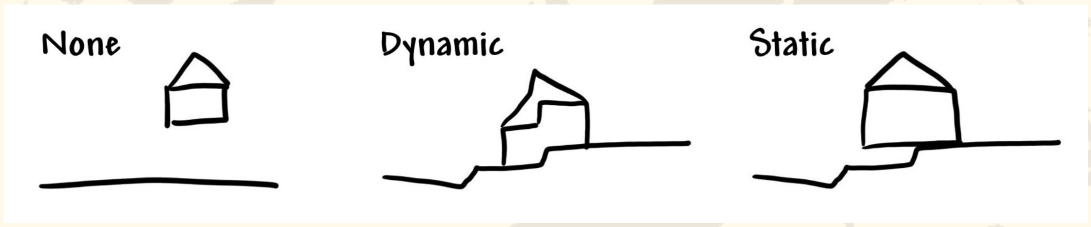
The values below are not usually needed as the default values should be sufficient for most custom blocks.

- %LIGHT_TRANSMISSION
  - Percentage of light that transmits through the block.
- %SELF_ILLUMINATION
  - How much light the block produces itself.
- =TRANSLUCENT
- =HAS_INTERIOR
  - Whether the textures of the block can be seen from both sides if you can see through the block.
- =HAS_ALPHA
  - Whether the block's textures have transparency
- =HAS_NORMALSPEC
  - If set to true: for each texture the block uses, the game will search for a normalspec texture as well. You can create one for your custom textures by giving it the same name but with "_normalspec" at the end, and putting it in the same directory.
- =HAS_COLLISION
- =CAN_BUILD_ON
- =UNBREAKABLE
- =GLOWS
  - Whether the block has fulbright or not.
- =DEFLECTS_LASERS
- =ALLOWS_ZOMBIE_SPAWNS
- =ALLOWS_SKELETON_SPAWNS
- =ALLOWS_ALIEN_SPAWNS
- =ALLOWS_DEMON_SPAWNS



# Exclusive Keywords to: Structure

- $FILE
  - Value must be a path to a *.structure* file (without file extension).
- #RADIUS
  - The distance a structure has to be from another of itself to spawn.
- %CHANCE
  - The chance of the structure spawning every radius.
- "ALIGNMENT
  - *None*
    - No alignment. The structure can spawn mid-air or underground.
  - *Dynamic*
    - The structure is dynamic and aligned to the surface.
  - *Static*
    - The structure is static and aligned to the surface.

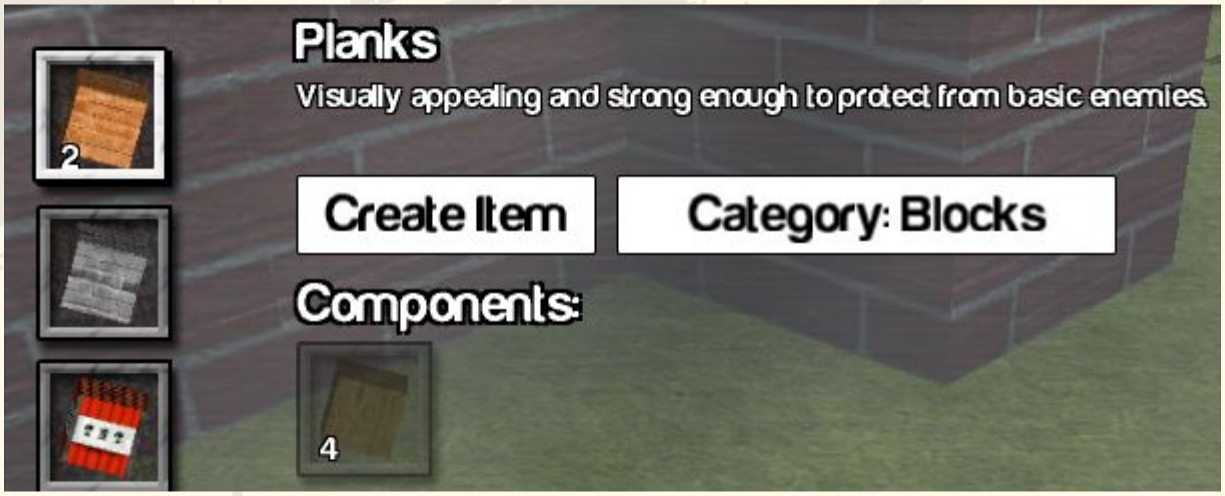


- #HEIGHT_TRANSLATION
  - The height translation of the structure from the point it spawns. Values can be positive or negative. For static aligned structures it is suggested to build a base for them and then translate them into the ground so it looks like they are on smooth terrain instead of floating.
- #MINIMUM_SPAWN_HEIGHT
- #MAXIMUM_SPAWN_HEIGHT
- #MINIMUM_SPAWN_DISTANCE
- #MAXIMUM_SPAWN_DISTANCE
- #MAXIMUM_SPAWNS
  - With this, you can limit how many times your structure can spawn. Don't set this keyword if you want it to spawn infinitely.
- $BLOCK_TO_SPAWN_INSIDE
  - This sets what blocks the structure is allowed to spawn inside of. Do not set if you want your structure to spawn normally. Useful for making dungeons.

## Exclusive Keywords to: Recipe

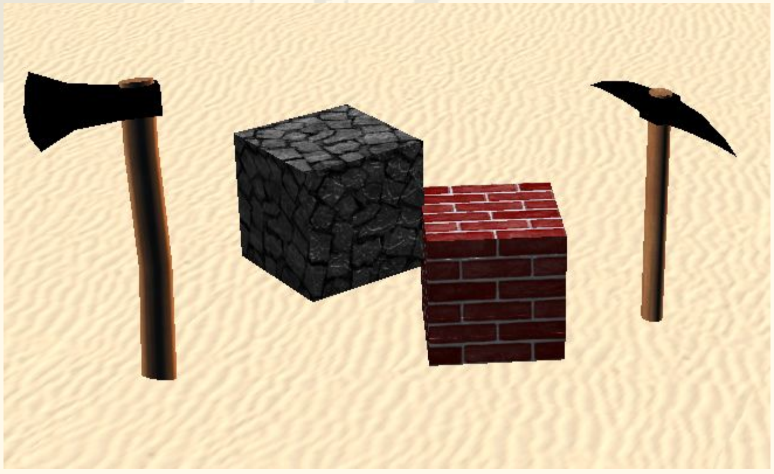
- OUTPUT
  - The item the recipe is for and how much of it is crafted.
  - Expected values look like this: *Diamond Pickaxe=1*
- INPUT
  - The items the recipe needs to craft
  - Expected values look like this: *Stick=2, Diamonds=3*

```
Recipe
OUTPUT: Space Rock Pickaxe=1
INPUT: Diamonds=5, Bloodstone=10, Space Rock=10, Stick=2
$AUTHOR: Salem(FNIX)
```


Planks
Visually appealing and strong enough to protect from basic enemies.
Create Item
Category: Blocks
Components:
2
4

## Exclusive Keywords to: Enemy Loot

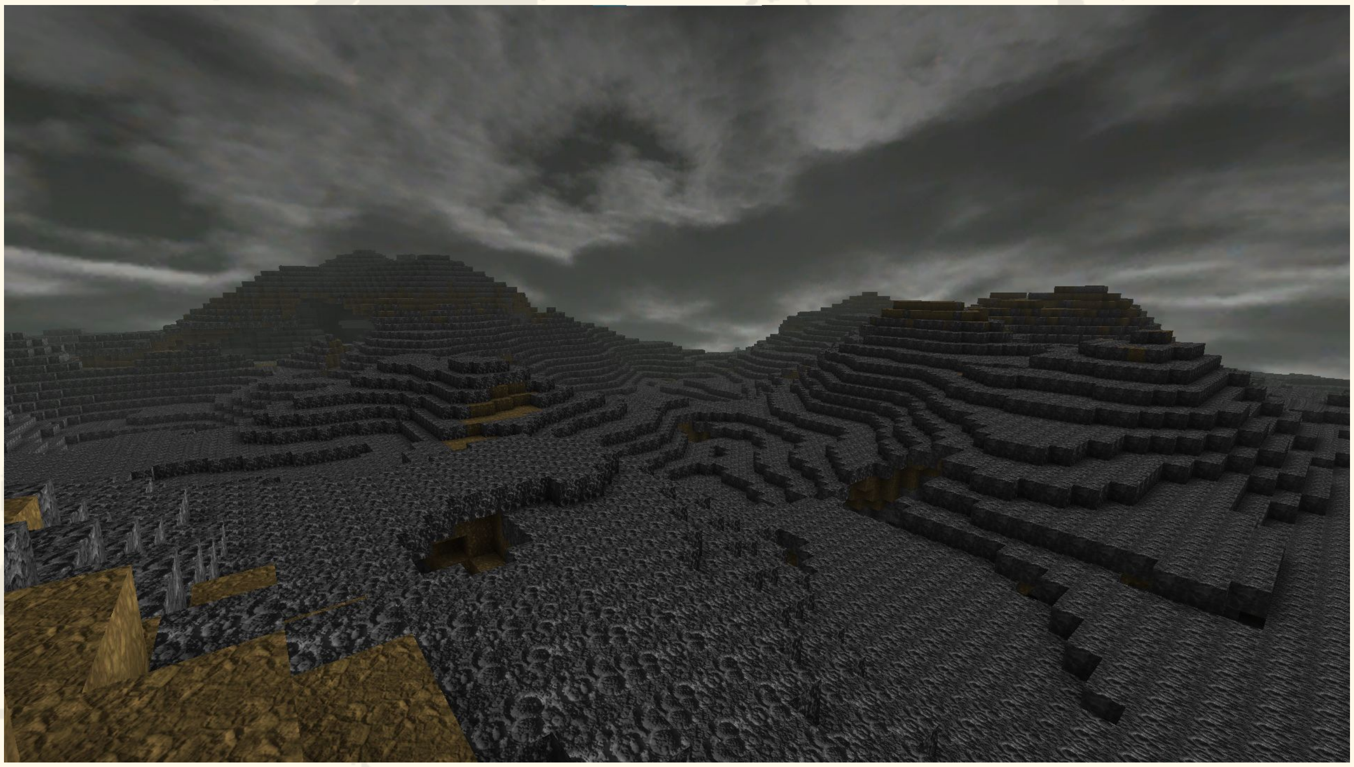
- "ENEMY
  - *Zombie*
  - *Skeleton*
  - *Alien*
  - *Demon*
  - *Fire Dragon*
  - *Forest Dragon*
  - *Sand Dragon*
  - *Ice Dragon*
  - *Undead Dragon*
- “ITEM
  - The name or ID of the item dropped.
- %CHANCE
- #MINIMUM_AMOUNT
- #MAXIMUM_AMOUNT



## Exclusive Keywords to: Block Replacement

Block Replacement tags will replace specific naturally generated blocks. This can be used to replace ores or customize biomes.

- $BLOCK_TO_REPLACE
    - Value must be a block name or ID.
- $BLOCK_TO_REPLACE_WITH
    - Value must be a block name or ID.
- #MINIMUM_SPAWN_HEIGHT
- #MAXIMUM_SPAWN_HEIGHT
- #MINIMUM_SPAWN_DISTANCE
- #MAXIMUM_SPAWN_DISTANCE



## Exclusive Keywords to: Item Blacklist

Blacklisted items will not be visible in the crafting menu, or drop from blocks or enemies. This is useful if you want to replace certain items or make a weapon rebalance pack.

- ITEMS
  - Expected values look like this: *Pistol, Iron Pistol, Stick, Iron Pickaxe*

## Exclusive Keywords to: Recipe Blacklist

Blacklisted recipes will not show up in the crafting menu. This is useful if you want to redo a vanilla item's recipe.

- ITEMS
  - Expected values look like this: *Pistol, Iron Pistol, Stick, Iron Pickaxe*

## Exclusive Keywords to: Loadout Override

Loadout Overrides are used to replace the default spawn loadout with a custom one.

- ITEMS
    - Expected values look like this: *Pistol=1, Iron Pickaxe=1, Torch=30*

## Exclusive Keywords to: Music Override

MusicOverrides are used to replace the default distance tracks with custom ones.

- TRACKS
  - Expected values look like this: *Song1=250, AwesomeSong=1200, SuperAwesomeSong=5000*
  - It’s FilePath=Distance. The first value is the path to the file from the current directory (without the file extension) and the second value is roughly the distance the track will trigger at. Commas separate each track, like other keywords that use similar formats to this (recipes, for example).

## Exclusive Keywords to: Music Packs

Music Packs work exactly like Music Overrides except they do not override vanilla songs and instead are added alongside them.

- TRACKS
  - Expected values look like this: *Song1=250, AwesomeSong=1200, SuperAwesomeSong=5000*
  - It's FilePath=Distance. The first value is the path to the file from the current directory (without the file extension) and the second value is roughly the distance the track will trigger at. Commas separate each track, like other keywords that use similar formats to this (recipes, for example).

Without using any tags, you can also add new Main Menu Themes. All you have to do is add a .MP3 file to your addons folder and start its name with "MENU_". The game will automatically play it as the menu theme. If you have multiple menu themes, the game will play a random one each time.

# Sound Quote List

- *GunShot1*
- *GunShot2*
- *GunShot3*
- *GunShot4*
- *LaserGun1*
- *LaserGun2*
- *LaserGun3*
- *LaserGun4*
- *LaserGun5*
- *RPGLaunch*
- *Reload*
- *AssaultReload*
- *ShotGunReload*
- *Place*
- *pickupitem*
- *Teleport*
- *FootStep*
- *GrenadeArm*
- *Arrow*
- *Beep*
- *SolidTone*
- *Click*
- *Error*
- *craft*
- *Award*
- *Hit*
- *Fall*
- *Douse*
- *Explosion*
- *GroundCrash*
- *CreatureUnearth*
- *ZombieCry*
- *ZombieDig*
- *ZombieGrowl*
- *Skeleton*
- *Alien*
- *Felguard*
- *DragonScream*
- *DragonFall*
- *WingFlap*
- *BulletHitDirt*
- *BulletHitHuman*
- *Fuse*
- *DoorOpen*
- *DoorClose*
- *LightSaberSwing*
- *punch*
- *Sand*
- *leaves*

# Creating Custom Assets

Looking through the keywords, you may notice some refer to external files such as models, textures, structures, or tracks. This page will help with getting those files set up. OpenClassic comes with a Modding Toolbox, which this page will be referring to. If you do not have it, you can download it here.

**Custom Models**

OpenClassic currently does not support custom animations, which means custom items have to reuse existing animations. That means that custom models need to be fitted to the existing model they are templating. In your toolbox you will find a Reference Models folder. Using a 3D Modelling program of your choice, align your custom model to the relevant reference model. You can name a mesh object COLOR1 or COLOR2 for the respective keywords to affect them.

Once your model is ready, export it as a .FBX. Then open the XNBConverter program included in the toolbox. Using the default settings you can convert your exported model to a .XNB file. Now your model is ready to use.

The “BarrelTip” bone included with the reference models is important, it’s where the bullet comes out from ingame.

**Custom Structures**

If you refer to Page 6: “Quick Tips”, you’ll learn about the *+editor* launch argument. Use this to enter the Structure Editor. In the Structure Editor you can build and save a structure easily, with the help of extra instructions ingame. Saved structures will go to the “OpenClassic Data” folder in your game directory (next to the addons folder). Simply move the file into your addons folder wherever you want it to be, and you can then refer to it in tags.

**Other**

Custom tracks for the main menu or ingame must be .MP3 files.

Custom block textures must be 128x128 .PNG files.

Texture packs can be created by editing the example pack in the toolbox. You can name texture packs anything you want as long as they start with “PACK_”, the game will automatically load them. You can create normalspec maps for texture packs by adding “_normalspec” to the end of the filename.

Custom menu backdrops can be created by adding “BACKDROP_” to the start of your image name and placing it anywhere in the addons folder.

Warning: Texture packs are still experimental and may cause visual bugs.

## More on Color Values

Color values support both RGB and vanilla material names. Available material names are listed below.

- Coal
- Wood
- Copper
- Stone
- Iron
- Gold
- Diamond
- Bloodstone
- Brass
- CopperOre
- IronOre