



Barony

[Tutto](#) [Discussioni](#) [Screenshot](#) [Immagini](#) [Trasmissioni](#) [Video](#) [Workshop](#) [Notizie](#) [Guide](#) [Recensioni](#)
[Barony > Guide > Guide di mistersnake](#)


Barony Official Translations Mod Tutorial

Di mistersnake e altri 1 collaboratori



(Servono più voti)

A step-by-step guide for creating a translation mod, replacing every piece of text in the game. We also cover Barony modding and Steam Workshop basics, so you can test and share your work with users or collaborators.

2



Aggiungi ai preferiti

Condividi

Introduction

Thank you for your interest in creating a Barony language translation!

We hope this guide serves as a one-stop-shop for all the requirements, process, and questions you might have about making your very own Barony language translation.

CREATO DA

mistersnake
OfflineWALL OF JUSTICE
Offline

As of update 4.0.2, all of game's text has been migrated into data. In other words, none of the game text is hard coded. This allows for complete translations using standard Barony content modding process!

(If you find text you can't replace, please let us know in the comments!)

Before You Begin

Text Editing Software

Your operating system's default text editor is all you need. For serious text data editing in Windows, we recommend Notepad++. It's a lightweight and feature-rich editor. The most obvious reason to use Notepad++ is its browser-like tab interface, allowing you to have all of your translation files open at once, and easy to navigate. It also numbers your text lines, allows for color schemes, and color formatting options, which is especially handy for modifying JSON files. This guide won't teach you how to use all of Notepad++'s features, but all the text editor screenshots shown in this guide are from Notepad++, so it may make it easier to follow along.

```
C:\Program Files (x86)\Steam\steamapps\common\Barony\lang\en.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
en.txt
1 # English language file
2 # all entries must contain a string number, followed by a space, followed by a string, ended by a pound character
3 # lines beginning with pound character are a comment
4 # blank lines are ignored
5
6 1 null#
7
8 # class names
9
10 1900 barbarian#
11 1901 warrior#
```

Notepad++ is free! Get it here: [Download Notepad++ \[notepad-plus-plus.org\]](http://notepad-plus-plus.org)

Visual Studio Code [\[code.visualstudio.com\]](https://code.visualstudio.com) is also suitable for editing the kind of text we're dealing with and is available for free on other platforms, including Mac and Linux.

Font Editing Software

Languages which use characters beyond the English latin character set (Chinese, Arabic, Hindi, etc.), or even those which require a few more accent markers, an updated font may be needed to display text correctly in Barony. While you may replace Barony's fonts with any of your choosing, the style and size may differ from the game's native font and as a result you may wish to edit Barony's fonts so that they can display your desired characters!

This guide won't teach you how to create and modify fonts, but if you intend to get into that,

Categoria: Creazione di mod o configurazione

Languages: Inglese

Pubblicato in data 27 set 2023, ore 9:22

Aggiornato in data 22 ott 2023, ore 20:47

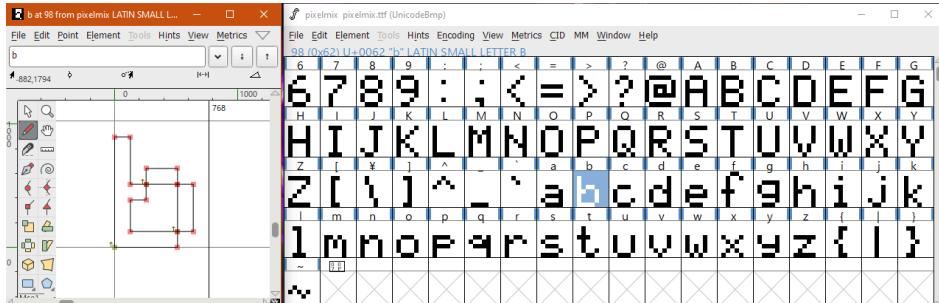
1,076 Visite uniche

24 Preferiti

INDICE DELLA GUIDA

Sommario

[Introduction](#)
[Setting Up Your Language Mod](#)
[Editing Text Data](#)
[Testing Your Translation Mod](#)
[Editing JSON Data](#)
[JSON Tools](#)
[Replacing Fonts](#)
[Language Content List](#)
[Giving Your Mod an Icon](#)
[Sharing Your Translation with the World!](#)
[Make Your Translation Part of Barony](#)
[FAQ](#)



Commenti

FontForge can help get you started. And of course, it is also free and will work on Windows, Mac & Linux!

Download FontForge here: [Download \[fontforge.org\]](http://fontforge.org)

Barony Translations Community

A translation may be a big project and we encourage you to join our community Discord if seeking contributors or other technical help. Join the Barony Discord Translations channel here!:

[Barony Discord Server - Translations General \[discord.gg\]](#)

The Barony Steam Discussion boards thread on this subject may also be a great place to reach out:

[Barony Translations Steam Discussion](#)

With those introductions out of the way, let's get to the game modding!

Setting Up Your Language Mod

Let's start by creating the bones of your translation project. These steps will allow you to organize the work that needs to be done, test your work along the way, and ultimately prepare for putting your completed translation on the Steam Workshop!

Note that these first steps are the same for any mod, no matter what kind of content you're replacing.

Creating the Mod's File Structure

- Launch Barony
- Navigate to the Main Menu and select "Start Modded Game"



- Select the "Local Mods" tab on the top-left of the menu
- Click the "Create Blank Mod Folder" button in the bottom left



- In the popup, enter the title for your translation



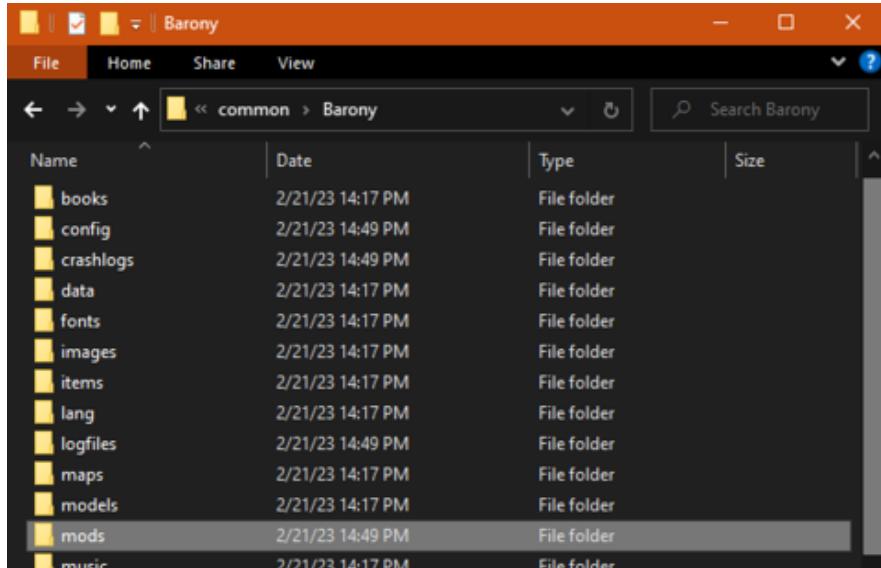
In your "/steamapps/common/Barony/mods/" directory, the game has now created a folder for

your mod, with the name you've entered.

To navigate quickly to your Barony mod folders:

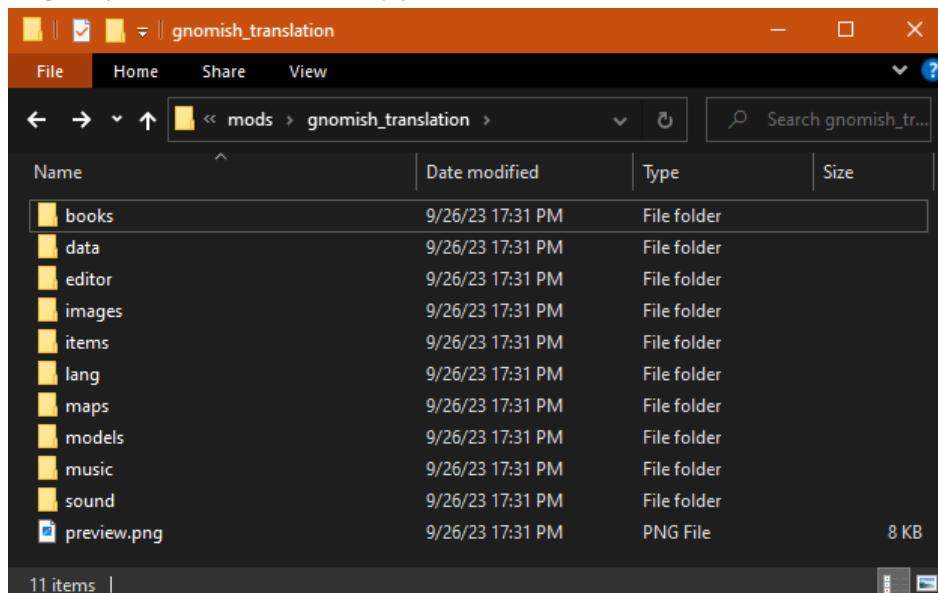
1. Right click on "Barony" in your Steam Library
2. Go to the "Manage" menu item
3. Select "Browse Local Files"

This opens the folder where Barony is installed!



4. Open the "mods" folder and find your mod folder within

Within your new mod folder, is a copy of the Barony file structure, and a placeholder preview image for your mod. It is otherwise empty!



If you want to tidy up a bit, you're welcome to delete any folders which you don't intend to translate, or don't have any content to translate! EG:
/editor, /maps, /models, /music

Our final setup step is to copy the game data that we want our translation to change!

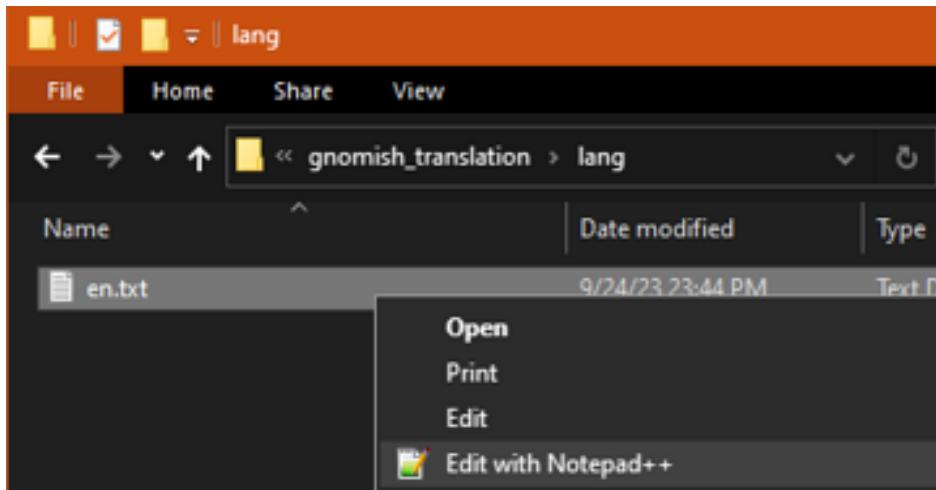
The process is simple. Just find the original file, copy it, and paste it into the duplicate file structure of your mod's folders. When the game loads your mod, it loads your duplicate in place of the original file.

Some modders may find it simpler to copy over one file at a time, making it obvious what translation work has been completed.

Editing Text Data

Let's start by changing the Pause Menu text, as that will be easy to test and confirm that your mod is working as expected! If you haven't already:

1. Locate `/Barony/lang/en.txt` and **copy it**
2. Go to `/Barony/mods/*yourmodname*/lang` and **paste** the `en.txt` file
3. Open your mod's `en.txt` file in your text editor



Let's edit the Pause Menu option called "**QUIT TO MAIN MENU**". To locate this text scroll down to line 7366 (displayed on the left side in Notepad++) or just **search** for the terms "QUIT TO MAIN MENU".

If you use the search / find tool, you may find multiple instances of "QUIT TO MAIN MENU" in the `en.txt` file. Sometimes there is redundant text from different parts of the game. Might as well replace them bo

```
7363 5768 RESTART GAME#
7364 5769 RESTART TRIAL#
7365 5770 RETURN TO TRIAL HUB#
7366 5771 QUIT TO MAIN MENU#
7367 5772 QUIT TO DESKTOP#
7368 5773 PLAY#
```

Alright, we found the line we wanted to change in our text file, so all we gotta do is replace the existing text with the text we want. In our sample mod, we're making a gnomish language translation, as shown:

Save your modded `en.txt` file, and now it's time for us to go test our change!

Let's talk about how to do that in the next section.

Testing Your Translation Mod

Once you have replaced some game content, it's time to make sure it shows up as you expect!

Back in the **Play Modded Game** menu, you should be able to find your translation efforts in the Local Mods list. Just click "**Load Mod**"!



You can load as many mods as you like! Just keep in mind that the most recent mod loaded takes priority if multiple mods change the same content.

With your mod loaded, select "**Start Modded Game**".



The game will quickly load your mod. Once it's done, any of the content you've replaced will be shown instead of the base game content. Your translation is now testable!

Keep in mind that cheats might help you test your content more quickly.

In our "`en.txt`" translation edit, we changed the "QUIT TO MAIN MENU" button in the Pause Menu to the Gnomish language, saying "JIB JEB JAB JIB JAB JUB" instead. Here's what it looks like in-game!

The good news is that our mod works! The bad news is that our last "JUB" is getting cut off. This is because the gnomish translation is too long for this button. When text is too long in Barony, it



will either get cut off, or run outside the boundaries of the text display area, and both aren't great!

This highlights the importance of using the existing language to help inform appropriate length, and if there are close-calls, testing is your best friend! We'll need to adjust the wording in Gnomish to make sure the game is still correct.

Perhaps JIB JAB JEB JUB will communicate the same idea in fewer words.

Keep the importance of testing in mind as you continue your translation efforts.

Editing JSON Data

Translating text in JSONs is similar to translating text in TXT files. While there is a little syntax mixed in which can change how your text is displayed, you don't need to be a seasoned programmer to follow the rules and get a good result. Most of what you'll deal with is just plain text, and can be translated like anything else.

We'll try to give you some examples and instruction to get you started.

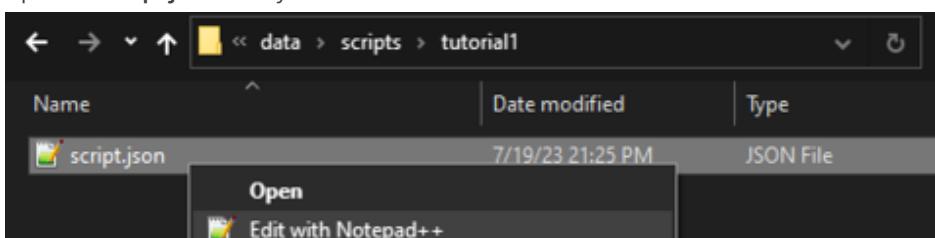
Tutorial Signs

Barony's hard enough without the tutorial signs being in a different language! Let's start with JSON editing by modifying Barony's first tutorial sign. If you haven't already, let's copy that JSON over to our mod.

Navigate to `/Barony/data/scripts/tutorial1` and copy `script.json`.

Next we'll go back to `/Barony/mods/*yourmodname*/data/scripts/tutorial1` and paste it there.

Open that **script.json** file in your text editor.



Let's scroll down to the first tutorial sign script definition, labeled "tutorial1z".

There is a whole lot of stuff you can ignore if your goal is just to change the text. The JSON starts by defining the colors of the font, and then proceeds to outline each sign as a script definition. You just need to find the sign you want to modify, and alter its text. Unless you're familiar with JSON, it's fine to just ignore the rest.

In fact, it's important to ignore the rest. Don't modify any text that you're not certain appears in the game! Altering variable names may prevent the JSON from loading properly, and translating them is not useful!

Before you start, it might also be helpful to know that JSON scripts use indentations, commas, and various brackets to help organize when sections of instructions start and finish. It's important to respect the beginnings and ends of script entries by paying attention to where your brackets and commas are. Modifying just the middles of scripts will help you avoid formatting errors that will prevent your JSON from working properly.

This JSON has the first tutorial's sign listed first (though they may be listed in any order).

```
30
31
32
33
34
35
36
37
38
    "tutorials": [
        "sign": [
            "Use$ keys",
            "to walk around.",
            "",
            "Move$ to look and turn.",
            "",
            "Press$ to interact with",
            "objects in the world."
```

Ah, we recognize this sign text! This is good to edit.

Translating this is simple, though we need to keep the size of the signs in mind so as not to run off the page.

off the ends. The quotes (" ") inside the script entry tells you where the text stops and starts. The comma (,) tells you when the line is over, and we move to the next line. By respecting the length of each line, and where empty lines ("") are, we can gain an understanding of how signs are spaced so that they are readable.

Unmodified, here's what that sign looks like (when using a mouse & keyboard):



Doing a simple translation into Gnomish, this is our result in the JSON:

```

31   "sign": [
32     "Jub$JabsJab$Jab$",
33     "Jib jab jib jub.",
34     "",
35     "Jub$ jub jab jab jib.",
36     "",
37     "Jub$ jeb jabjub jab",
38     "jib jab jib jub jeb."
39   ],

```

You'll see \$ symbols in the text. These are used to signify where glyphs (icons) are being placed within the lines of text. Each \$ is replaced with an image, in the order by which each glyph is defined in the "**variables**" section of the script entry. If this sounds like gibberish, that's OK.

The important takeaway is that you can adjust wording between each \$ (**glyph**), but if you want to change the order that each glyph is displayed, you will need to modify its order of appearance in the **variables** section of the script entry. If you don't want to get into that, we recommend you just leave each \$ in its current order, and work your translation around that!

Placing some gnomish language between the glyphs of the original text, we can see that our simple replacement translation has worked well!

However, it looks like we need to take note of where text is **highlighted** on the sign. Where previously each **highlighted** word was highlighted according to a word which indicated it was a gameplay term, those highlights are not correctly ordered when translated to Gnomish!

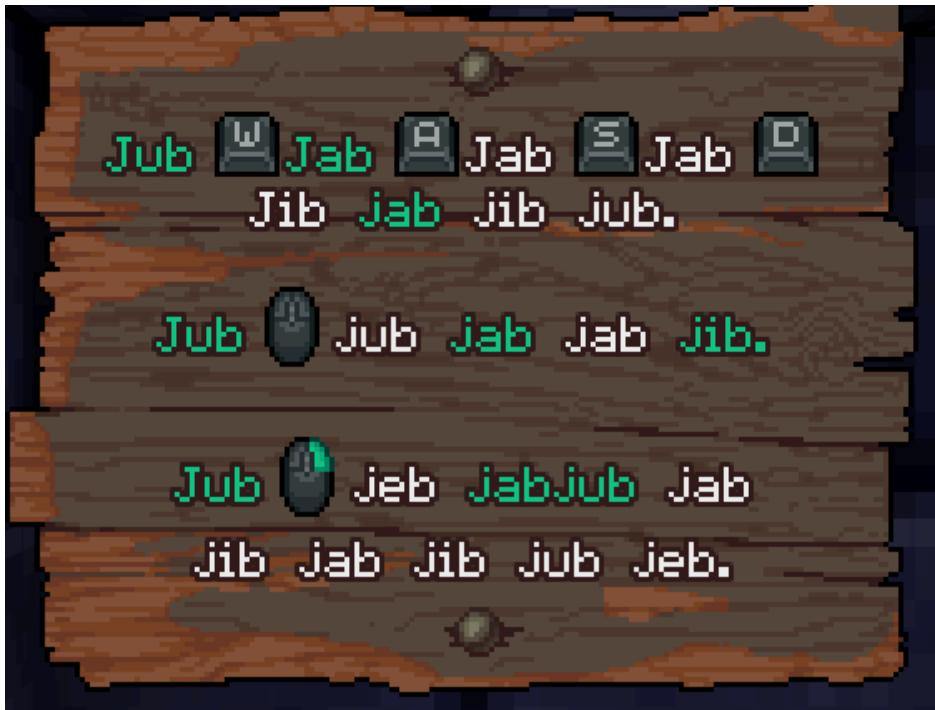
For the purposes of this tutorial, Gnomes use the word "jub" when describing a gameplay mechanism, and that should always be highlighted to help draw the attention of the player. Let's fix this sign with that in mind. The place to correct this is the "**word highlights**" section of the script entry.

If you wanted to avoid the hassle of managing highlights, you could simply delete the **word highlights** section (from line 47 to line 54) and then no words would be highlighted

The "**word_highlights**" section replicates the format of each line of the sign text, but uses a list of **bracketed numbers** to define which word we are highlighting.

The JSON defines which word is highlighted in each line within the "**word highlights**" section by listing the number of the word we want to highlight.

To fix the highlights, the first thing to know is that counting in Barony starts from 0, not 1. So when we see the number 0 in the word highlights section of the JSON, we're highlighting the first word. Glyphs don't count as words, so based on what our sign says, we want to highlight word 0, 1, 2 & 3 in the first row (since each of those words is "jub"!).



```

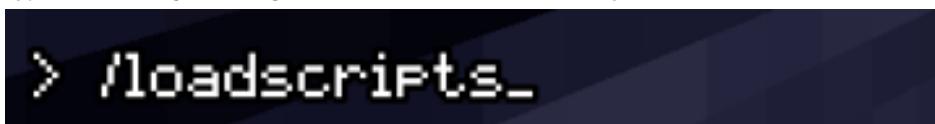
47   "word_highlights": [
48     [0,1],
49     [1],
50     [],
51     [0,2,4],
52     [],
53     [0, 2]
      "word_highlights": [
        [0,1,2,3],

```

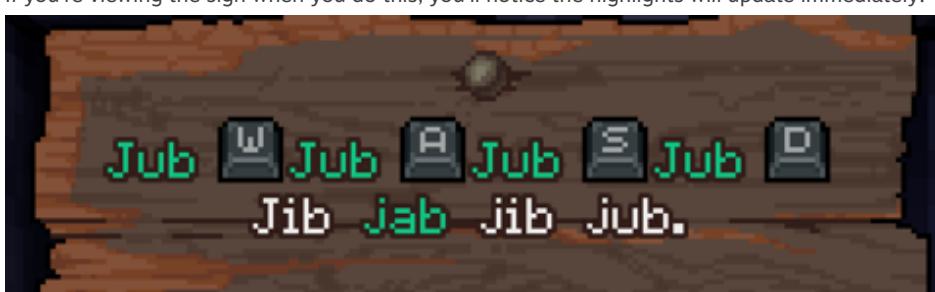
Now that we've updated the "word_highlights" section for line 1, let's save this JSON, and then check it out in-game.

We don't have to close and relaunch the game to reload the script! Let's use a console command (or "cheat") to reload the game's scripts.

Type the following into the game's chat and hit enter: `/loadscripts`



If you're viewing the sign when you do this, you'll notice the highlights will update immediately!



Looks like it worked perfectly! Let's change our highlights so that only **jub** is highlighted for the entire sign.

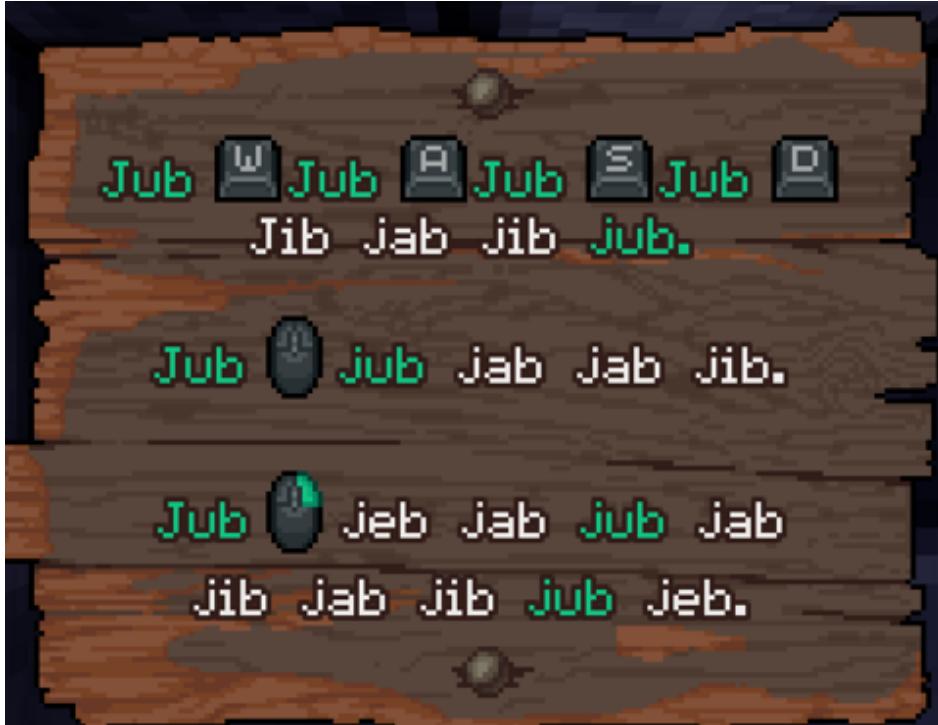
Here is what our highlights now look like in our JSON for this sign's script entry:

```

47   "word_highlights": [
48     [0,1,2,3],
49     [3],
50     [],
51     [0,1],
52     [],
53     [0, 3],
      "word_highlights": [
        [3],

```

And after doing another `/loadscripts` command, here's what it looks like in-game!



Perfect!

This sign has now been perfectly translated from English into Gnomish. We just need to repeat the process with the other signs in the tutorial script to complete our tutorial translation.

Some signs may have two highlight colors. The first color refers to the `word_highlights` section to determine which words are highlighted, as we've already covered in this tutorial. The second color refers to `word_highlights2`. Number each word in its respective section to assign which words is assigned with which color! But the process is the same for each.

Other JSON Formats

Other JSON menus may format text a little differently, but aren't too far off what we described from the signs. A good clue is to look for the plain text within quotes, modify it, and check the results. For example, this excerpt from the `follower_wheel.json`:

```
"action": "tinker_detect_metal",
"id": 5,
"path": "#*images/ui/FollowerWheel/icons/tinker/btn_tinker_detect-scrap_off.png",
"path_hover": "#*images/ui/FollowerWheel/icons/tinker/btn_tinker_detect-scrap_hover.png",
"path_active": "#*images/ui/FollowerWheel/icons/tinker/btn_tinker_detect-scrap_on.png",
"path_active_hover": "#*images/ui/FollowerWheel/icons/tinker/btn_tinker_detect-scrap_hover-on.png",
"text_maps": [
    {
        "default": [
            {
                "text": "Detect items with: [Metal Scrap]",
                "word_highlights": [3,4]
            }
        ]
    }
]
```

The area labeled "`text`": is where you can find your plain language to replace, with the word highlights following on the next line. Let us know if you run into any confusing JSON formatting in the comments and we'll be happy to help and/or update the guide!

JSON Tools

Assistance Debugging JSON

If you're not a code monkey, a lot of the syntax for JSON might be a little confusing, and it's normal to make some mistakes from time to time. If you try to use `/loadscripts` and your sign stops displaying altogether, it means your JSON formatting has an error and can't load properly. It may be wise to simply undo your most recent change, and try your adjustment again, being careful to avoid messing with commas or brackets.

If you keep running into a problem, however, it may be useful to use a **JSON validator**.

By using a JSON validator tool, you can catch some errors that might otherwise be pretty obtuse.

Here's a link to one we use from time to time:

[Online JSON Validator / Editor \[jsoneditoronline.org\]](https://jsoneditoronline.org/)

To make use of the validator, simply copy/paste the contents of your JSON file into the text field,

and a red box will appear at the bottom, showing you where an error is detected. It won't always give you the solution, but it can be a good place to start!

Here the JSON validator detects where I've mistakenly added a comma that breaks the script. The

Auto Repair button correctly removes it and the JSON works great again!

```

844 v
845     {
846         "text": "Command unavailable for %s",
847         "word_highlights" : [0,1,2,3,4,5,6,7,8,9],
848     },
849 v
850     "rest_cooldown" :
851         {
852             "text": "You may only use this command once per floor",
853             "word_highlights" : [0,1,2,3,4,5,6,7,8,9]
854 v
855         },
856         "tinker_quality_low" :
857             {
858                 "text": "Higher quality %s required",
859                 "word_highlights" : [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
860             }
861 }

```

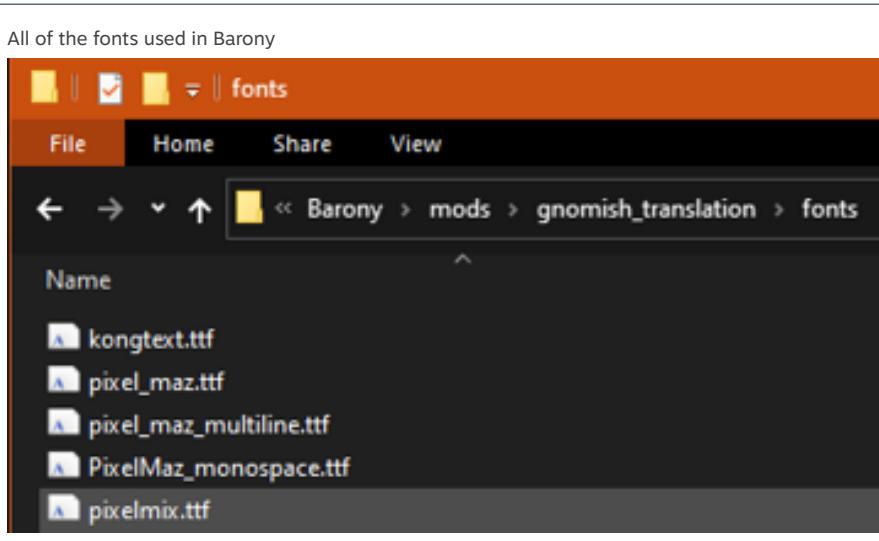
Line: 846 Column: 47

⚠ Quoted object key expected but got ')' at line 847 column 4

Auto repair Show me

Replacing Fonts

In order to replace the game's fonts, you may need to manually create the "fonts" folder in your mod's directory. From there, you may take any font and rename it to replace an existing font within Barony. Just match the name of the existing font, and you're good to go!



By replacing a font like "pixel_maz.ttf" with a system font like Arial or Times New Roman, you'll immediately gain support for non-English characters. However you may find that the size of the replacement font doesn't fit correctly within the constraints of window or button borders, or that the style might be off-putting.

Customizing the game's font directly with a tool like FontForge may be the best way to achieve high-quality results!

Language Content List

Now that you know how to modify TXT and JSON data files to translate Barony, refer to the following list for a summary of all the game's accessible readable and audible language data:

General Game Text
/Barony/lang/en.txt

Item Names and Tooltips
/Barony/lang/item_names.json
/Barony/items/item_tooltips.json

Cutscene Story Data
/Barony/data/story/ (all *.json files)

Books

/Barony/books/ (all *.txt files)

Menu Text Data

/Barony/data/callout_wheel.json
 /Barony/data/charsheet.json
 /Barony/data/class_descriptions.json
 /Barony/data/follower_wheel.json
 /Barony/data/HUD_settings.json
 /Barony/data/monster_data.json
 /Barony/data/race_descriptions.json
 /Barony/data/skillsheet_entries.json
 /Barony/data/skillsheet_leadership_entries.json
 /Barony/data/status_effects.json
 /Barony/data/tutorial_strings.json

Tutorial Sign Text

/Barony/data/scripts/tutorial_hub/script.json
 /Barony/data/scripts/tutorial1/script.json
 /Barony/data/scripts/tutorial2/script.json
 /Barony/data/scripts/tutorial3/script.json
 /Barony/data/scripts/tutorial4/script.json
 /Barony/data/scripts/tutorial5/script.json
 /Barony/data/scripts/tutorial6/script.json
 /Barony/data/scripts/tutorial7/script.json
 /Barony/data/scripts/tutorial8/script.json
 /Barony/data/scripts/tutorial9/script.json
 /Barony/data/scripts/tutorial10/script.json

Fonts

/Barony/fonts/kongtext.ttf
 /Barony/fonts/pixel_maz.ttf
 /Barony/fonts/pixel_maz_multiline.ttf
 /Barony/fonts/PixelMaz_monospace.ttf
 /Barony/fonts/pixelmix.ttf

Voice Acting

/Barony/sound/erudyce/ (all *.ogg files)
 /Barony/sound/orpheus/ (all *.ogg files)
 /Barony/sound/ (all Herx-*.ogg files)

Once the most important of these files are replaced with the proper language, you have yourself a completed translation!

Giving Your Mod an Icon

When we created your translation's basic structure, we got a placeholder "**preview.png**" file in your **"/Barony/mods/*yourmodname*** directory. This placeholder is just an image of the Barony Shield icon. This image will display in the Steam Workshop, and in the mod list when your mod is loaded in-game.

It's a good idea to replace this with an image that represents your mod well!

Many translations have used the Barony logo with the orange shield painted to reflect the country of the language represented, or have simply placed a flag behind the Barony shield.

Using your favorite image editing software, simply replace the 512x512 PNG file with your mod's icon, and it'll be properly represented when you upload your finished mod!

Sharing Your Translation with the World!

Once you've completed your translation, tested it, and are happy with the results, it's a good time to share it with the community! Return to the "**Play Modded Game**" menu, select the "**My Workshop Items**" tab.

This displays a list of all the Workshop items you've created (if any). On the bottom right of that menu, select the "**New Workshop Mod**" button.

The game will then open up a popup menu asking you to:

- Select the folder you want to upload from (this is your **Barony/mods/*yourmodname*** folder)

- Give the mod a name
- Give the mod a brief description (modifiable directly from its Steam Workshop page)
- Select appropriate tags for your mod (likely just the "translations" checkbox)

With those details accounted for, you can now upload your mod to the Workshop!

For your mod to be fully visible, you'll want to go to the page where it lives on the Steam Workshop and share it with the world. From here you can also upload images, assign videos, or share it with a smaller audience. This is especially useful if you are collaborating with multiple people on the same translation project!

Once your mod is available on the Workshop, you can choose to enable / disable either the **Local Mods** version (good for continued local editing), or the **Workshop** version and Barony will load either of them as an active mod.

Make Your Translation Part of Barony

If you're satisfied with your mod and would like to consider offering Turning Wheel LLC permission to distribute it as part of the main game, please let us know in this thread. We also regularly check our Translations Discussion on Discord [discord.gg] .

FAQ

Some messages in Barony do not support proper grammar in my language for translation. What can I do?

We currently have no consistent solution to account for this. Barony's log messages, in particular, may use "Mad Libs" style sentence construction, intended for English grammar rules. The best we can suggest right now would be to rewrite these lines so that they are as good as they can be within your language's grammatical constraints.

9 commenti Segui discussione (?)



Aggiungi un commento



Engicop 20 nov 2025, ore 19:31

Ohh thank you!



mistersneak [autore] 20 nov 2025, ore 19:02

Yes you can find all of the Compendium language entries in:
"\Barony\lang\compendium_lang"



Engicop 20 nov 2025, ore 17:59

Is the compendium translatable? I couldn't find the files for it.



Andrea h2o 1 lug 2025, ore 17:27

Is there any Italian translation?



Tinga 22 feb 2025, ore 7:58

Would making a mod that edits the en.txt file conflict with any other active mods that also make edits to the en.txt file?

I'm replacing herx's voicelines and want to make the text match if that helps. I already have a couple mods installed that likely make some changes to en.txt so I'm wondering if it'd even be worth the effort of matching the text.
(sorry for crossposting)



Rid_ll 18 nov 2024, ore 6:08

Thanks for the reply



mistersneak [autore] 17 nov 2024, ore 22:01

Sorry Rid, those are created in the level file. This is something we should have corrected on our end by referring instead to a JSON. I think it would be wise to leave alone signs like those until we fix them. You can modify the level file yourself but that will invalidate achievements since the level will be modded. I will find a time to correct it eventually!



Rid_ll 16 nov 2024, ore 22:57

Could you please tell me how to translate these labels? I couldn't find what controls this in the game files.

<https://imgur.com/a/NJWFxwl>



Rafael M. 15 mar 2024, ore 20:02

Thanks man!!!!!! Helped a lot!



© Valve Corporation. Tutti i diritti riservati. Tutti i marchi appartengono ai rispettivi proprietari negli Stati Uniti e in altri Paesi. Alcuni dati geospatiali su questo sito sono forniti da geonames.org.

[Informativa sulla privacy](#) | [Informazioni legali](#) | [Accessibilità](#) | [Contratto di sottoscrizione a Steam](#) | [Cookie](#)