

Localization Tool

Documentation

Quick instructions

There are 3 main elements in the tool:

- The *Editor Tool* window
- The *Localization Manager* prefab
- The *Localization Object* component

The *Editor Tool* window can be found in Tools/ArthemeyDevelopment/LocalizationTool/EditorTool, where you can create, open and modify language files. The files must be in JSON or CSV format. You can also add the *Localization Object* component to the objects in your scene and assign a *key*.

The *Localization Manager* prefab is where you will drag and drop your language file.

The file must be in the *Streaming Assets* folder. You can create the file with the *Editor Tool* or with external tools.

The *Localization Object* component is in charge of localize every element in your game, you need to add it to every object that is going to be localized and set the *key*, you can do this manually or with the *Editor Tool*.

After adding the component you need to select the type of object you want to localize and different options will be shown to you depending on the object you select, for a detail explanation on every option go to the *Localization Object* section in the documentation.

If you have any issue or question about the tool, want to reuse the code of my tool to create your own or have any suggestion please contact me at arthemydevelopment@gmail.com.

Editor Tool

The Editor Tool window is divided into two sections, File Management and Object Management.

In File Management you can create, load, or save the language file you will use in your project. If you are going to load a file, you must first select the file format and make sure it is in the Streaming Assets folder.

▼ Add or modify a new file

File Format Json

Create New Language FileLoad Language Data

After that, a list of the elements in the file will appear. The left column contains the keys, this are the reference values used to define the objects. The right columns contains the values, this is the text that will be set in the objects, each language has its own column. The first element is reserved for the language list.

▼ Data Management

▼ Localization Data

▼ LLItems

Key	Value	English	Spanish	Italian	New Language
Menu_Title	Value	Example Scene	Escena de ejemplo	Scena di esempio	
Menu_Img	Value	ArthemDevelopment/ExampleImage_EN	ArthemDevelopment/ExampleImage_ES	ArthemDevelopment/ExampleImage_IT	
Menu_Start	Value	Start	Comenzar	Inizio	
Menu_Continue	Value	Continue	Continuar	Continua	
Menu_Options	Value	Options	Opciones	Opzioni	
Menu_Credits	Value	Credits	Creditos	Titoli di coda	
Menu_Copyright	Value	Copyright ArthemDevelopment 2021	Copyright ArthemDevelopment 2021	Copyright ArthemDevelopment 2021	

Add New Language

Remove Last Language

Save Language Data

If you already have a loaded file, or you create one, the “Add New Language”, “Remove Last Language” and “Save Language Data” buttons will appear.

The first and second buttons are used to define how many languages the file will hold. This can also be change with external tools.

The third one allows you to save your changes into the same file or a new one.

The “Remember Configuration” of the language selection option and the “Trigger Automatically” of the custom event option in the “Localization Object” store these options in a PlayerPrefs. You can see these values or reset them in the Debugging foldout.

▼ Debugging

For testing purposes you will need to reset the default Index language and the Custom Event Trigger setting, otherwise, if you set the modifier of remembering the language selection or the automatically trigger option, the custom events for the buttons will trigger automatically every time.

Clear Language indexClear Custom Event trigger

Current default language option: Spanish

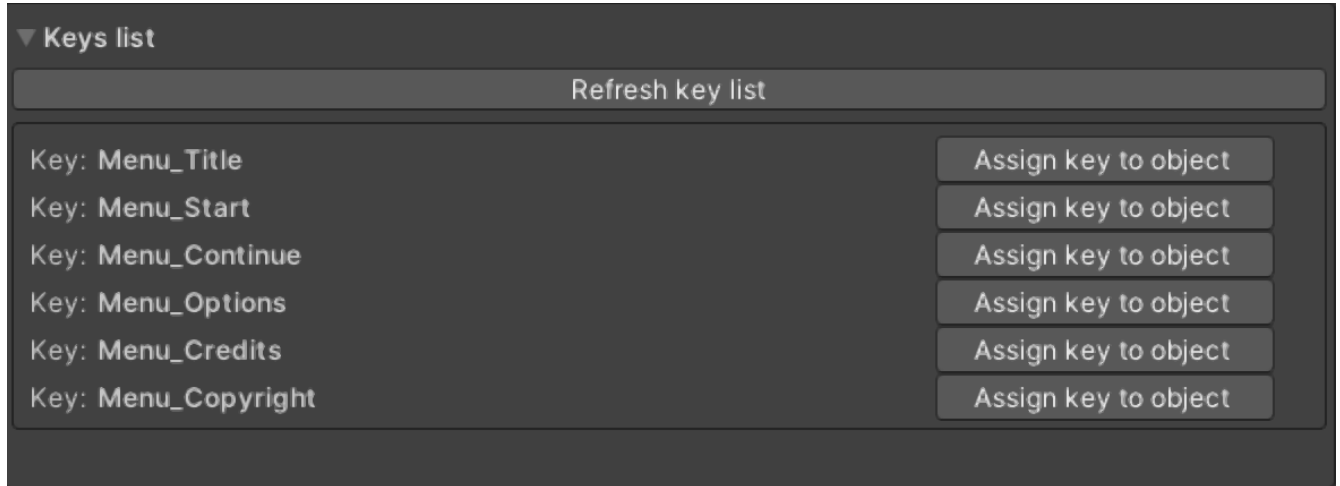
Custom Trigger index 0Is custom event trigger: false



Editor Tool

In Object Management you can assign the Localization Object component and a specific key to any object in your scene.

After you create a file or load one, you can update the key list. Here you can see all the existing keys in your file.



To add the Localization Object component and assign the key select the object in your scene and press the Assign key to object button.



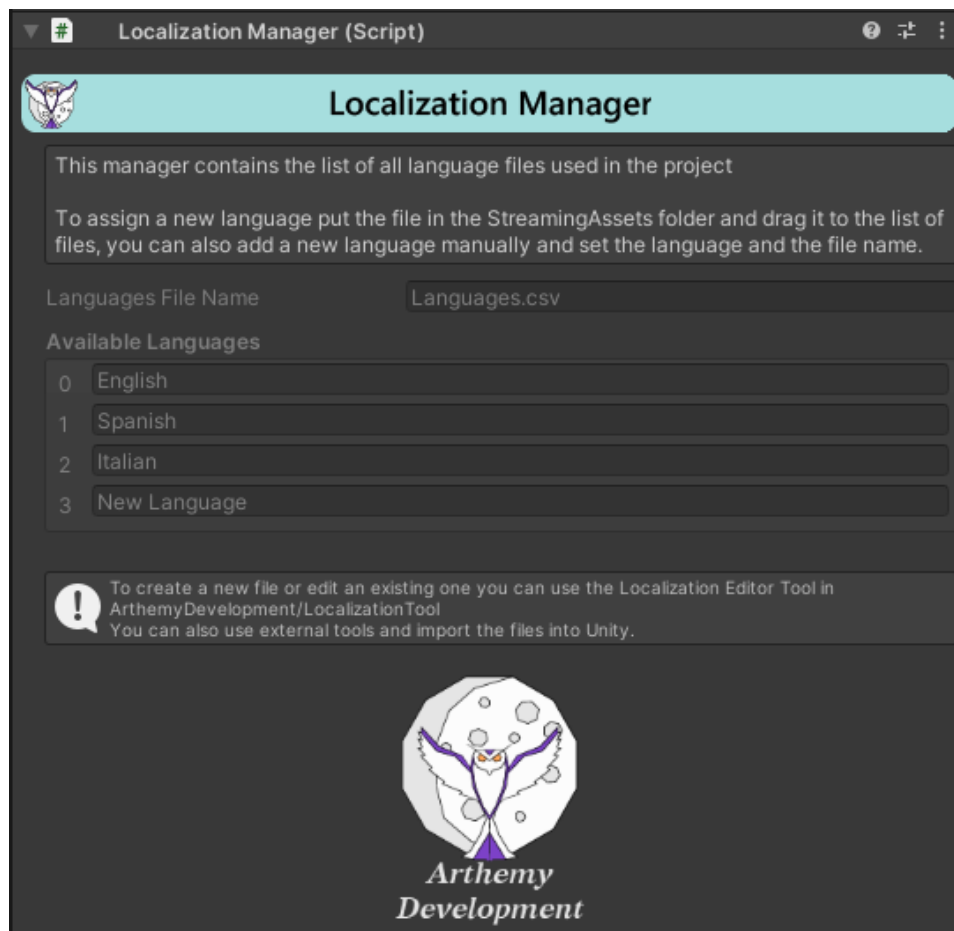
Localization Manager

The Localization Manager prefab contains the Localization Manager component, you can add this component to another object or put the prefab in your scene. The component has the DontDestroyOnLoad behavior and must be instantiated in the first scene of your game.

There are two main values that must be set here.

The first one is the "Language File Name", this is the name of the language file you should have in the StreamingAssets folder.

The second one is the list of languages your file support. Each language has an index value that is used to set the active language on runtime.



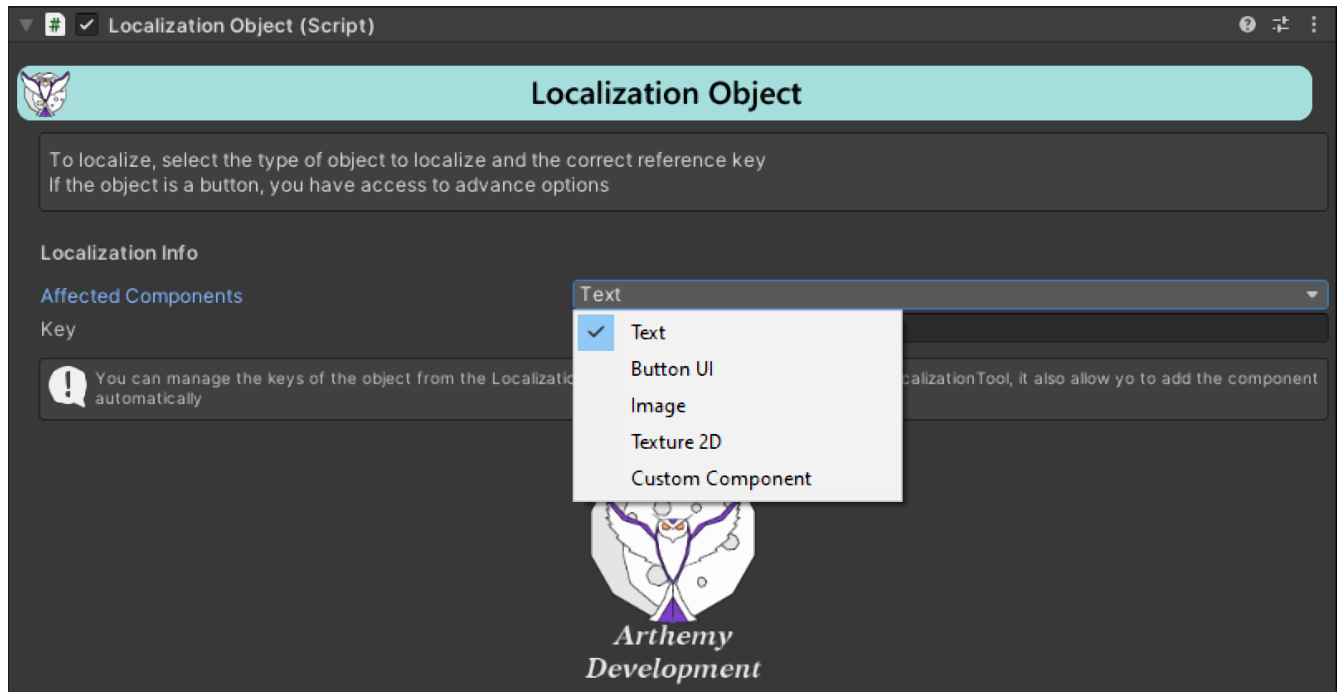
To set the values drag and drop the language file from the StreamingAssets folder into the component. The language file name and all the available languages on the file will automatically be detected.



Localization Object

The Localization Object component is the main element in the tool, this component is where you will configure everything to localize the objects in your scene.

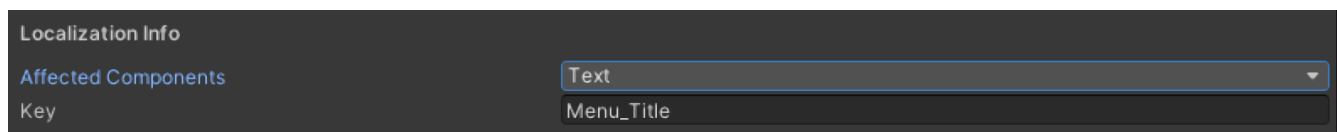
The first thing you need to select is the type of object you are localizing and the reference key.



The supported object types are

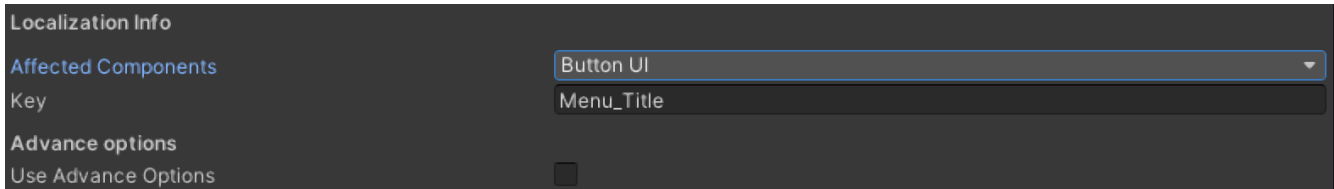
- TMPro/Unity Text
- UI Button
- UI Image
- Texture 2D
- Custom Component

For the TMPro/Unity Text, the value of the key must be the translated text you want to use.



Localization Object

For UI Button the value of the key must be the translated text that is gonna be on the TMPro/Unity text as a child in the button object.



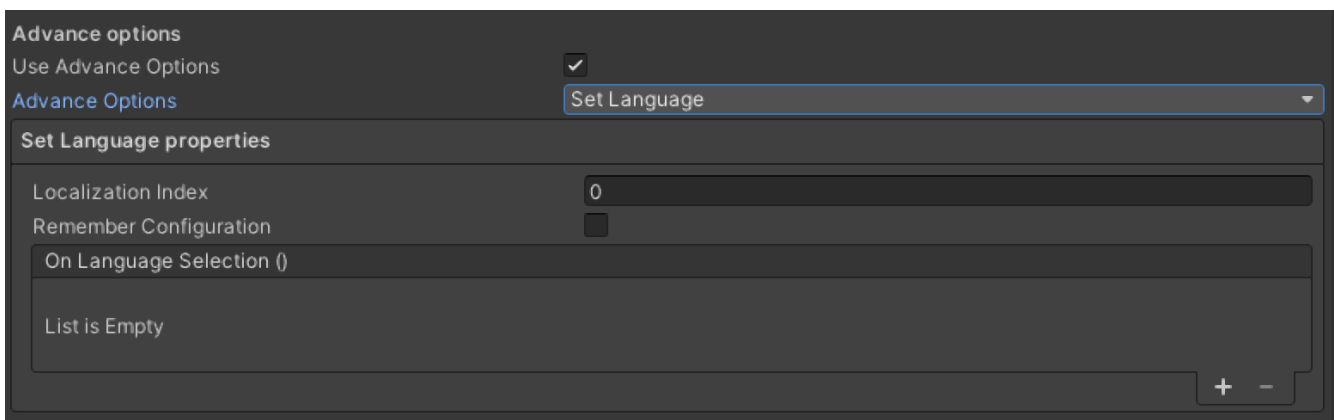
You will also have access to advance options for the button, these options are:

Language Selection: This option allow you to use the button to set the active language in your game, with this you don't need a reference key and the text in the button will be the language name that shows up on the Localization Manager component, instead you need to use the language index value. You can also set it to remember the selected language between sessions, this will automatically set the language when the button is enabled, also you can assign methods to be triggered in the `OnLanguageSelection()` event.

The difference between this event and the `OnClick()` event on the button component is:

- All the methods in the `OnLanguageSelection()` will be triggered at the end of the `OnClick()` event.
- In case you check the Remember Configuration options, only the events in the `OnLanguageSelection()` will be triggered when the button is enabled. The `OnClick()` events of the button will not be triggered.

To change the language in runtime, you will need to call the `ClearPlayerPrefKey` method in the Localization Object script.

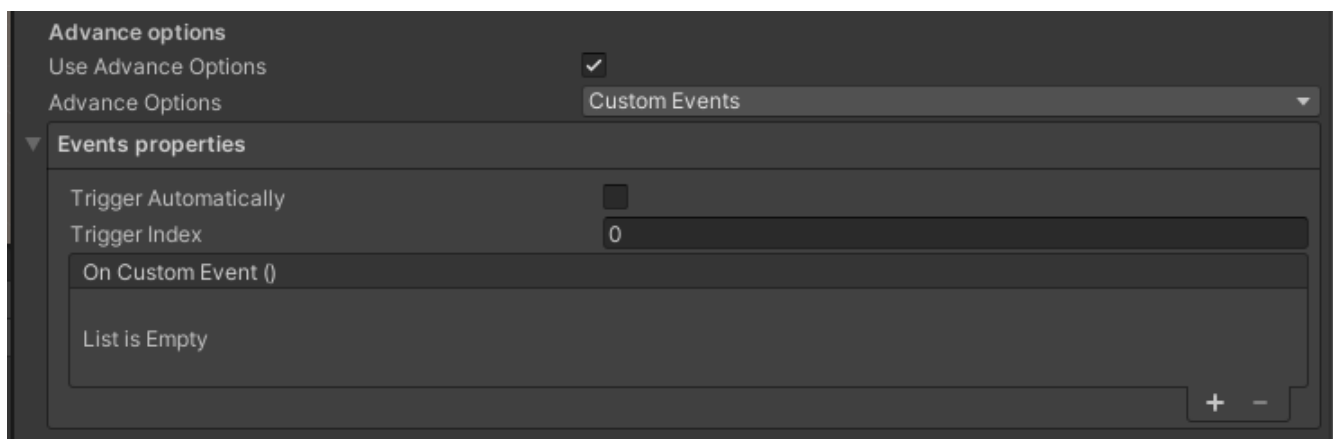


Localization Object

Custom Events: This option works similar to the Language Selection option, but allows you to have only the custom event option, this event works the same as the `OnLanguageSelection()` event of the Language Selection option.

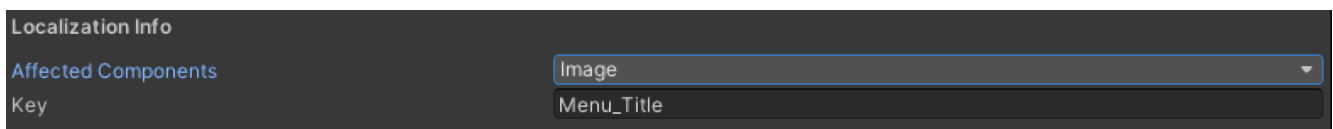
If you set **Trigger Automatically**, the events in `OnCustomEvent` will trigger as soon as the object is active based in the index, this allows you to have multiple custom events every one with it's own automatic trigger, you can also give the same index to multiple components if you want them to be linked.

To reset the **Trigger Automatically** value in runtime you need to call the `ClearCustomEventTrigger` method in the Localization Object script



For the Image, the image you are going to localize needs to be in the Resources folder and the reference key must be the file name in the folder, if you are using a sub-folder, you need to add them to the key as well.

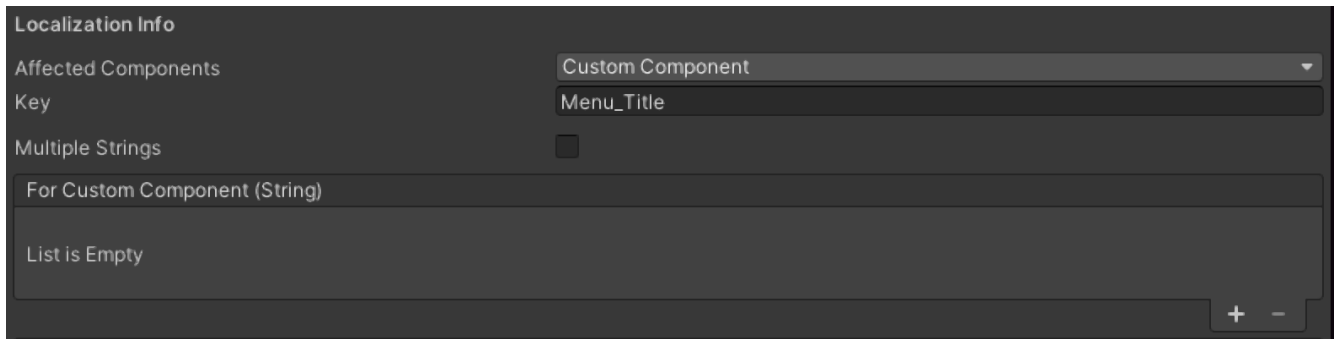
For example, if the image you are going to localize is a png file named Title in a sub-folder named Menu, the keys value must be "Menu/Title". The image format is not needed



Localization Object

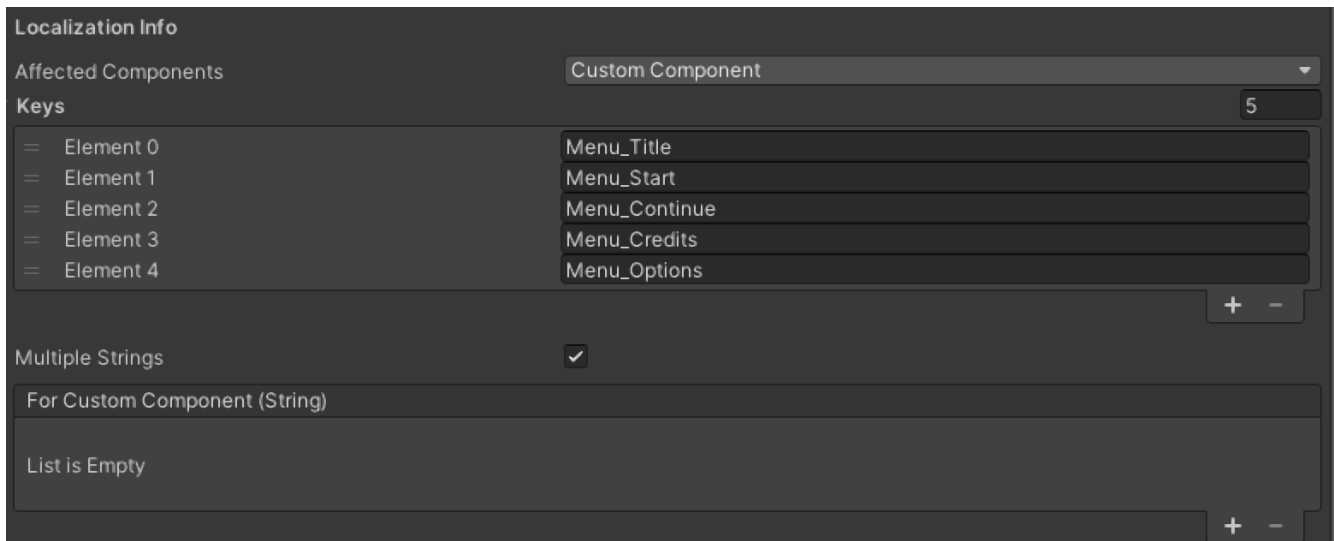
The last option is the Custom Component type, this option allows you to integrate this tool with a custom system, a dialog system for example, easily.

To use it, you need to have a method in your script with a string parameter that needs to be added to the ForCustomComponent() event.



The screenshot shows the 'Localization Info' panel. The 'Affected Components' dropdown is set to 'Custom Component'. The 'Key' field contains 'Menu_Title'. The 'Multiple Strings' checkbox is unchecked. Below the 'For Custom Component (String)' section, there is a text area that says 'List is Empty' and a '+ -' button at the bottom right.

In case you need to add multiple values, you can toggle the Multiple String option, this allows you to have a list of keys instead of a single one, the keys can be added manually or with the Editor Tool window just like any other Localization Object component.



The screenshot shows the 'Localization Info' panel with the 'Multiple Strings' checkbox checked. The 'Affected Components' dropdown is still 'Custom Component'. The 'Keys' section now shows a list of five elements: 'Element 0' (Menu_Title), 'Element 1' (Menu_Start), 'Element 2' (Menu_Continue), 'Element 3' (Menu_Credits), and 'Element 4' (Menu_Options). The 'For Custom Component (String)' section still shows 'List is Empty' and the '+ -' button.

If you are using this last option, be aware that you will need a way to clear the values in your system when you change the active language



Tips and other info

This tool is not meant to be used in big projects, however it could perfectly work no matter the size of your game, just make sure everything is organized and you keep track of every key and every object that will be localized.

I recommend to have one specific member of your team to handle the tool and take care of its implementation in your project.

If you need to modify the tool to adapt it to your systems, feel free to do it, I tried to make it modular and compatible with other systems, but is impossible without making it more complex and with a lot more of work that I could afford to.



Update your language files

If you want to update your language files from previous versions to make them compatible with the new version. Here are the steps I recommend doing.

First of all I suggest to have all the language files on CSV format. This will make the update more simple.

Once you have all your files as CSV files, open them with an external tool, like Microsoft Excel or similar.

Choose the file that will become the final language file and add a row at the top to assign the languages of each column. This step can also be done at the end if it makes easier coping and pasting the values.

Then copy all the localized values on each file and paste them in different columns on your language file, remember to check that all the values are correctly aligned and in the right position.

Once your done you can save your file and close the external tool. Now use the editor tool to check that the file is correctly read by the tool. If all the elements shows up as intended, you can now drag this new file onto the "LocalizationManager" component. If your files were originally on JSON format, you can save the data as JSON from the editor tool.

No other change on the objects is needed. The internal system works exactly the same, only the file reading system was updated.

