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 ArthemyDevelopment/LocalizationTool/EditorTool, where you can create, open and modify language files. The files must be in JSON, CSV, or STRINGS formats. 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 add your language files to the tool, these files must be in the *Streaming Assets* folder. To add files to the list you can drag & drop them or write the language and the file name manually. You can create these files with the *Editor Tool* or with external tools.

The *Localization Object* component will do all the work for you, 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 diferent options will be shown to you deppending on the object you select, for a detail explanation on every option go to the *Localization Object* section in the documentation.

As soon as you import the tool in to your project, you will need to move the "*_StreamingAsset*" folder inside "*ArthemyDevelopment*" to the *Assets* folder and delete the "*_*". This is because Unity don't let you include the folder in the store package.
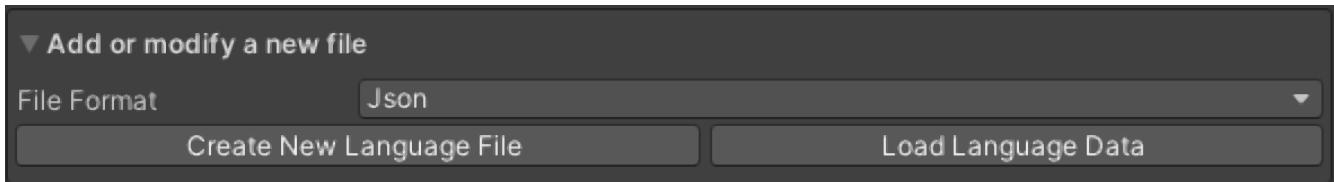If you download the tool from Itch.io, this is not required.

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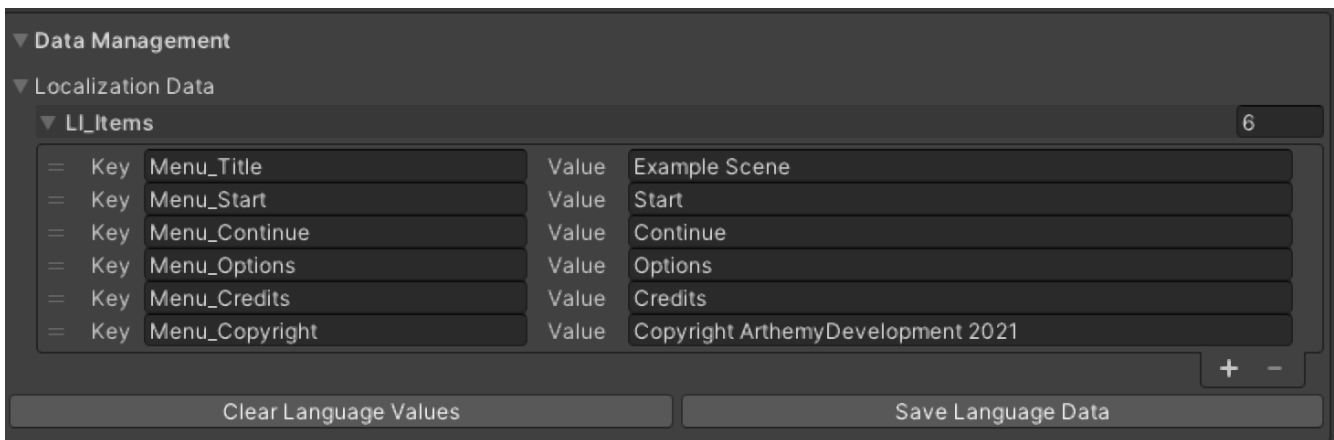
In File Management you can create, load, or save the language files you will use in your project. If you are going to load a file, you must first select the file format and make sure it is in the Streaming Assets folder.

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 column contains the values, this is the text that will be set in the objects.
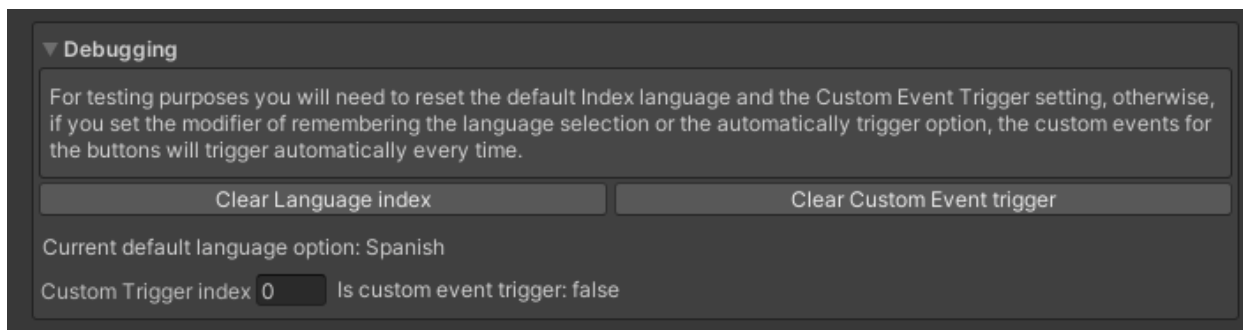

If you already have a loaded file, or you create one, the Clear Language Values and Save Language Data buttons will appear, the first one erase all the values from the file, but the keys will remain. With this, you can create template files for future languages. The second one allows you to save your 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 PlayerPref. You can see these values or reset them in the Debugging foldout.

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.
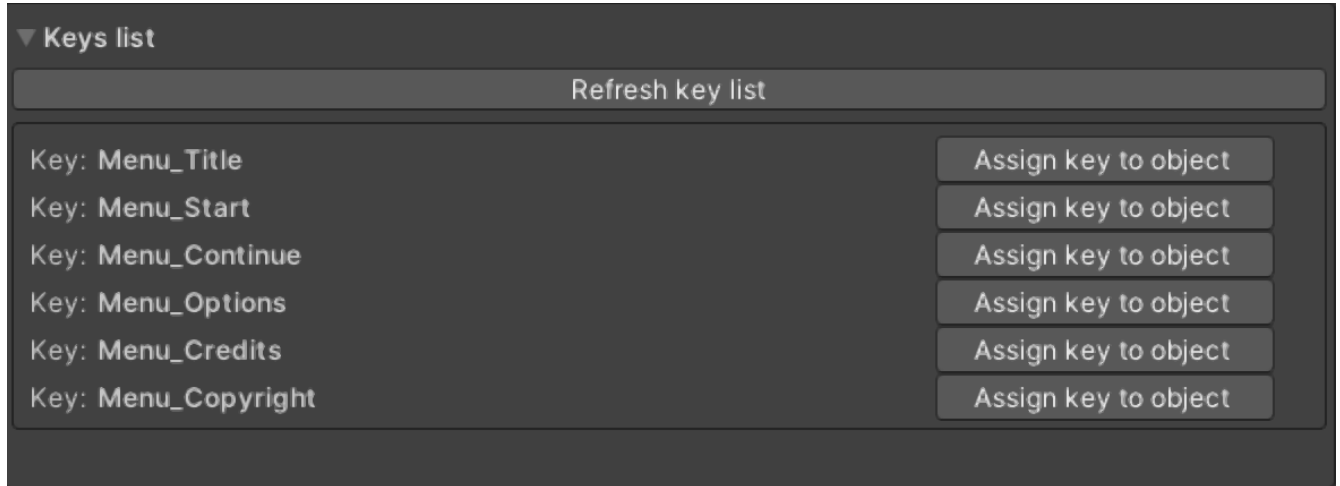

To add the Localization Object component and assign the key select the object in your scene and press the Assign key to object button.

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 DontDestroyOn-Load behavior and must be instantiated in the first scene of your game.

In the language list, every file contains an index value, this value is used to set the active language with the Localization Object component, the name of the language, and the file name in the Streaming Asset folder.
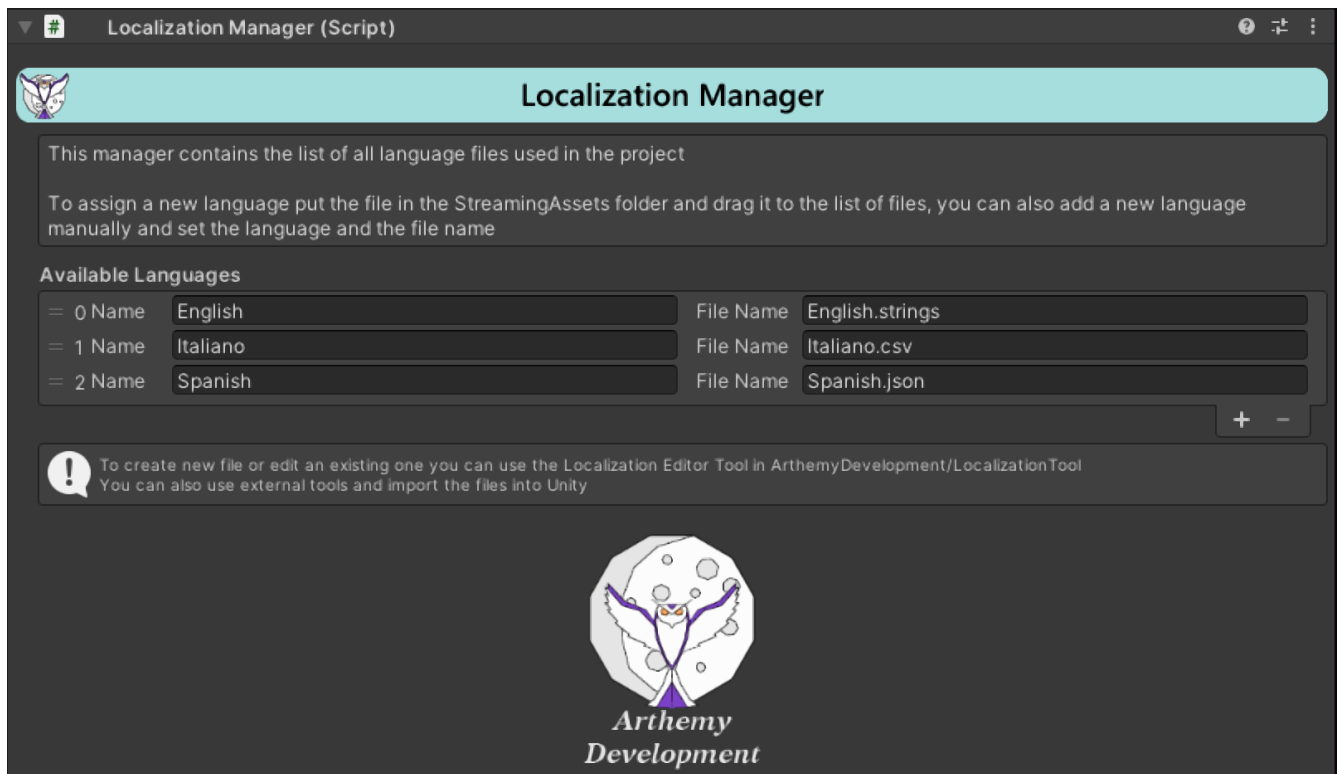
To add language files you can drag & drop them from the Streaming Asset folder or write the language and the file name manually.

If you wish you can reorder the languages index dragging them in the list. This is useful if you want to have a specific index value order for the languages.

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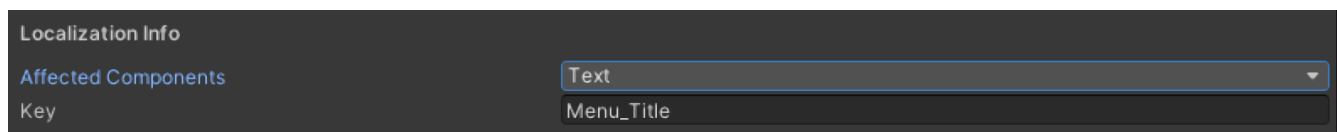
The first thing you need to select is the type of object you are localizing and the reference key.
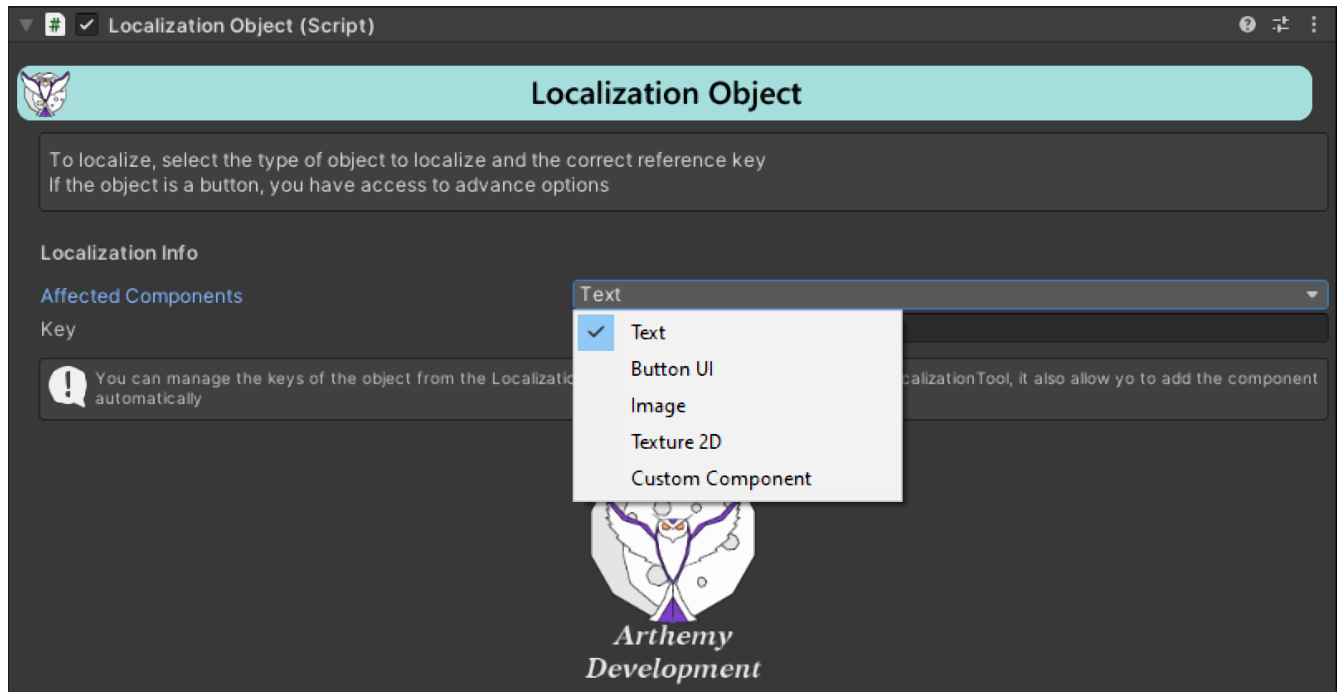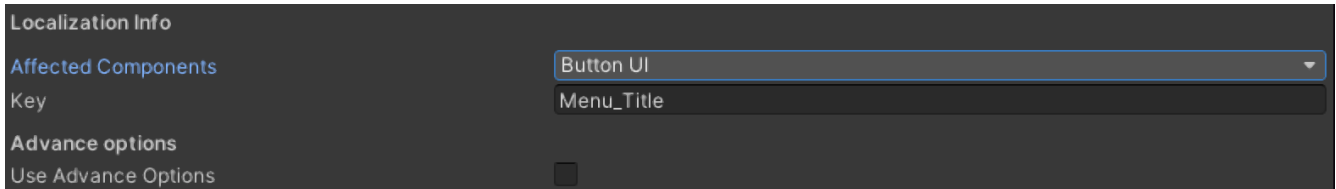

The supported object types are
- TMPro Text
- UI Button
- UI Image
- Texture 2D
- Custom Component

For the TMPro 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 text as a child in the button object.

You will also have acces to advance options for the button, these options are:
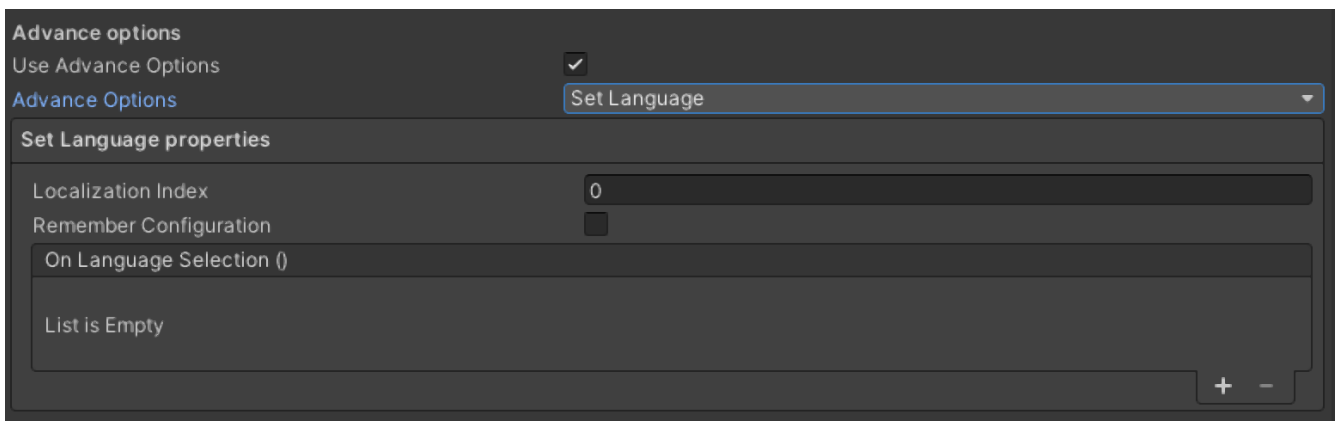
Language Selection: This option allow you to use the button to set the active language in your game, with this you don't need a reference key and the text in the button will be the language name you put in the Localization Manager component, instead you need to use the language index value. You can also 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.

If you will let the Remember Configuration option disable, you can use this event to have a better organization in your project if you want.

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
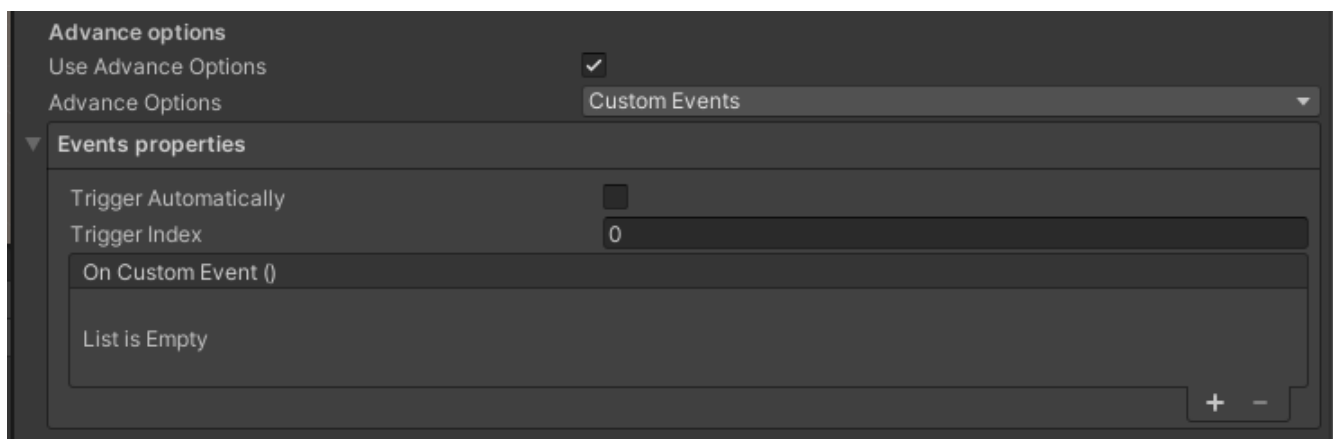
For the Image, the image you are going to localize needs to be in the Resources folder and the reference key must be the file name in the folder, if you are using a subfolder, you need to add them to the key as well.
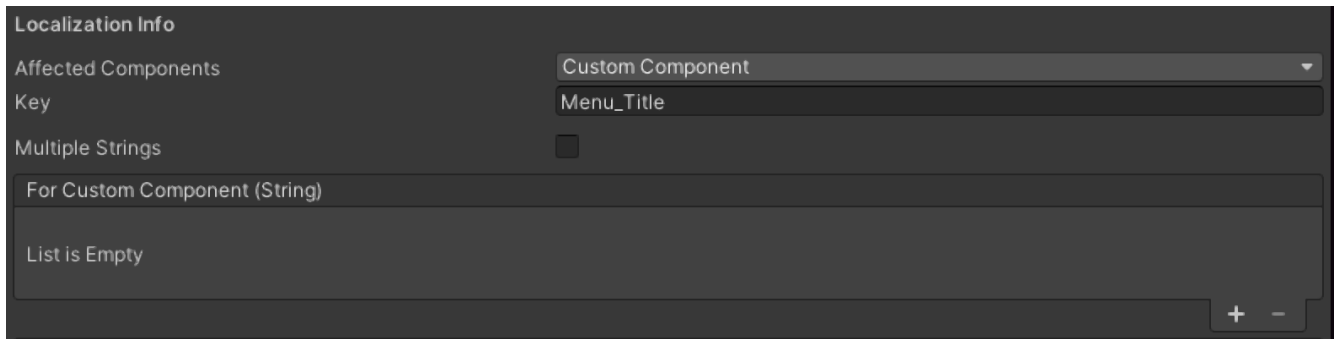
For example, if the image you are going to localize is a png file named Title in a subfolder named Menu, the keys value must be "Menu/Title". The image format is not needed
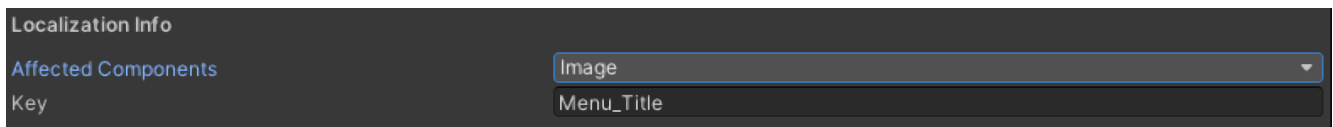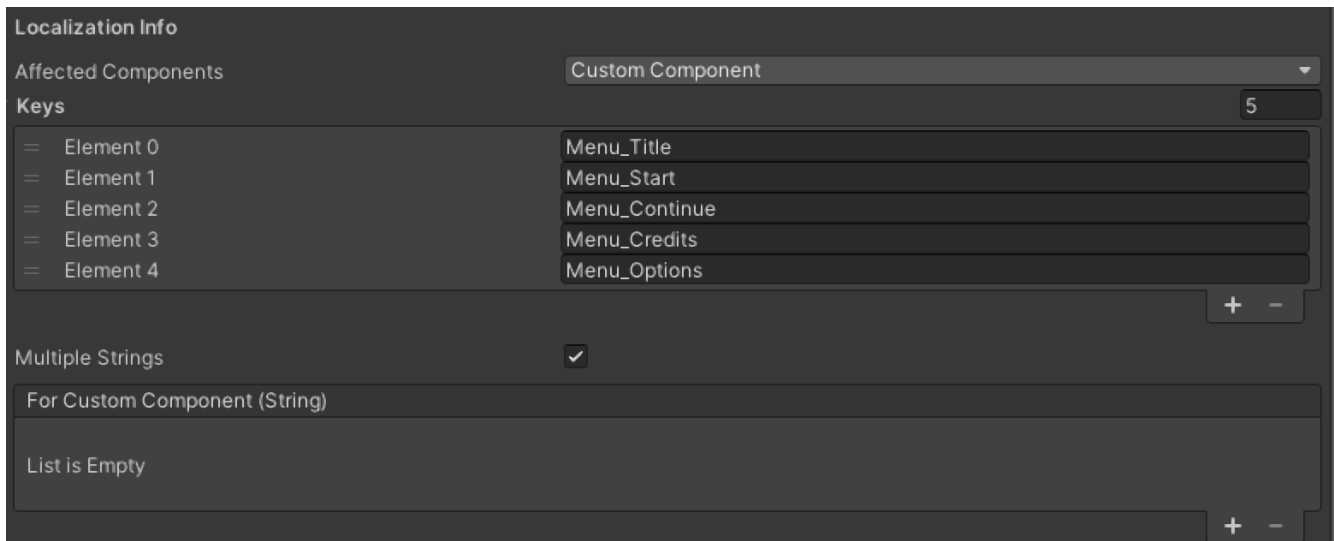
Localization Object

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.

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.

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

For the reference keys, I recomend using an ENUM for the values, this will prevent any typo and let you keep track of every key you are using an make sure all the files contain that key.

This tool is not ment to be used in big projects, however it could perfectly work no mater 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.

In case you are using CSV or STRINGS format, consider that the tool read only the first two columns, that means that you could use a third one for the reference text to translate if you need it, you could also use values only for organization and ignore them in the tool, for example, the first value of the columns to specify what that column is.
However, when saving the file, only the values that the tool reads are the values it will save.

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 imposible without making it more complex and with a lot more of work that I could aford to.