LOCALIZATION PRO

DOCUMENTATION FOR UNITY3D

RELEASE VERSION 1.0

CREATED BY: 3Y3.NET

INDEX

About Localization Pro

Localization Pro is, by far, the best localization system for Unity. You can use it from small indie projects to large professional projects where the team is spread all over the world.

With Localization Pro you can automatically localize all kinds of objects in your project: **text, images, audio clips, materials and even the properties of your own custom objects**.

The asset is highly configurable and allows you to make the same task in different ways. There is a pipeline designed for small projects where localization can be done as long as you are creating your game scenes and there is another one designed for large projects where all localization values are exported to files, send to translators and then imported back to the final project. Any way you can change from one to another pipeline wherever you wish.

Localization Pro provides powerful tools that **saves tons of hours of work** for any project.

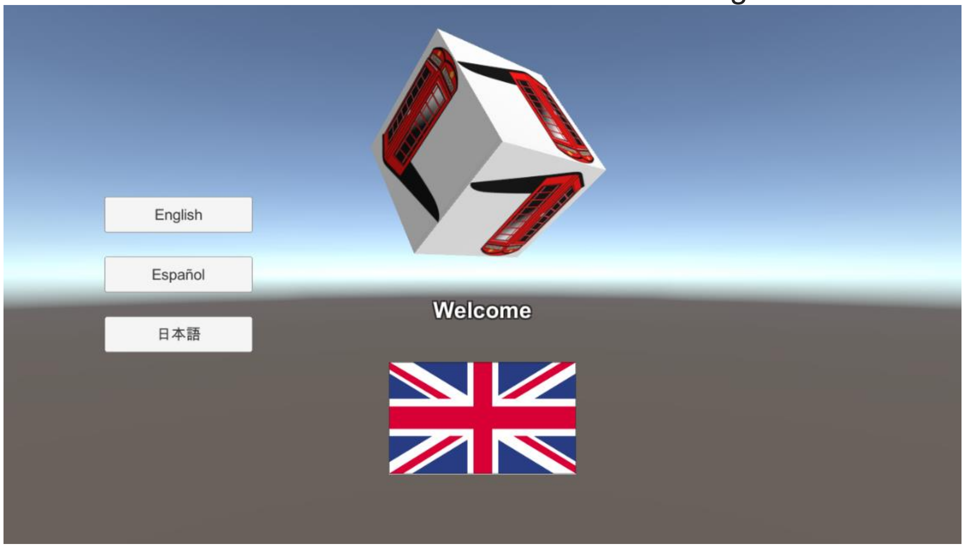
Quick Start

For an ultra-fast quick start follow this steps, but take in mind that most of the powerful features of Localization Pro are not covered in this quick start. We encourage you to see the tutorial videos and/or read the manual.

This tutorial is available in video at:
https://www.youtube.com/watch?v=rO_ErdpKRsU

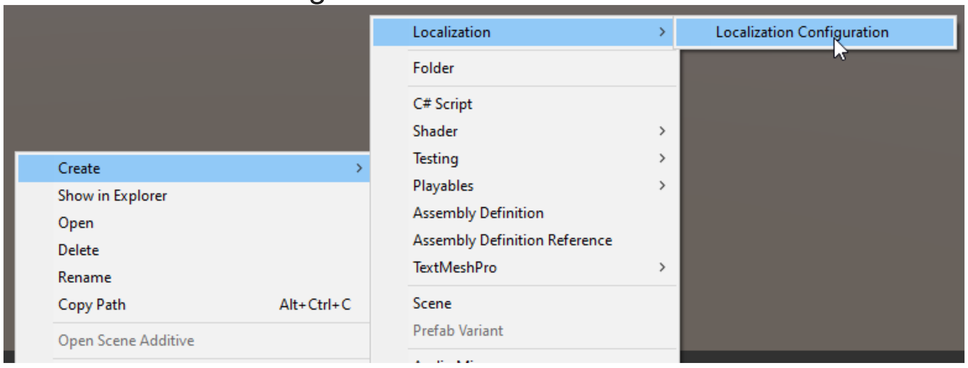
This quick start will cover the localization of **Text, Image and Materia**l components in a single scene.

1) Open the scene ‘Common Game Tools/Sample scenes/ Localization Tutorial/LocalizationTutorial’. You will see something like this:

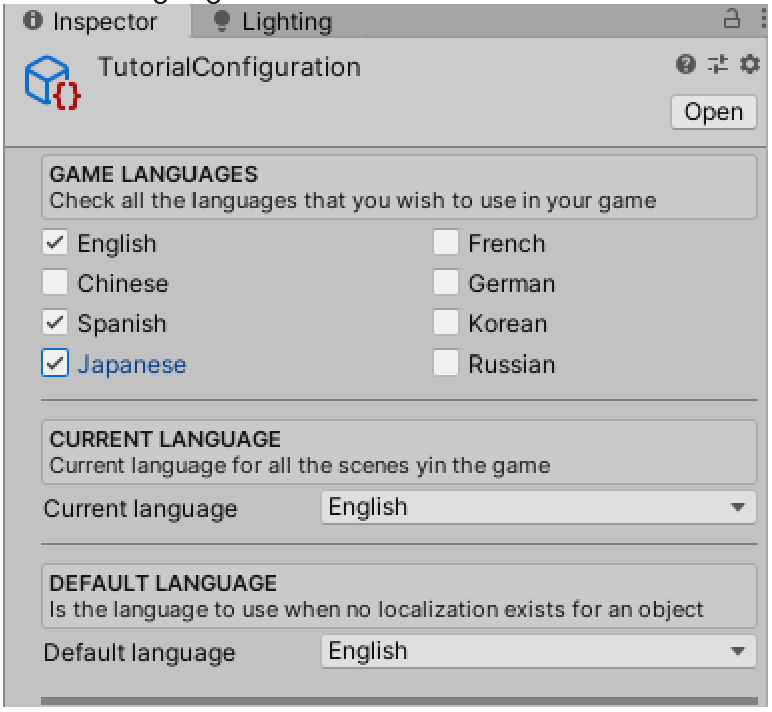


2) Localization in the scene is not implemented yet, so let’s do it! In the scene folder right click anywhere and select

‘Create->Localization->Localization Configuration’. Name the object created ‘TutorialConfiguration’



3) Select the object ‘TutorialConfiguration’ in the project folder and in the top section of the inspector (GAME LANGUAGES) check the English, Spanish and Japanese checkboxes. Leave ‘English’ as current and default language.



TutorialConfiguration

GAME LANGUAGES
Check all the languages that you wish to use in your game

[✓] English   [ ] French
[ ] Chinese   [ ] German
[✓] Spanish   [ ] Korean
[✓] Japanese   [ ] Russian

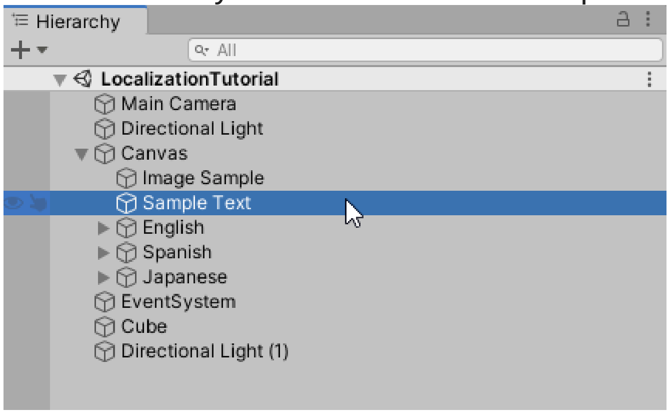
CURRENT LANGUAGE
Current language for all the scenes yin the game

Current language: English

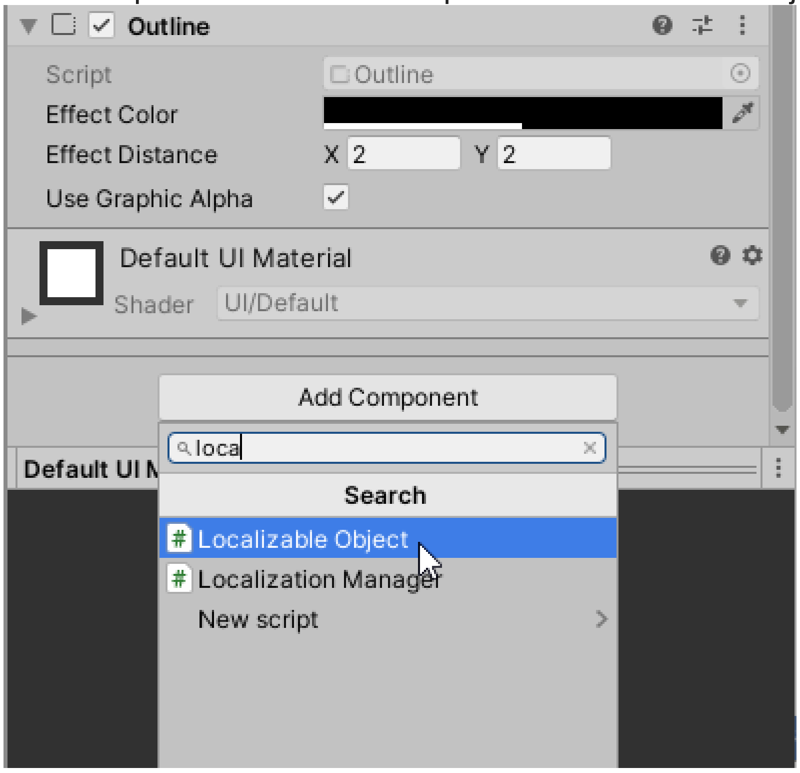
DEFAULT LANGUAGE
Is the language to use when no localization exists for an object

Default language: English

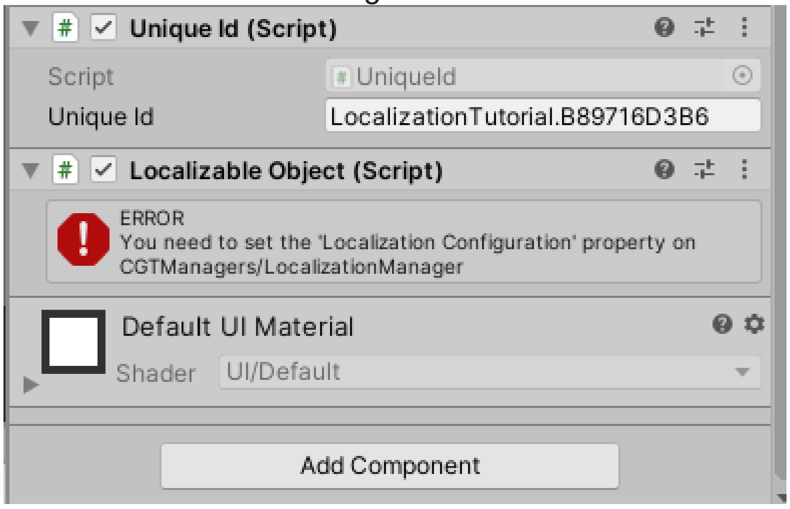
4) In the Hierarchy click in the Canvas/Sample Text



5) In the inspector click ‘Add Component’->Localizable Object

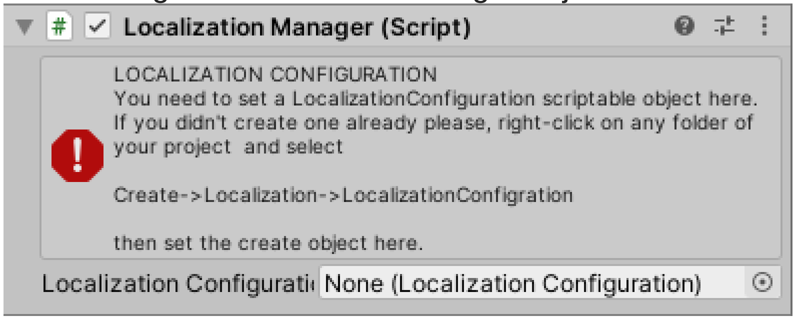


You should see something like this:



ERROR
You need to set the 'Localization Configuration' property on CGTManagers/LocalizationManager

The error tells that we need to set a parameter in the CGTManagers/LocalizationManager. If we take a look at the Hierarchy we can notice that a new object has been created. Click in the CGTManagers/LocalizationManager object.

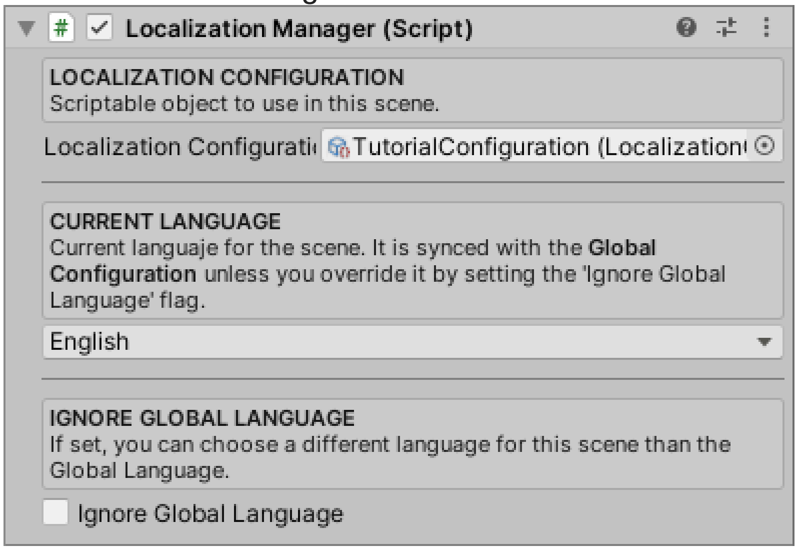

Localization Manager (Script)
Localization Configurati
None (Localization Configuration)

LOCALIZATION CONFIGURATION
You need to set a LocalizationConfiguration scriptable object here.
If you didn't create one already please, right-click on any folder of your project and select

Create->Localization->LocalizationConfigration

then set the create object here.

Simply set the Localization Configuration parameter to the previously created TutorialConfiguration.


Localization Manager (Script)

LOCALIZATION CONFIGURATION
Scriptable object to use in this scene.

Localization Configurati TutorialConfiguration (Localization

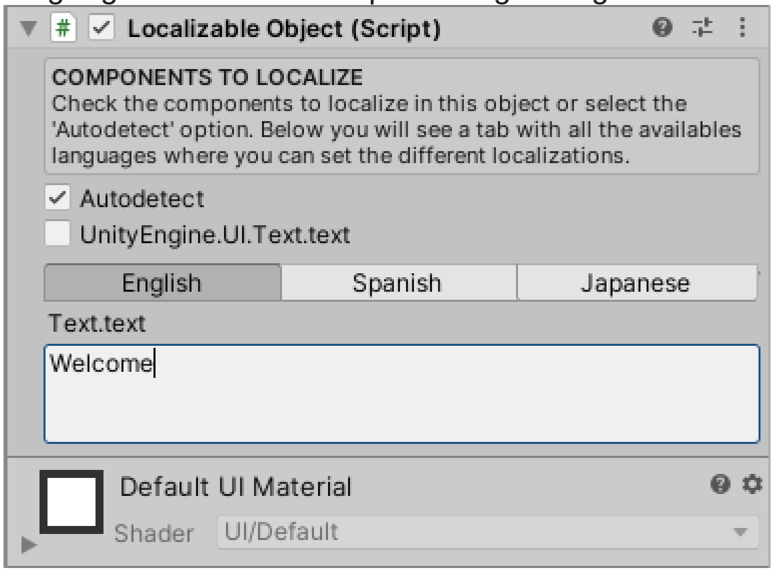
CURRENT LANGUAGE
Current languaje for the scene. It is synced with the **Global Configuration** unless you override it by setting the 'Ignore Global Language' flag.

English

IGNORE GLOBAL LANGUAGE
If set, you can choose a different language for this scene than the Global Language.

☐ Ignore Global Language

6) Go back and click in the hierarchy at Canvas/Sample Text fill all languages with the correspondant greeting


Localizable Object (Script)

COMPONENTS TO LOCALIZE
Check the components to localize in this object or select the 'Autodetect' option. Below you will see a tab with all the availables languages where you can set the different localizations.

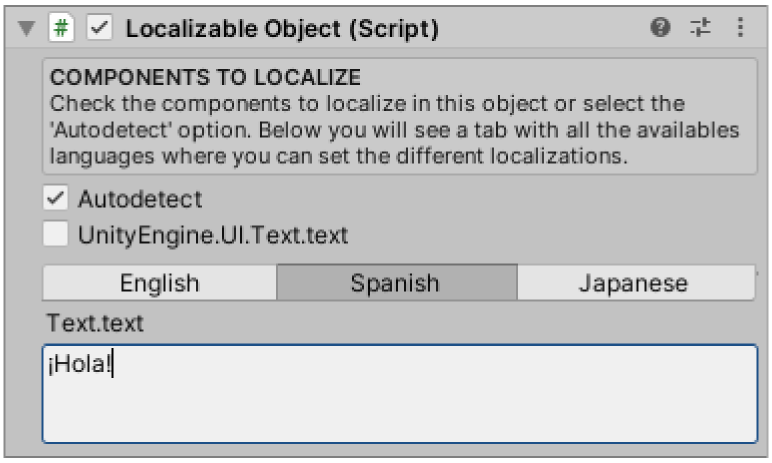
☑ Autodetect
☐ UnityEngine.UI.Text.text

English | Spanish | Japanese

Text.text

Welcome

Default UI Material
Shader UI/Default


Localizable Object (Script)

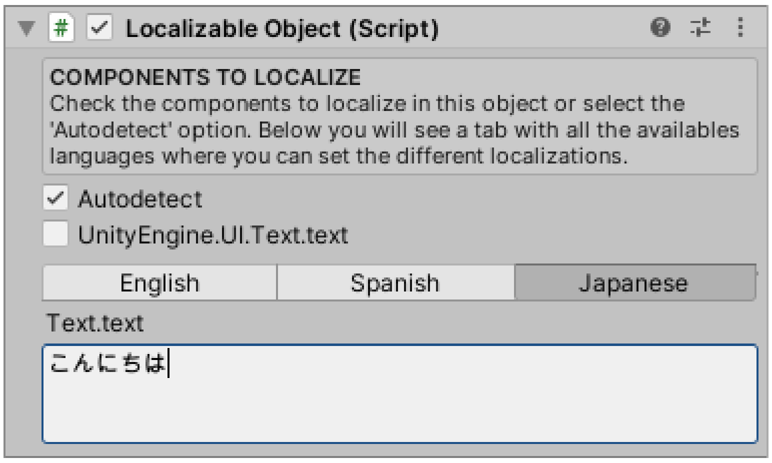
COMPONENTS TO LOCALIZE
Check the components to localize in this object or select the 'Autodetect' option. Below you will see a tab with all the availables languages where you can set the different localizations.

☑ Autodetect
☐ UnityEngine.UI.Text.text

English | Spanish | Japanese

Text.text

¡Hola!



Localizable Object (Script)

COMPONENTS TO LOCALIZE
Check the components to localize in this object or select the 'Autodetect' option. Below you will see a tab with all the availables languages where you can set the different localizations.

- [✓] Autodetect
- [ ] UnityEngine.UI.Text.text

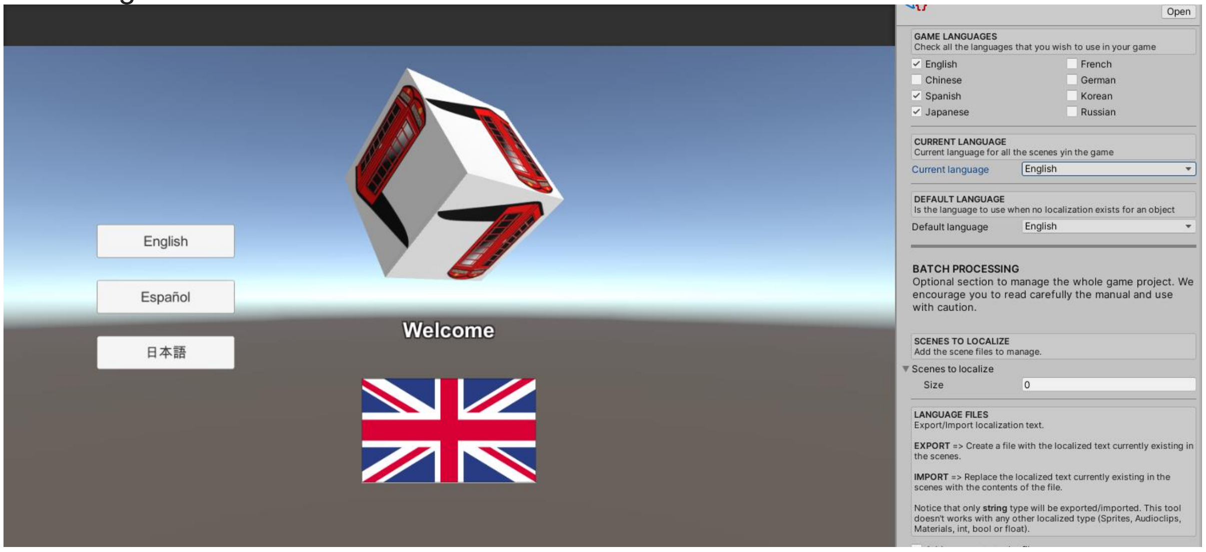
Text.text

こんにちは

7) Make the same with the Canvas/Image Sample object, that is add a Localizable Object component and set the Image.SourceImage to the correspondent sprites (uk, spain, japan)

8) Now click in the 'Localizable Cube' and add a Localizable Object component. Set the material to the correspondent ones (phone, bull and samurai)

9) DONE!! Now you can test the localization even in the editor mode. Select the TutorialConfiguration and change the current language to English, Spanish or Japanese. Automatically the text, images and materials will be changed.



GAME LANGUAGES
Check all the languages that you wish to use in your game

- [✓] English
- [ ] Chinese
- [✓] Spanish
- [✓] Japanese
- [ ] French
- [ ] German
- [ ] Korean
- [ ] Russian

CURRENT LANGUAGE
Current language for all the scenes yin the game

Current language: English

DEFAULT LANGUAGE
Is the language to use when no localization exists for an object

Default language: English

BATCH PROCESSING
Optional section to manage the whole game project. We encourage you to read carefully the manual and use with caution.

SCENES TO LOCALIZE
Add the scene files to manage.

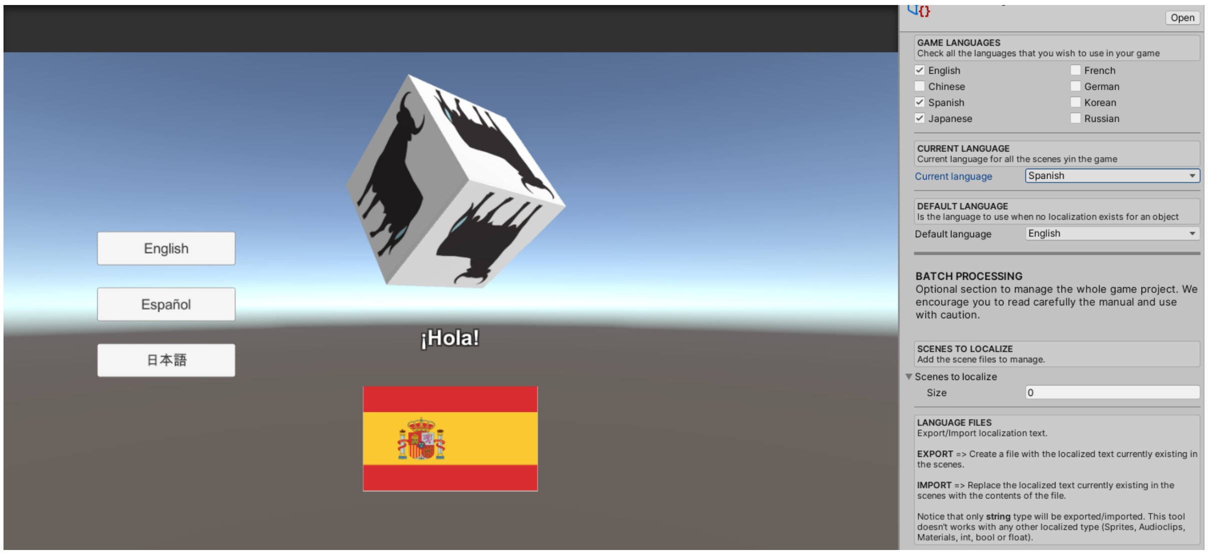
Scenes to localize
Size: 0

LANGUAGE FILES
Export/Import localization text.

EXPORT => Create a file with the localized text currently existing in the scenes.

IMPORT => Replace the localized text currently existing in the scenes with the contents of the file.

Notice that only **string** type will be exported/imported. This tool doesn't works with any other localized type (Sprites, Audioclips, Materials, int, bool or float).



GAME LANGUAGES
Check all the languages that you wish to use in your game
✓ English French
Chinese German
✓ Spanish Korean
✓ Japanese Russian

CURRENT LANGUAGE
Current language for all the scenes yin the game
Current language: Spanish

DEFAULT LANGUAGE
Is the language to use when no localization exists for an object
Default language: English

BATCH PROCESSING
Optional section to manage the whole game project. We encourage you to read carefully the manual and use with caution.

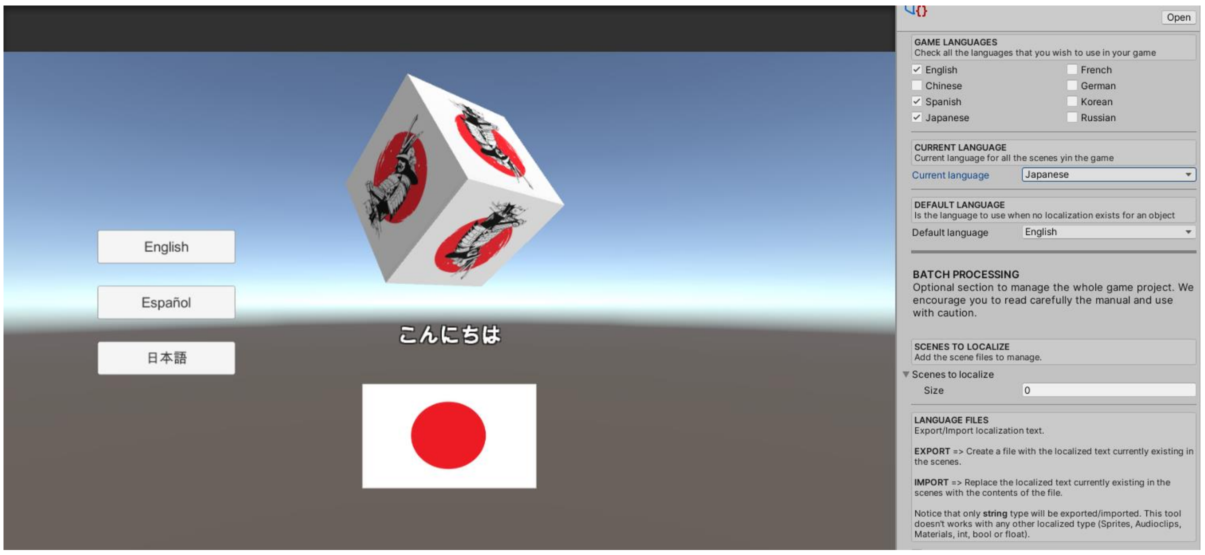
SCENES TO LOCALIZE
Add the scene files to manage.
Scenes to localize
Size: 0

LANGUAGE FILES
Export/Import localization text.

EXPORT => Create a file with the localized text currently existing in the scenes.

IMPORT => Replace the localized text currently existing in the scenes with the contents of the file.

Notice that only **string** type will be exported/imported. This tool doesn't works with any other localized type (Sprites, Audioclips, Materials, int, bool or float).



GAME LANGUAGES
Check all the languages that you wish to use in your game
✓ English French
Chinese German
✓ Spanish Korean
✓ Japanese Russian

CURRENT LANGUAGE
Current language for all the scenes yin the game
Current language: Japanese

DEFAULT LANGUAGE
Is the language to use when no localization exists for an object
Default language: English

BATCH PROCESSING
Optional section to manage the whole game project. We encourage you to read carefully the manual and use with caution.

SCENES TO LOCALIZE
Add the scene files to manage.
Scenes to localize
Size: 0

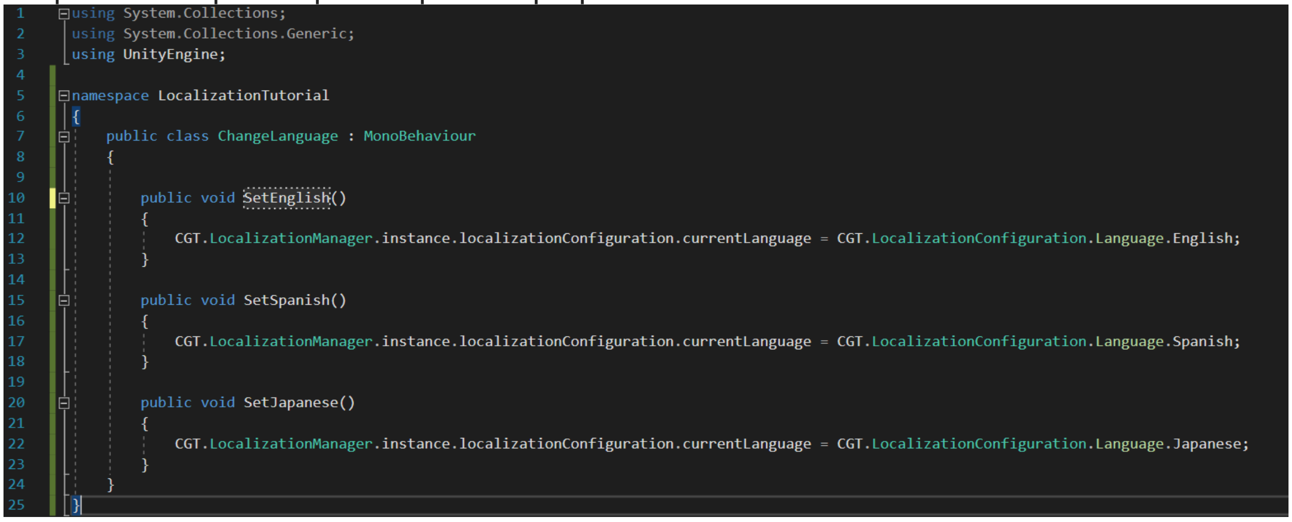
LANGUAGE FILES
Export/Import localization text.

EXPORT => Create a file with the localized text currently existing in the scenes.

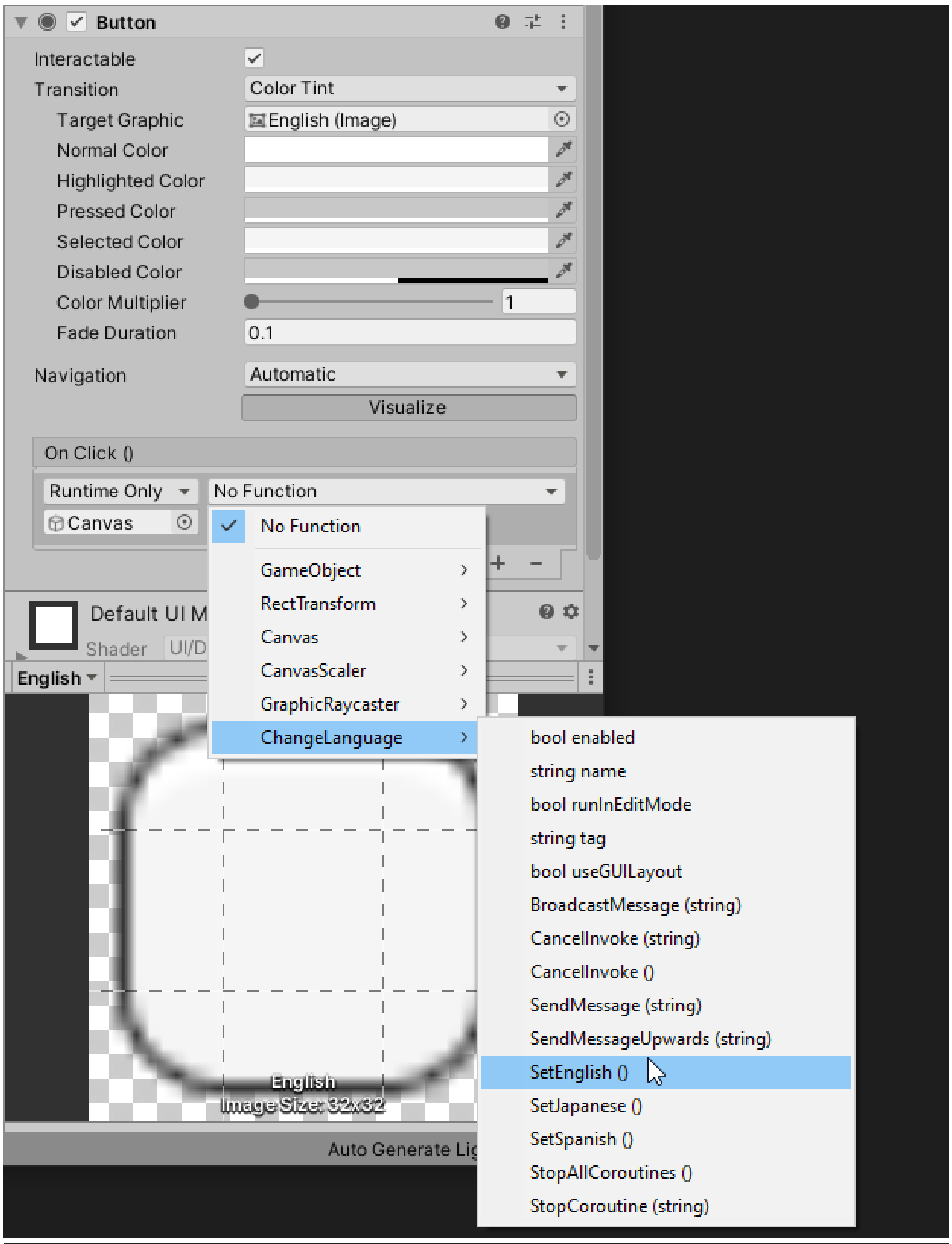
IMPORT => Replace the localized text currently existing in the scenes with the contents of the file.

Notice that only **string** type will be exported/imported. This tool doesn't works with any other localized type (Sprites, Audioclips, Materials, int, bool or float).

The only thing left is to attach the buttons to the prepared 'ChangeLanguage' script. The script is super simple and prepared for this tutorial:


```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

namespace LocalizationTutorial
{
    public class ChangeLanguage : MonoBehaviour
    {

        public void SetEnglish()
        {
            CGT.LocalizationManager.instance.localizationConfiguration.currentLanguage = CGT.LocalizationConfiguration.Language.English;
        }

        public void SetSpanish()
        {
            CGT.LocalizationManager.instance.localizationConfiguration.currentLanguage = CGT.LocalizationConfiguration.Language.Spanish;
        }

        public void SetJapanese()
        {
            CGT.LocalizationManager.instance.localizationConfiguration.currentLanguage = CGT.LocalizationConfiguration.Language.Japanese;
        }
    }
}
```

Add the script where you want in the scene. For example, in the Camera and set the buttons to the respective function, for example in the 'English' button:


Button

Interactable ✓

Transition: Color Tint

Target Graphic: English (Image)

Normal Color

Highlighted Color

Pressed Color

Selected Color

Disabled Color

Color Multiplier: 1

Fade Duration: 0.1

Navigation: Automatic

On Click ()

Runtime Only | No Function

Canvas

English

Make the same with the other two buttons and now you can test you scene in runtime and see how the localization changes.

As we previously said this Quick Start only covers the very basic functionality of the asset. We encourage you to see the rest of the options, especially the Batch Utilities and the Custom Object localization.

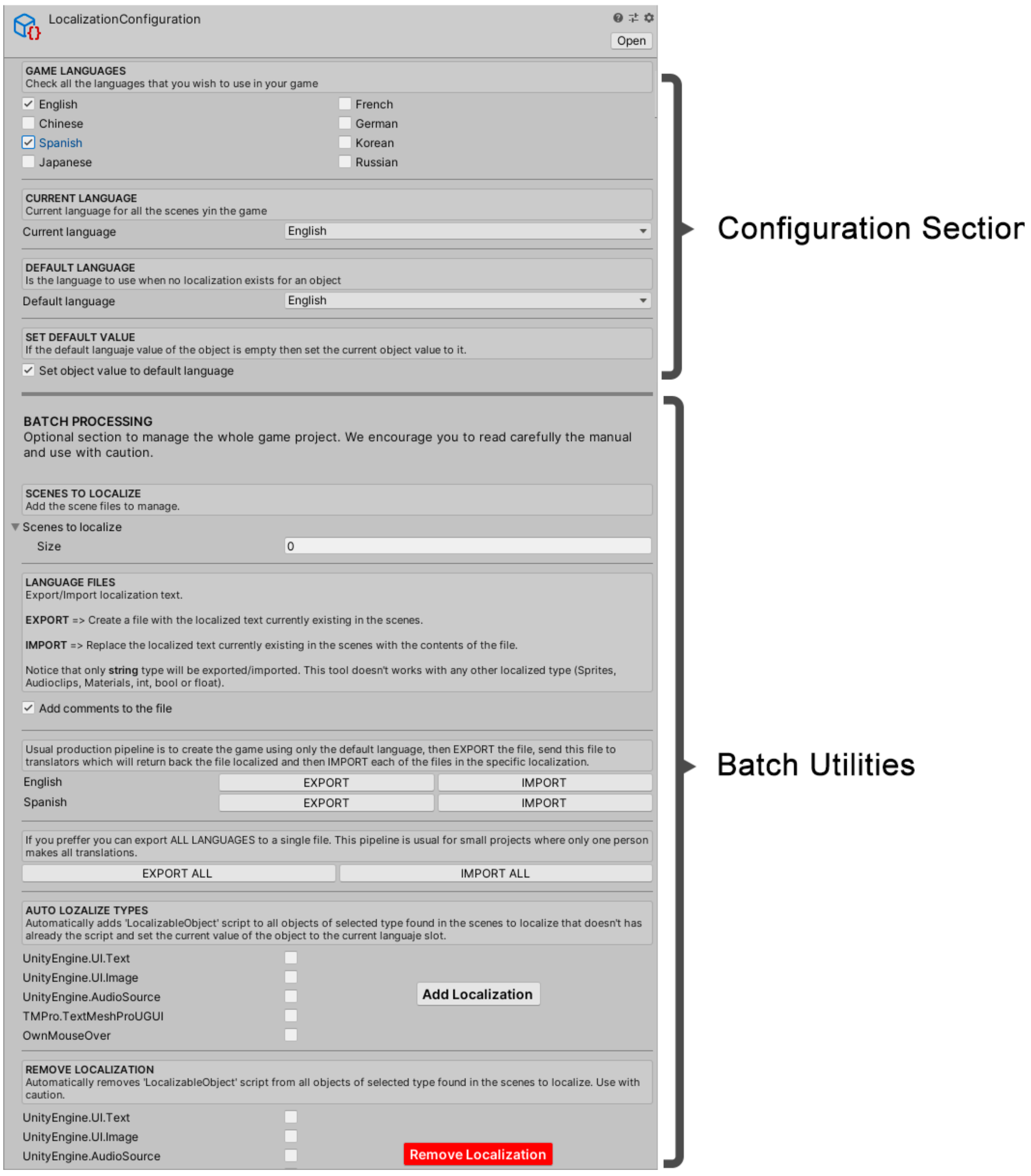
Localization Configuration

The first step in any project is create a ‘Localization Configuration’ object. To do simply right click in the folder where you wish the object to be created (can be anyone you want) and select ‘Create->Localization->Localization Configuration’.

A new 'Localization Configuration' object will be created. This object is where all configuration of your project is managed and also where you can perform all the batch operations for large projects. You can create as many 'Localization

Configuration' objects as you wish but you should only use one at a time for your project.

If you select the 'Localization Configuration' object in in the folder you can see two well differenced sections, the first one is related to project configuration and the second one contains all the batch utilities for large projects.



LocalizationConfiguration

Open

GAME LANGUAGES
Check all the languages that you wish to use in your game

✓ English	French
Chinese	German
✓ Spanish	Korean
Japanese	Russian

CURRENT LANGUAGE
Current language for all the scenes yin the game

Current language: English

DEFAULT LANGUAGE
Is the language to use when no localization exists for an object

Default language: English

SET DEFAULT VALUE
If the default languaje value of the object is empty then set the current object value to it.

✓ Set object value to default language

BATCH PROCESSING
Optional section to manage the whole game project. We encourage you to read carefully the manual and use with caution.

SCENES TO LOCALIZE
Add the scene files to manage.

Scenes to localize
Size: 0

LANGUAGE FILES
Export/Import localization text.

EXPORT => Create a file with the localized text currently existing in the scenes.

IMPORT => Replace the localized text currently existing in the scenes with the contents of the file.

Notice that only **string** type will be exported/imported. This tool doesn't works with any other localized type (Sprites, Audioclips, Materials, int, bool or float).

✓ Add comments to the file

Usual production pipeline is to create the game using only the default language, then EXPORT the file, send this file to translators which will return back the file localized and then IMPORT each of the files in the specific localization.

English	EXPORT	IMPORT
Spanish	EXPORT	IMPORT

If you preffer you can export ALL LANGUAGES to a single file. This pipeline is usual for small projects where only one person makes all translations.

EXPORT ALL | IMPORT ALL

AUTO LOZALIZE TYPES
Automatically adds 'LocalizableObject' script to all objects of selected type found in the scenes to localize that doesn't has already the script and set the current value of the object to the current languaje slot.

UnityEngine.UI.Text
UnityEngine.UI.Image
UnityEngine.AudioSource
TMPro.TextMeshProUGUI
OwnMouseOver

Add Localization

REMOVE LOCALIZATION
Automatically removes 'LocalizableObject' script from all objects of selected type found in the scenes to localize. Use with caution.

UnityEngine.UI.Text
UnityEngine.UI.Image
UnityEngine.AudioSource

Remove Localization

Configuration Section

The configuration section is where you define the languages used and where you must set the current language within your game.

Game Languages

In this section you can select all the languages you are going to use in your project. By default Localization Pro includes the 8 more spoken languages in the world but you can add any other language in a single step (see section **Adding New Languages** in this manual)

Current Language

This parameter is the language currently used in your project. You must set this parameter in runtime to change from a localization to other. It also works in editor mode, that is, you can set this parameter in editor mode a see how all localized objects changes from one language to other.

Default Language

This parameter is the default language of your project. When a localized object has no input for a specific language the asset will choose automatically the localization of the default language.

Set Object Value to Default Language

This is a very useful option to boost your game localization. If you set to 'on' then whenever you add a 'Localizable Script' to a game object the value of the 'Default Language' will be set to the current value of the GameObject. For example, if you are localizing a Text in 3 languages and the default is English, then the current value of the text will be set into the English placeholder.

Batch Utilities

Batch section is optional and is only used when you wish to import/export localization files or you wish to add/remove localization globally to any kind of objects. There is a tutorial explaining all Batch Processing, We encourage you to see the tutorial at:

https://www.youtube.com/watch?v=cWe0Zjqw9EA

Scenes to Localize

This parameter is the list of scenes to work with when you use the batch utilities. You can add as many scenes as you wish.

Language Files

In this section you can export and import language files for any of the languages or for the whole project. Usual production pipeline is to create the game using only the default language, then EXPORT the file, send this file to translators which will return back the file localized and then IMPORT each of the files in the specific localization.

You can export one file per language or, directly, export al languages to the same file. It depends on you translation structure, that is, if you are using only one translator or if you have one translator per language and need to send different files.

NOTE: Only string/text type will be exported/imported. This tool doesn't works with any other localized type (Sprites, Audioclips, Materials, int, bool or float).

The import/export file is a CSV comma ',' separated fully compatible with plain text editors, Google Spreadsheet, Excel and any other spread sheet. The CSV format is standard CSV format. Quotation of fields " is optional unless you have some special character in the field. In that case quotation is mandatory. Quotes are escaped by double quoting and new line is considered a special character.

Each line of the CSV correspond to an object/languaje in the project and has 5 fields:

First field: Ordinal created for user convenience. Any change in this field has no effect in the import/export.

Second field: Fully qualified object unique identifier. This field is generated automatically by the tool and MUST not be changed. Is what the asset uses to match the file entries with the actual objects and scenes. Do not touch.

Third field: Is the name of the game object in the scene. Is used only for user convenience. Any change in this field has no effect in the import/export.

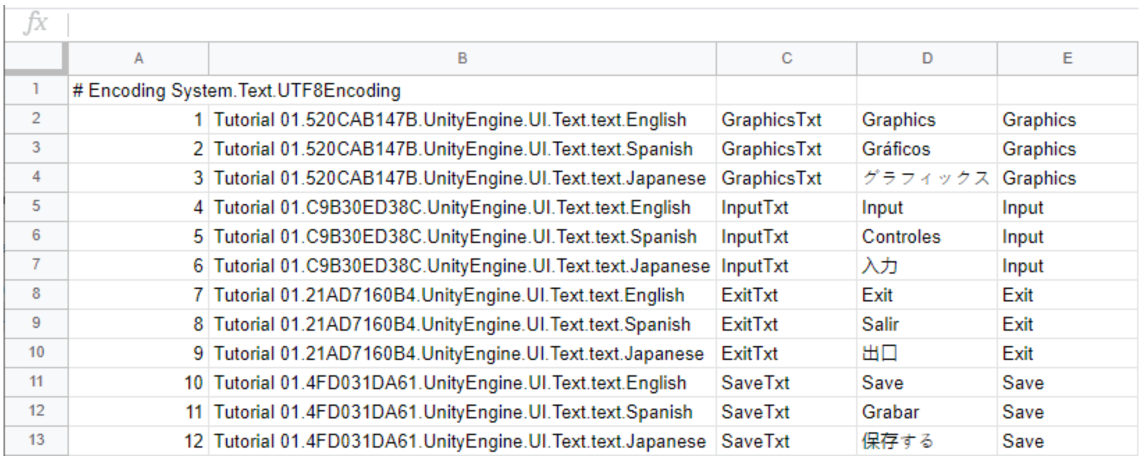
Fourth field: Translated text. This is the only field your translator should really fill. As a CSV field accepts multi-line, and any character.

Fifth Field: The current value of the game object. Usually will be the default language. This field is informational for the translator and has no effect in the localization.

For your convenience it is recommended to edit complex localization (for example a localization with a full HTML content or those which has lot of especial characters) in a spread sheet and then export to CSV. Or at least, open the CSV with a spread sheet to verify that all characters are properly escaped.

Example CSV

In this case is a file created with 'EXPORT ALL' as it contains all the languages:



	A	B	C	D	E
1	# Encoding System.Text.UTF8Encoding				
2	1	Tutorial 01.520CAB147B.UnityEngine.UI.Text.text.English	GraphicsTxt	Graphics	Graphics
3	2	Tutorial 01.520CAB147B.UnityEngine.UI.Text.text.Spanish	GraphicsTxt	Gráficos	Graphics
4	3	Tutorial 01.520CAB147B.UnityEngine.UI.Text.text.Japanese	GraphicsTxt	グラフィックス	Graphics
5	4	Tutorial 01.C9B30ED38C.UnityEngine.UI.Text.text.English	InputTxt	Input	Input
6	5	Tutorial 01.C9B30ED38C.UnityEngine.UI.Text.text.Spanish	InputTxt	Controles	Input
7	6	Tutorial 01.C9B30ED38C.UnityEngine.UI.Text.text.Japanese	InputTxt	入力	Input
8	7	Tutorial 01.21AD7160B4.UnityEngine.UI.Text.text.English	ExitTxt	Exit	Exit
9	8	Tutorial 01.21AD7160B4.UnityEngine.UI.Text.text.Spanish	ExitTxt	Salir	Exit
10	9	Tutorial 01.21AD7160B4.UnityEngine.UI.Text.text.Japanese	ExitTxt	出口	Exit
11	10	Tutorial 01.4FD031DA61.UnityEngine.UI.Text.text.English	SaveTxt	Save	Save
12	11	Tutorial 01.4FD031DA61.UnityEngine.UI.Text.text.Spanish	SaveTxt	Grabar	Save
13	12	Tutorial 01.4FD031DA61.UnityEngine.UI.Text.text.Japanese	SaveTxt	保存する	Save

Example of escaped fields

```
1;"SC1.21F0C59FF5.MyClass.text.English";"Example 01";"Text with a "" quote"
2;"SC1.21F0C59FF6.MyClass.text.English";"Example 02";"This is a text
several
new lines"
3;"SC1.21F0C59FF7.MyClass.text.English";"Example 02";"This is a text
several ""quotes"" and
new lines"
```

Export/Import files

As explained before, usual production pipeline is to create the game using only the default language, then EXPORT the file, send this file to translators which will return back the file localized and then IMPORT each of the files in the specific localization.

A good practice is to have backup copies of the language files (or the 'all languages' file if is the case) just in case you miss or delete values by error. You can import any file whenever you wish.

In the other hand you can export any language file when you wish either to have a backup or for make modifications and import again.

Auto-localization of objects

An interesting feature of Localization Pro is that you can automatically add or remove localization capabilities to an entire set of objects in the selected scenes. This will save lot of time and accelerate a lot your production pipeline.

Add localization

In this section you can add localization capabilities to all object that contains the localization components you select. In the left you can see a list with all localizable components. By default the asset identify Text, TextMeshPro, Images and Audio Clips as potentially localizable.

You can add your own components to the list in a single step (see section **Adding Own Components** in this manual)

Once selected press the "Add Localization" button and the asset will add a 'Localizable Object' script to all game objects which contains any on the selected types if it doesn't already has the script.

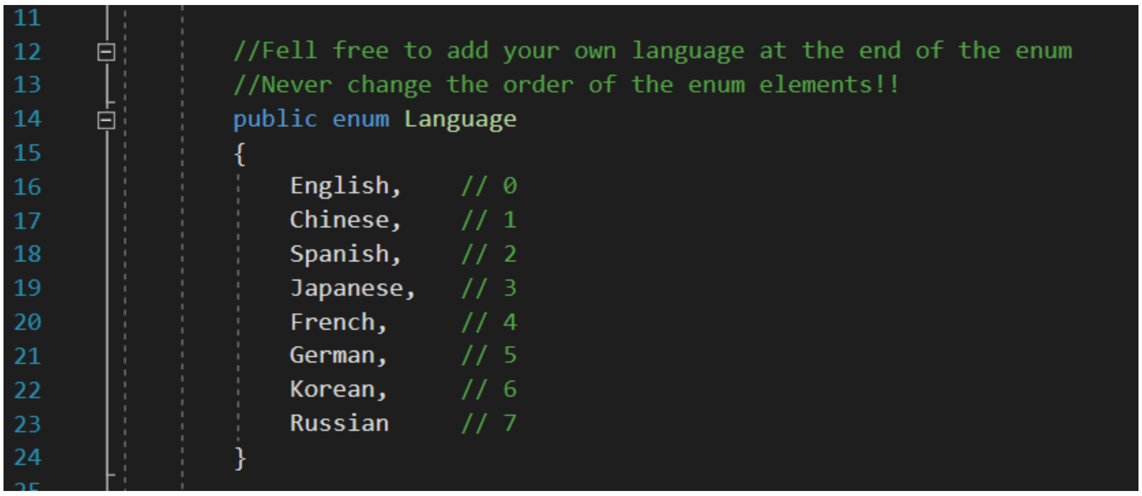
Remove Localization

In the other hand you can remove localization capabilities to a entire set of objects. It works pretty in the same way that 'Add Localization' but in this case it removes the 'Localizable Object' script.

NOTE: Use with caution as this tool can remove large amount of localizations to the project. A good practice is to export all languages before removing just to have a backup of the data.

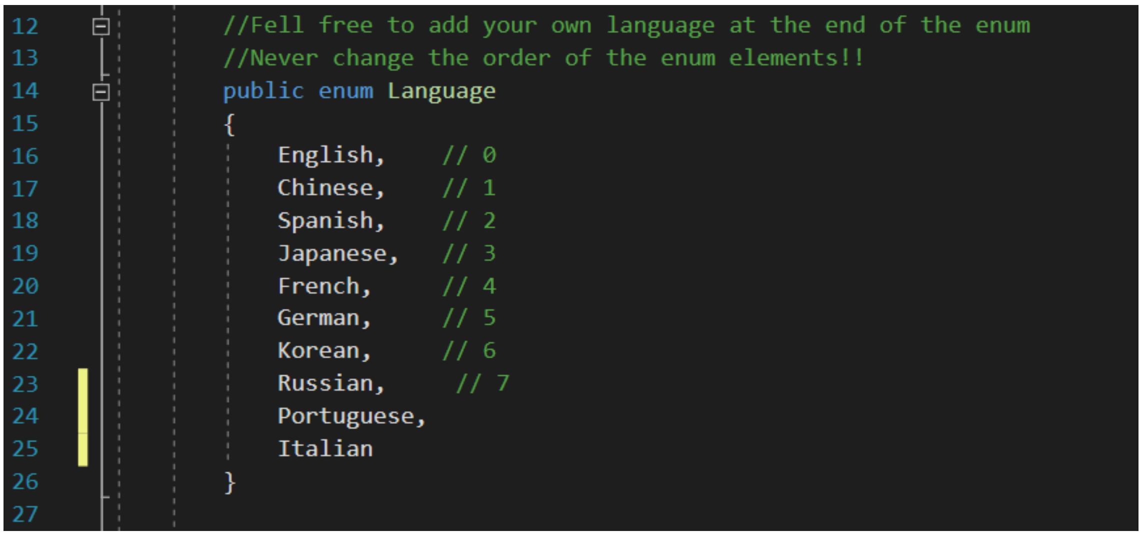
Adding New Languages

You can add new languages to the list of available ones by editing the file Common Game Tools/Scripts/ ScriptableObjects /LocalizationConfiguration.cs



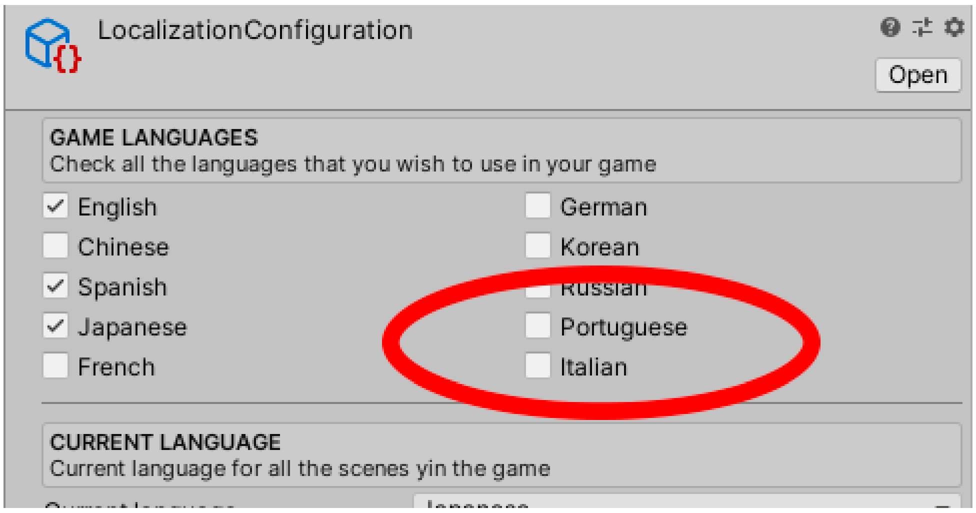
```
11
12      //Fell free to add your own language at the end of the enum
13      //Never change the order of the enum elements!!
14      public enum Language
15      {
16          English,    // 0
17          Chinese,    // 1
18          Spanish,    // 2
19          Japanese,   // 3
20          French,     // 4
21          German,     // 5
22          Korean,     // 6
23          Russian     // 7
24      }
```

In the line 14 you can see an enum with the 8 more spoken languages in the world. You can add here your own languages simply by adding and entry ant the end. For example, in this case I will add the Portuguese and Italian languages to the list.



```
12      //Fell free to add your own language at the end of the enum
13      //Never change the order of the enum elements!!
14      public enum Language
15      {
16          English,    // 0
17          Chinese,    // 1
18          Spanish,    // 2
19          Japanese,   // 3
20          French,     // 4
21          German,     // 5
22          Korean,     // 6
23          Russian,     // 7
24          Portuguese,
25          Italian
26      }
27
```

As you can see, both languages are immediately shown in the available languages.


LocalizationConfiguration

Open

GAME LANGUAGES
Check all the languages that you wish to use in your game

[✓] English
[ ] Chinese
[✓] Spanish
[✓] Japanese
[ ] French
[ ] German
[ ] Korean
[ ] Russian
[ ] Portuguese
[ ] Italian

CURRENT LANGUAGE
Current language for all the scenes yin the game

Custom Object Localization

You can add new components to the list of available components to localize. By default the asset can automate the localization of Text, Audio Source, Materials and Images. The tool will look for this components in all Game Objects in the scene but if, for example, you have your own MonoBehavior script which contains a string and an AudioClip and you wish to localize it you can do it by modifying the file Common Game Tools/Scripts/ ScriptableObjects /LocalizationConfiguration.cs

Look for the Array typesToLocalize (around line 35) and add your own object as a typeof (see example). Then look for the Array atributesToLocalize (around line 55) and add the types of your objects that you wish to localize.

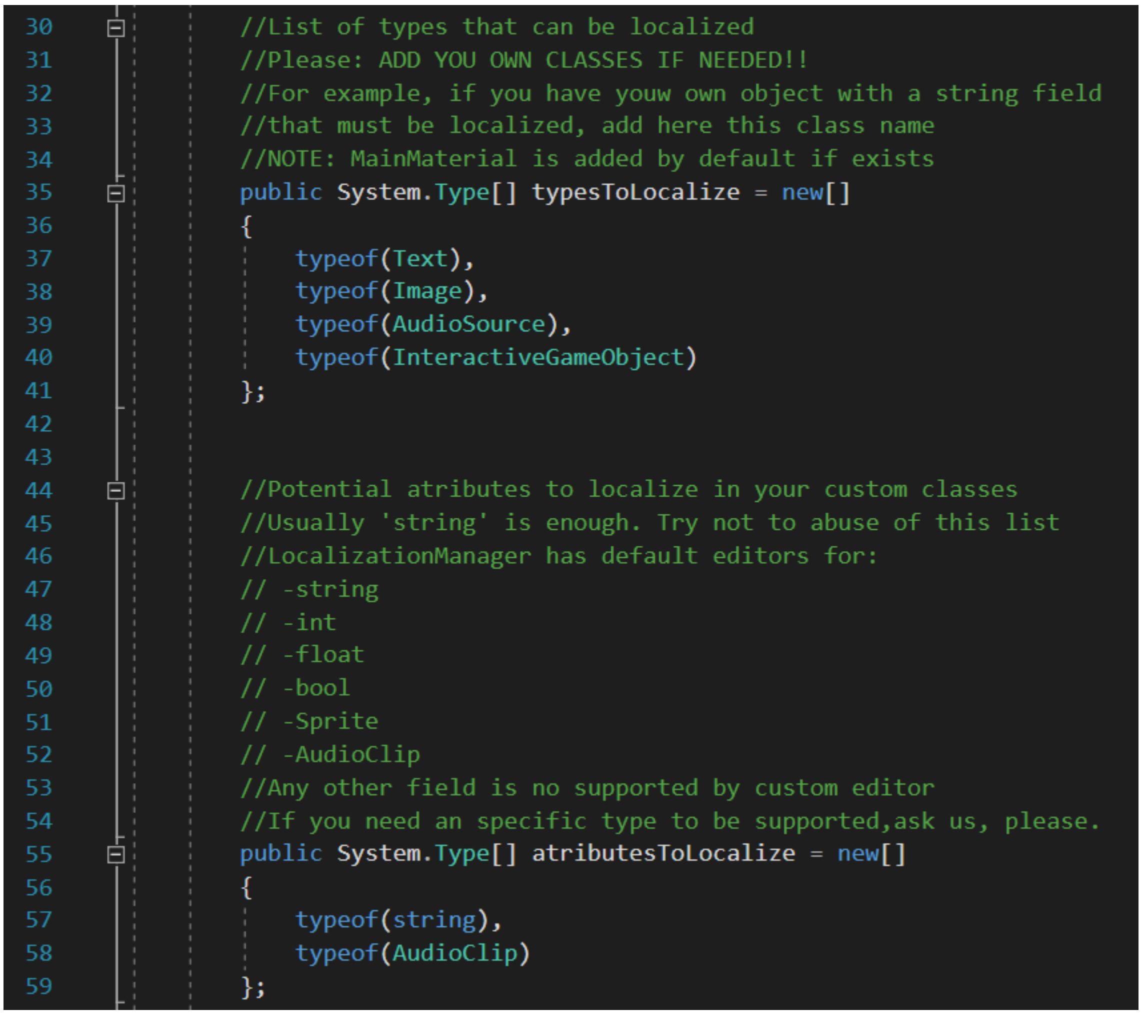
In this example I have added my own InteractiveGameObject class and in the types I've added the string and AudioClip types.

From now in advance all the string and AudioClips public parameters of my class InteractiveGameObject have been added to the Localization pipeline.

This feature is **incredibly powerful** and **saves tons of hours** of work in any project. There is a tutorial explaining all Batch Processing, We encourage you to see the tutorial at:

https://www.youtube.com/watch?v=7Q75082x9Wg

By default Localization Pro provides localization capabilities for custom parameters of type string, int, float, bool, Sprite and AudioClip. If you need a specific new type to be added, please write us an email and we will try to add to the roadmap of evolutives of the asset.



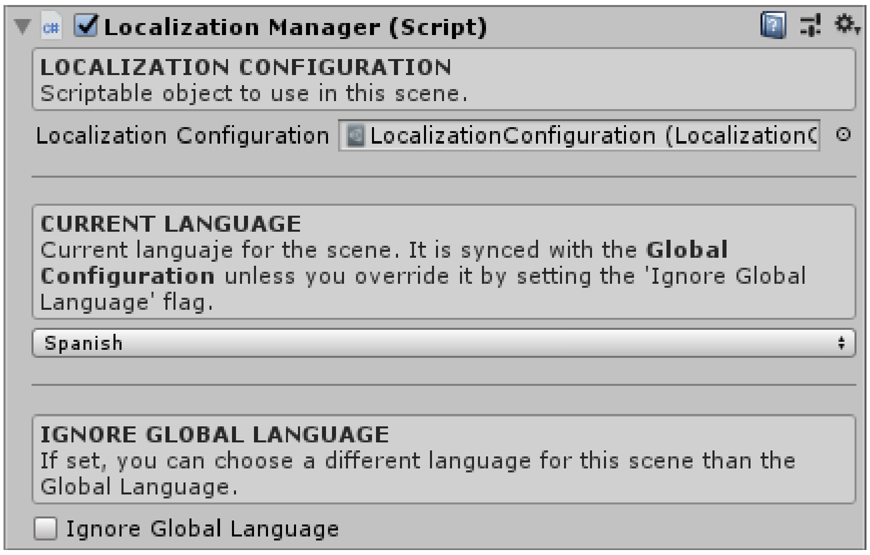
```
        //List of types that can be localized
        //Please: ADD YOU OWN CLASSES IF NEEDED!!
        //For example, if you have youw own object with a string field
        //that must be localized, add here this class name
        //NOTE: MainMaterial is added by default if exists
        public System.Type[] typesToLocalize = new[]
        {
            typeof(Text),
            typeof(Image),
            typeof(AudioSource),
            typeof(InteractiveGameObject)
        };


        //Potential atributes to localize in your custom classes
        //Usually 'string' is enough. Try not to abuse of this list
        //LocalizationManager has default editors for:
        // -string
        // -int
        // -float
        // -bool
        // -Sprite
        // -AudioClip
        //Any other field is no supported by custom editor
        //If you need an specific type to be supported,ask us, please.
        public System.Type[] atributesToLocalize = new[]
        {
            typeof(string),
            typeof(AudioClip)
        };
```

Localization Manager Component

If the Localization Configuration controls the global behavior of Localization Pro (that is, number of languages, default and current language, etc…) the Localization Manager component is in charge of set the language of all game objects within a scene.

This component is automatically created when you localize the first object. You only need to set the Localization Configuration you wish to use, as we saw in the quick start tutorial.

The language of the whole scene can be changed setting the parameter ‘Current Language’. Usually this parameter is automatically set by the Localization Configuration component but you can override the global configuration and set a specific language only for a specific scene by setting the ‘Ignore Global Language’ flag.



Localization Manager (Script)

LOCALIZATION CONFIGURATION
Scriptable object to use in this scene.

Localization Configuration: LocalizationConfiguration (LocalizationC

CURRENT LANGUAGE
Current languaje for the scene. It is synced with the **Global Configuration** unless you override it by setting the 'Ignore Global Language' flag.

Spanish

IGNORE GLOBAL LANGUAGE
If set, you can choose a different language for this scene than the Global Language.

☐ Ignore Global Language

Best Practices

In general, Localization Pro allows you to localize your game in several ways, each one more adequate for each kind of project.

Small Projects or prototypes

This projects usually consists in one single scene with a few game objects to localize. In this case I suggest to use the 'on line' localization as described in the Quick Start. That is, add localizable objects as long as you are creating the game and set the appropriate values to each language.

Medium projects

By medium projects I mean those with many scenes and game objects but where localization will be done only by one person or a small group of persons located in the same headquarter. In this case I suggest to use the batch options only at the end of the game creation process, once your game is finished and it will have only little changes.

Export all languages into a single file and make there the translations. Import then back into the project.

Large projects

This projects are similar to previous one except that translators may be located all around the world. In this case I suggest to export each language separately, send to the translators and the import back into the project.

Global API

Localization Pro can be easily integrated with third party assets. It provides an open API to set the main parameters in runtime.

In fact the API is quite simple and expose only four methods. That is all you need to integrate Localization Pro with your game manager. All the API is in a single static class for your convenience, the LocalizatinoProAPI class.

So, to invoke the methods from any script you only need to call LocalizationProAPI.*methodName()*. The methos are:

SetCurrentLanguage

```
public static void SetCurrentLanguage(LocalizationConfiguration.Language
language)
```

Set the current language for the project to the indicated language. Before change the language the method check that the language is in the list of available languages. For example, to change current language to Spanish use:

```
LocalizationProAPI.SetCurrentLanguage(LocalizationConfiguration.Language.Spanish)
```

SetDefaultLanguage

```
public static void SetDefaultLanguage(LocalizationConfiguration.Language
language)
```

Set the default language for the project to the indicated language. Before change the language the method check that the language is in the list of available languages. For example, to change defauylt language to Spanish use:

```
LocalizationProAPI.SetDefaultLanguage(LocalizationConfiguration.Language.Spanish)
```

GetCurrentLanguage

```
public static LocalizationConfiguration.Language
GetCurrentLanguage(LocalizationConfiguration.Language language)
```

Returns the current language of the project. Returns a LocalizationConfiguration.Language enum member. For example, to get current language use:

```
LocalizationProAPI.GetCurrentLanguage()
```

GetDefaultLanguage

```
public static LocalizationConfiguration.Language
GetDefaultLanguage(LocalizationConfiguration.Language language)
```

Returns the default language of the project. Returns a LocalizationConfiguration.Language enum member.For example, to get default language use:

```
LocalizationProAPI.GetDefaultLanguage()
```

Included Scenes

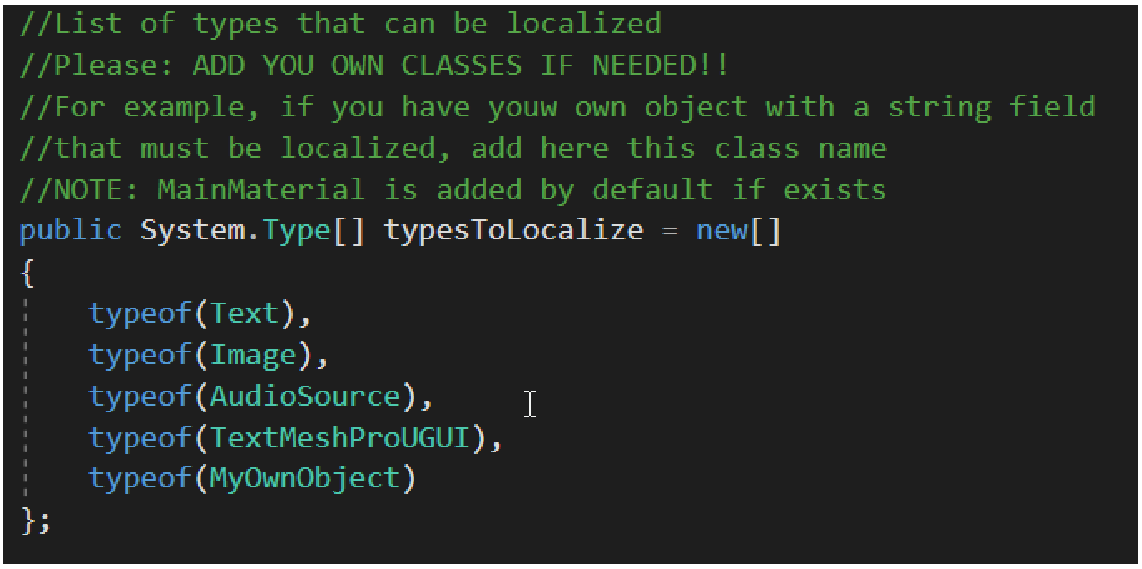
The project comes with three scenes:

Localization Tutorial

This scene is the one you need to open to follow the quick start tutorial. There are two versions of this scene, one with the scene without any localization and the other once the localization has been done. Open the first one and follow the tutorial, you can compare the final result with the finished scene.

Localization Own Object

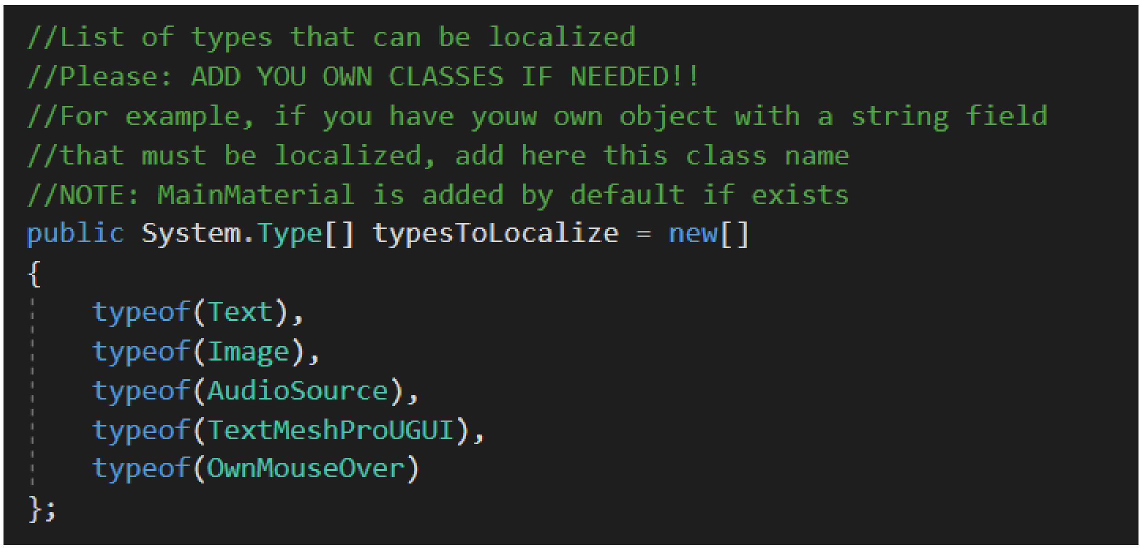
This scene has custom object to be localized. To make it works make sure that the typesToLocalize list in the LocalizationConfiguration.cs script contains a line with typeof(MyOwnObject), just as sown in the image below:



```
//List of types that can be localized
//Please: ADD YOU OWN CLASSES IF NEEDED!!
//For example, if you have youw own object with a string field
//that must be localized, add here this class name
//NOTE: MainMaterial is added by default if exists
public System.Type[] typesToLocalize = new[]
{
    typeof(Text),
    typeof(Image),
    typeof(AudioSource),
    typeof(TextMeshProUGUI),
    typeof(MyOwnObject)
};
```

Localization Batch Tutorial

This tutorial has two scenes and shows how to use the Batch Processing options on Localization Pro. To make it works make sure that the typesToLocalize list in the LocalizationConfiguration.cs script contains a line with typeof(OwnMouseOver), just as sown in the image below:



```
//List of types that can be localized
//Please: ADD YOU OWN CLASSES IF NEEDED!!
//For example, if you have youw own object with a string field
//that must be localized, add here this class name
//NOTE: MainMaterial is added by default if exists
public System.Type[] typesToLocalize = new[]
{
    typeof(Text),
    typeof(Image),
    typeof(AudioSource),
    typeof(TextMeshProUGUI),
    typeof(OwnMouseOver)
};
```