Translations Lab

Introduction

A lot of created products using Unity require to be localized. This plugin will help you in it. The plugin will collect all localized strings and will prepare localization files. Then, you can use Google Drive and load files to Google Sheet for further send to translators.Another script will help you export translations back to Unity. Plugin works with standard Unity Text class and also works with TextMeshPro. You also can use this plugin for localization strings in your scripts code - this plugin will collect these values too. For more information see documentation (https://docs.google.com/document/d/1-RqM8NqIyiSAutwrJF39qIHfWeENlFuZRSVdRybW7DE/edit?usp=sharing). If you have any questions or suggestions - email chillout.labs@gmail.com .

Installation

Add package from Asset store to your project. Note that this plugin depends on 2 external libraries: TextMeshPro and YamlDotNet (https://assetstore.unity.com/packages/tools/integration/yamldotnet-for-unity-36292). YamlDotNet included directly in Asset. If you have another version of YamlDotNet you can remove *TranslationsLab/Plugins/YamlDotNet* folder and use version that you have in project.

Note: If your version of YamlDotNet is older than version included in TranslationLab - highly recommend to use version from TranslationLab library

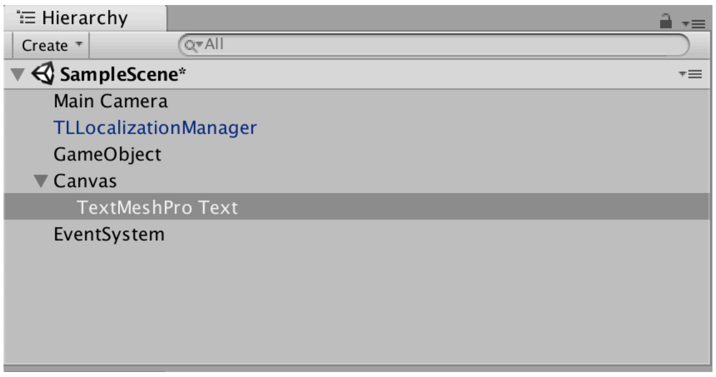
Note: if you use Unity 2017 and did not use TextMeshPRO before, you should to comment rows **4, 14 - 20, 26** in file *LocalizedText.cs* (Unity removed TextMeshPro from asset store and you will not able to download it). **Update:** Since **version 2.0.0** for Unity 2017 you need to uncomment define of USE_TEXT_MESH_PRO in row 1 in file *TextMeshProWrapper.cs* . For Unity higher than 2017 TMP supports automatically. For Unity 2018 do not forget to execute *Window -> TextMeshPro -> Import TMP Essential Resources*. After importing of TMP finished call *Window -> TextMeshPro -> Project Files GUID Remapping Tool*. After that you should be able to run demo

Note: Script uses .Net Runtime version 4. You can change it in Player settings. Go to *Player Settings -> Settings for PC, Mac & Linux -> Other settings -> Configuration and select* Scripting Runtime version to .Net v4. *Note that this may require changes to your existing code!*

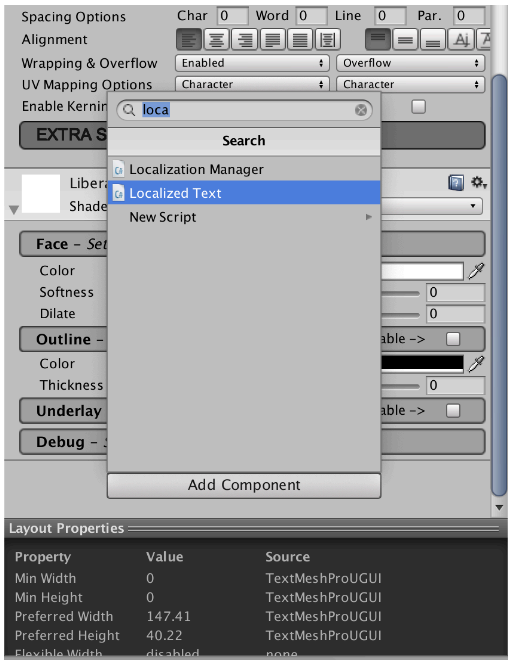
Adding translations value

In scene game object

To textMeshPro:
1) Create textMeshPro object (Create -> UI -> TextMestPro - text)

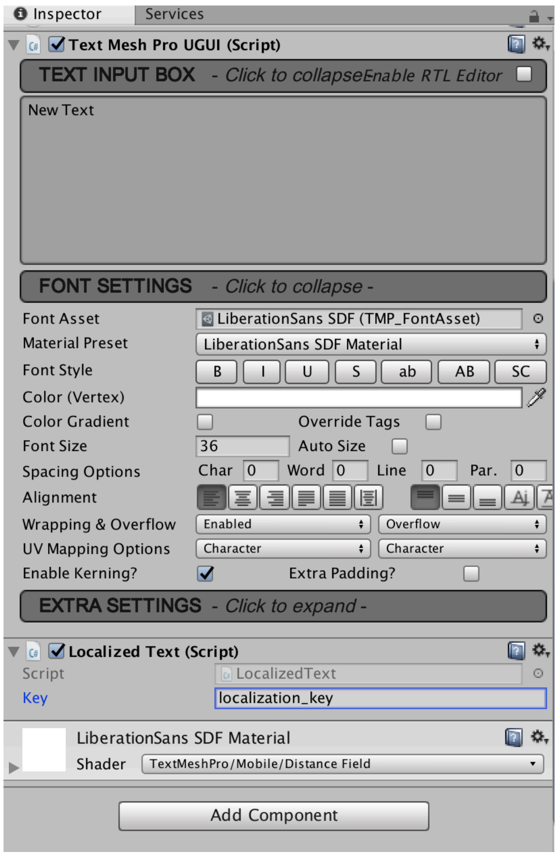


2) In Inspector find LocalizedText and add it to object


Spacing Options: Char 0, Word 0, Line 0, Par. 0
Wrapping & Overflow: Enabled, Overflow
UV Mapping Options: Character, Character

Property	Value	Source
Min Width	0	TextMeshProUGUI
Min Height	0	TextMeshProUGUI
Preferred Width	147.41	TextMeshProUGUI
Preferred Height	40.22	TextMeshProUGUI
Flexible Width	disabled	none

3) Setup localization key


Inspector | Services

Text Mesh Pro UGUI (Script)

TEXT INPUT BOX - *Click to collapse* *Enable RTL Editor* ☐

New Text

FONT SETTINGS - *Click to collapse -*

Font Asset: LiberationSans SDF (TMP_FontAsset)
Material Preset: LiberationSans SDF Material
Font Style: B I U S ab AB SC
Color (Vertex)
Color Gradient ☐ Override Tags ☐
Font Size: 36 Auto Size ☐
Spacing Options: Char 0 Word 0 Line 0 Par. 0
Alignment
Wrapping & Overflow: Enabled, Overflow
UV Mapping Options: Character, Character
Enable Kerning? ✓ Extra Padding? ☐

EXTRA SETTINGS - *Click to expand -*

Localized Text (Script)
Script: LocalizedText
Key: localization_key

LiberationSans SDF Material
Shader: TextMeshPro/Mobile/Distance Field

Add Component

For Unity Text class the same usage is correct, but on the first step you need to add Unity Text object.

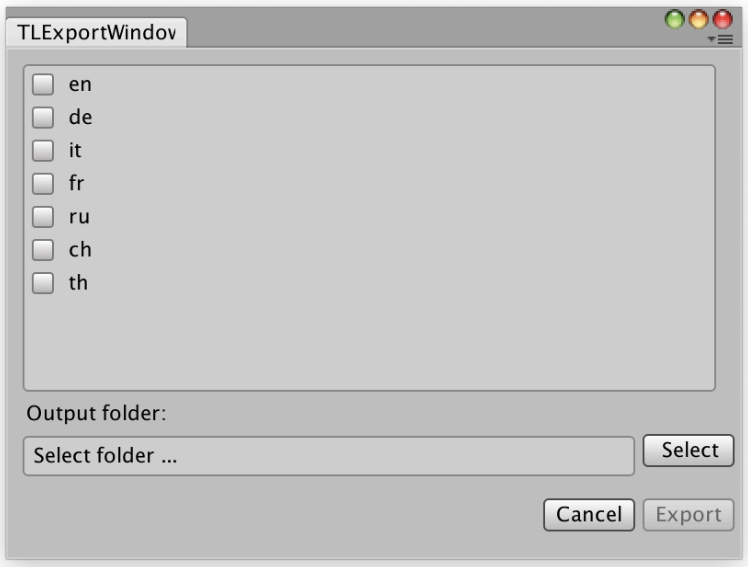
In scripts code

You can use a plugin to localize your strings in code. Export script will automatically collect these strings and add them to localization files. Usage example:

```
public void FuncWithTranslations() {
	var tr1 = LocalizationManager.GetLocalizedString("String to localize");
	var tr2 = LocalizationManager.GetLocalizedString("String with ) closing bracket");
	var tr3 = LocalizationManager.GetLocalizedString("String with ( opening bracket");
	var tr4 = LocalizationManager.GetLocalizedString("String with \" quotation mark");
	var tr5 = LocalizationManager.GetLocalizedString("String with \") quotation mark
and bracket");
	var tr6_7 = func(LocalizationManager.GetLocalizedString("String to localize1"),
LocalizationManager.GetLocalizedString("String to localize2"));
	var tr8 = LocalizationManager.GetLocalizedString(func("String to localize"));
}
```

Export

You can export localization files in JSON Format. To export localizations go to Window -> TranslationsLab -> Export Localizations. You will see the next window:



To export translations you should select languages and output folder.

Import

To import your translations to project, select *Window -> TranslationsLab -> Import translations*. Then find prepared *.json* files and select one, tap *Open* button. This operation will import translations in *Streaming Assets* folder. To see imported files in Unity editor - unfocus Unity editor and focus again (to cause assets importing) or reopen Unity editor.

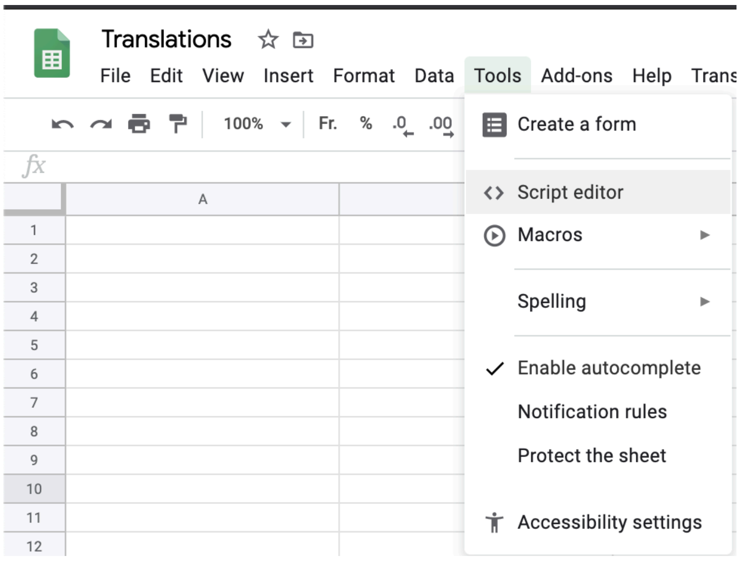
Google Sheet scripts usage

For translations exchange between developer and translator Google scripts were added. You can import json files, send link or invitations to translators. After translations are finished, you can export translations in JSON format and import them back to Unity. All scripts located at *TranslationsLab -> Editor -> GoogleScripts*

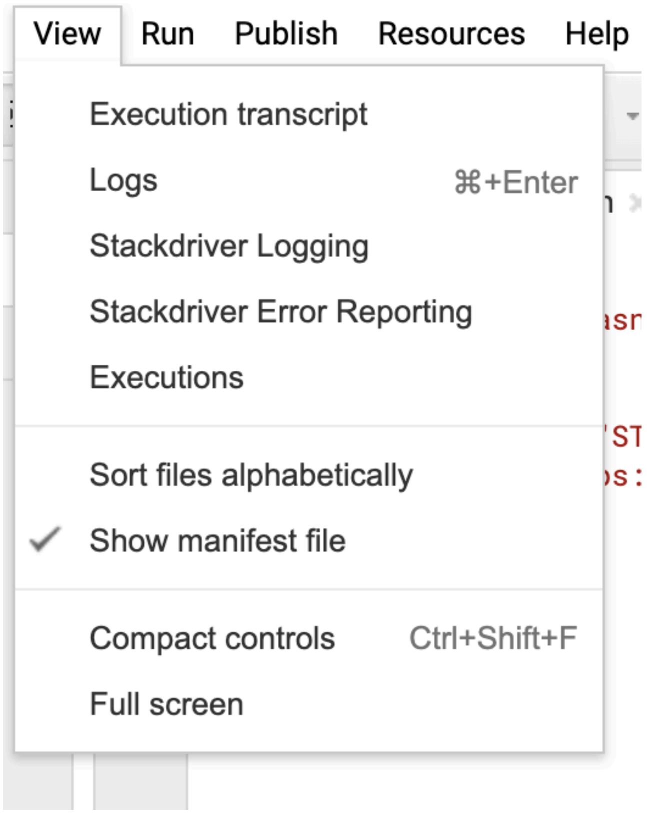
Prepare translations

1) Go to Google Drive and create a separate folder for translations.

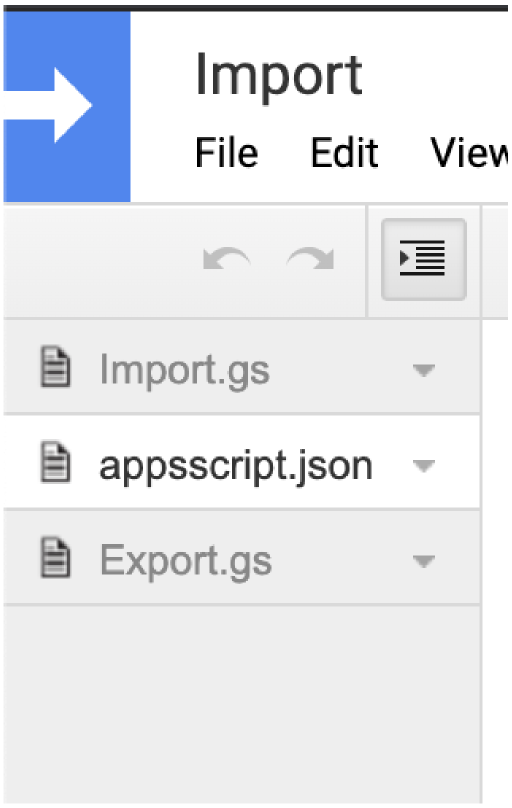
2) In this folder create ‘Input’ and ‘Output’ directories
3) Create Google Sheet document and open it
4) Go to Tools -> Scripts Editor



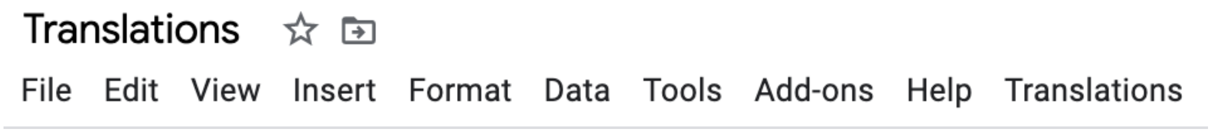
5) In opened tab go to View -> Show manifest



6) In script editor, in file ‘appsscript.json’ copy content of ‘appsscript.json’ from reporitory and save.
7) In the left files menu, create files ‘Import.gs’ and ‘Export.gs’. Copy content of files with the same names from repository.



8) Save changes and close scripts editor
9) If all went fine - you should see new tab in top panel with name ‘Translations’

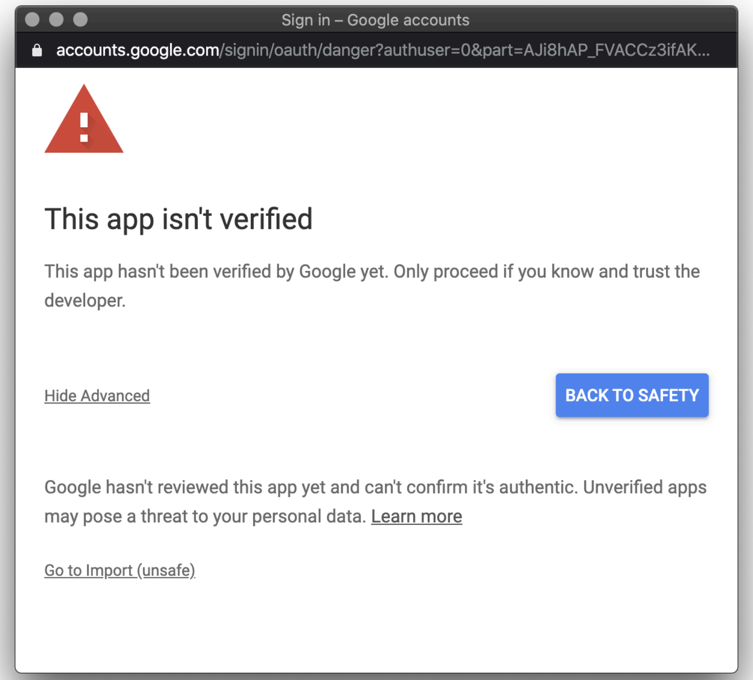

Translations
File Edit View Insert Format Data Tools Add-ons Help Translations

10) If you do not see this tab - try to reopen Google sheet. If a problem occurs after reopen - recheck all setup steps. If nothing helps - write an email on the support address.

Import translations

1) Export translations from Unity. You should have some json files for translations.
2) Put this files in ‘Input’ folder which you created on preparation step
3) Go to Google Sheet. Then go to ‘Translations -> Import translations...’
4) Then scripts will ask you for permissions. Because the app (script) is not published, it will be shown as a warning. Firstly, select an account for script usage.

5) In opened window select ‘Advanced’ and then ‘Go to *project name*’


Sign in – Google accounts

accounts.google.com/signin/oauth/danger?authuser=0&part=AJi8hAP_FVACCz3ifAK...

This app isn't verified

This app hasn't been verified by Google yet. Only proceed if you know and trust the developer.

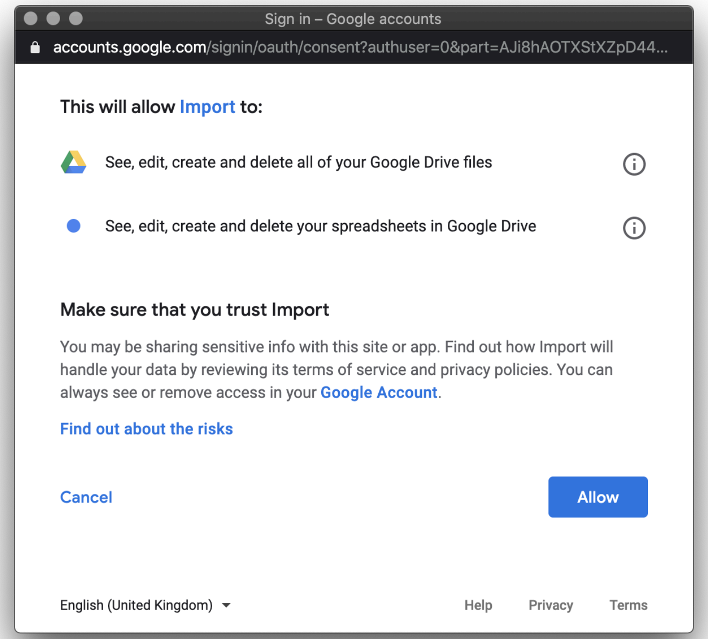
Hide Advanced

BACK TO SAFETY

Google hasn't reviewed this app yet and can't confirm it's authentic. Unverified apps may pose a threat to your personal data. Learn more

Go to Import (unsafe)

6) Then allow access for drive


Sign in – Google accounts

accounts.google.com/signin/oauth/consent?authuser=0&part=AJi8hAOTXStXZpD44...

This will allow Import to:

See, edit, create and delete all of your Google Drive files

See, edit, create and delete your spreadsheets in Google Drive

Make sure that you trust Import

You may be sharing sensitive info with this site or app. Find out how Import will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.

Find out about the risks

Cancel

Allow

English (United Kingdom)

Help Privacy Terms

After this steps you should be able to import translations

Export translations

1) Open Google Sheet with finished translations
2) Go to ‘Translations -> Export translations...’
3) App will ask you permissions, same as for import
4) After the script is finished you can take ready json files in the ‘Output’ directory. Import these files back to Unity project

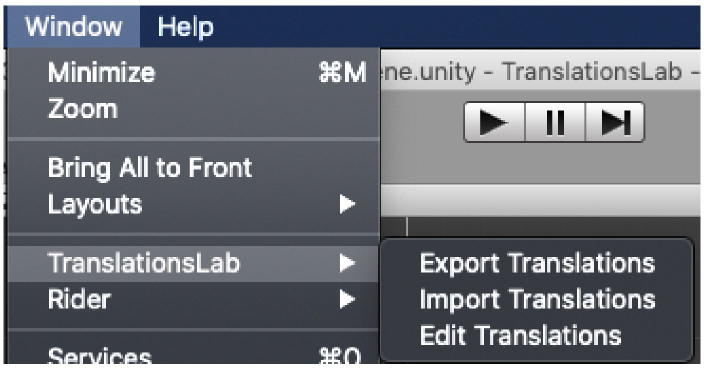
Settings

PossibleLanguages	List of possible languages. It will be shown in the export window (you can select it). If you want a special language you can add it to this property.
SelectedLanguages	Selected languages in the last export
ShowWarningIfDuplicatedKeyFound	Prints a warning to log if a duplicated key is found
PathsToExclude	List of excluding paths. For future releases
PathToLocalizationsScripts	Path to LocalizationManager and LocalizedText script. Used when exporting localization from scenes. WARNING: If you move these files - fix this path to correct
OutputFolder	Subdirectory, in which files will be exported

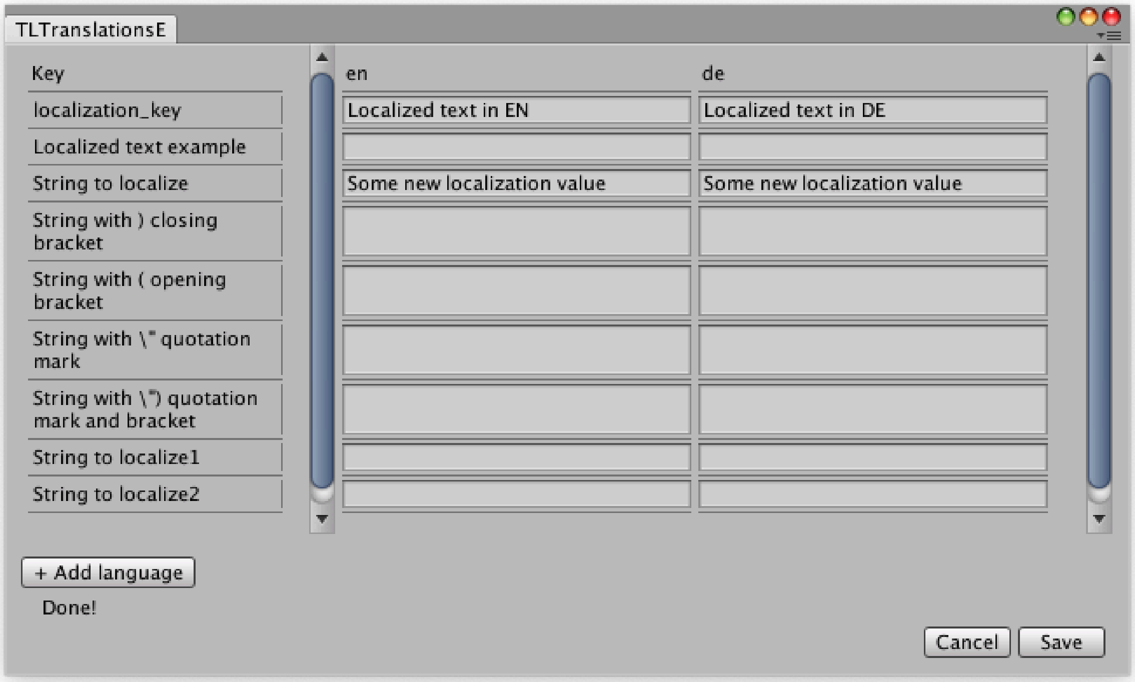
Translations edit

Note: In version lower 2.0.1 there were some troubles with translation edit. To fix them, set up languages using Window -> Translations Lab -> Project Settings.

In **version 1.1.0** translations edit were added directly in Unity Editor. You can edit translations without exporting them. You can find translations editing in *Window -> TranslationsLab -> Edit Translations* .

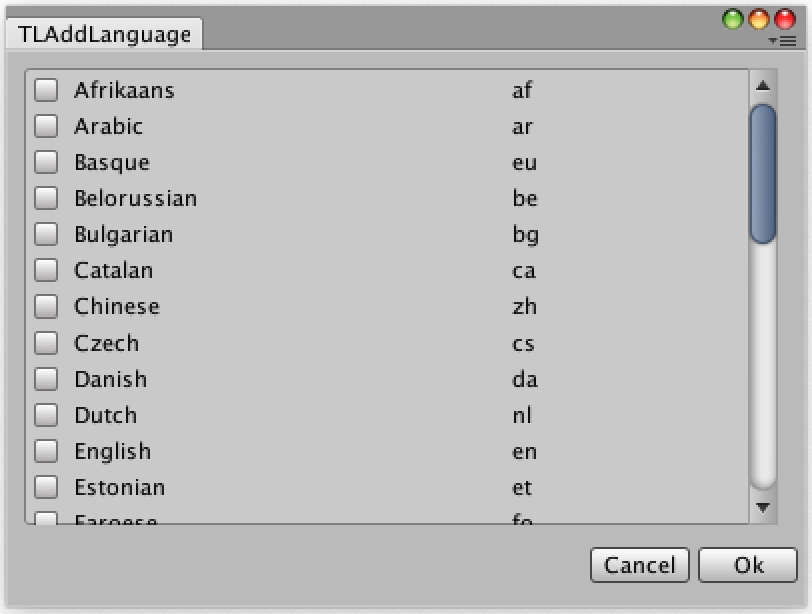


When you select “Edit translations”, some translations gather process will run and takes some time. If you don’t see recent added translations - try save scene or re-launch Unity. Progress of this process you can see in status label under *“+ Add language”* button.


TLTranslationsE

Key	en	de
localization_key	Localized text in EN	Localized text in DE
Localized text example		
String to localize	Some new localization value	Some new localization value
String with) closing bracket		
String with (opening bracket		
String with \" quotation mark		
String with \") quotation mark and bracket		
String to localize1		
String to localize2		

After adding translations, you can tap *“Save”* button and save your edited data. It will be saved in *Streaming Assets* folder. Also you can add language from list of supported languages. Just tap *“+ Add language”* button and select language which you needed from the list.



☐ Afrikaans	af
☐ Arabic	ar
☐ Basque	eu
☐ Belorussian	be
☐ Bulgarian	bg
☐ Catalan	ca
☐ Chinese	zh
☐ Czech	cs
☐ Danish	da
☐ Dutch	nl
☐ English	en
☐ Estonian	et
☐ Faroese	fo

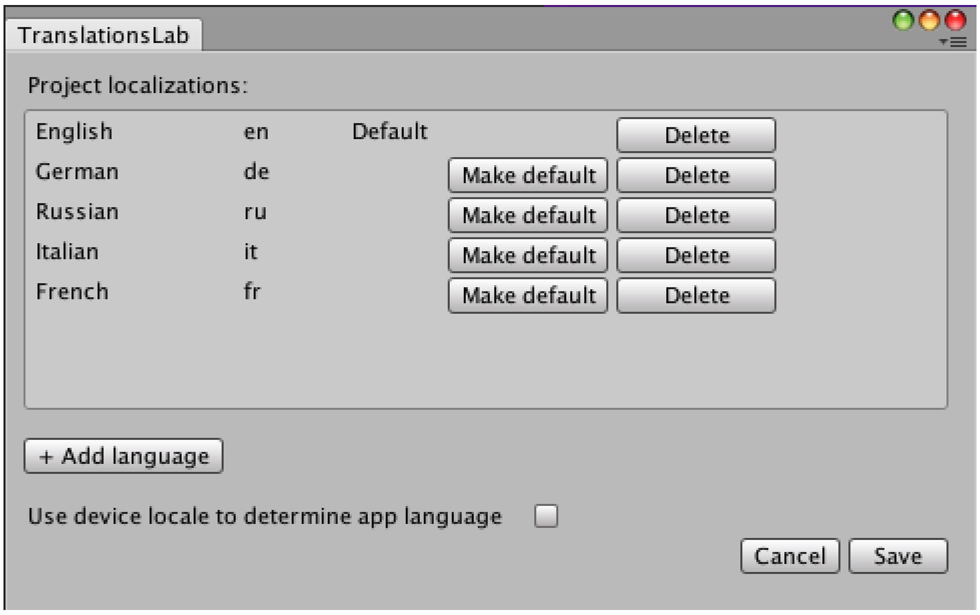
In game language change support

We implement in-game language change very often in our products. In case of in-game language change we need mechanism to update translations in existing objects. Since **version 2.0.0** we added this support. You can implement language change screen in your game, setup button language handler and call:

```
LocalizationManager.GetInstance().ChangeLanguage(language);
```

If you used TranslationsLab asset - all texts in objects will update automatically.

Also we added project translations settings to setup your project translations more clear and comfortable. You can open setting in *Window -> TranslationsLab -> Project settings*. Then you can see this window:


TranslationsLab

Project localizations:

English	en	Default		Delete
German	de		Make default	Delete
Russian	ru		Make default	Delete
Italian	it		Make default	Delete
French	fr		Make default	Delete

+ Add language

Use device locale to determine app language ☐

Cancel Save

Here you can add and remove languages, setup any language as default, which will be shown in game if device language not supported. Also you can setup flag, which will make LocalizedManager firstly to check device locale, and if language supported - use it. It comfortable to use if you do not want to add in-game language change and use device locale. If you do not set this settings, default language will be used.

Run demo

Since **version 2.0.0** asset includes Demo scene. It located at *TranslationsLab/Demo*. TranslationsLab uses *StreamingAssets* folder as resources for translations. But Unity project can contain only one such folder. So, you need to copy translations from *TranslationsLab/Demo/LocalizationResources* to your *StreamingAssets* folder. If you do note make this - demo scene will not contain translations and will work incorrectly. Sorry for this uncomfortable situation, but we working on this issue and try to make our product better. Thanks for understanding