Documentation

website: https://aspermind.com

follow us: https://twitter.com/Aspermind

email us: support@aspermind.com

Presentation

Thank you for buying UniLang! The purpose of this plug-in is to make the translation of your games and tools a breeze. This document will help you get started with UniLang, and will show you how to create your text libraries and insert them into your program.

Note that UniLang needs Unity 4.6 or higher to work.

Table of content

Presentation 2
What's in the package 3
The Plug-in DLL 3
The Prefabs 3
The demo 3
Getting started with UniLang 4
The library of translated texts 4
The Translation Manager component 5
The Translated Text component 7
Using UniLang in code 8
Change the active language 8
Trigger an event at language change 9
Get a line of text 10
Script reference 11
Translation Manager 11
Translation Item 12

What's in the package

The Plug-in DLL

UniLang is available as a `.dll` file in the `UniLang/Plugins` directory. As soon as you import the package, UniLang will be available in your project.

The Prefabs

In the `UniLang/Prefabs` directory, you will find 2 prefabs that contain the 2 components offered by UniLang.

- **Translation Manager:** This component contains the UniLang core and must be in your scene for UniLang to work.
- **Translated Text:** this component is for use in your UI canvas and allows you to display translated text in a `Text` component.

The demo

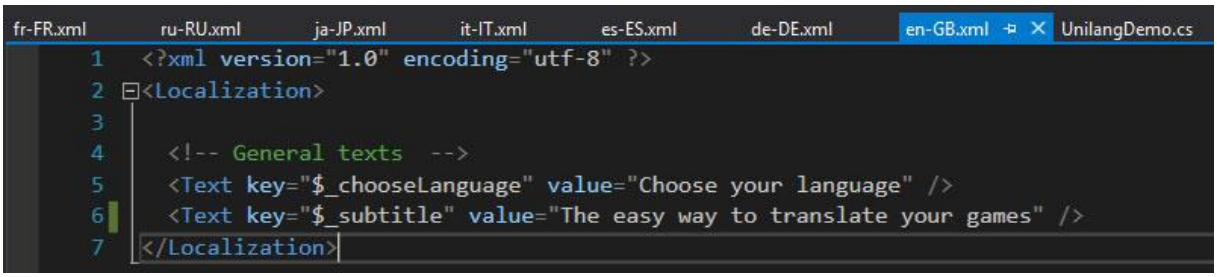
UniLang contains a simple demo with commented source code to serve as a working example.

To try it, open the scene in the `UniLang/Demo` directory.

Getting started with UniLang

The library of translated texts

Create an XML file, and format it as follows:



Code :

```
<?xml version="1.0" encoding="utf-8" ?>
<Localization>
  <Text key="$_chooseLanguage" value="Choose your language" />
  <Text key="$_subtitle" value="The easy way to translate your
games" />
</Localization>
```

The file must start with

```
<?xml version="1.0" encoding="utf-8" ?>
```

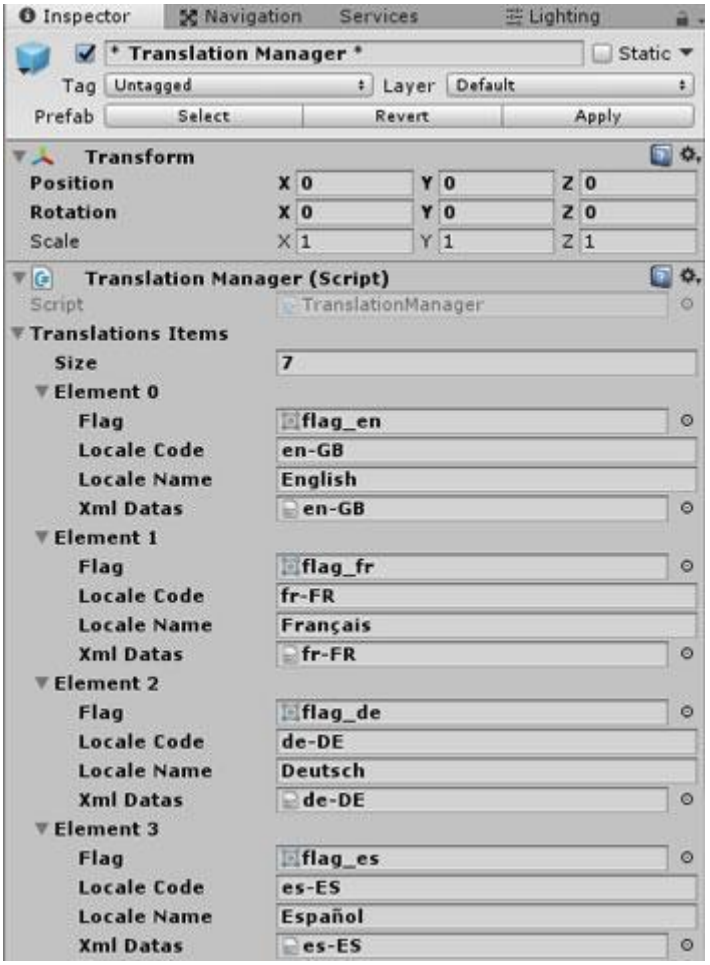
Otherwise, you will get an error when starting the application.

Then, within the "`Localization`" node, for each line of text in your application, create a "`Text`" node with a key / value pair, the key being the code name of the line, and the value being the text to display.

Make an XML file by language, and name it so that it can be found without any problem.

The Translation Manager component

In your scene, create a `GameObject` and add the `Translation Manager` component.



The component offers you to fill in a list of objects of type "`Translation Item`". The size of this collection is the number of languages you want to offer in your software. You can start with one, then add as your development cycle progresses.

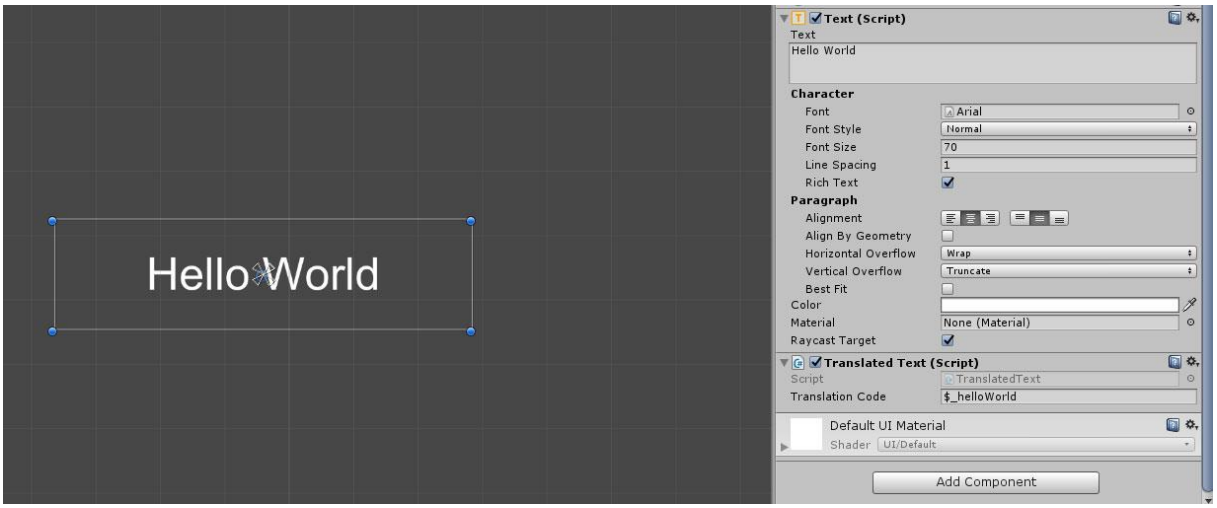
Each element contains 4 fields:

- **`Flag`** *(facultative):* a sprite containing the flag associated with that language. Useful for making a language selection screen (as in the demo).
- **`Locale Code`** *(facultative):* the ISO 639x code of the language. May be useful in some applications or in combination with other plug-ins.
- **`Locale Name`** *(facultative)***:** the name of this language.
- **`Xml Datas`:** the XML file containing your text library as key / value pairs.

The Translated Text component

This component, very useful for creating your user interfaces, dynamically changes the text of a `Text` component according to the active language.

In your scene, create a Canvas UI and create an element of type `Text`. On this object, add the `Translated Text` component.



In the `Translation Code` field, insert the key of one of your texts.

Et voilà! This item will change its text automatically according to the language selected in the `Translation Manager`.

Using UniLang in code

There are several things you can do by code, such as triggering an event when the language changes (for example, to refresh your interface).

The first thing to do, is to declare the reference:

```
using UniLang;
```

Change the active language

To change the active language, you must call `TranslationManager.instance.ChangeLanguage()` with as parameter the index number of the "Translation Item" list.

Exemple :

```
public void NextLanguage()
{
   this.currentLang++;

   if (this.currentLang == this.maxLang)
      this.currentLang = 0;

   TranslationManager.instance.ChangeLanguage(this.currentLang);
}
```

Trigger an event at language change

To trigger an event at language change, you must subscribe a method to the event `TranslationManager.instance.OnChangeLanguage`.

Exemple:

```
void Start()
{
      TranslationManager.instance.OnChangeLanguage +=
      ChangeLanguage;
}

private void ChangeLanguage()
{
      this.imgFlag.sprite =
      TranslationManager.instance.CurrentFlag;
}

private void OnDestroy()
{
      TranslationManager.instance.OnChangeLanguage -=
      ChangeLanguage;
}
```

This code changes the sprite of an image with the flag of the selected language.

Do not forget to unsubscribe the method to the event at the destruction of the `GameObjet`, so you won't encounter any errors.

Get a line of text

To get a line of text in your text library, just call the method with the text key as parameter.

Exemple:

```
this.txtSubtitle.text =
TranslationManager.instance.GetString("$_subtitle");
```

Script reference

Translation Manager

Accessors

CurrentLangCode	Get	string	Returns the ISO 639x code of the active language.
CurrentLangName	Get	string	Returns the name of the active language.
CurrentFlag	Get	Sprite	Returns the sprite of the active language flag.
CurrentLangCode	Get	List<TranslationItem>	Returns the list of languages as <TranslationItem> objects
OnChangeLanguage	Get/Set	ChangeLanguageEvent	Returns the delegate for the event triggered at language change

Methods

ChangeLanguage(int languageId)	void	Change the active language for the language provided as parameter
GetString(string strCode)	string	Returns the matching text with the key provided as a parameter

Translation Item

Accessors

LocaleId	Get	string	Returns the ISO 639x code of the language.
LocaleName	Get	string	Returns the name of the language.
Flag	Get	Sprite	Returns the sprite of the language flag.
XmlDatas	Get	TextAsset	Returns the TextAsset object containing the text library xml file