Contents

Demo

Import the asset and run **LocalisationExample** scene for features demonstration.

With the scene loaded, you can click on one of the three buttons to change UI to the desired language.

By default, if a localisation is not set to an optional language, it will display the content for the main language.

After testing, feel free to delete the entries on **LanguageManager** and **LocalisationManager**.

TextMeshPro Support

Base asset only support Text components, for those who are using TextMeshPro asset, there is a built-in extension **L&T_TMProExtension**.

If you want TMPro support, load up the unity package located on **Utilities** folder and import the files, replacing existing ones.

How to use it?

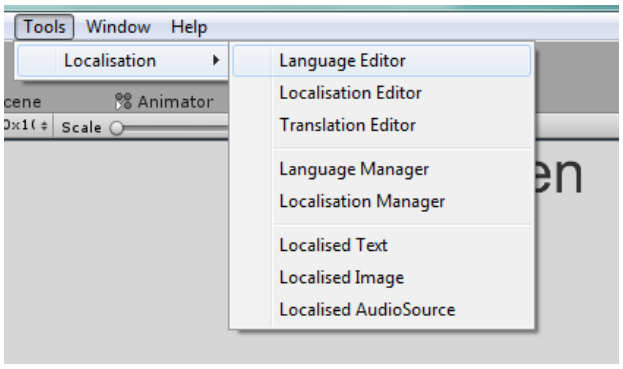
Manage Languages

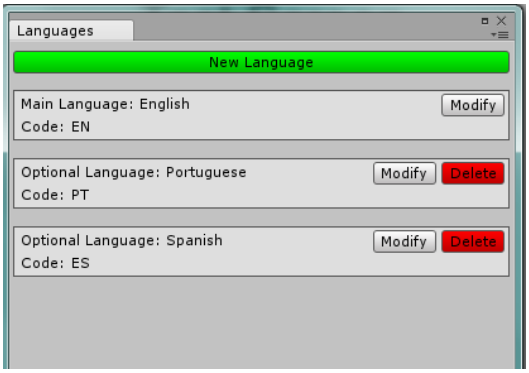
TLDR:

1) Open **Language Editor** at Tools/Localisation/Language Editor;
2) Add new languages with New Language button;
3) Edit existing languages with Modify button;
4) Remove existing languages with Delete button;

Note: Name and code must be unique.

To manage languages that will be available in your game, open the **Language Editor** on Unity menu bar Tools/Localisation/Language Editor.





By default, English is the main language; you can change to whatever you want through the 'Modify' button.

To create additional languages click the 'New Language' button. You must give a unique name and a code for it.

You can also remove a language if you do not want it anymore by clicking on the 'Delete' button.

Set Translations

TLDR:

1) Add a **GUITranslator** to the scene by clicking on Tools/Localisation/GUITranslator;
2) Add a **Localised Text** to the scene by clicking on Tools/Localisation/Localised Text;
3) Give a unique key to the newly created localised text.

To have a text be translated during runtime, first add the script **GUITranslator** to an empty object on the scene. To do this you can click on GUITranslator on menu Tools/Localisation/GUITranslator or manually add the script to a GameObject. Note that doing manually, you have to both LanguageManager and LocalisationManager assets do the GUITranslator.

After setting up the GUITranslator, you can create a new localized text clicking on **Localised Text** on menu Tools/Localisation/Localised Text. Alternatively, you can add the LocalisedText component to an existing text object.

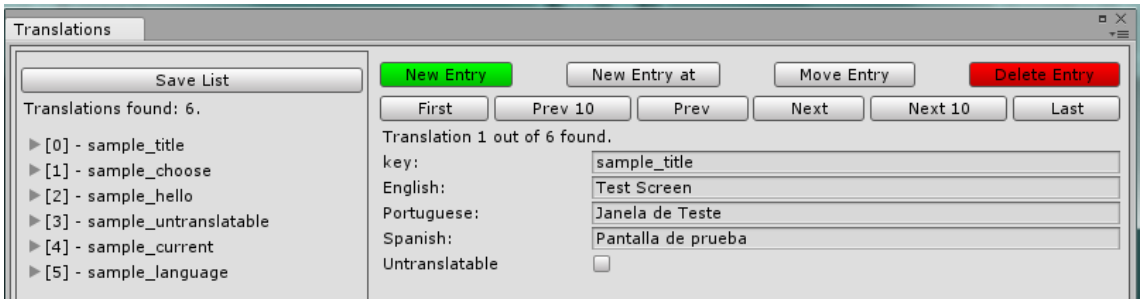
Manage Translations

TLDR:

1) Open **Translator Editor** at Tools/Localisation/Translator Editor;
2) Add new entries with 'New Entry' or 'New Entry At' buttons;
3) Move existing entries to another position with 'Move Entry' button;
4) Remove entries with 'Delete Entry' button;
5) Set Untranslatable to use main language translation only;
6) Save recent changes with 'Save List' button;

Note: Key must be unique.

To manage your translations, open the **Translation Editor** on menu bar Tools/Localisation/Translation Editor.



On the left side, there is a list of all entries created. You can click on any entry to start modifying it on the right side of the screen.

With an entry selected, on the right side it will appear a list of current languages available for translation. Here you can change the key for a new unique one.

It is possible to set this entry as **Untranslatable** if you decide to not having it being translated. Doing this is similar to a text component without a LocalisedText component, but it gives the benefit of easily switching on/off.
To add new entries, click the 'New Entry' Button at the top of the screen. It is also possible to create a new entry in a given position with the 'New Entry at' button.
To move entries to a different position the list, click on the 'Move Entry' button.
Finally, you can remove entries by clicking on the "Delete Entry' button.
After editing, click the 'Save List' button to save all recent changes.

Set Image/Audio Localisations

TLDR:

1) Add a **GUITranslator** to the scene by clicking on Tools/Localisation/GUITranslator;
2) Add a **Localised Image** to the scene by clicking on Tools/Localisation/Localised Image;
3) Add a **Localised AudioSource** to the scene by clicking on Tools/Localisation/Localised AudioSource;
4) Give a unique key to the newly created localised objects.

Note: Different type objects can have the same key.

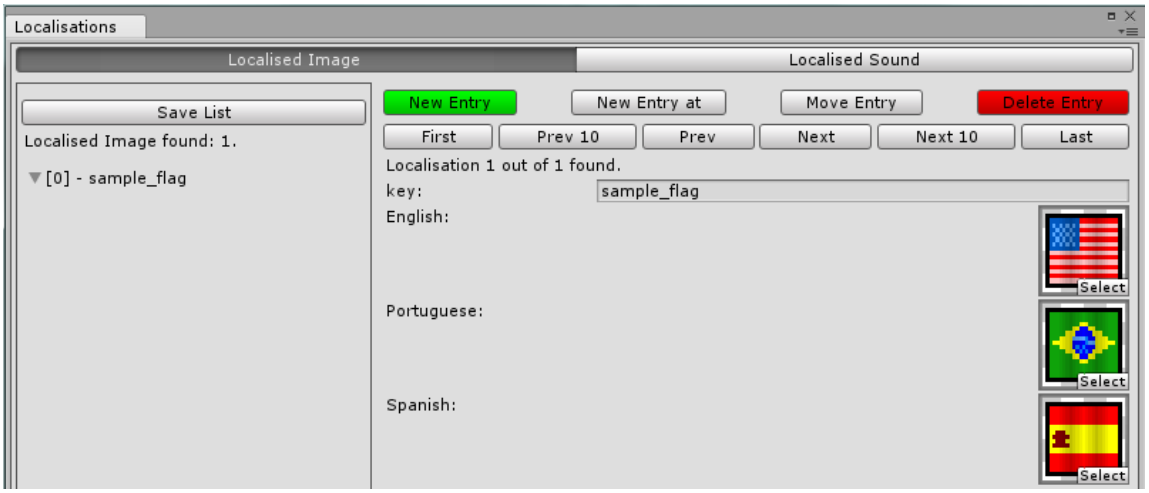
It is possible to localize audio and image as well. First, add the script **GUITranslator** to an empty object on the scene. To do this you can click on GUITranslator on menu Tools/Localisation/GUITranslator or manually add the script to a GameObject. Note that doing manually, you have to both LanguageManager and LocalisationManager assets do the GUITranslator.
After setting up the GUITranslator, you can create a new localized image clicking on **Localised Image** on menu Tools/Localisation/Localised Image. Alternatively, you can add the LocalisedImage component to an existing image object.
Similarly, you can create a localised audio by clicking on **Localised Audio** on menu Tools/Localisation/Localised Audio. Doing this, it will automatically create a new AudioSource object with a LocalisedAudio component.

Manage Image/Audio Localisations

TLDR:

1) Open **Localisation Editor** at Tools/Localisation/Localisation Editor;
2) Select between **Localised Image** and **Localised Audio** to edit;
3) Add new entries with ‘New Entry’ or ‘New Entry At’ buttons;
4) Move existing entries to another position with ‘Move Entry’ button;
5) Remove entries with ‘Delete Entry’ button;
6) Save recent changes with ‘Save List’ button;

To manage your translations, open the **Localisation Editor** on menu bar Tools/Localisation/Localisation Editor.



On top, there is a toolbar where you can select between **Localised Image** and **Localised Audio** to edit.

On the left side, there is a list of all entries created. You can click on any entry to start modifying it on the right side of the screen.

With an entry selected, on the right side it will appear a list of current languages available for localisation. Here you can change the key for a new unique one, also note that the same key can be used once one each type of file.

To add new entries, click the ‘New Entry’ Button at the top of the screen. It is also possible to create a new entry in a given position with the ‘New Entry at’ button.

To move entries to a different position the list, click on the ‘Move Entry’ button.

Finally, you can remove entries by clicking on the “Delete Entry’ button.

After editing, click the ‘Save List’ button to save all recent changes.

Calling GUITranslator on scripts

TLDR:

1) Add a **GUITranslator** to the scene by clicking on Tools/Localisation/GUITranslator;
2) Add a reference of GUITranslator on the desired script or use the singleton GUITranslator.Instance;
3) Call GUITranslator.**CheckLanguageAndCode** to automatically update GUI to application's language;
4) Call GUITranslator.**UpdateGUI** to update GUI with a given language Code;
5) Call GUITranslator.**GetLocalisedText** to get translation of given key on current language;

Methods:

- public void **CheckLanguageAndCode** ( )
- public void **UpdateGUI** (string **code**)
- public string **GetLocalisedText** (string **key**)
- public Sprite **GetLocalisedImage** (string **key**)
- public AudioClip **GetLocalisedAudio** (string **key**)

The script **TranslationSample.cs** demonstrates how to call GUITranslator.cs to update language settings.

To call GUITranslator methods you can use the singleton GUITranslator**.Instance**.

```
10      void Start ()
11      {
12          guiTranslator = GUITranslator.Instance;
13
14          //Automatically sets language based on application's language
15          guiTranslator.CheckLanguageAndCode ();
```

Using GUITranslator.**CheckLanguageAndCode** it will automatically set current language to the application's language and update the GUI.

Each button on the **SampleLocalisation** scene pass a string with the code of the designated language as seen below:

```
21      //Select language with given code
22      public void ButtonLanguage (string code)
23      {
24          //Updates GUI with code for new language
25          guiTranslator.UpdateGUI (code);
26
27          //Manually updates expecific texts on the scene
28          UIDisplayUpdate ();
29      }
```

Using GUITranslator.**UpdateGUI**, it will update the GUI to the language with given code.

You can get current translations of a key directly using GUITranslator.**GetLocalisedText**, as seen below:

```
31      private void UIDisplayUpdate ()
32      {
33          //It is possible to get the translation of a single text instead of using a LocalisedObject.
34          //Get translation for given key in the current language
35          string current = guiTranslator.GetLocalisedText ("sample_current");
36          string language = guiTranslator.GetLocalisedText ("sample_language");
37          string code = guiTranslator.Code;
38
39          //Display current language for demonstration purpose
40          displayCurrentLanguage.text = string.Format ("{0} : {1} ({2})", current, language, code);
41      }
```

Make sure the key and translations exists on Translation Editor.

Translating Dropdown items

Add the **Localised Dropdown** component to an existing dropdown or by clicking on Tools/Localisation/Localised Dropdown.

On the Localised Dropdown component, add a key for each item present at the dropdown list.

Note: Make sure to have the same number of keys and items.

Word wrapping

To break a text into multiple lines add **\n** in your text exactly where you want to become a new line.

For example, the following text:

> “Lorem ipsum dolor sit amet,\nconsectetur adipiscing elit.”

It will be displayed as:

> “Lorem ipsum dolor sit amet
> consectetur adipiscing elit.”

Saving Current Language

By default, the current language will be stored on **PlayerPrefs** with the key “language”.

On Start, GUITranslator checks the last used language on PlayerPrefs to update the GUI. If none was found, then it sets to the main language.

Every time the language is changed, GUITranslator updates the PlayerPrefs.

Questions, Suggestions and Feedback

Feel free to email me at justkrated@gmail.com.