Using UniTranslate

Thank you for choosing UniTranslate!

UniTranslate is a localization library for Unity which allows you to create containers for localized resources, called Translation Assets. Those Translation Assets contain multiple keys which identify strings or other values. UniTranslate offers several features which help you create translation keys and migrate existing hardcoded strings in UI Texts or 3D Text Meshes.

1 Getting Started

After you imported the asset store package, you can take a look into the samples folder. Feel free to delete it if you do not need it anymore.

If you use other extensions with the JSON.NET (Newtonsoft.Json.dll) library, your project may break. Please delete one of the duplicate libraries before you continue.

To integrate UniTranslate into your project, you need to create at least one Translation Asset. Translation Assets are created using the Assets – Create – Translation menu or with a right click on the folder you want to place the asset in.

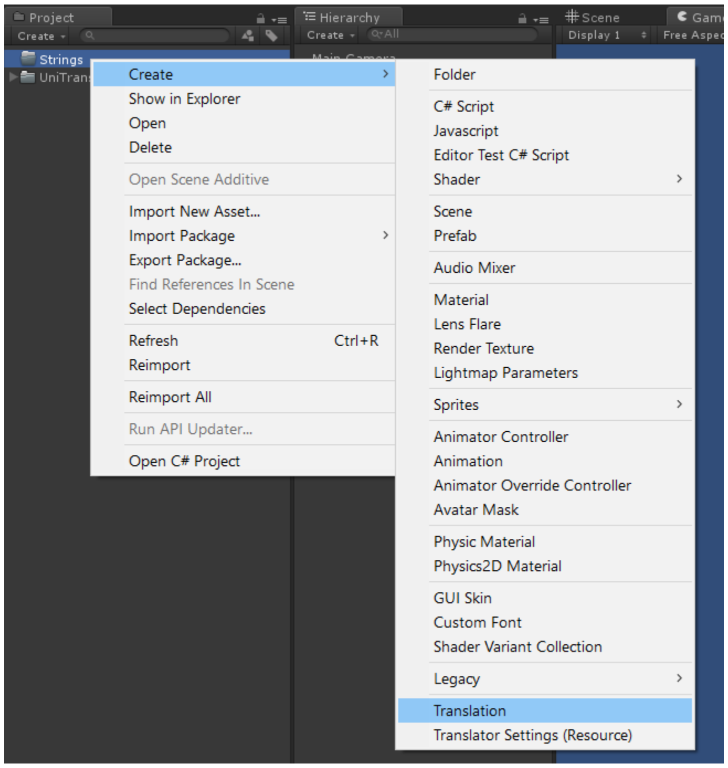
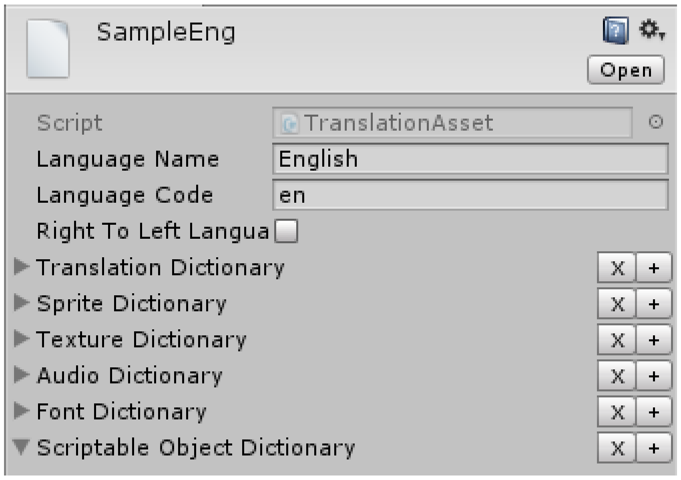


Figure 1: Adding Translation Assets

After creating an asset, you need to assign a language name and code in the inspector:



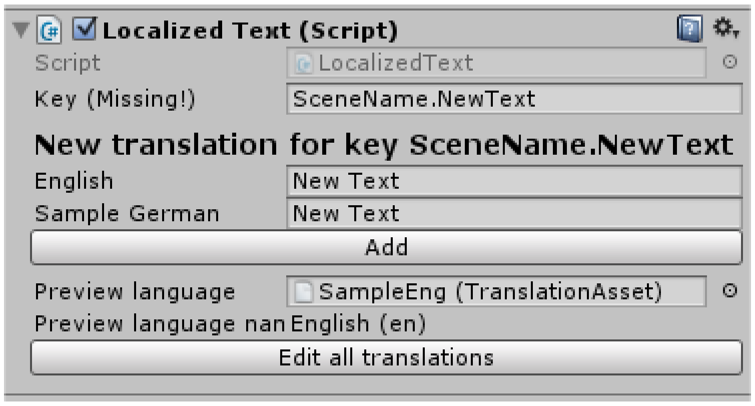
SampleEng
Open

Script	TranslationAsset
Language Name	English
Language Code	en
Right To Left Langua	☐

- ▶ Translation Dictionary X +
- ▶ Sprite Dictionary X +
- ▶ Texture Dictionary X +
- ▶ Audio Dictionary X +
- ▶ Font Dictionary X +
- ▼ Scriptable Object Dictionary X +

Figure 2: Configuring a Translation Asset

Now add a canvas and a UI text as its child. After you set your text object up, use the **Add Component** button to add a **LocalizedText** component. If you instead want to use UniTranslate with 3D TextMeshes, add a **LocalizedTextMesh** component. UniTranslate will try to generate a key for your translation based on the scene name and your existing text. Now edit the strings assigned to the languages and click **Add**.



Localized Text (Script)

Script	LocalizedText
Key (Missing!)	SceneName.NewText

New translation for key SceneName.NewText

English	New Text
Sample German	New Text

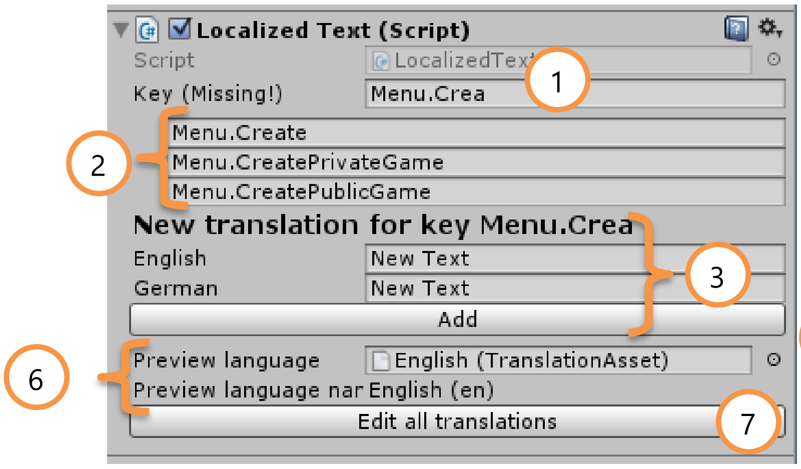
Add

Preview language	SampleEng (TranslationAsset)
Preview language nan	English (en)

Edit all translations

Figure 3: Adding keys inside the LocalizedText component

2 The In-component Key Editor



Localized Text (Script)

Script: LocalizedText

Key (Missing!): Menu.Crea

Menu.Create

Menu.CreatePrivateGame

Menu.CreatePublicGame

New translation for key Menu.Crea

English: New Text

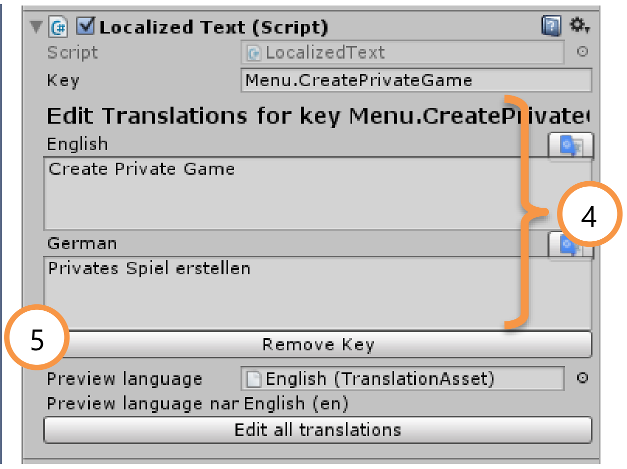
German: New Text

Add

Preview language: English (TranslationAsset)

Preview language nar English (en)

Edit all translations



Localized Text (Script)

Script: LocalizedText

Key: Menu.CreatePrivateGame

Edit Translations for key Menu.CreatePrivate

English: Create Private Game

German: Privates Spiel erstellen

Remove Key

Preview language: English (TranslationAsset)

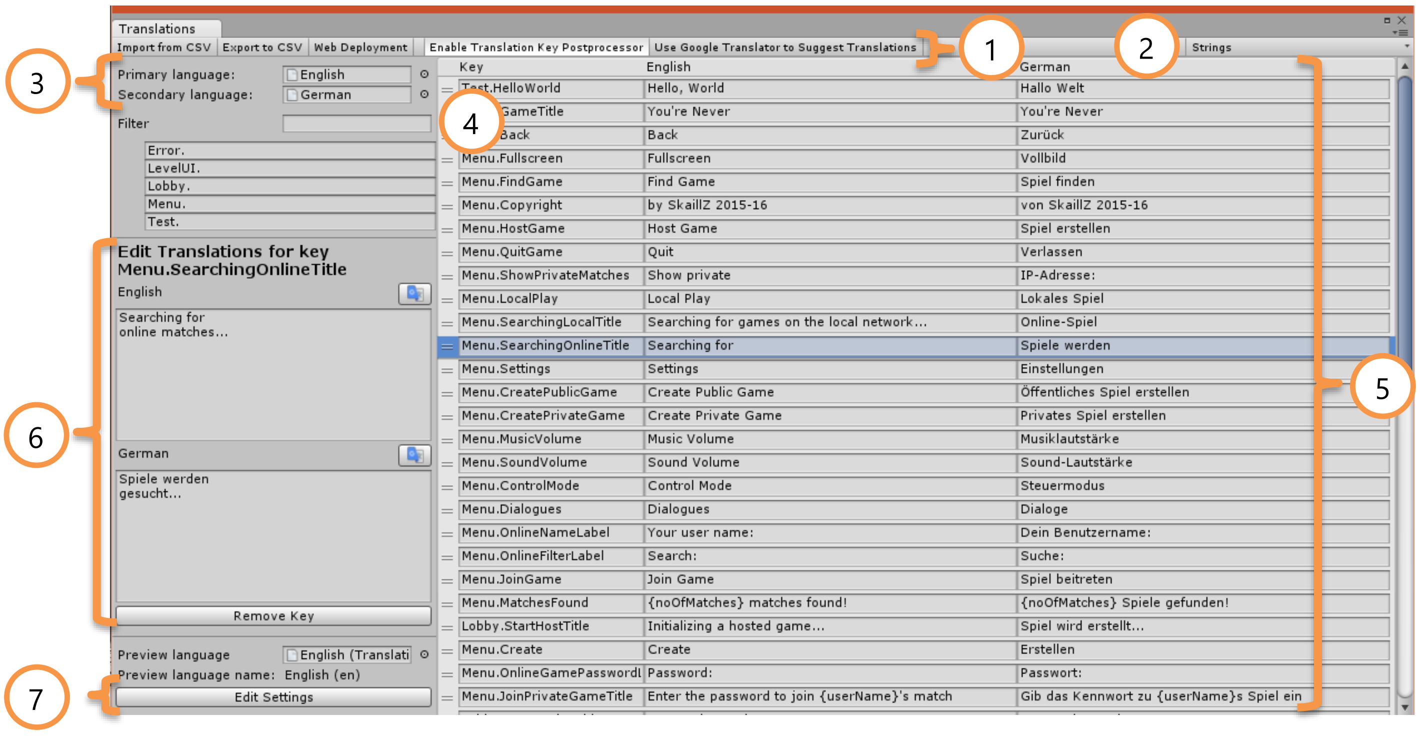
Preview language nar English (en)

Edit all translations

The in-component key editor supports your workflow by allowing you to use many features of UniTranslate inside the inspector of the currently edited translatable component.

1) Enter the translation key you want to use or create here. By convention, keys in UniTranslate contain a super key and a sub key, separated by a dot symbol.
2) This list shows the available sub keys for the entered super key, or available super keys if no super key is entered.
3) You can create a new key by assigning the values for each key and clicking the **Add** button or pressing the **Return** key two times. Press Tab to quickly navigate between the text fields.
4) If the specified key exists, the edit view is shown. In the edit view, you can edit multi-line values of the keys.
5) The **Delete Translation** button removes the translation key and values from all translation assets.
6) The preview language which is currently shown in the editor.
7) This button opens the localization editor window.

3 The Localization Editor Window


Translations
Import from CSV
Export to CSV
Web Deployment
Enable Translation Key Postprocessor
Use Google Translator to Suggest Translations
Strings
1
2
3
4
5
6
7

Primary language: English
Secondary language: German
Filter

Error.
LevelUI.
Lobby.
Menu.
Test.

Edit Translations for key Menu.SearchingOnlineTitle
English
Searching for
online matches...
German
Spiele werden
gesucht...
Remove Key

Preview language: English (Translati
Preview language name: English (en)
Edit Settings

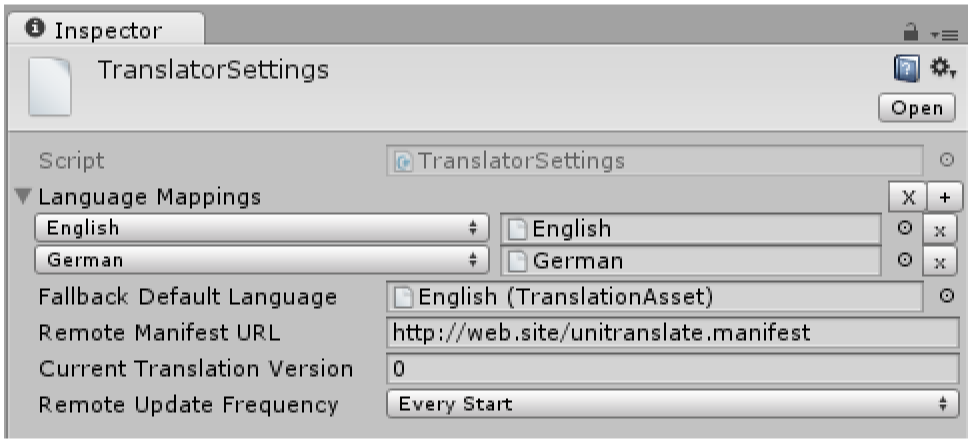
Key	English	German
Test.HelloWorld	Hello, World	Hallo Welt
GameTitle	You're Never	You're Never
Back	Back	Zurück
Menu.Fullscreen	Fullscreen	Vollbild
Menu.FindGame	Find Game	Spiel finden
Menu.Copyright	by SkaillZ 2015-16	von SkaillZ 2015-16
Menu.HostGame	Host Game	Spiel erstellen
Menu.QuitGame	Quit	Verlassen
Menu.ShowPrivateMatches	Show private	IP-Adresse:
Menu.LocalPlay	Local Play	Lokales Spiel
Menu.SearchingLocalTitle	Searching for games on the local network...	Online-Spiel
Menu.SearchingOnlineTitle	Searching for	Spiele werden
Menu.Settings	Settings	Einstellungen
Menu.CreatePublicGame	Create Public Game	Öffentliches Spiel erstellen
Menu.CreatePrivateGame	Create Private Game	Privates Spiel erstellen
Menu.MusicVolume	Music Volume	Musiklautstärke
Menu.SoundVolume	Sound Volume	Sound-Lautstärke
Menu.ControlMode	Control Mode	Steuermodus
Menu.Dialogues	Dialogues	Dialoge
Menu.OnlineNameLabel	Your user name:	Dein Benutzername:
Menu.OnlineFilterLabel	Search:	Suche:
Menu.JoinGame	Join Game	Spiel beitreten
Menu.MatchesFound	{noOfMatches} matches found!	{noOfMatches} Spiele gefunden!
Lobby.StartHostTitle	Initializing a hosted game...	Spiel wird erstellt...
Menu.Create	Create	Erstellen
Menu.OnlineGamePasswordL	Password:	Passwort:
Menu.JoinPrivateGameTitle	Enter the password to join {userName}'s match	Gib das Kennwort zu {userName}s Spiel ein

You can open the localization editor window in the menu Window – UniTranslate – Translation Editor

1) **Import from CSV:** Opens the CSV import wizard, which allows you to import localized strings from a CSV file.
   Export to CSV: Allows you to export your string localizations in comma-seperated values format
   Web deployment: Opens the web deployment wizard, which allows you to deploy and change the settings for string localization upgrades during run-time.
   Enable Translation Key Postprocessor: If enabled, UniTranslate will search for missing keys after a scene is loaded in the editor or processed in a build. The post processor might affect performance in the editor, but is not included in builds.
   Use Google Translator to Suggest Translations: If enabled, UniTranslate will use Google Translator to suggest translations whenever a new key is added on a text component based on its content. Ensure that your language codes are right, so that they are recognized by Google Translator.
2) Use this menu to set the type of localizations you want to edit. UniTranslate supports **strings**, **sprites**, **textures**, **audio clips**, **fonts** and **Scriptable Objects**.
3) In order to use the translation editor, you will need to select a **primary editing language**. You can also assign a **secondary language** to edit two languages simultaneously. Please note that all keys existing in the primary language are visible, so if you need to find missing keys, use the one where those keys exist as the primary language.

4) Use the **filter** field to filter translation keys by their starting characters
5) A list with all the localizations matching the filter. Please note that you can only reorder localizations if the filter is empty. Scroll to the bottom to add or delete an element from this list. You can also press the Del key to delete your selection.
6) You can edit strings with long multi-line content here. Click the **Google Translate** button next to the title of a language to automatically translate the string with Google Translator. Auto translations always use the preview language as the base language and overwrite the existing content of the string.
   Auto translations are not supported in the Web Player target due to platform limitations.
7) Click the **Edit Settings** button to select the Translator Settings asset in the inspector.

4 TranslationSettings



Script	TranslatorSettings
Language Mappings	
English	English
German	German
Fallback Default Language	English (TranslationAsset)
Remote Manifest URL	http://web.site/unitranslate.manifest
Current Translation Version	0
Remote Update Frequency	Every Start

Language Mappings: Click the plus icon on the right to add a new mapping, then assign a Translation Asset to the selected language. This way, your application will automatically determine which language should be used at startup. Please note that a mapping set to null could lead to

Fallback Default Language: This Translation Asset is used if the language mappings are empty or the user's language does not exist in the mappings.

The other settings are only used if web deployment is enabled:

Remote Manifest URL: The URL to the manifest with version data. Managed automatically by the Web Deployment Wizard (refer to 3. The Translation Editor window).

Current Translation Version: The version of the bundled localization strings. Updates are only downloaded if their version is higher than this number.

Remote Update Frequency: Determines how often UniTranslate checks for localization updates.

5 Translator Initialization

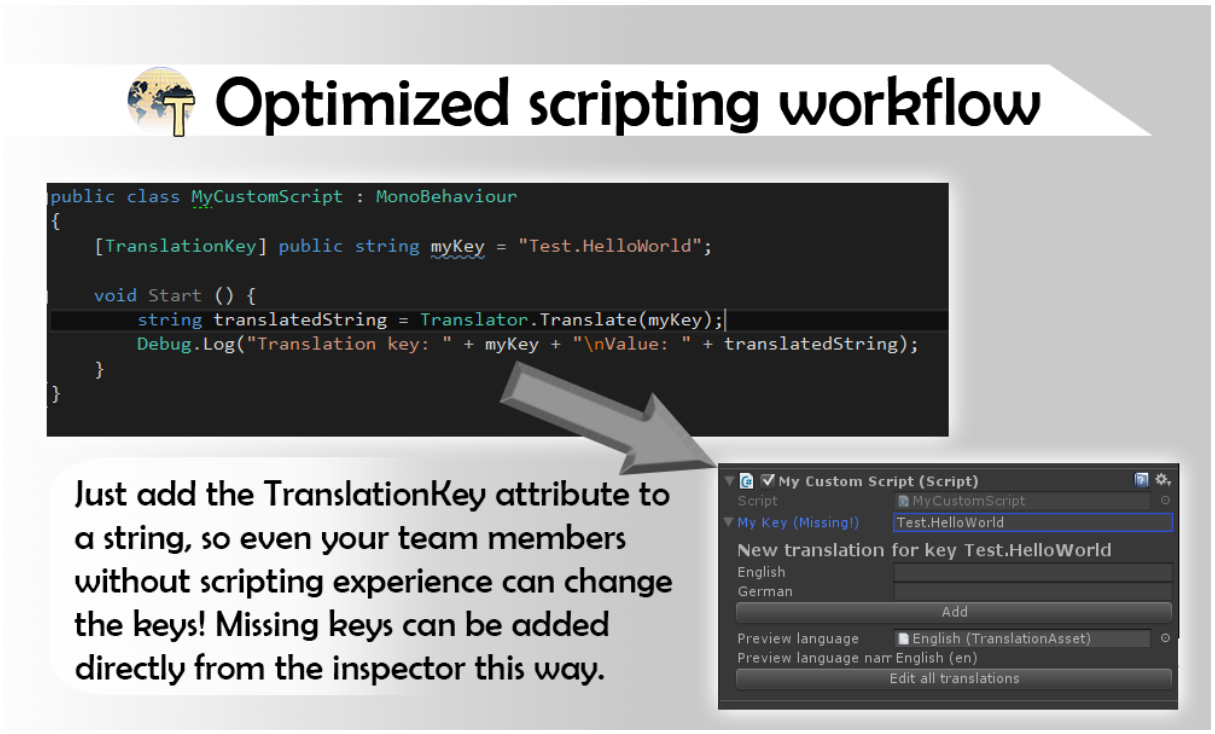
At runtime, UniTranslate will instantiate an invisible object called the Runtime Translator, which stores a reference to the current Translation Asset and resolves all API calls. By default, it is instantiated when the Translator instance is accessed the first time (which is the case with most API calls, e.g. the `Translator.Translate()` method). Moreover, UniTranslate will start checking for new updates right after the initialization.

This behavior works fine in the most cases, but you may want to bundle UniTranslates initialization process with your other initialization logic. To initialize the Translator manually, just call the `Translator.Initialize()` method before your first translation query.

6 Scripting Guide

This section is not a complete scripting reference, but a guide to the most important methods. Instead refer to the full documentation for detailed information.

Localizations can be accessed in scripts by calling `the Translator.Translate(string key)` function. You could just pass a string as the key, but that is not the recommended approach. Instead, use the `[TranslationKey]` attribute to mark a string as a translation key, so your staff can use the in-component editor (see Page 3) to assign keys.



```
public class MyCustomScript : MonoBehaviour
{
    [TranslationKey] public string myKey = "Test.HelloWorld";

    void Start () {
        string translatedString = Translator.Translate(myKey);
        Debug.Log("Translation key: " + myKey + "\nValue: " + translatedString);
    }
}
```

Just add the TranslationKey attribute to a string, so even your team members without scripting experience can change the keys! Missing keys can be added directly from the inspector this way.

My Custom Script (Script)
Script MyCustomScript
My Key (Missing!) Test.HelloWorld
New translation for key Test.HelloWorld
English
German
Add
Preview language English (TranslationAsset)
Preview language nam English (en)
Edit all translations

Translating sprites or other types in scripts

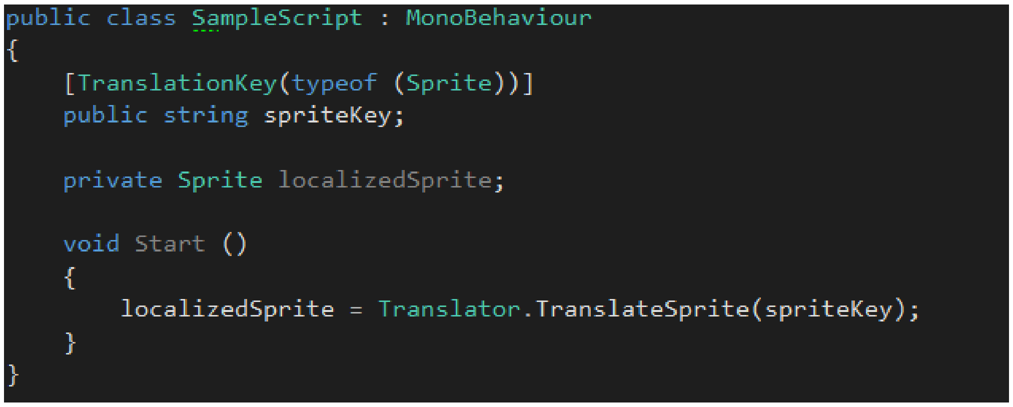
In order to provide type-safe access to translation values, different methods are used to get different kinds of values, like for example the `TranslateSprite(string key)` method for sprites.

static Sprite	**TranslateSprite** (string key)
static Texture	**TranslateTexture** (string key)
static AudioClip	**TranslateAudio** (string key)
static Font	**TranslateFont** (string key)
static ScriptableObject	**TranslateScriptableObject** (string key)

In contrast to the method for strings, those methods **return a null value if the specified key does not exist**, so you might find the following functions useful (or just want to check if the translated reference is null):

static bool	**SpriteExists** (string key)
static bool	**TextureExists** (string key)
static bool	**AudioExists** (string key)
static bool	**FontExists** (string key)
static bool	**ScriptableObjectExists** (string key)

If you use the `[TranslationKey]` attribute like before, you may encounter issues where correct type of key cannot be found in the editor. To solve this problem, just pass the desired type in the constructor of the attribute, like this:



```
public class SampleScript : MonoBehaviour
{
    [TranslationKey(typeof (Sprite))]
    public string spriteKey;

    private Sprite localizedSprite;

    void Start ()
    {
        localizedSprite = Translator.TranslateSprite(spriteKey);
    }
}
```

Tip: Examine the LanguageDropdown component to learn how to change languages at runtime. It is located in UniTranslate/Scripts.