Google Cloud Translation

- Intro:

Google Cloud Translation provides a simple programmatic interface for translating an arbitrary string into any supported language using state-of-the-art Neural Machine Translation. Translation API is highly responsive, so websites and applications can integrate with Translation API for fast, dynamic translation of source text from the source language to a target language (e.g., French to English). Language detection is also available in cases where the source language is unknown. The underlying technology pushes the boundary of Machine Translation and is updated constantly to seamlessly improve translations and introduce new languages and language pairs.

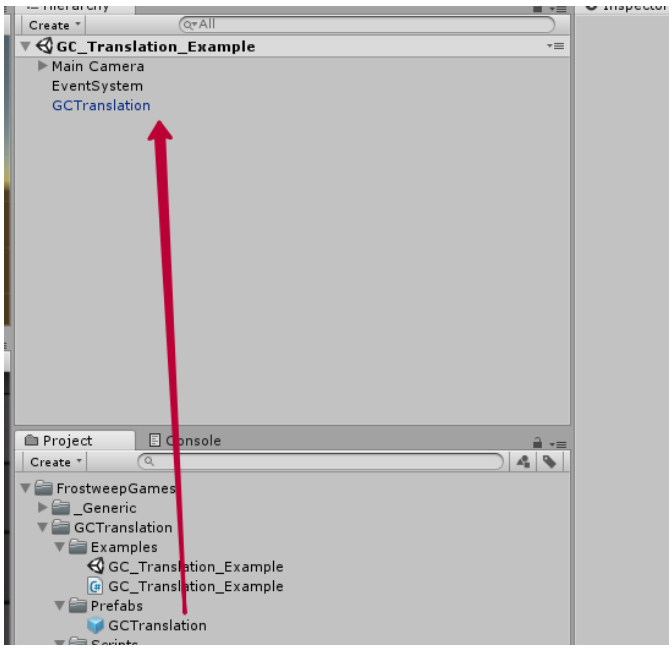
How to use:

Create you first an app example:

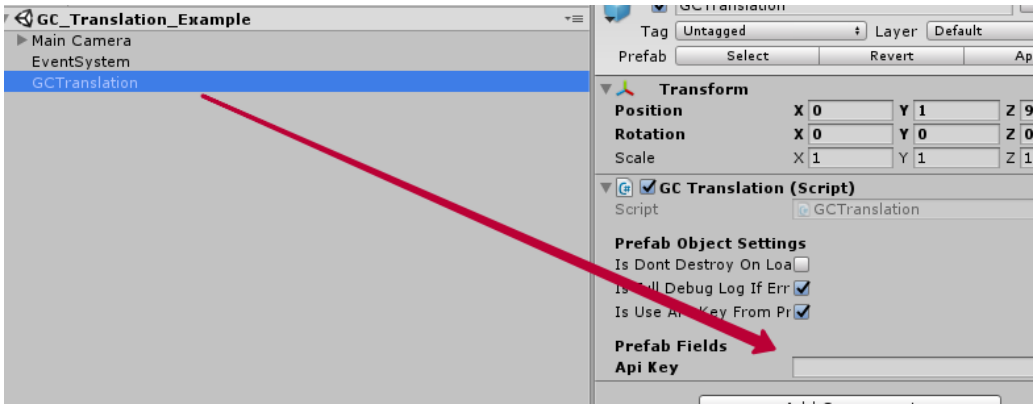
Create the script with and name it 'TutorialExample':

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

public class Example : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Drag n Drop the Prefab of GCTranslation into the scene:



Insert your own Google Cloud API Key into this field:



Create the variable for the GCTranslation and get an instance of an object:

```
public class GC_Translation_Example : MonoBehaviour
{
    private GCTranslation _gcTranslation;


    private void Start()
    {
        _gcTranslation = GCTranslation.Instance;
```

Then you should subscribe on the events:

```
        _gcTranslation.TranslateSuccessEvent += TranslateSuccessEventHandler;
        _gcTranslation.DetectLanguageSuccessEvent += DetectLanguageSuccessEventHandler;
        _gcTranslation.GetLanguagesSuccessEvent += GetLanguagesSuccessEventHandler;

        _gcTranslation.TranslateFailedEvent += TranslateFailedEventHandler;
        _gcTranslation.DetectLanguageFailedEvent += DetectLanguageFailedEventHandler;
        _gcTranslation.GetLanguagesFailedEvent += GetLanguagesFailedEventHandler;

        _gcTranslation.ContentOutOfLengthEvent += ContentOutOfLengthEventHandler;
```

```
    // handlers
    private void TranslateSuccessEventHandler(TranslationResponse value)
    {
    }

    private void DetectLanguageSuccessEventHandler(DetectLanguageResponse value)
    {
    }

    private void GetLanguagesSuccessEventHandler(LanguagesResponse value)
    {
    }

    private void TranslateFailedEventHandler(string value)
    {
    }

    private void DetectLanguageFailedEventHandler(string value)
    {
    }

    private void GetLanguagesFailedEventHandler(string value)
    {
    }

    private void ContentOutOfLengthEventHandler()
    {
    }
```

Create the Translate request:

```
_gcTranslation.Translate(new TranslationRequest()
{
    q = "my text",
    source = Enumerators.TextLanguage.EN.ToString(),
    target = Enumerators.TextLanguage.DE.ToString(),
    format = Enumerators.TextFormatType.text.ToString(),
    model = Enumerators.ModelType.@base.ToString()
});
```

Where **q** is the text content for the translation, **source** is the source language, **target** is the target language,

format is text format (can be *html* or *text*), **model** is the model of the translation (can be *nmt* (`Neural Machine Translation`) or *base* (`Phrase-Based Machine Translation`)).

When the Translate request will be successful, will be fire the `TranslateSuccessEvent`.

Handle the translation (for example in some input field):

```
// handlers
private void TranslateSuccessEventHandler(TranslationResponse value)
{
    foreach (var translation in value.data.translations)
        _textNetworkResultInputField.text += translation.translatedText + "\n---------\n";
}
```

```
private InputField _textNetworkResultInputField;
```

When the Translate request will be failed, will be fire the `TranslateFailedEvent`.

Handle the event:

```
private void TranslateFailedEventHandler(string value)
{
    _textNetworkResultInputField.text = value;
}
```

For Detect language of the text you have to create request method:

```
_gcTranslation.DetectLanguage(new DetectLanguageRequest()
{
    q = _textInputField.text
});
```

```
    private InputField _textInputField,
                       _textNetworkResultInputField;
```

```
_textNetworkResultInputField = transform.Find("Canvas/InputField_Output").GetComponent<InputField>();
_textInputField = transform.Find("Canvas/InputField_Input").GetComponent<InputField>();
```

Subscribe on events:

```
_gcTranslation.TranslateSuccessEvent += TranslateSuccessEventHandler;
_gcTranslation.DetectLanguageSuccessEvent += DetectLanguageSuccessEventHandler;


_gcTranslation.TranslateFailedEvent += TranslateFailedEventHandler;
_gcTranslation.DetectLanguageFailedEvent += DetectLanguageFailedEventHandler;
```

Our Handlers are:

```
private void DetectLanguageSuccessEventHandler(DetectLanguageResponse value)
{
    foreach (var detection in value.data.detections)
    {
        foreach (var item in detection)
            _textNetworkResultInputField.text += "language: " + item.language + "\n" +
                                                 "isReliable: " + item.isReliable + "\n" +
                                                 "confidence: " + item.confidence + "\n---------\n";
    }
}
```

```
private void DetectLanguageFailedEventHandler(string value)
{
    _textNetworkResultInputField.text = value;
}
```

You can insert parameters:

q – text content for detection

For Getting Languages for the translation you have to create request method:

```
private void GetLanguagesButtonOnclickHandler()
{
    _gcTranslation.GetLanguages(new LanguagesRequest()
    {
        target = _textInputField.text,
        model = ((Enumerators.ModelType) modelType.value).ToString()
    });
}
```

Where **target** is target language code, **model** is the model of the translation (can be *nmt* (`Neural Machine Translation`) or *base* (`Phrase-Based Machine Translation`)).

Subscribe on new events:

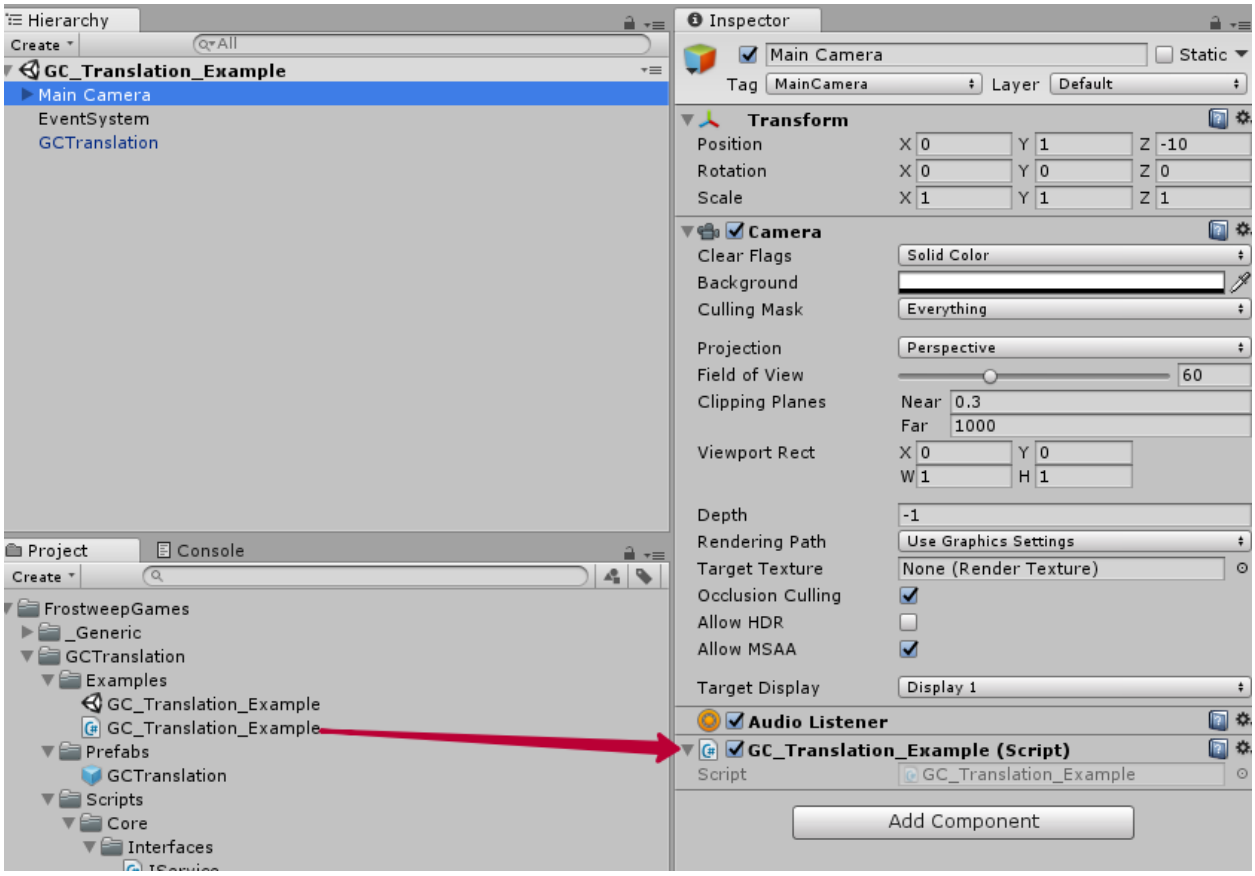
```
_gcTranslation.TranslateSuccessEvent += TranslateSuccessEventHandler;
_gcTranslation.DetectLanguageSuccessEvent += DetectLanguageSuccessEventHandler;
_gcTranslation.GetLanguagesSuccessEvent += GetLanguagesSuccessEventHandler;

_gcTranslation.TranslateFailedEvent += TranslateFailedEventHandler;
_gcTranslation.DetectLanguageFailedEvent += DetectLanguageFailedEventHandler;
_gcTranslation.GetLanguagesFailedEvent += GetLanguagesFailedEventHandler;
```

Make handlers:

```
private void GetLanguagesSuccessEventHandler(LanguagesResponse value)
{
    foreach(var language in value.data.languages)
    {
        _textNetworkResultInputField.text += "name: " + language.name + "\n" +
                                             "language: " + language.language + "\n---------\n";
    }
}
```

```
private void GetLanguagesFailedEventHandler(string value)
{
    _textNetworkResultInputField.text = value;
}
```

Then drag n drop your script on the object in the scene:



Then enjoy your project!

- Note:
  1) The plugin does not cover the cost of the Google Cloud Service
  2) Be sure to read the terms of service of Google Cloud Translation API

- Versions changes:

  1.1

- Updated Google Cloud Translation API
- Improved Core
- Unified networking
- Fixed bugs
- Improved example