Google Cloud Natural Language

- Intro:

Google Cloud Natural Language API reveals the structure and meaning of text by offering powerful machine learning models in an easy to use REST API. You can use it to **extract information** about people, places, events and much more, mentioned in text documents, news articles or blog posts. You can use it to **understand sentiment** about your product on social media or **parse intent** from customer conversations happening in a call center or a messaging app. You can **analyze text uploaded in your request** or integrate with your document storage on Google Cloud Storage.

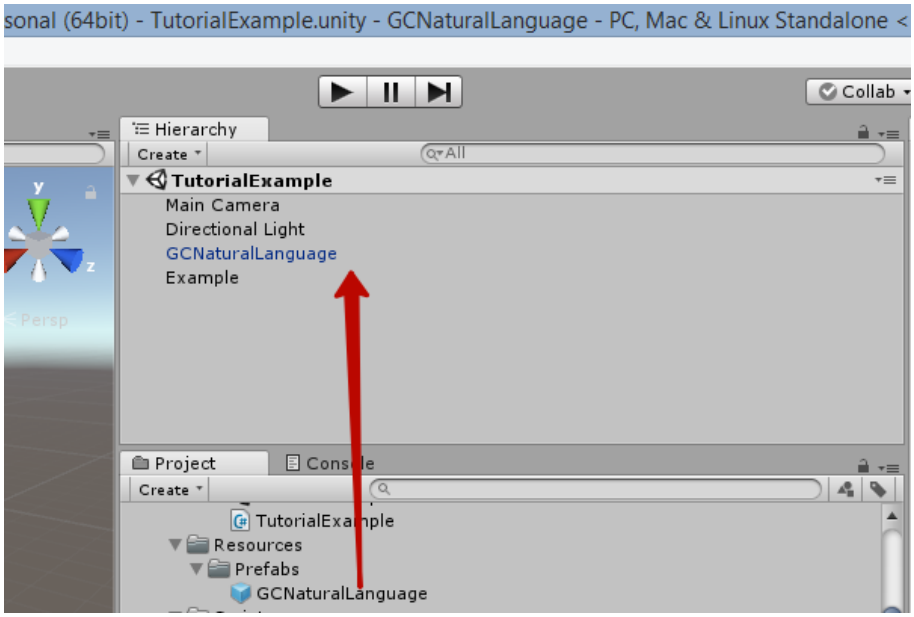
- How to use:

Create you first an app example:

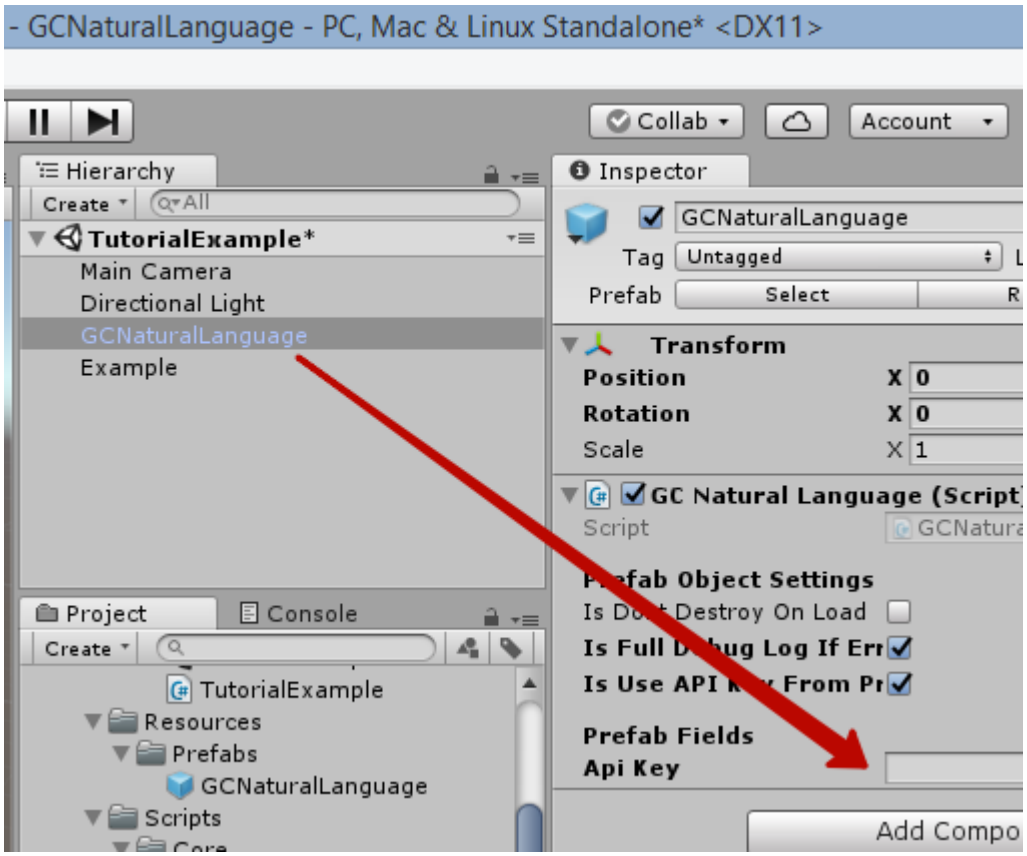
Create the script with and name it 'TutorialExample':

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

0 references
public class TutorialExample : MonoBehaviour {

    // Use this for initialization
    0 references
    void Start () {

    }

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

    }
}
```

Drag n Drop the Prefab of GCNaturalLanguage into the scene:



Insert your own Google Cloud API Key into this field:



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

```
using UnityEngine;
using FrostweepGames.Plugins.GoogleCloud.NaturalLanguage;
using System;

0 references
public class TutorialExample : MonoBehaviour
{
    private GCNaturalLanguage _gcNaturalLanguage;

    0 references
    private void Start()
    {
        _gcNaturalLanguage = GCNaturalLanguage.Instance;
```

Then you should subscribe on the events:

```
    private void Start()
    {
        _gcNaturalLanguage = GCNaturalLanguage.Instance;

        _gcNaturalLanguage.AnnotateTextSuccessEvent += _gcNaturalLanguage_AnnotateTextSuccessEvent;
        _gcNaturalLanguage.AnalyzeEntitySentimentSuccessEvent += _gcNaturalLanguage_AnalyzeEntitySentimentSuccessEvent;
        _gcNaturalLanguage.AnalyzeSentimentSuccessEvent += _gcNaturalLanguage_AnalyzeSentimentSuccessEvent;
        _gcNaturalLanguage.AnalyzeSyntaxSuccessEvent += _gcNaturalLanguage_AnalyzeSyntaxSuccessEvent;
        _gcNaturalLanguage.AnalyzeEntitiesSuccessEvent += _gcNaturalLanguage_AnalyzeEntitiesSuccessEvent;
        _gcNaturalLanguage.ClassifyTextSuccessEvent += _gcNaturalLanguage_ClassifyTextSuccessEvent;


        _gcNaturalLanguage.AnnotateTextFailedEvent += _gcNaturalLanguage_AnnotateTextFailedEvent;
        _gcNaturalLanguage.AnalyzeEntitySentimentFailedEvent += _gcNaturalLanguage_AnalyzeEntitySentimentFailedEvent;
        _gcNaturalLanguage.AnalyzeSentimentFailedEvent += _gcNaturalLanguage_AnalyzeSentimentFailedEvent;
        _gcNaturalLanguage.AnalyzeSyntaxFailedEvent += _gcNaturalLanguage_AnalyzeSyntaxFailedEvent;
        _gcNaturalLanguage.AnalyzeEntitiesFailedEvent += _gcNaturalLanguage_AnalyzeEntitiesFailedEvent;
        _gcNaturalLanguage.ClassifyTextFailedEvent += _gcNaturalLanguage_ClassifyTextFailedEvent;
```

Create the Annotate Text request:

```
public void AnnotateText(string text, Enumerators.Language lang)
{
    _gcNaturalLanguage.Annotate(new AnnotateTextRequest()
    {
        encodingType = Enumerators.EncodingType.UTF8,
        features = new Features
        {
            extractDocumentSentiment = true,
            extractEntities = true,
            extractEntitySentiment = true,
            extractSyntax = true
        },
        document = new LocalDocument()
        {
            content = text,
            language = _gcNaturalLanguage.PrepareLanguage(lang),
            type = Enumerators.DocumentType.PLAIN_TEXT
        }
    });
}
```

Where encoding type if UTF8, with all features, and LocalDocument:

content is the text data

language is the text data language

type is the text data type

Then you can call the method with text data and text data language:

```
string content = "Religious texts (also known as scripture, or scriptures, from the Latin scriptura, meaning a writing)";

AnnotateText(content, Enumerators.Language.en);
```

When the Annotation Text request will be successful, will be fire the *AnnotateTextSuccessEvent.*

Handle the Text Detection:

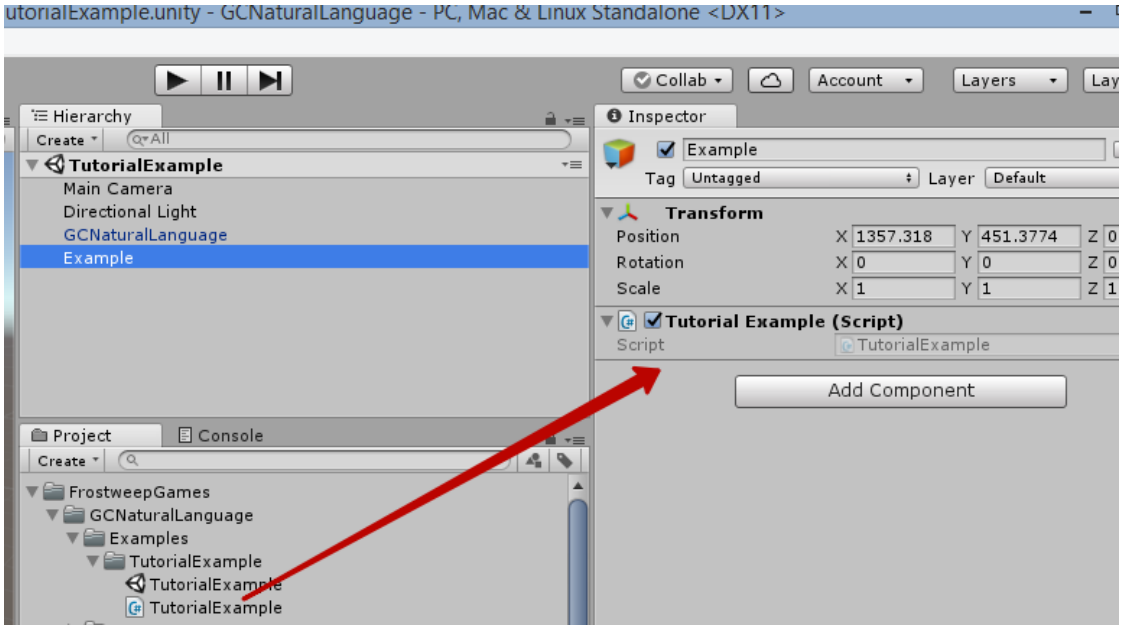
```
private void _gcNaturalLanguage_AnnotateTextSuccessEvent(AnnotateTextResponse obj)
{
    string result = string.Empty;

    foreach (var sentence in obj.sentences)
        result += sentence.text.content + Environment.NewLine;

    Debug.Log("sentences :" + result);
}
```

When the Annotation request will be failed, will be fire the *AnnotateTextFailedEvent.*

Handle the event:

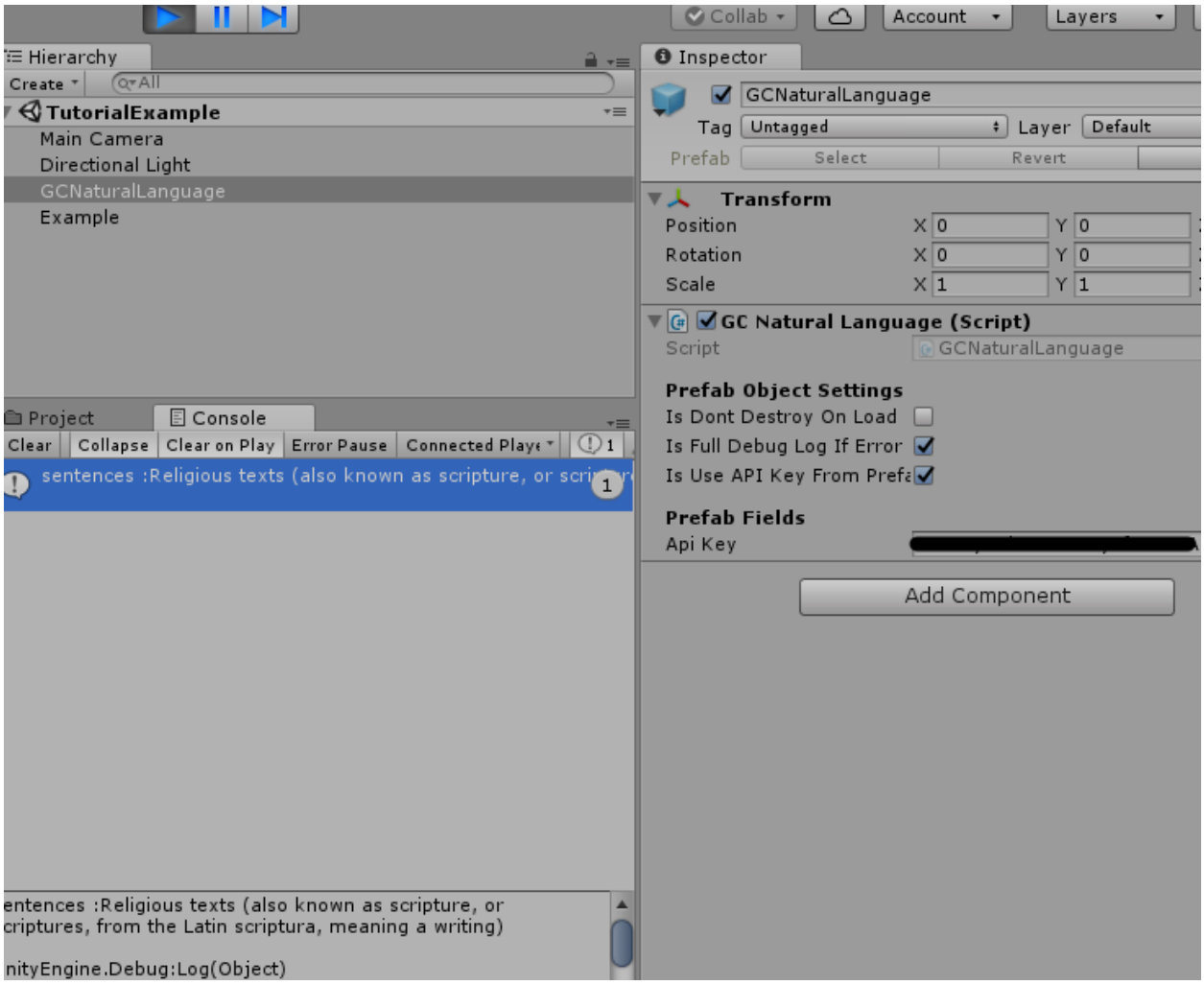
```
private void _gcNaturalLanguage_AnnotateTextFailedEvent(string obj)
{
    Debug.Log(obj);
}
```

Create an object in the scene and attach the TutorialExample script into this object:



Then click on the Play button.

Waiting for the result and get:



That’s all! So you can make your own text analyze using our plugin!

- 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 Natural Language API
- Versions changes:

  1.0 – Implemented Google Cloud Natural Language API