Google Cloud Video Intelligence

- Intro:

  Google Cloud Video Intelligence API has pre-trained machine learning models that automatically recognize a vast number of objects, places, and actions in stored and streaming video. It's highly efficient for common use cases and improves over time as new concepts are introduced.

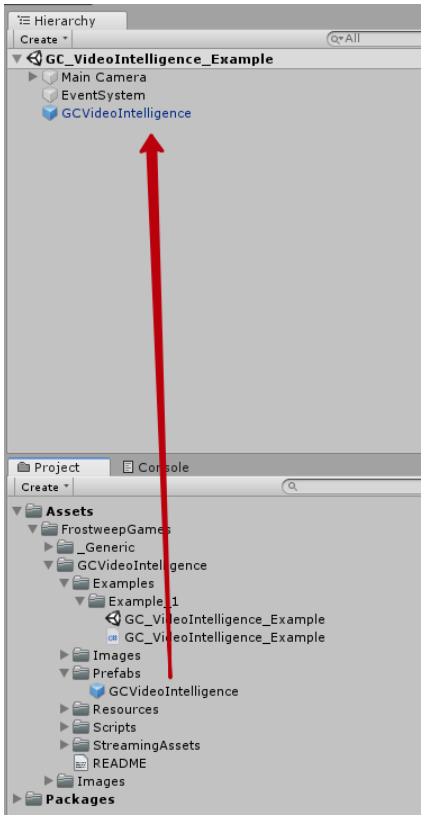
- How to use:

  Create you first an app example:

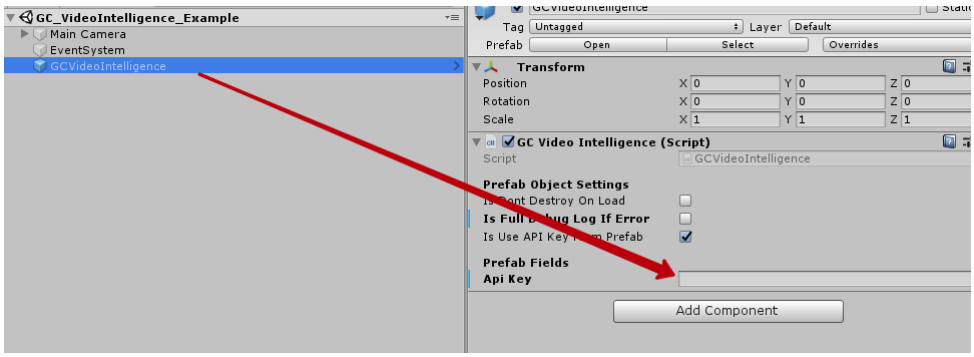
  Create the script with and name it 'Example':

```
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 *GCVideoIntelligence* into the scene:



  Insert your own Google Cloud API Key into this field:



Create a variable for the *GCVideoIntelligence* and get an instance of an object:

```
ссылок: 0
public class GC_VideoIntelligence_Example : MonoBehaviour
{
    private GCVideoIntelligence _gcVideoIntelligence;

    ссылок: 0
    private void Start()
    {
        _gcVideoIntelligence = GCVideoIntelligence.Instance;
    }
```

Then you have to subscribe on the events:

```
    _gcVideoIntelligence.AnnotateSuccessEvent += AnnotateSuccessEventHandler;
    _gcVideoIntelligence.AnnotateFailedEvent += AnnotateFailedEventHandler;


    _gcVideoIntelligence.GetSuccessEvent += GetSuccessEventHandler;
    _gcVideoIntelligence.GetFailedEvent += GetFailedEventHandler;

    _gcVideoIntelligence.ListSuccessEvent += ListSuccessEventHandler;
    _gcVideoIntelligence.ListFailedEvent += ListFailedEventHandler;

    _gcVideoIntelligence.CancelSuccessEvent += CancelSuccessEventHandler;
    _gcVideoIntelligence.CancelFailedEvent += CancelFailedEventHandler;

    _gcVideoIntelligence.DeleteSuccessEvent += DeleteSuccessEventHandler;
    _gcVideoIntelligence.DeleteFailedEvent += DeleteFailedEventHandler;
```

```
private void AnnotateSuccessEventHandler(AnnotateResponse response, long arg2)
{
}

ссылка: 1
private void GetSuccessEventHandler(Operation response, long arg2)
{
}

ссылка: 1
private void ListSuccessEventHandler(ListOperationResponse response, long arg2)
{
}

ссылка: 1
private void CancelSuccessEventHandler(string response)
{
}

ссылка: 1
private void DeleteSuccessEventHandler(string response)
{
}

ссылка: 1
private void GetFailedEventHandler(string arg1, long arg2)
{
}

ссылка: 1
private void ListFailedEventHandler(string arg1, long arg2)
{
}

ссылка: 1
private void CancelFailedEventHandler(string arg1, long arg2)
{
}

ссылка: 1
private void DeleteFailedEventHandler(string arg1, long arg2)
{
}

ссылка: 1
private void AnnotateFailedEventHandler(string arg1, long arg2)
{
}
```

Create Annotate request:

Declare button and input field

```
public Button annotateButton;
```

```
public InputField urlInputField;
```

Connect listener

```
annotateButton.onClick.AddListener(AnnotateButtonOnClickHandler);
```

Create *annotate* request

```
private void AnnotateButtonOnClickHandler()
{
    AnnotateVideo(VideoConvert.Convert(urlInputField.text));
}
```

VideoConvert.Convert converts byte array or path to local stored file into base64 string.

Let's handle *Annotate* response:

Declare input field

```
public InputField nameInputField;
```

Fill *annotate* request handler

```
ссылка: 1
private void AnnotateSuccessEventHandler(AnnotateResponse response, long arg2)
{
    nameInputField.text = response.name;
}
```

Received parameter **name** we will use in ***Get***, ***Cancel***, ***Delete*** requests

Let's create ***List*** request:

Declare input fields

```
public InputField filterInputField;
public InputField pageSizeInputField;
public InputField pageTokenInputField;
```

Declare button and connect listener

```
public Button listButton,
```

```
listButton.onClick.AddListener(ListButtonOnClickHandler);
```

Create *list* request

```
private void ListButtonOnClickHandler()
{
    _gcVideoIntelligence.List(
        nameInputField.text,
        filterInputField.text,
        double.Parse(pageSizeInputField.text),
        pageTokenInputField.text);
}
```

*note: sometimes double.Parse don't parse text field with dots. There you need implement CulltureInfo provider.

List request require **name** of *region*. **filter**, **page size** and **next page token**.

By default, filter and page token could be empty.

Fill *list* response handler

```
private void ListSuccessEventHandler(ListOperationResponse response, long arg2)
{
    pageTokenInputField.text = response.nextPageToken;

    if (response.operations != null)
    {
        foreach (var operation in response.operations)
        {
            Debug.Log(Newtonsoft.Json.JsonConvert.SerializeObject(operation));
        }
    }
}
```

Here we handled operations. We will see list of *annotation* operations.

Let's create ***Get*** request:

Declare button and connect listener

```
    public Button getButton,
```

```
getButton.onClick.AddListener(GetButtonOnClickHandler);
```

Create **Get** request

```
    private void GetButtonOnClickHandler()
    {
        _gcVideoIntelligence.Get(nameInputField.text);
    }
```

Get request require **name** of operation

Fill *Get* response handler

Here we will have two types of handling: metadata – during annotation process and actual response with annotation results.

```
private void GetSuccessEventHandler(Operation response, long arg2)
{
    if (response.response != null)
    {
        if (response.response.annotationResults != null)
        {
            foreach (var result in response.response.annotationResults)
            {
                foreach (var frame in result.explicitAnnotation.frames)
                {
                    Debug.Log(frame.pornographyLikelihood + " | " + frame.timeOffset);
                }
            }
        }
    }
    else if (response.metadata != null)
    {
        foreach (var progress in response.metadata.annotationProgress)
        {
            Debug.Log("progress: " + progress.progressPercent + " ; start Time" + progress.startTime + " ; update Time" + progress.updateTime);
        }
    }
}
```

Same operations you can do for Delete and Cancel requests.

```
ссылка: 1
private void CancelButtonOnClickHandler()
{
    _gcVideoIntelligence.Cancel(nameInputField.text);
}
```

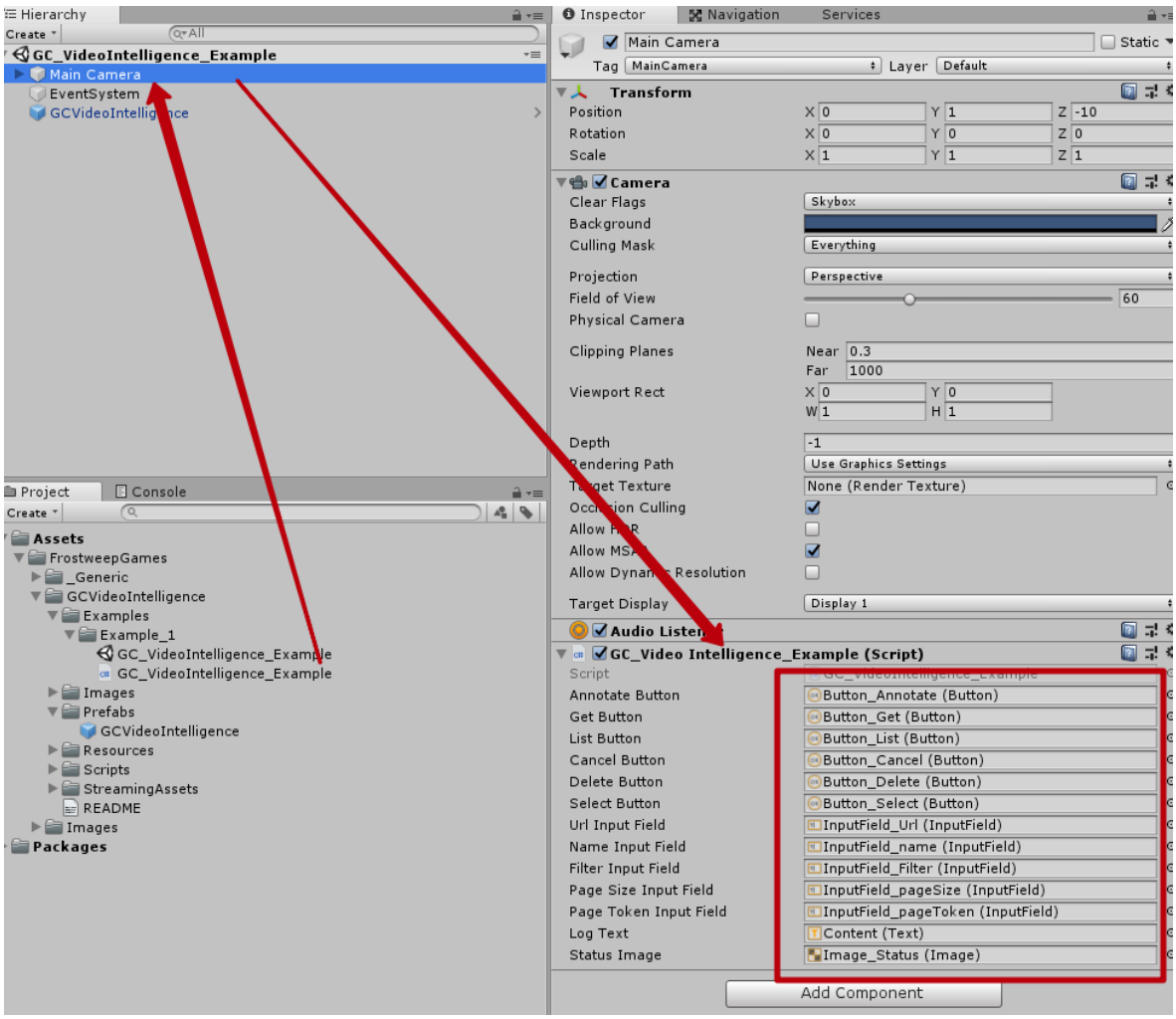
```
private void DeleteButtonOnClickHandler()
{
    _gcVideoIntelligence.Delete(nameInputField.text);
}
```

```
private void CancelSuccessEventHandler(string response)
{
    Debug.Log("CancelSuccessEventHandler: " + response);
}

ссылка: 1
private void DeleteSuccessEventHandler(string response)
{
    Debug.Log("DeleteSuccessEventHandler: " + response);
}
```

After all buttons and handlers are filled you could start working on next step.

Create scene.

Create an object in the scene and attach *Example* script on this object:



Then connect all buttons, input fields, text fields and images from the scene into script via dragNdrop.

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 Video Intelligence API

- Versions changes:

  1.0 – Implemented Google Cloud Video Intelligence API