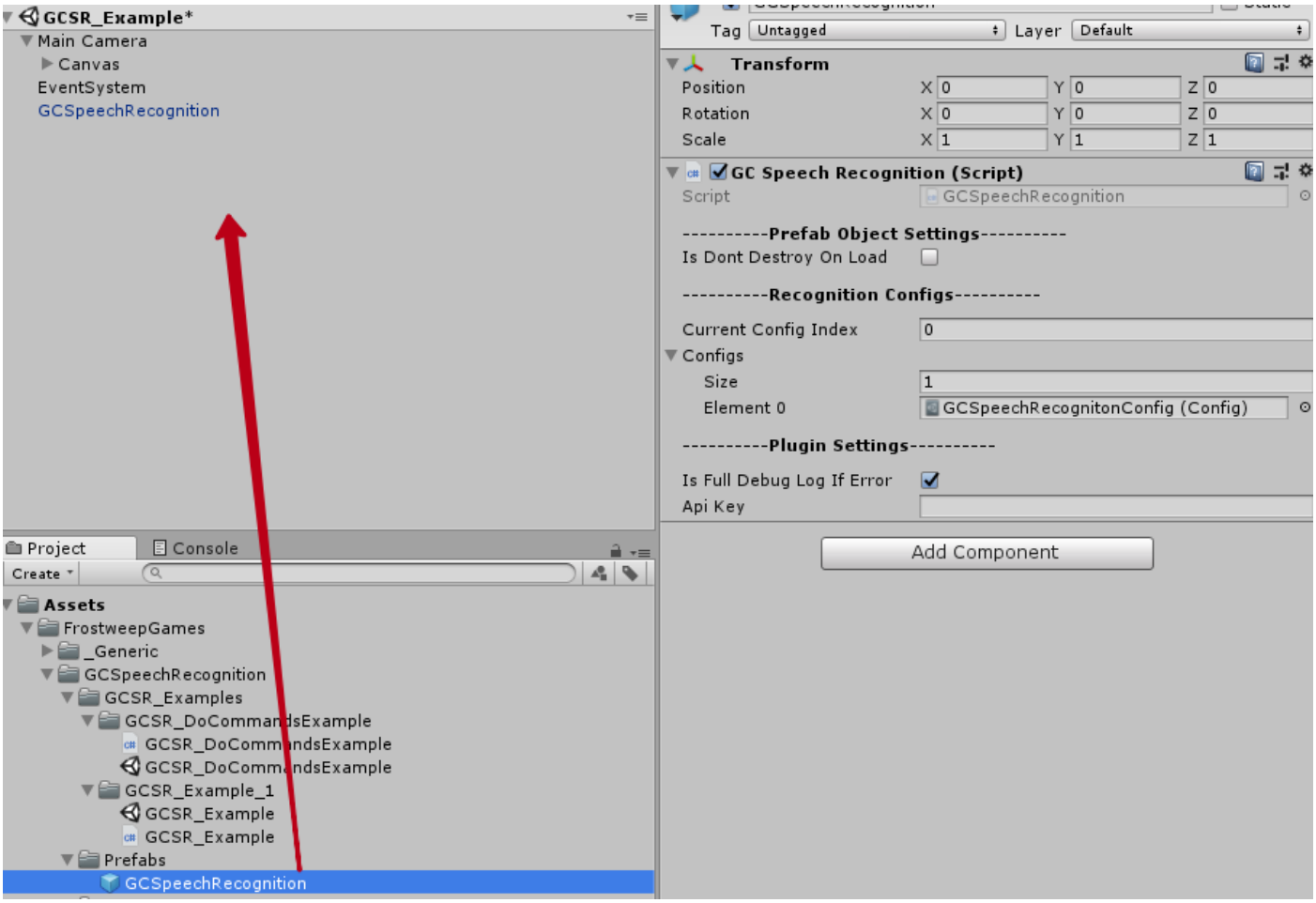
Google Cloud Speech Recognition

How to use

Lets create a new scene.

Drag and drop GCSpechRecognition prefab from Prefabs folder of our asset.



Now you can use API of an asset.

Lets create an Example script and name as `GCSR_Example`

Open it in IDE(Visual Studio for example).

Now we can write code there. But lets back again to the scene and prepare it.

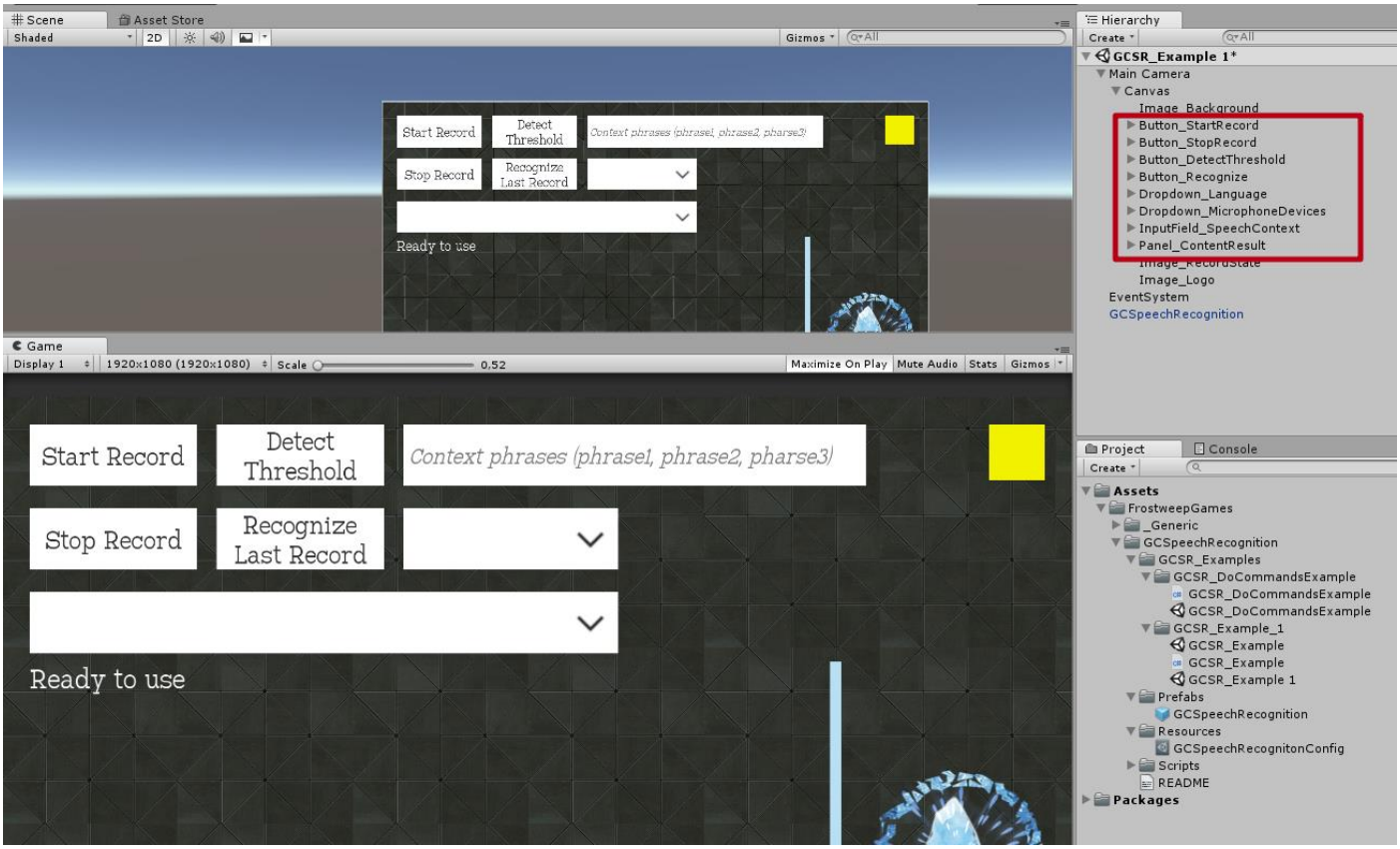
Create a Canvas and add few **buttons**: StartRecord, StopRecord, DetectThreshold, Recognize.

Add **InputField** for the Context phrases. Also add two **Dropdowns** for selecting language and microphone device.

Then add **Text** to show the results.

Name created objects like as you wish.

Save scene and lets work with **code**.

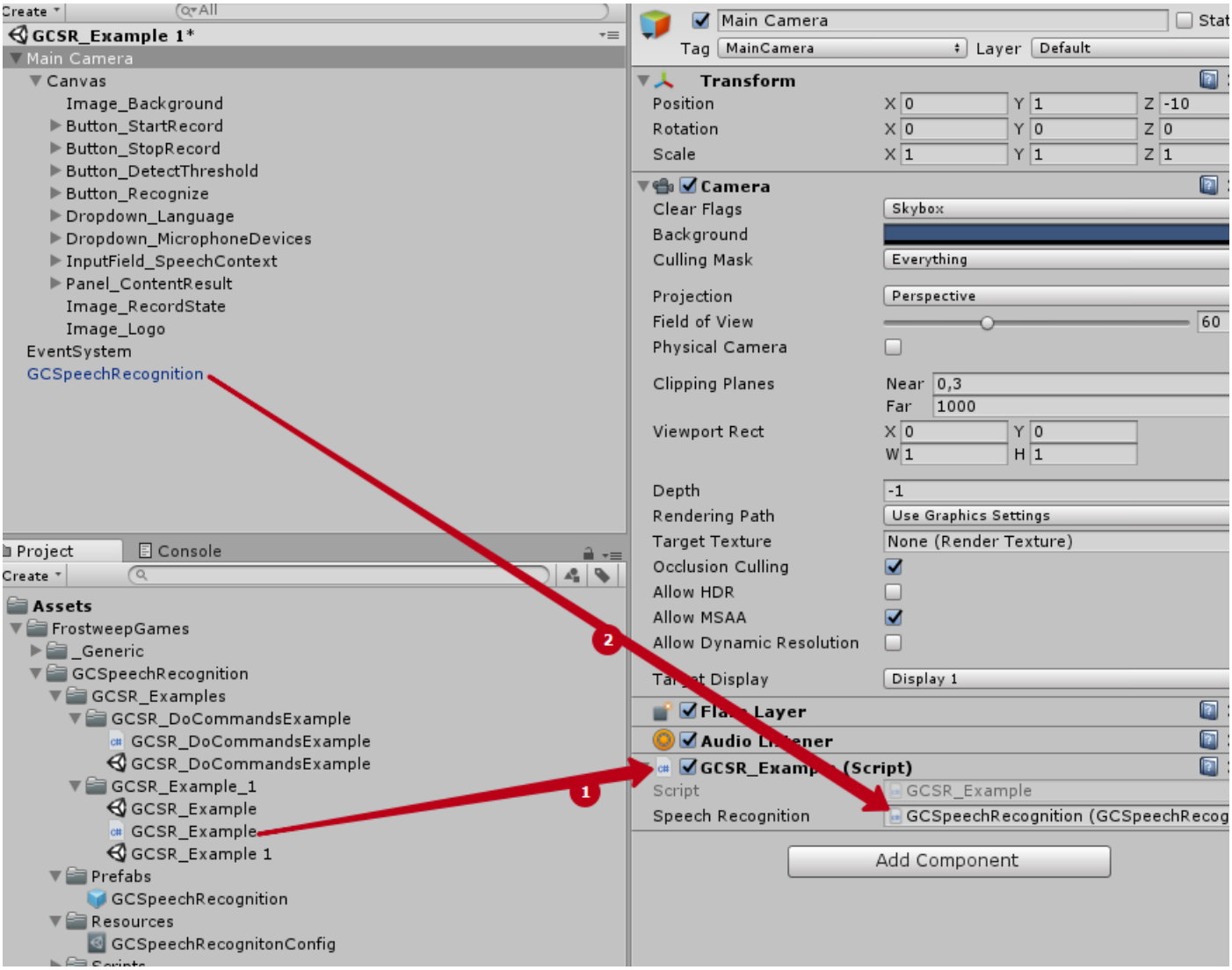


Now open the IDE and lets write initial setup of the script.

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace FrostweepGames.Plugins.GoogleCloud.SpeechRecognition.Examples
{
    ссылок: 0
    public class GCSR_Example : MonoBehaviour
    {
        [SerializeField]
        private GCSpeechRecognition _speechRecognition;

        ссылок: 0
        private void Start()
        {

        }

        ссылок: 0
        private void OnDestroy()
        {

        }
    }
}
```

We have two functions **Start** and **OnDestroy**, - there we will write our subscriptions and initializations of the fields and cleanup of them.

Lets open the Unity and attach our created script to the MainCamera object, then drag and drop **GCSpeechRecognition** object from the scene into the empty field of our attached script.

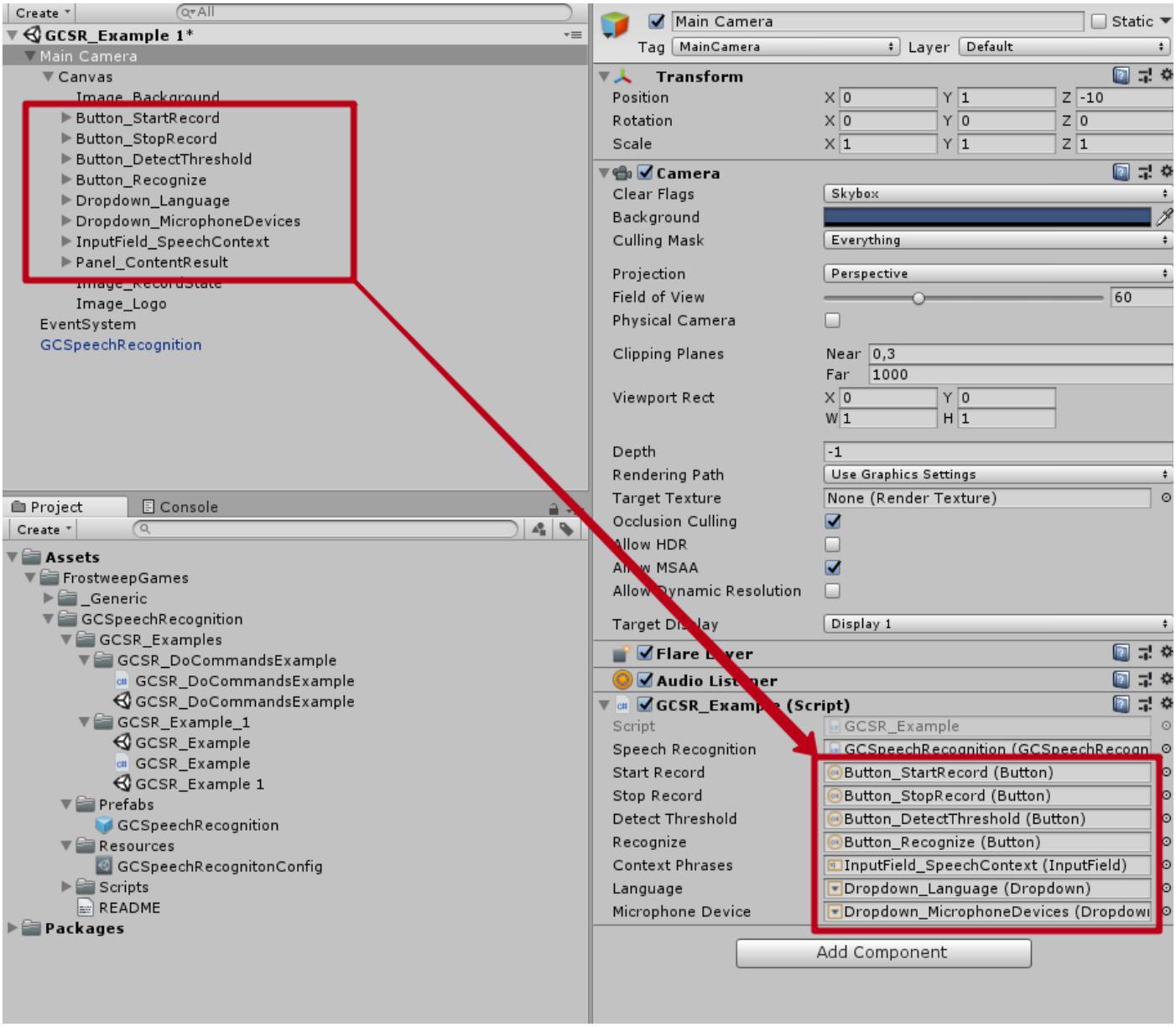


Now our variable in script functional and we can use API.

But first of all we need to create list of variables for the created buttons, inputField, dropdowns, etc..

```
using System;
using UnityEngine;
using UnityEngine.UI;

namespace FrostweepGames.Plugins.GoogleCloud.SpeechRecognition.Examples
{
    ссылок: 0
    public class GCSR_Example : MonoBehaviour
    {
        [SerializeField]
        private GCSpeechRecognition _speechRecognition;

        [SerializeField]
        private Button _startRecord,
                       _stopRecord,
                       _detectThreshold,
                       _recognize;

        [SerializeField]
        private InputField _contextPhrases;

        [SerializeField]
        private Dropdown _language,
                         _microphoneDevice;


        ссылок: 0
        private void Start()
        {

        }

        ссылок: 0
        private void OnDestroy()
        {

        }
    }
}
```

Then attach objects to script via drag and drop in scene.



Ok, now we can create handlers to our objects:

```
ссылок: 0
private void Start()
{
    _startRecord.onClick.AddListener(StartRecordButtonOnClickHandler);
    _stopRecord.onClick.AddListener(StopRecordButtonOnClickHandler);
    _detectThreshold.onClick.AddListener(DetectThresholdButtonOnClickHandler);
    _recognize.onClick.AddListener(RecognizeButtonOnClickHandler);
}

ссылок: 0
private void OnDestroy()...

ссылка: 1
private void StartRecordButtonOnClickHandler()
{

}

ссылка: 1
private void StopRecordButtonOnClickHandler()
{

}

ссылка: 1
private void DetectThresholdButtonOnClickHandler()
{

}

ссылка: 1
private void RecognizeButtonOnClickHandler()
{

}
```

After lets write initialization of dropdowns:

```
private void Start()
{
    _startRecord.onClick.AddListener(StartRecordButtonOnClickHandler);
    _stopRecord.onClick.AddListener(StopRecordButtonOnClickHandler);
    _detectThreshold.onClick.AddListener(DetectThresholdButtonOnClickHandler);
    _recognize.onClick.AddListener(RecognizeButtonOnClickHandler);

    _language.ClearOptions();

    for (int i = 0; i < Enum.GetNames(typeof(Enumerators.LanguageCode)).Length; i++)
    {
        _language.options.Add(new Dropdown.OptionData(((Enumerators.LanguageCode)i).Parse()));
    }

    _language.value = _language.options.IndexOf(_language.options.Find(x => x.text == Enumerators.LanguageCode.en_GB.Parse()));

    _microphoneDevice.ClearOptions();

    for (int i = 0; i < _speechRecognition.GetMicrophoneDevices().Length; i++)
    {
        _microphoneDevice.options.Add(new Dropdown.OptionData(_speechRecognition.GetMicrophoneDevices()[i]));
    }

    //smart fix of dropdowns
    _microphoneDevice.value = 1;
    _microphoneDevice.value = 0;
}
```

Ok, we have initialization of each our object. So let subscribe on events and then write handlers of events.

```
        //smart fix of dropdowns
        _microphoneDevice.value = 1;
        _microphoneDevice.value = 0;

        _speechRecognition.RecognizeSuccessEvent += RecognizeSuccessEventHandler;
        _speechRecognition.RecognizeFailedEvent += RecognizeFailedEventHandler;

        _speechRecognition.StartedRecordEvent += StartedRecordEventHandler;
        _speechRecognition.RecordFailedEvent += RecordFailedEventHandler;

        _speechRecognition.EndTalkigEvent += EndTalkigEventHandler;
    }

    ссылок: 0
    private void OnDestroy()
    {
        _speechRecognition.RecognizeSuccessEvent -= RecognizeSuccessEventHandler;
        _speechRecognition.RecognizeFailedEvent -= RecognizeFailedEventHandler;
        _speechRecognition.StartedRecordEvent -= StartedRecordEventHandler;
        _speechRecognition.RecordFailedEvent -= RecordFailedEventHandler;

        _speechRecognition.EndTalkigEvent -= EndTalkigEventHandler;
    }
```

And also, handler for microphone dropdown

```
        _recognize.onClick.AddListener(RecognizeButtonOnClickHandler);

        _microphoneDevice.onValueChanged.AddListener(MicrophoneDevicesDropdownOnValueChangedEventHandler);

        _language.ClearOptions();

        for (int i = 0; i < Enum.GetNames(typeof(Enumerators.LanguageCode)).Length; i++)...

        _language.value = _language.options.IndexOf(_language.options.Find(x => x.text == Enumerators.LanguageCo

        _microphoneDevice.ClearOptions();

        for (int i = 0; i < _speechRecognition.GetMicrophoneDevices().Length; i++)...

        //smart fix of dropdowns
        _microphoneDevice.value = 1;
        _microphoneDevice.value = 0;

        _speechRecognition.RecognizeSuccessEvent += RecognizeSuccessEventHandler;
        _speechRecognition.RecognizeFailedEvent += RecognizeFailedEventHandler;

        _speechRecognition.StartedRecordEvent += StartedRecordEventHandler;
        _speechRecognition.RecordFailedEvent += RecordFailedEventHandler;

        _speechRecognition.EndTalkigEvent += EndTalkigEventHandler;
    }

    ссылок: 0
    private void OnDestroy()...

    ссылка: 1
    private void MicrophoneDevicesDropdownOnValueChangedEventHandler(int value)
    {
    }
```

Ok, now we have all subscriptions and handlers and we can use API.

Lets use API of asset for start record, stop record, detect threshold and recognize.

```
ссылка: 1
private void StartRecordButtonOnClickHandler()
{
    _speechRecognition.StartRecord(false);
}

ссылка: 1
private void StopRecordButtonOnClickHandler()
{
    _speechRecognition.StopRecord();
}

ссылка: 1
private void DetectThresholdButtonOnClickHandler()
{
    _speechRecognition.DetectThreshold();
}

ссылка: 1
private void RecognizeButtonOnClickHandler()
{
    if (_speechRecognition.LastRecordedClip == null || _speechRecognition.LastRecordedRaw == null)
        return;

    RecognitionConfig config = RecognitionConfig.GetDefault();
    config.languageCode = ((Enumerators.LanguageCode)_languageDropdown.value).Parse();
    config.speechContexts = new SpeechContext[]
    {
        new SpeechContext()
        {
            phrases = _contextPhrasesInputField.text.Replace(" ", string.Empty).Split(',')
        }
    };
    config.audioChannelCount = _speechRecognition.LastRecordedClip.channels;

    GeneralRecognitionRequest recognitionRequest = new GeneralRecognitionRequest()
    {
        audio = new RecognitionAudioContent()
        {
            content = _speechRecognition.LastRecordedRaw.ToBase64()
        },
        config = config
    };

    _speechRecognition.Recognize(recognitionRequest);
}
```

In StartRecord handler we call **StartRecord** function with false parameter – it mean that we don't use Runtime Voice Detection Function.

Detect Threshold function needed for a **Runtime Voice Detection** feature. It detect noise at background and changes detection threshold. We don't need to use it in our example but its quite good tool to know about it.

And our main function for the recognition 😊 We need to create an instance of a RecognitionConfig class. You can use GetDefault static function to simple fill of the data in class variable. Then you can modify in it what you want. We need to use different language in each Recognize request so lets get its value from our language dropdown value.

Then we need to create an array of SpeechContext's, - it's a great feature of a Google service to make recognition more specified to a context of the speech. Then we can modify count of sound channels.

And our main request instance is GeneralRecognitionRequest in which we need to inser config and audio. For the audio we need to create an instance of a RecognitionAudioContent and in content field insert converted to base64 raw data of an audio clip.

Then simply call Recognize function with a recognitionRequest parameter.

For the microphone dropdown we have handler in which we should select Microphone Device.

```
ссылка: 1
private void MicrophoneDevicesDropdownOnValueChangedEventHandler(int value)
{
    if (!_speechRecognition.HasConnectedMicrophoneDevices())
        return;
    _speechRecognition.SetMicrophoneDevice(_speechRecognition.GetMicrophoneDevices()[value]);
}
```

Here we use three API functions:

HasConnectedMicrophoneDevies – it uses to know do we have microphones connected to a device.

SetMicrophoneDevice – which uses for setting microphone in plugin

GetMicrophoneDevices – which uses for receiving list of microphone devices.

Okay, now we have most usable API calls and we need to handle results. So, lets write handlers:

```
ссылок: 2
private void StartedRecordEventHandler()
{
    _result.text = "StartedRecordEventHandler";
}

ссылок: 2
private void RecordFailedEventHandler()
{
    _result.text = "RecordFailedEventHandler";
}

ссылок: 2
private void EndTalkigEventHandler(AudioClip clip)
{
    _result.text = "EndTalkigEventHandler";
}

ссылок: 2
private void RecognizeFailedEventHandler(string error)
{
    _result.text = "Recognize Failed: " + error;
}
```

Here we add simple logging of events,

Lets also fill handler of Recognition Success Event:

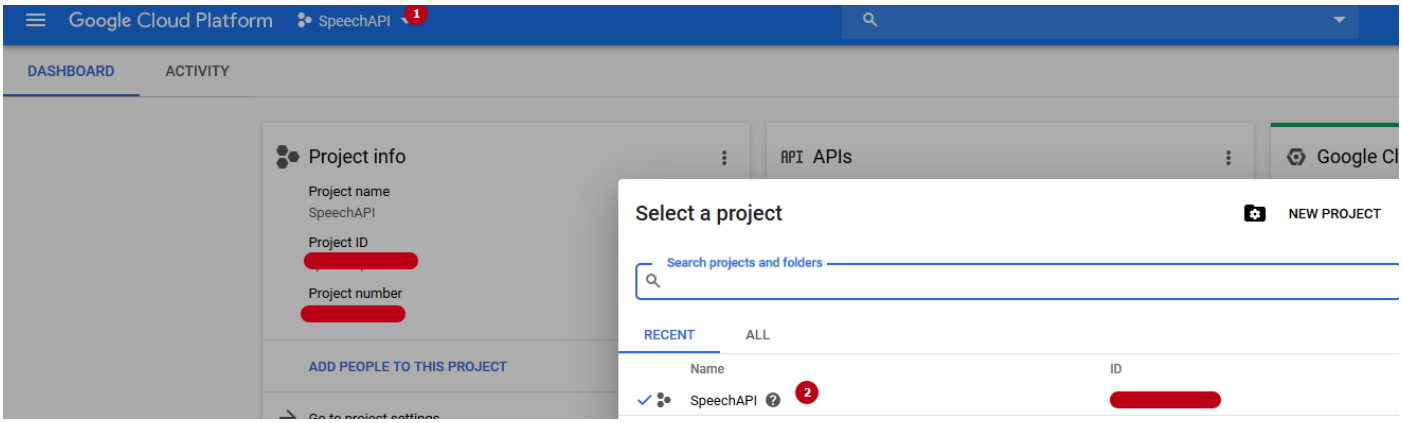
```
private void RecognizeSuccessEventHandler(RecognitionResponse recognitionResponse)
{
    _result.text = "Recognize Success.";

    if (recognitionResponse == null || recognitionResponse.results.Length == 0)
    {
        _result.text = "\nWords not detected.";
        return;
    }

    _result.text += "\n" + recognitionResponse.results[0].alternatives[0].transcript;

    var words = recognitionResponse.results[0].alternatives[0].words;

    if (words != null)
    {
        string times = string.Empty;

        foreach (var item in recognitionResponse.results[0].alternatives[0].words)
        {
            times += "<color=green>" + item.word + "</color> - start: " + item.startTime + "; end: " + item.endTime + "\n";
        }

        _result.text += "\n" + times;
    }

    string other = "\nDetected alternatives: ";

    foreach (var result in recognitionResponse.results)
    {
        foreach (var alternative in result.alternatives)
        {
            if (recognitionResponse.results[0].alternatives[0] != alternative)
            {
                other += alternative.transcript + ", ";
            }
        }
    }

    _result.text += other;
}
```

Here we handle: if recognition response not null we can proceed with it. **Response** has array of **results** which has array of **alternatives**. **Alternative** has **transcript** – it's a result of recognition, **words** – that have separated words and timing of them. In our function we check all results, all alternatives and insert in result text.

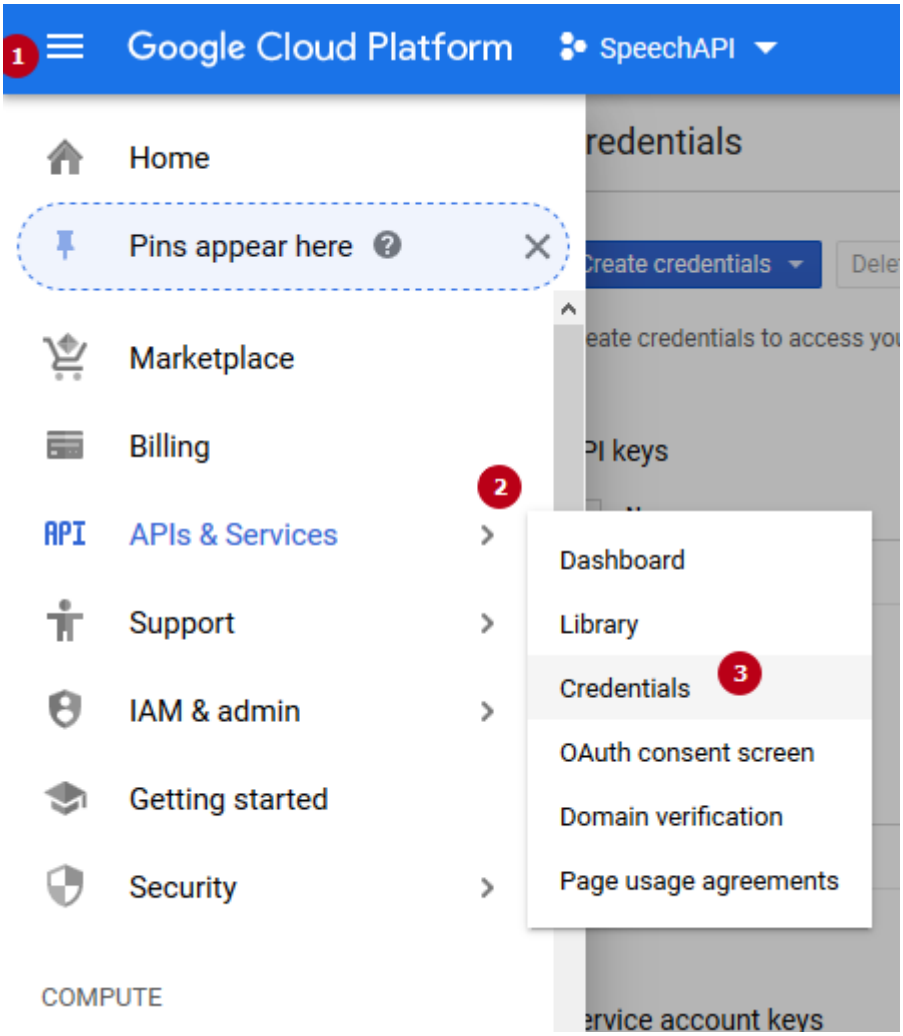
Now we have configured scene and example script to use them.

Lets move to the **Google Service** and setup dashboard:

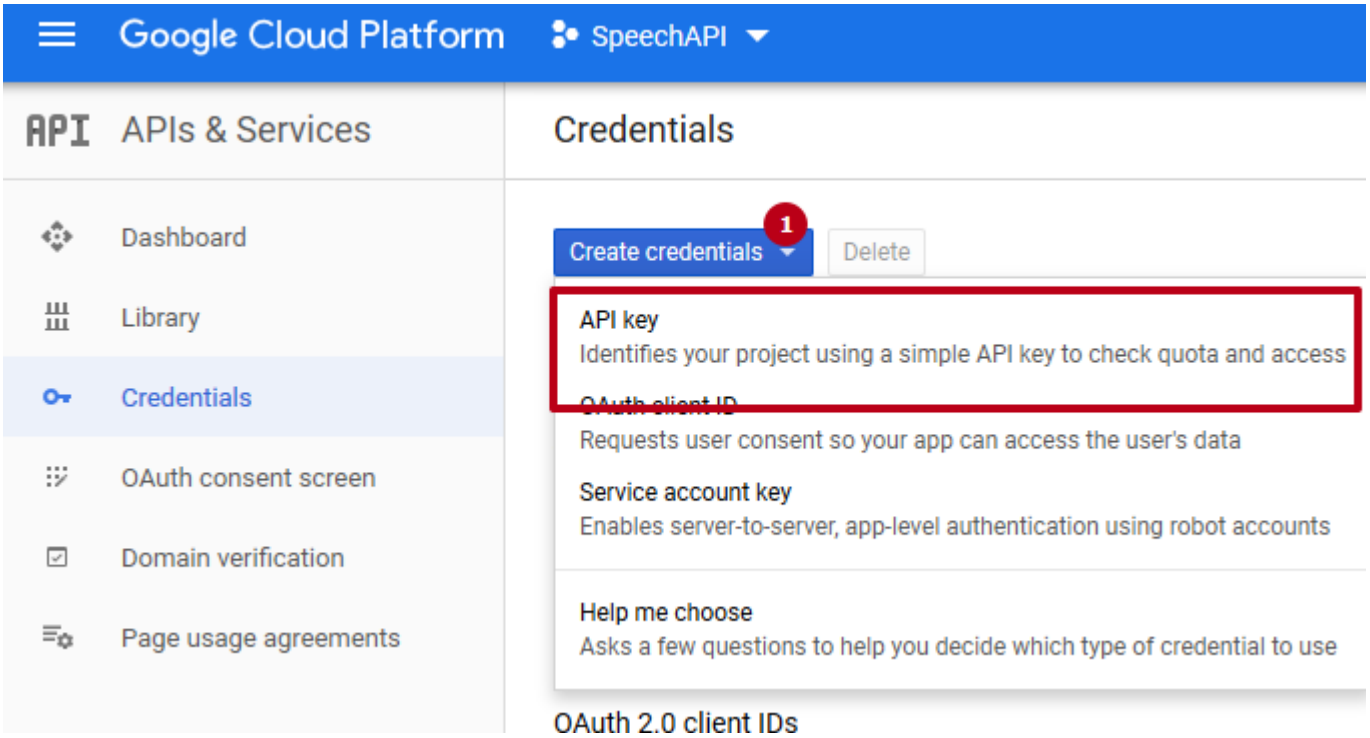
Open the https://console.cloud.google.com/ create a first project (or use already created) and select it.



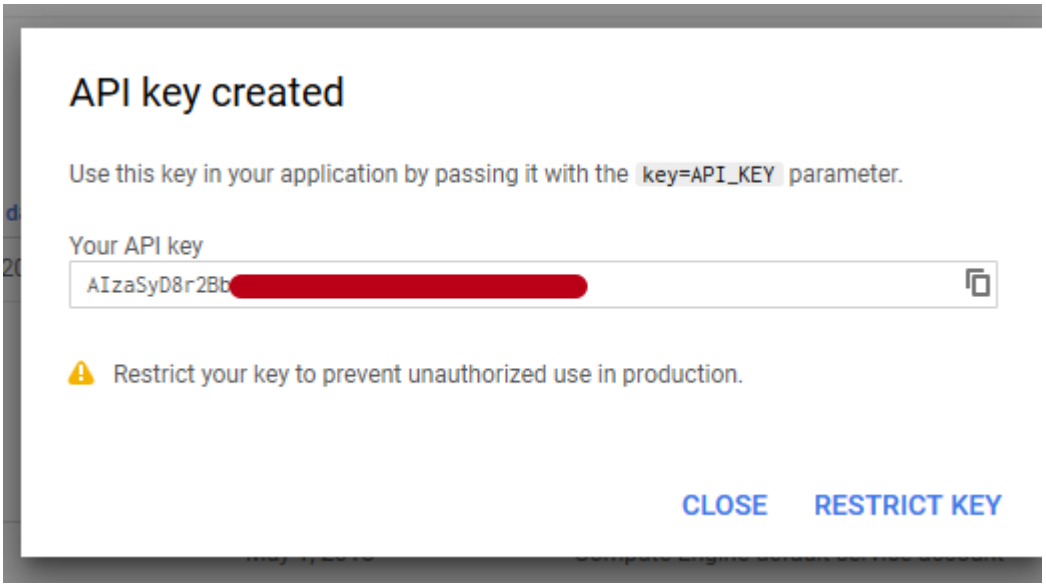
Then select **Menu**, then **Api & Services**, then **Credentials**

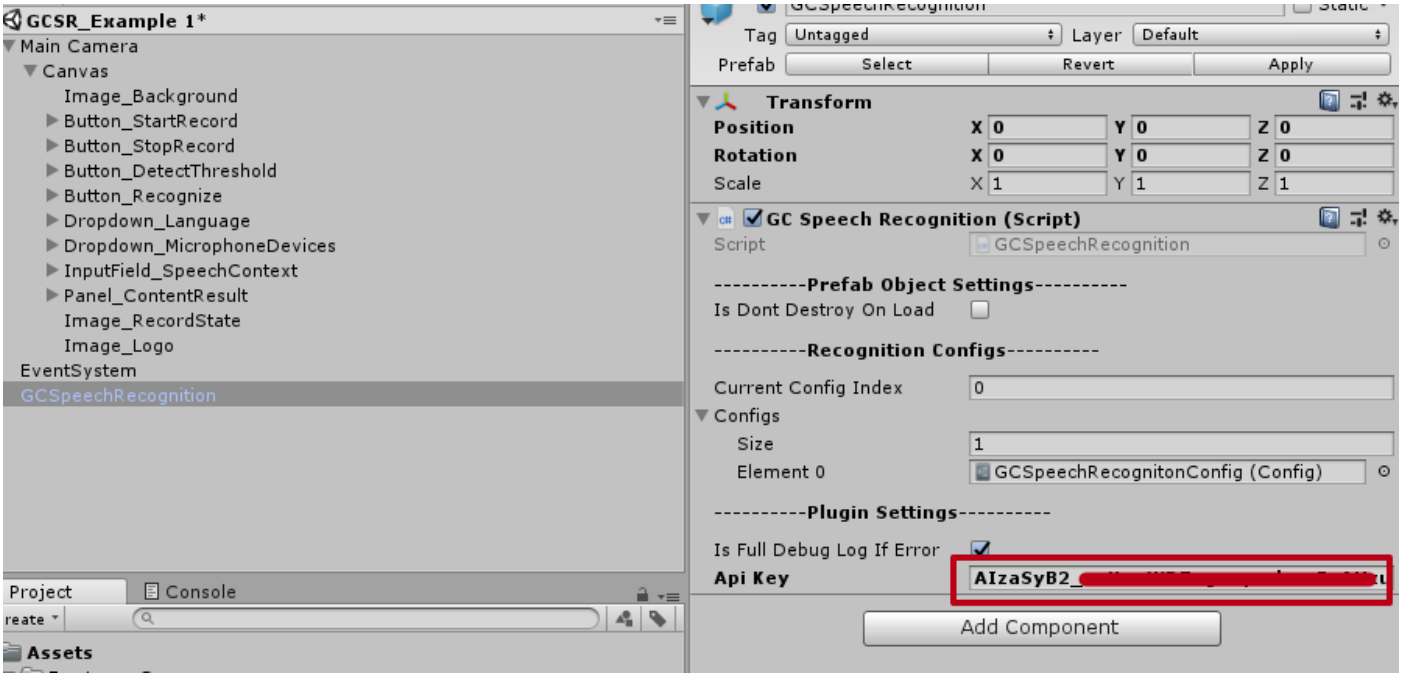


After that in opened screen **select Create credentials** and select **API Key**

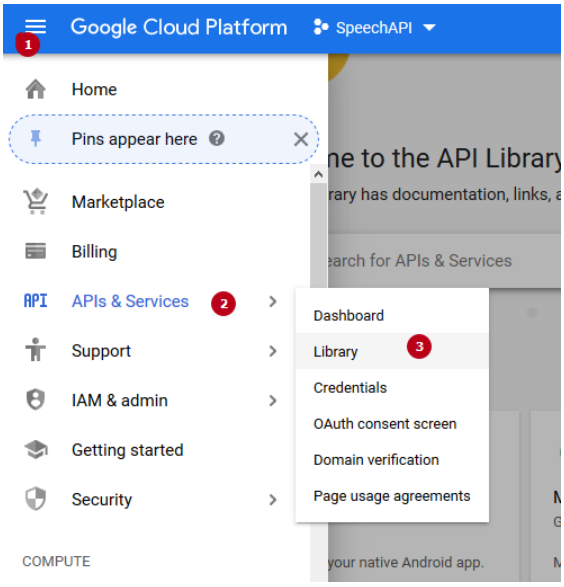


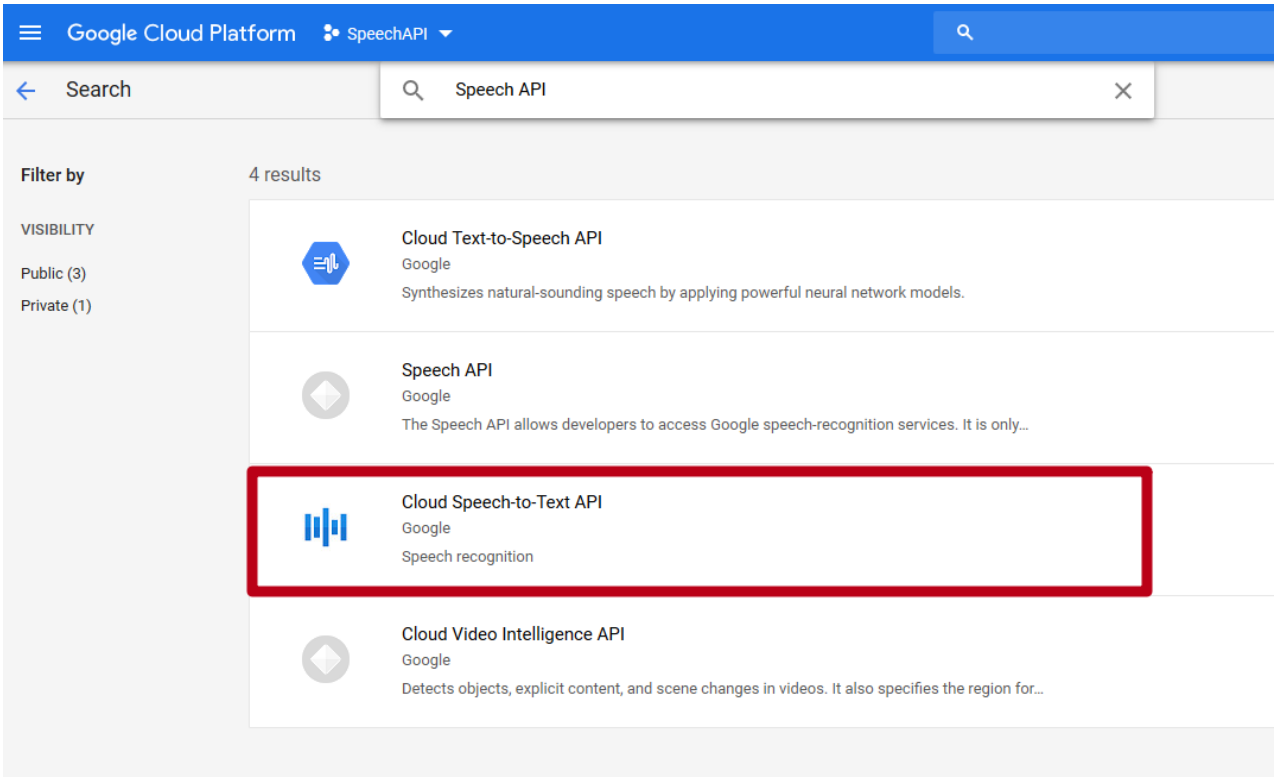
Now copy this key and in sert in **GCSpeechRecognition** prefab in scene:

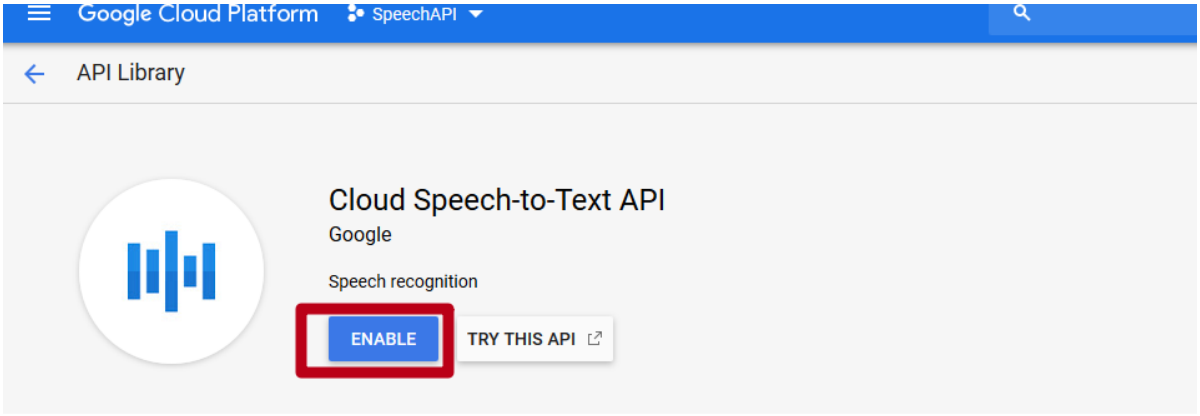




Then we need to enable Speech Recognitioon API. Open the APIs page and find **Speech API**







Now you ready to sue our asset fully! Open the Unity and click on Play button and Enjoy!

Special functions:

GetOperation – gets operation info about long running operation

GetListOperations – gets list of long running operations

CancelAllRequests – cancels all requests in progress

ReadyToRecord – detects if record is possible at this moment

SaveLastRecordedAudioClip – saves last record audio clip in .wav format at path

LongRunningRecognize – begins long running recognition at google service.

CancelRequest – cancels request with specific id that gets during creation of a request

HasMicrophonePermission – returns if app granted for microphone permission or not

RequestMicrophonePermission – requests android permission on devices such as Android

Note

- Working with il2cpp and mono
- Works with Unity Cloud Build

- Plugin partially support WebGL

Version Updates

• 4.0
- updated networking
- improved core
- fixed bugs
- implemented new features
- updated examples

• 3.2.x
- implemented word-specific information for recognized words
- added support of 'checking key signature of the android certificate'
- fixed bugs
- improved code

• 3.1
- Implemented Full Long Recognize Api
- Implemented All languages that the Google Speech Recognition supporting
- Fixed bugs
- Added Newtonsoft.Json Serializer\Deserializer
- Improvements in general

• 3.0
- Updated API to the latest 1.0 version
- New code architecture (SOA)
- Added async networking (Unity Web Requests)
- Removed support of WebGL (will be added in future updates)
- Not Fully implemented Long Recognize API
- Improved Examples
- Improved Config Prefab
- Improved runtime voice detection service
- Improved media service
- Improved audio converter
- Added audio tools

• 2.1
- implemented new features
- updated and improved example
- removed 3rd party libraries
• 2.0
- UPDATED Speech Recognition API to the latest Google Cloud Speech API
- implemented new features
- implemented speech detection threshold
- changed namespaces
- fixed bugs

• 1.1
- Changed Code Namespace with FrostweepGames.SpeechRecognition on FrostweepGames.SpeechRecognition.Google
- Implemented Runtime Speech Detection Utility