Recognissimo

Recognissimo is a cross-platform offline speech recognition plugin.

Check out the online documentation and demo.

For all questions write to bluezzzy.dev@gmail.com.

Features:

- No internet connection required
- Supports 21+ languages and dialects and more to come
- Fast and lightweight, written in C++
- Setup everything in Editor without code
- Load languages from the web or local storage
- Automatic storage and permissions management
- Use speech data from a microphone, audio clip or Unity scene
- Includes speech recognizer, voice activity detector and voice control with regular expressions support
- Extend using simple API

Supported platforms:

- Windows (x86, x64)
- macOS (x64, ARM64)
- Linux (x64)
- Android (ARMv7, ARM64, x86, x86_64)
- iOS (ARM64, x64)
- WebGL (HTTPS required)

Supported Unity editors:

- 2021.2 and above

Supported languages and dialects:

- Arabic
- Chinese
- English
- French

- German
- Italian
- Portuguese
- Russian
- Spanish
- Catalan, Czech, Dutch, Esperanto, Farsi, Filipino, Hindi, Indian English, Japanese, Kazakh, Swedish, Turkish, Ukrainian, Vietnamese

Recognissimo uses Vosk as a speech recognition backend, so you can use any Vosk-compatible models.

Known issues

Uncompressed AudioClip cannot be read in WebGL

Symptom:

> (Error in browser console) Cannot get data on compressed samples for audio clip "*clipName*". Changing the load type to DecompressOnLoad on the audio clip will fix this.

Fix: Update your version of Unity to one of the following versions

Cannot download remote file in WebGL

Symptom:

> Failed to download *xxx*. Reason: Unknown Error.

Fix: Setup CORS policy. More info here

Platform notes

Android

Recognissimo components require RECORD_AUDIO, READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE permissions:

- `MicrophoneSpeechSource` requires RECORD_AUDIO.
- `RemoteLanguageModelProvider` requires READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE to download and read language models.
- `StreamingAssetsLanguageModelProvider` requires READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE:
  - to extract language models from StreamingAssets folder in APK to persistent storage;
  - to extract language models from StreamingAssets folder in OBB to persistent storage if it fails to mount OBB.

The developer does not need to set permissions manually:

- RECORD_AUDIO permission is added to the application manifest by Unity;
- READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE permissions are added to the application manifest by Recognissimo at build time.

If you are using *Android App Bundle (AAB)* and `StreamingAssetsLanguageModelProvider` component, make sure asset pack containing StreamingAssets is loaded (more info here) before using Recognissimo components.

macOS and iOS

Before building, specify "Microphone Usage Description" (Unity may also require "Camera Usage Description" to be set even though Recognissimo does not use the camera) in project settings, otherwise the build will fail.

Migrating from previous version

Recognissimo 2 has breaking API changes.

> It is recommended to backup and remove Recognissimo from the project before upgrade.

Use the component-by-component changelog below to migrate existing code to the new version:

- SpeechRecognizer
  - `SpeechRecognizer.StartRecognizing()` and `SpeechRecognizer.StopRecognizing()` removed, use `SpeechRecognizer.StartProcessing()` and `SpeechRecognizer.StopProcessing()`.
  - `SpeechRecognizer.vocabulary` became property and renamed to `SpeechRecognizer.Vocabulary`.
  - Struct `Vocabulary` has been removed. The corresponding property is of type `List<string>`.
  - `SpeechRecognizer.enableDetailedResultDescription` became property and renamed to `SpeechRecognizer.EnableDetails`.
  - `SpeechRecognizer.allowEmptyPartialResults` removed.
  - `SpeechRecognizer.alternatives` became property and renamed to `SpeechRecognizer.Alternatives`.
  - `SpeechRecognizer.IsRecognizing` removed, use `SpeechRecognizer.State == SpeechProcessorState.Processing` .
  - Partial results within a utterance are now unique and no longer repeated.
  - UnityEvent fields `SpeechRecognizer.partialResultReady`, `SpeechRecognizer.resultReady` and `SpeechRecognizer.finished` became properties and renamed to `SpeechRecognizer.PartialResultReady`, `SpeechRecognizer.ResultReady` and `SpeechRecognizer.Finished`.

- VoiceControl
  - `VoiceControl.recognizer` removed. `VoiceControl` does not use `SpeechRecognizer` as a dependency. It inherits from `SpeechProcessor` class instead and use internal speech recognizer.
  - `VoiceControl.commands` became property and renamed to `VoiceControl.Commands`.
  - `VoiceControl.autoStart` became property and renamed to `VoiceControl.AutoStart`.
  - `VoiceControl.SetupAsync` removed.
  - `VoiceControl.StartControl` and `VoiceControl.StopControl` removed, use `VoiceControl.StartProcessing` and `VoiceControl.StopProcessing`.

- VoiceActivityDetector
  - `VoiceActivityDetector.recognizer` removed. `VoiceActivityDetector` does not use `SpeechRecognizer` as a dependency. It inherits from `SpeechProcessor` class instead and use internal speech recognizer.
  - `VoiceActivityDetector.autoStart` became property and renamed to `VoiceActivityDetector.AutoStart`.
  - UnityEvent fields `VoiceActivityDetector.spoke` and `VoiceActivityDetector.silenced` became properties and renamed to `VoiceActivityDetector.Spoke` and `VoiceActivityDetector.Silenced`.
  - `VoiceActivityDetector.StartDetection` and `VoiceActivityDetector.StopDetection` removed, use `VoiceActivityDetector.StartProcessing` and `VoiceActivityDetector.StopProcessing`.

- LanguageModelProvider
  - `LanguageModelProvider` renamed to `StreamingAssetsLanguageModelProvider`
  - `LanguageModelProvider.speechModels` renamed to `StreamingAssetsLanguageModelProvider.languageModels`.
  - `LanguageModelProvider.defaultLanguage` renamed to `StreamingAssetsLanguageModelProvider.language`.
  - `LanguageModelProvider.LoadLanguageModel` and its async version removed, use `StreamingAssetsLanguageModelProvider.Initialize` or let `SpeechProcessor` initialize it implicitly.
  - `StreamingAssetsLanguageModelProvider.Initialize` returns `IEnumerator`.
  - `LanguageModelProvider.InitializeAsync` removed, use `StreamingAssetsLanguageModelProvider.Initialize` or let `SpeechProcessor` initialize it implicitly.

- MicrophoneSpeechSource
  - `MicrophoneSpeechSource.microphoneSettings` removed, use `MicrophoneSpeechSource.DeviceName` and `MicrophoneSpeechSource.TimeSensitivity` .
  - `MicrophoneSpeechSource.recordOnAwake` removed, recording starts when `MicrophoneSpeechSource.Initialize` or `SpeechProcessor.StartProcessing` called.
  - `MicrophoneSpeechSource.StartMicrophone` and `MicrophoneSpeechSource.StopMicrophone` removed, use `MicrophoneSpeechSource.IsPaused` after recording is started.
  - `MicrophoneSpeechSource.StartProduce` and `MicrophoneSpeechSource.StopProduce` renamed to `MicrophoneSpeechSource.StartProducing` and

MicrophoneSpeechSource.StopProducing

Comparison of typical use case implementations:

RECOGNISSIMO 1	RECOGNISSIMO 2

```
using UnityEngine;
using Recognissimo.Components;
using Recognissimo.Core;

public class SpeechRecognitionExample : MonoBehaviour
{
    [SerializeField]
    private SpeechRecognizer recognizer;

    [SerializeField]
    private LanguageModelProvider modelProvider;

    [SerializeField]
    private MicrophoneSpeechSource mic;

    private async void Start()
    {
        // Setup microphone.
        mic.microphoneSettings.deviceIndex = 0;
        mic.microphoneSettings.sampleRate = 16000;
        mic.microphoneSettings.timeSensitivity = 0.25f;
        mic.microphoneSettings.maxRecordingTime = 1;

        // Start microphone explicitly.
        mic.StartMicrophone();

        // Setup model provider.
        modelProvider.speechModels.Add(
            new LanguageModelProvider.ModelStreamingAssetsPath{
                modelPath = "LanguageModels/en-US",
                language = SystemLanguage.English}
        );

        await modelProvider.InitializeAsync();
        await modelProvider.LoadLanguageModelAsync(SystemLanguage.English);

        // Setup and start recognizer.
        recognizer.speechSource = mic;
        recognizer.modelProvider = modelProvider;
        recognizer.partialResultReady.AddListener(OnPartialResult);
        recognizer.resultReady.AddListener(OnResult);
        recognizer.enableDetailedResultDescription = false;
        recognizer.StartRecognition();
    }

    public async void SwitchLanguage(SystemLanguage language)
    {
        recognizer.StopRecognition();
        await modelProvider.LoadLanguageModelAsync(language);
        recognizer.StartRecognition();
    }

    private void OnPartialResult(PartialResult partialResult)
    {
        Debug.Log($"<color=yellow>{partialResult.partial}</color>");
    }

    private void OnResult(Result result)
    {
        Debug.Log($"<color=green>{result.text}</color>");
    }
}
```

```
using UnityEngine;
using Recognissimo.Components;

public class SpeechRecognitionExample : MonoBehaviour
{
    [SerializeField]
    private SpeechRecognizer recognizer;

    [SerializeField]
    private StreamingAssetsLanguageModelProvider modelProvider;

    [SerializeField]
    private MicrophoneSpeechSource mic;

    // Initialization is handled internally,
    // no need to make the method async.
    private void Start()
    {
        // Setup microphone.
        mic.DeviceName = null;
        // Sample rate is detected automatically.
        mic.TimeSensitivity = 0.25f;
        // Max recording time is detected automatically.
        // Microphone is initialized and started automatically
        // when recognizer.StartProcessing() called.

        // Setup language model provider.
        modelProvider.languageModels.Add(
            new StreamingAssetsLanguageModel{
                path = "LanguageModels/en-US",
                language = SystemLanguage.English}
        );

        // Language model is initialized automatically
        // when recognizer.StartProcessing() called.

        // Setup and start recognizer.
        recognizer.SpeechSource = mic;
        recognizer.LanguageModelProvider = modelProvider;
        recognizer.PartialResultReady.AddListener(OnPartialResult);
        recognizer.ResultReady.AddListener(OnResult);
        recognizer.EnableDetails = false;
        recognizer.StartProcessing();
    }

    public void SwitchLanguage(SystemLanguage language)
    {
        recognizer.StopProcessing();
        modelProvider.language = language;
        recognizer.StartProcessing();
    }

    private void OnPartialResult(PartialResult partialResult)
    {
        Debug.Log($"<color=yellow>{partialResult.partial}</color>");
    }

    private void OnResult(Result result)
    {
        Debug.Log($"<color=green>{result.text}</color>");
    }
}
```

Glossary

The speech processing application consists of 3 components:

- Speech processor
- Language model provider
- Speech source

Speech Processor

Speech processor receives the audio data from the **speech source** and decodes it using the language model provided by the **language model provider**. Further actions are determined by the algorithm used (for example, speech recognition, voice control or voice activity detection). **Speech source** and **language model provider** are called **speech processor dependencies**.

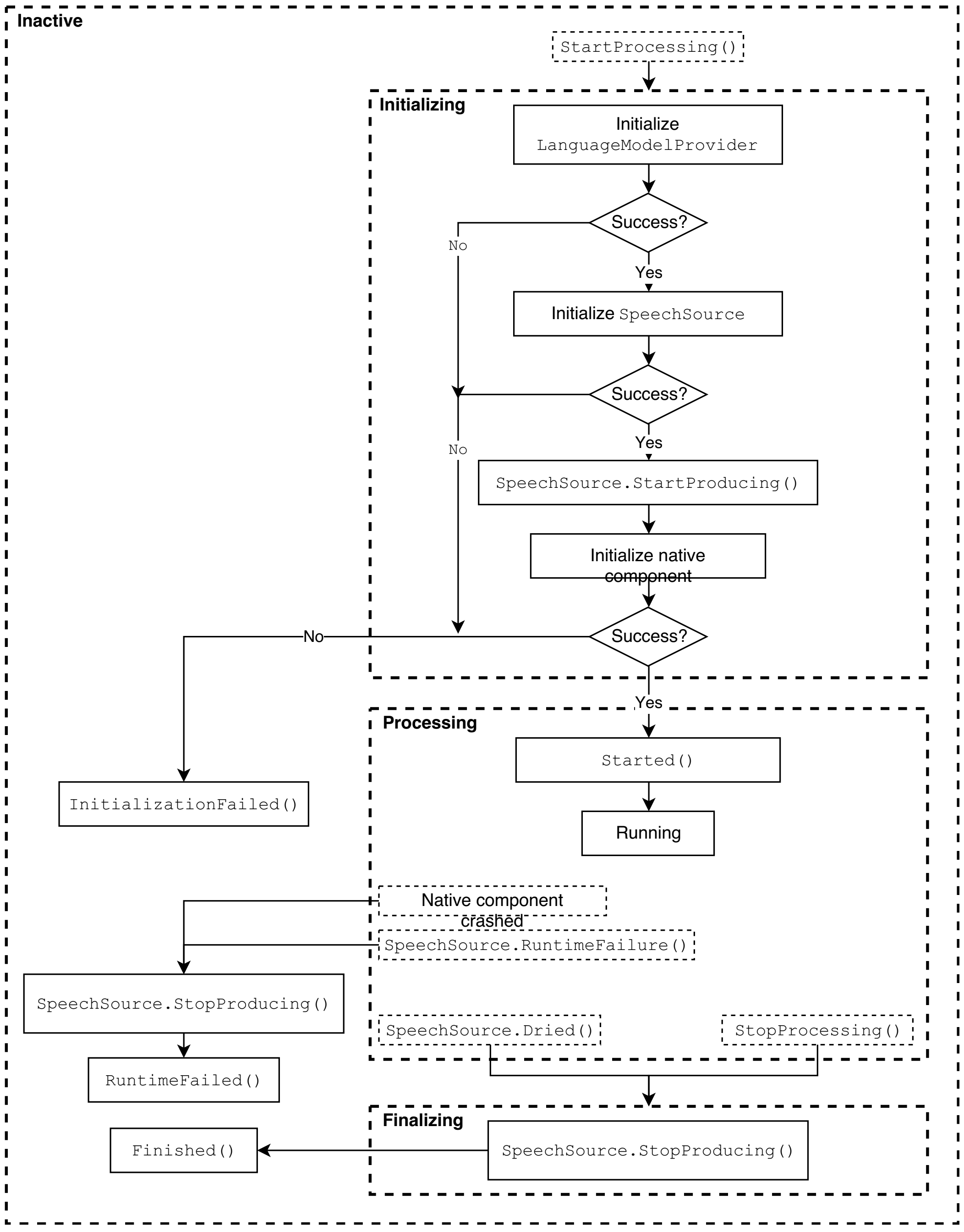
Speech Source

Speech source provides speech data. Recognissimo supports single-channel PCM audio at 16kHz and higher sampling rates.

Language Model Provider

Language model provider provides a language model - a set of files that are used by the speech processor to convert speech data to text. Recognissimo uses Vosk language models.

Speech Processor lifecycle

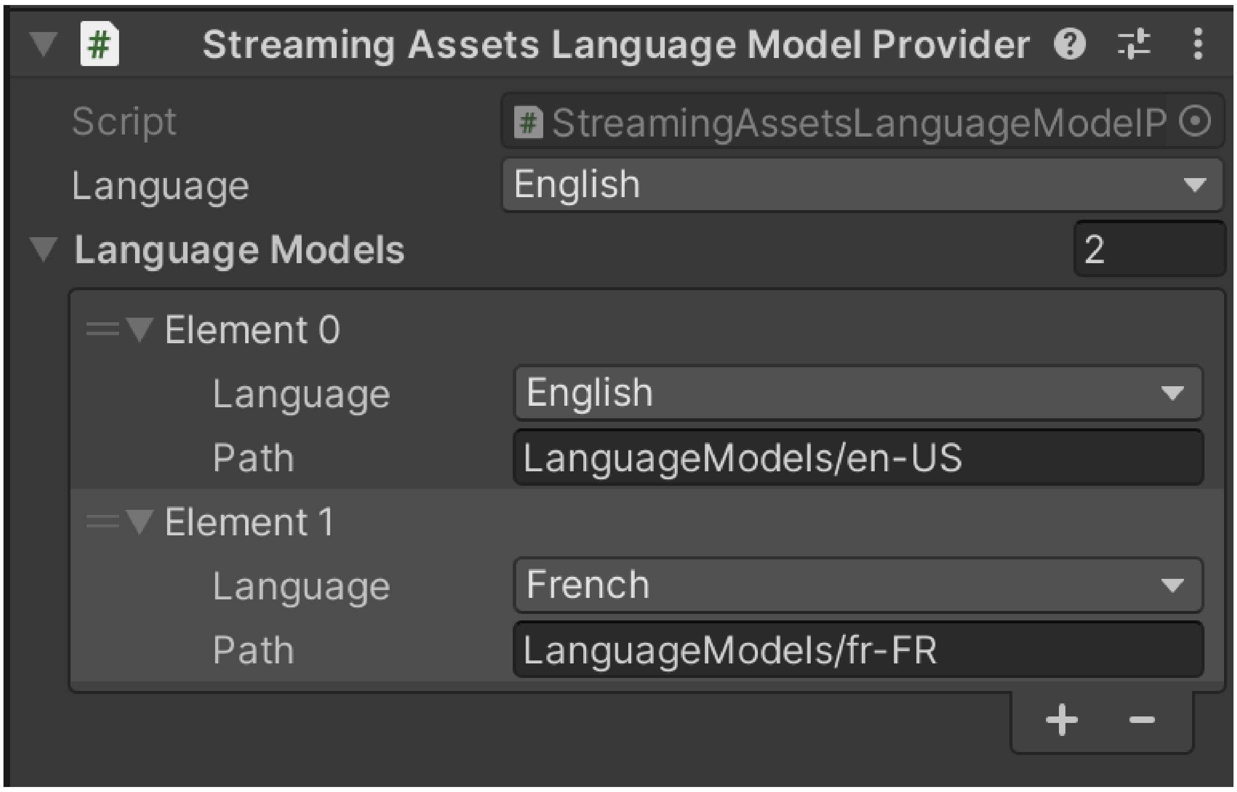


Setup Speech Processor dependencies

Editor

- Add a language model provider component to the scene.
  - If you want to load language model from StreamingAssets, add the `Streaming Assets Language Model Provider` component. Specify the path to the language models relative to the *StreamingAssets* folder and select the default language from the `Language` pop-up menu.

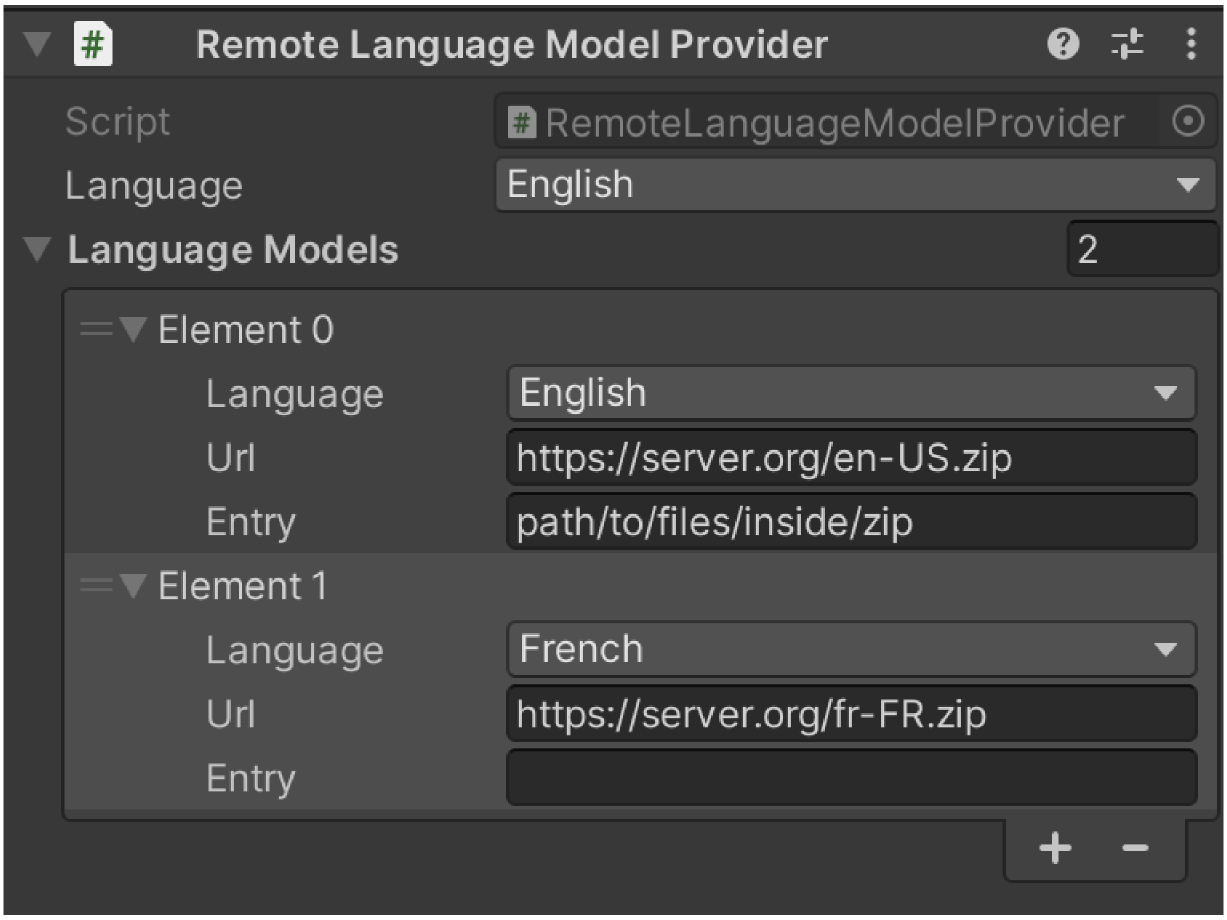
    > To add a new language model, download it (e.g. from here), extract it to the Unity StreamingAssets directory and specify path to its content and its language in `Streaming Assets Language Model Provider` settings.

    Streaming Assets Language Model Provider

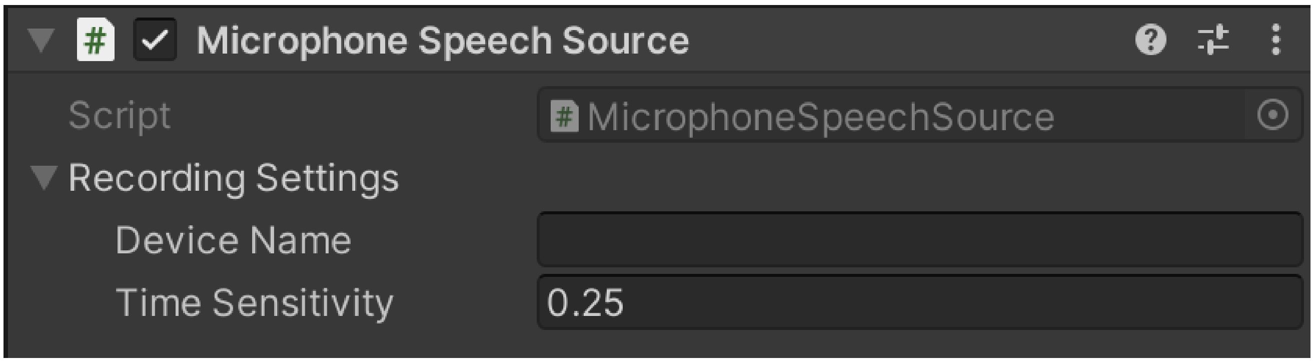
    Script: StreamingAssetsLanguageModelP
    Language: English
    Language Models: 2
    Element 0
    Language: English
    Path: LanguageModels/en-US
    Element 1
    Language: French
    Path: LanguageModels/fr-FR

  - If you want to download remote zipped language model, add the `Remote Language Model Provider` component. Specify the address of the remote zip containing language model files and select the default language from the `Language` pop-up menu. If the contents of the language model (folders `am`, `conf`, `graph` etc.) are located in the root of the zip, leave the `Entry` field empty, otherwise specify the path to the contents of the language model inside the zip.


Remote Language Model Provider

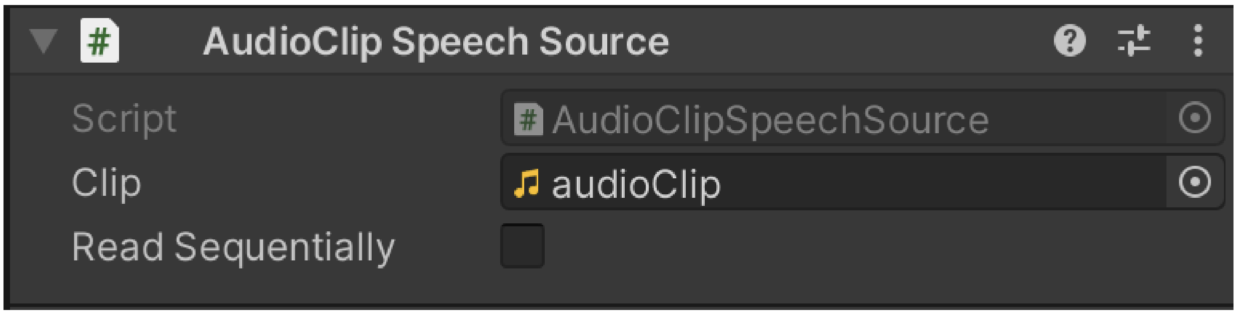
Script	RemoteLanguageModelProvider
Language	English
Language Models	2
Element 0	
Language	English
Url	https://server.org/en-US.zip
Entry	path/to/files/inside/zip
Element 1	
Language	French
Url	https://server.org/fr-FR.zip
Entry	

- Add speech source component to the scene.
  - If you want to use microphone, add `Microphone Speech Source` component.


Microphone Speech Source

Script	MicrophoneSpeechSource
Recording Settings	
Device Name	
Time Sensitivity	0.25

  - If you want to use audio clip, add `Audio Clip Speech Source` component and assign an audio clip to the `Clip` field. Use uncompressed mono audio (go to audioclip import settings and set `Force To Mono` to true, `Load Type` to `Decompress On Load`).


AudioClip Speech Source

Script	AudioClipSpeechSource
Clip	audioClip
Read Sequentially	☐

  - If you want to use AudioListener as a speech source, add `AudioListener Speech Source` component and specify channel from which you want the audio data to be streamed.

Explicit initialization

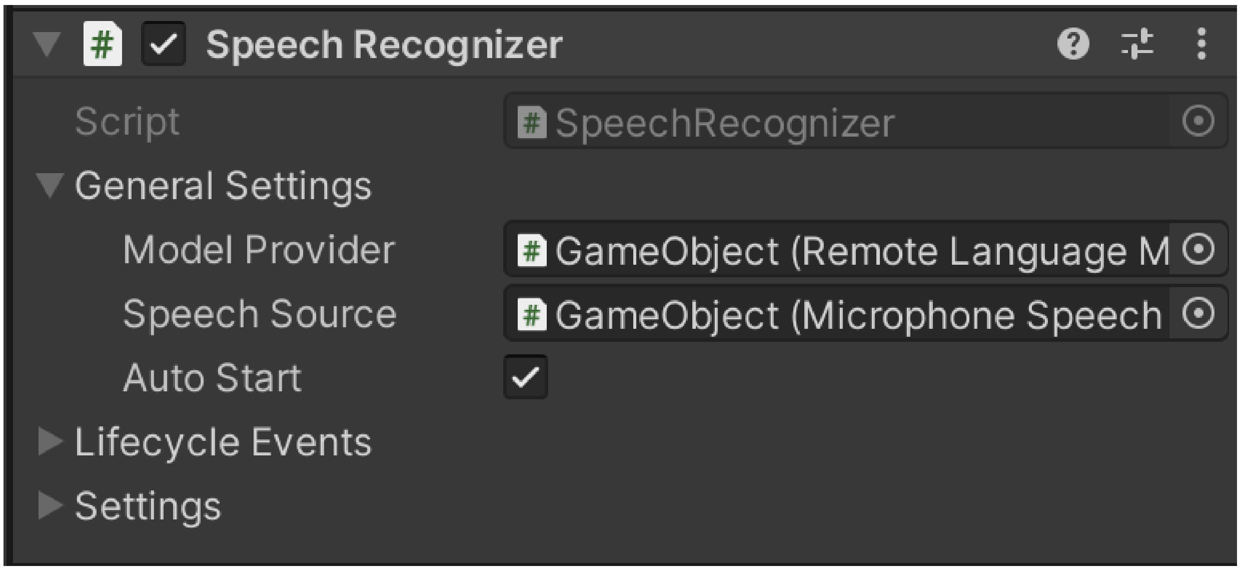
Speech processor implicitly initializes its dependencies when started. To initialize dependency explicitly, use `SpeechProcessorDependency.Initialize()` method. For example:

```
using System;
using System.Collections;
using Recognissimo;
using UnityEngine;

public class ExplicitInitializationExample : MonoBehaviour
{
    // SpeechSource inherits SpeechProcessorDependency base class.
    [SerializeField]
    private SpeechSource speechSource;

    private IEnumerator Start()
    {
        yield return speechSource.Initialize(HandleTaskStarted, HandleInitializationFail);
    }

    private void HandleTaskStarted(string taskName, bool isLongRunning)
    {
        // Print only for coroutines.
        if (isLongRunning)
        {
            Debug.Log($"Starting task {taskName}");
        }
    }

    private void HandleInitializationFail(string taskName, Exception exception)
    {
        Debug.Log($"Task {taskName} failed with error {exception.Message}");
    }
}
```

Setup Speech Recognizer

Editor

1. Add `Speech Recognizer` component to the scene, enable flag `Auto Start` and connect language model provider and speech source to it.



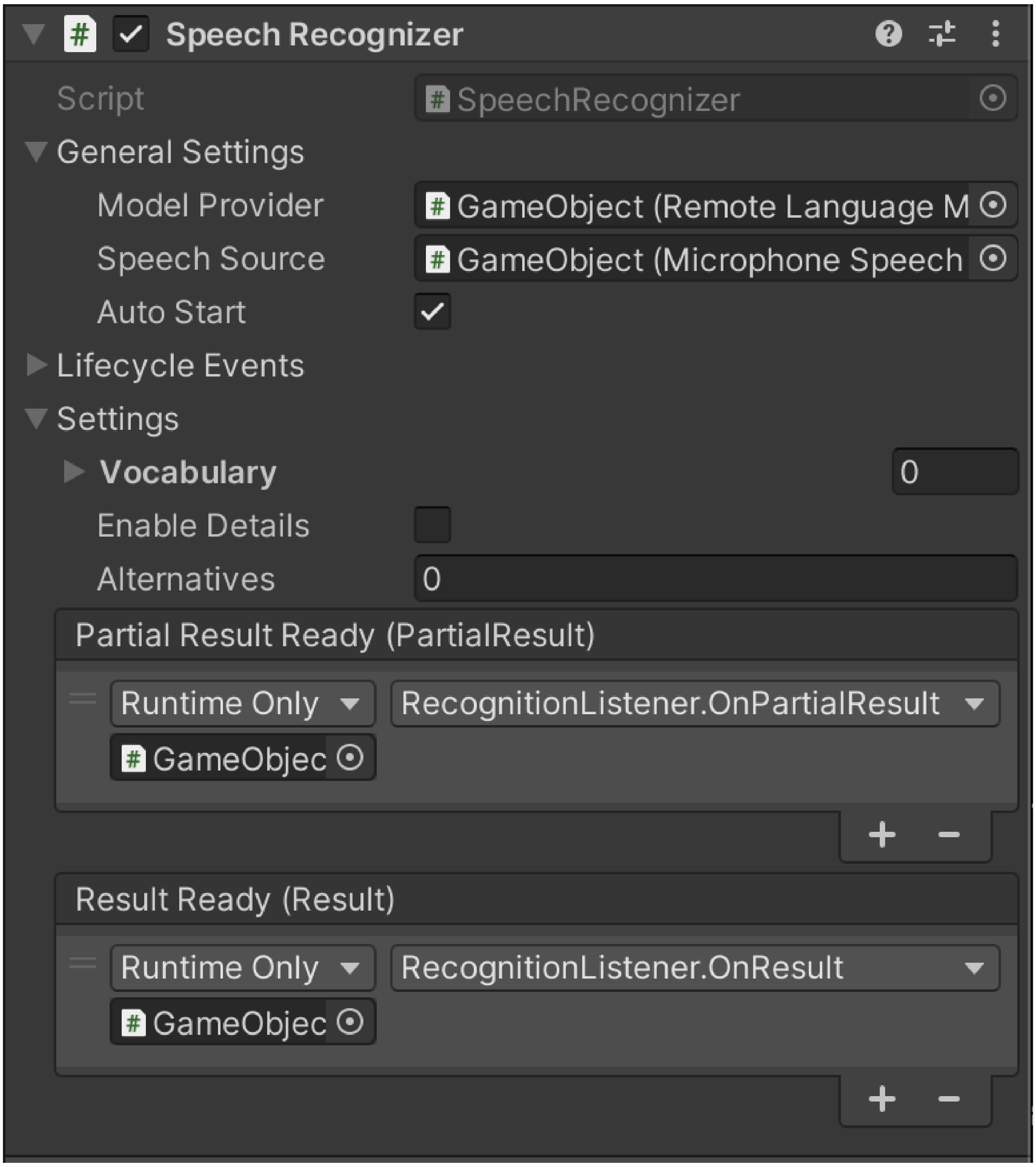
Speech Recognizer
Script: SpeechRecognizer
General Settings
Model Provider: GameObject (Remote Language M
Speech Source: GameObject (Microphone Speech
Auto Start: ✓
Lifecycle Events
Settings

2. Now let's get the output:

    1. Create a script called **RecognitionListener.cs**

    ```
    using Recognissimo.Components;
    using UnityEngine;

    public class RecognitionListener : MonoBehaviour
    {
        public void OnPartialResult(PartialResult partialResult)
        {
            Debug.Log($"<color=yellow>{partialResult.partial}</color>");
        }

        public void OnResult(Result result)
        {
            Debug.Log($"<color=green>{result.text}</color>");
        }
    }
    ```

    2. Add the `Recognition Listener` component and connect it to the `Speech Recognizer` events



Speech Recognizer

Script: SpeechRecognizer

General Settings

- Model Provider: GameObject (Remote Language M
- Speech Source: GameObject (Microphone Speech
- Auto Start: ✓

Lifecycle Events

Settings

- Vocabulary: 0
- Enable Details:
- Alternatives: 0

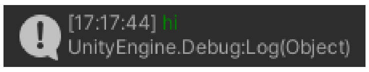
Partial Result Ready (PartialResult)

Runtime Only | RecognitionListener.OnPartialResult
GameObjec

Result Ready (Result)

Runtime Only | RecognitionListener.OnResult
GameObjec

3. Press **Play**. In the console window you should see the output



```
[17:17:44] hi
UnityEngine.Debug:Log(Object)
```

Scripting

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

public class SpeechRecognizerExample : MonoBehaviour
{
    private void Awake()
    {
        // Create components.
        var speechRecognizer = gameObject.AddComponent<SpeechRecognizer>();
        var languageModelProvider = gameObject.AddComponent<StreamingAssetsLanguageModelProvider>();
        var speechSource = gameObject.AddComponent<MicrophoneSpeechSource>();

        // Setup StreamingAssets language model provider.
        // Set the language used for recognition.
        languageModelProvider.language = SystemLanguage.English;
        // Set paths to language models.
        languageModelProvider.languageModels = new List<StreamingAssetsLanguageModel>
        {
            new() {language = SystemLanguage.English, path = "LanguageModels/en-US"},
            new() {language = SystemLanguage.French, path = "LanguageModels/fr-FR"}
        };

        // Setup microphone speech source. The default settings can be left unchanged, but we will do it as an example.
        speechSource.DeviceName = null;
        speechSource.TimeSensitivity = 0.25f;

        // Bind speech processor dependencies.
        speechRecognizer.LanguageModelProvider = languageModelProvider;
        speechRecognizer.SpeechSource = speechSource;

        // Handle events.
        speechRecognizer.PartialResultReady.AddListener(res => Debug.Log(res.partial));
        speechRecognizer.ResultReady.AddListener(res => Debug.Log(res.text));

        // Start processing.
        speechRecognizer.StartProcessing();
    }
}
```

How to use vocabulary

> This feature **may not be supported** by some language models.

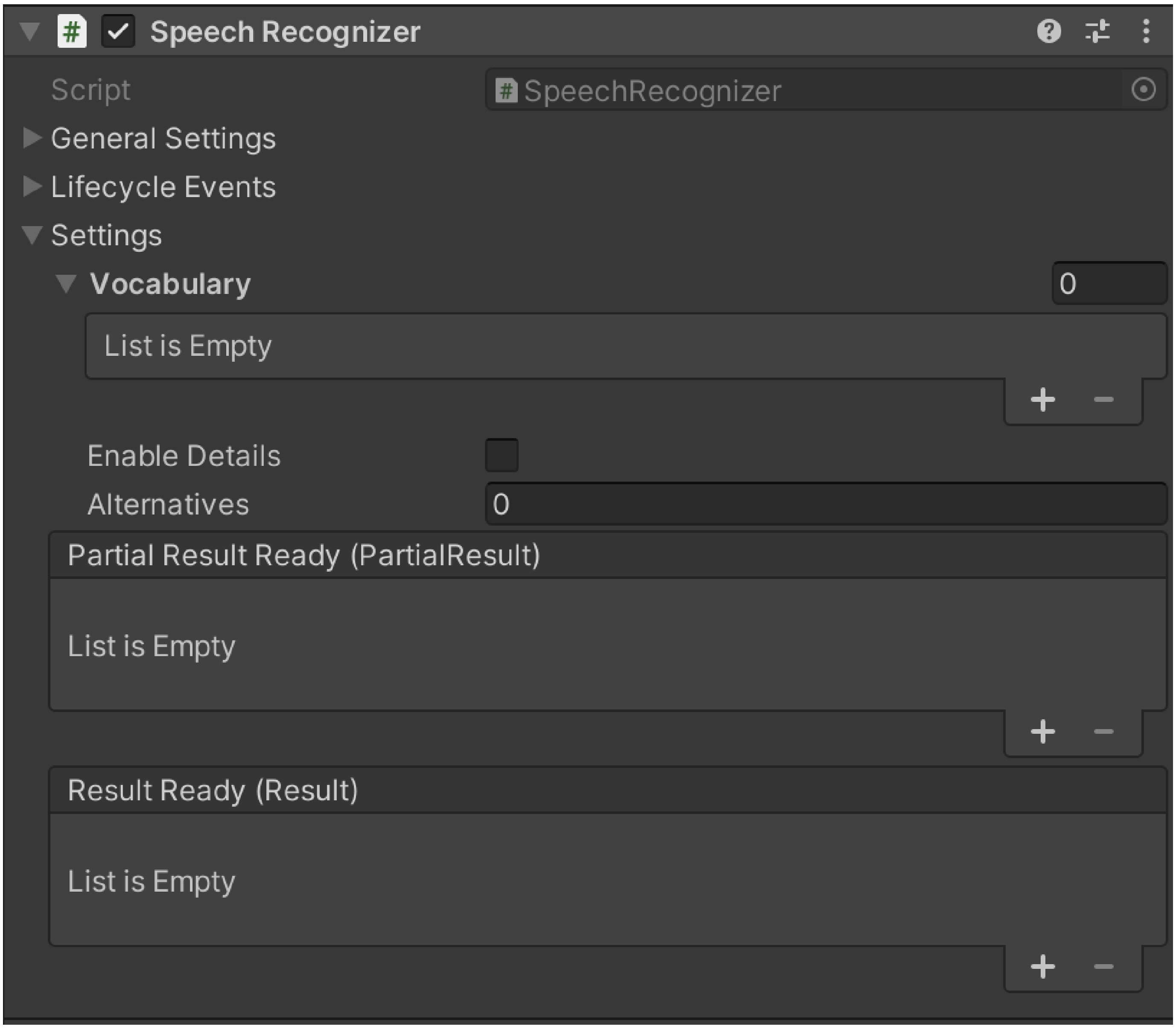
Vocabulary is a list of words available for speech recognizer. It is used to:

- simplify the recognition process by limiting the list of available words
- make speech recognizer output more predictable
- remove homophones

However, as the vocabulary definition implies, the speech recognition engine will try to match each spoken word with a word from the vocabulary, which is usually undesirable. To avoid this behavior, use the special word *"[unk]"* which means *"unknown word"*. Then every spoken word that cannot be recognized with the existing dictionary will be marked as *"[unk]"* in the resulting string.

You can set vocabulary using:

- UI (Speech Recognizer component)


Speech Recognizer

Script: SpeechRecognizer

General Settings

Lifecycle Events

Settings

Vocabulary: 0

List is Empty

Enable Details: ☐

Alternatives: 0

Partial Result Ready (PartialResult)

List is Empty

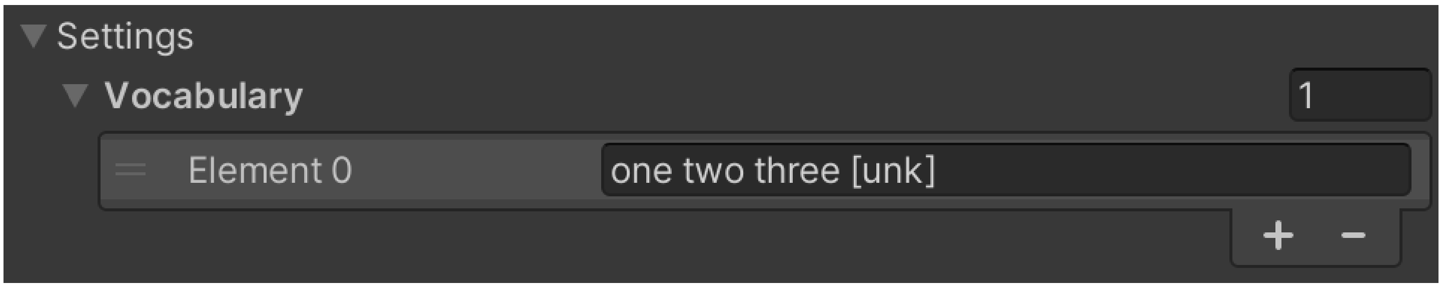
Result Ready (Result)

List is Empty

- script

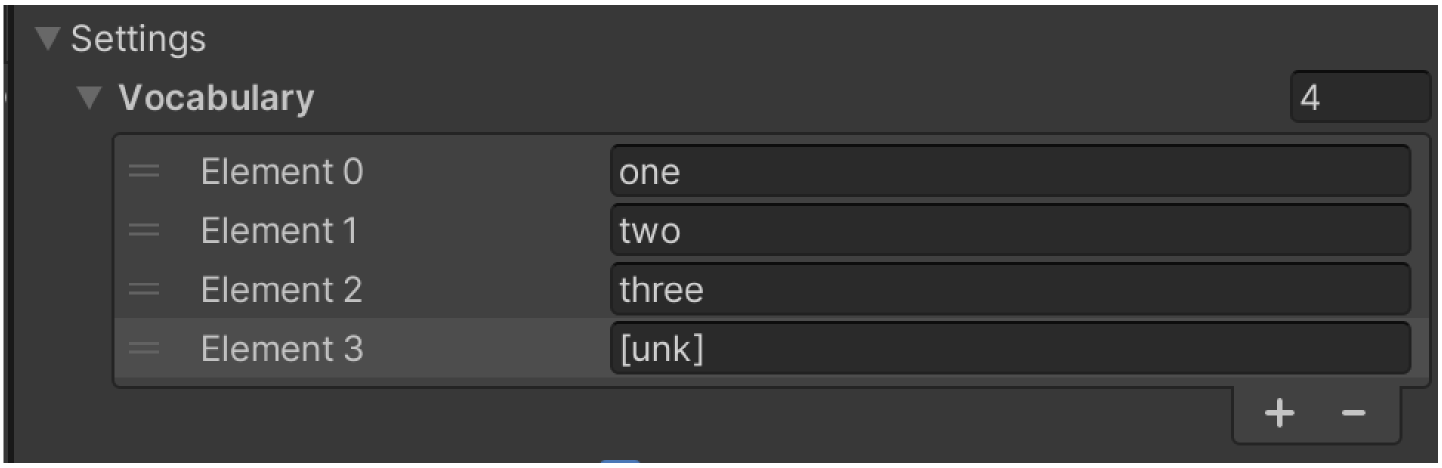
```
speechRecognizer.Vocabulary = new List<string>
{
   "one", "two three", "[unk]"
};
```

The order of the words doesn't matter. You can also use single string or multiple strings to describe the vocabulary. For example, the next vocabularies are the same:


Settings

Vocabulary: 1

Element 0: one two three [unk]



Settings

Vocabulary 4

Element 0	one
Element 1	two
Element 2	three
Element 3	[unk]

Setup Voice Activity Detector

Editor

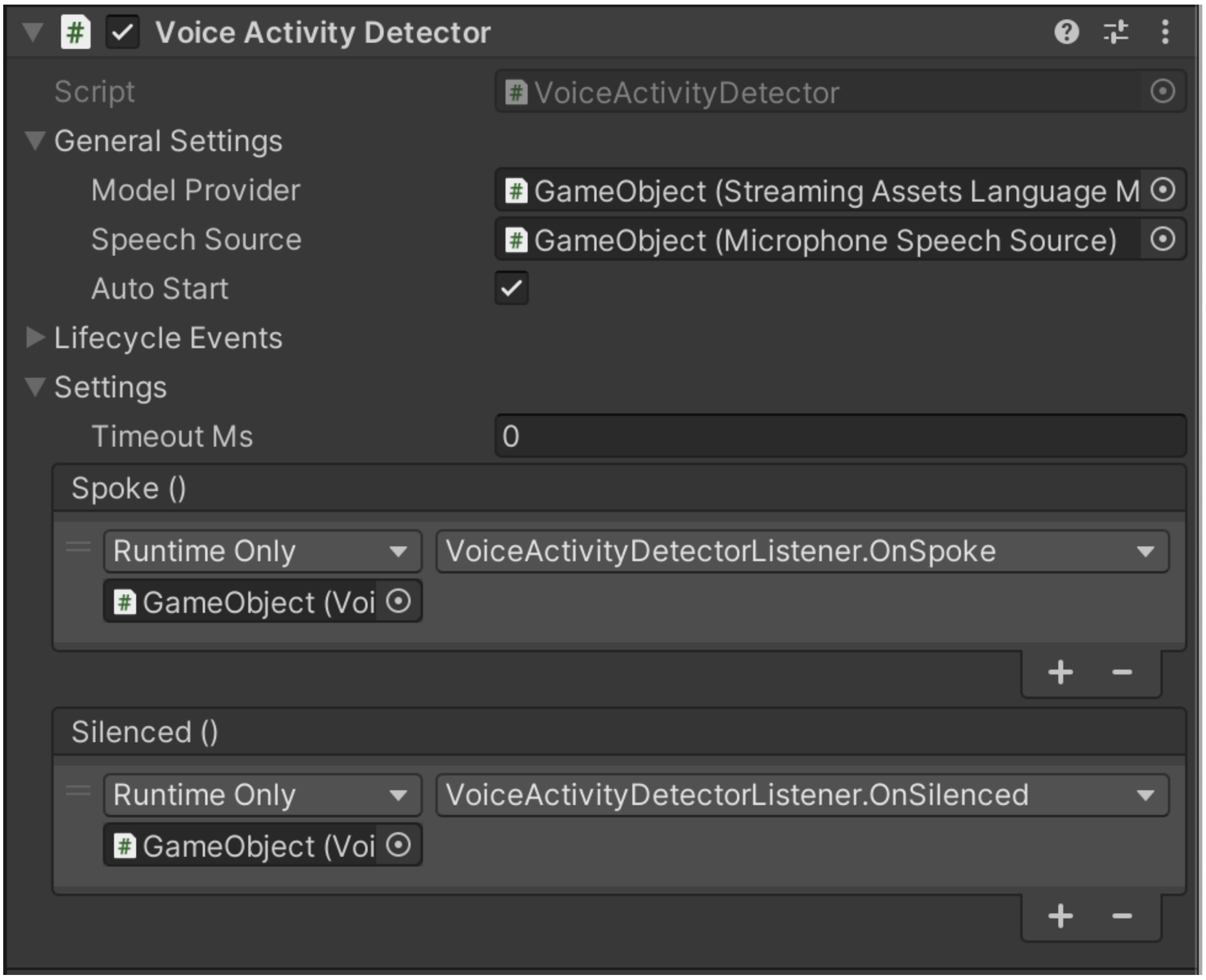
1. Add `Voice Activity Detector` component to the scene, enable flag `Auto Start` and connect language model provider and speech source to it.

2. Test voice activity detector.

    1. Create a script called **VoiceActivityDetectorListener.cs**.

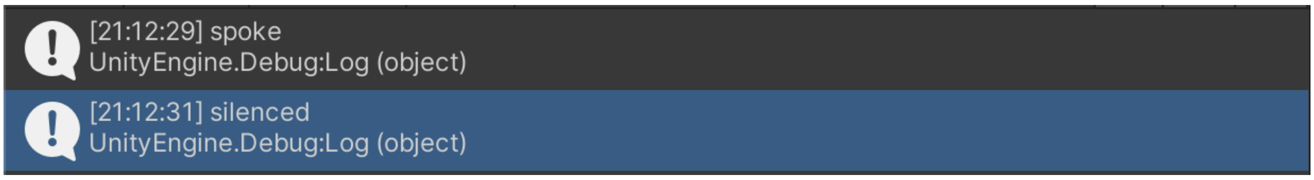
    ```
    using UnityEngine;

    public class VoiceActivityDetectorListener : MonoBehaviour
    {
        public void OnSpoke()
        {
            Debug.Log("spoke");
        }

        public void OnSilenced()
        {
            Debug.Log("silenced");
        }
    }
    ```

    2. Add the `Voice Activity Detector Listener` script and connect it to the `Voice Activity Detector` events.



Voice Activity Detector
Script: VoiceActivityDetector
General Settings
Model Provider: GameObject (Streaming Assets Language M
Speech Source: GameObject (Microphone Speech Source)
Auto Start: ✓
Lifecycle Events
Settings
Timeout Ms: 0
Spoke ()
Runtime Only | VoiceActivityDetectorListener.OnSpoke
GameObject (Voi
Silenced ()
Runtime Only | VoiceActivityDetectorListener.OnSilenced
GameObject (Voi

3. Press **Play**.



```
[21:12:29] spoke
UnityEngine.Debug:Log (object)
[21:12:31] silenced
UnityEngine.Debug:Log (object)
```

Scripting

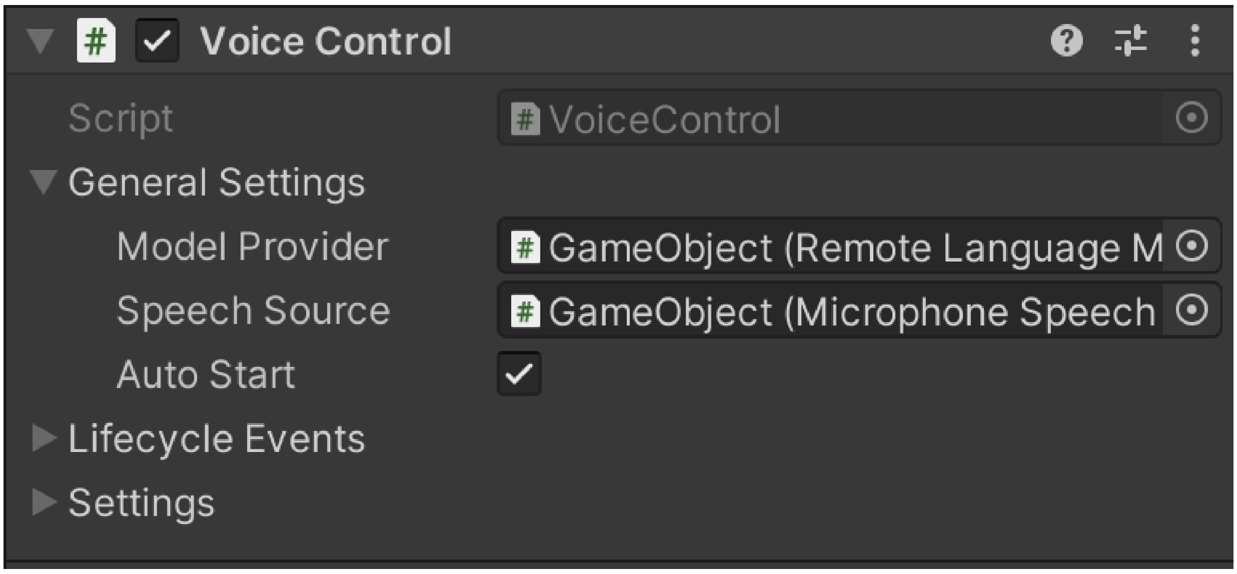
```
using Recognissimo.Components;
using UnityEngine;

public class SpeechSourceExample : MonoBehaviour
{
    private void Awake()
    {
        // Create components.
        var vad = gameObject.AddComponent<VoiceActivityDetector>();
        var languageModelProvider = gameObject.AddComponent<StreamingAssetsLanguageModelProvider>();
        var speechSource = gameObject.AddComponent<MicrophoneSpeechSource>();

        // Setup speech source and language model provider as in the previous example.
        // ...

        // Bind speech processor dependencies.
        vad.LanguageModelProvider = languageModelProvider;
        vad.SpeechSource = speechSource;

        // Setup voice control
        vad.TimeoutMs = 200;

        vad.Spoke.AddListener(() => Debug.Log("Spoke"));
        vad.Silenced.AddListener(() => Debug.Log("Silenced"));

        vad.StartProcessing();
    }
}
```

Setup Voice Control

Editor

1. Add Voice Control component to the scene, enable flag Auto Start and connect language model provider and speech source to it

2. Setup voice commands. Each command is a phrase and an event that is triggered when the phrase is spoken. The figure below shows an example of 2 commands that are activated when you speak "start" and "stop"



Voice Control

Script: VoiceControl

General Settings

Model Provider: GameObject (Remote Language M

Speech Source: GameObject (Microphone Speech

Auto Start: ✓

Lifecycle Events

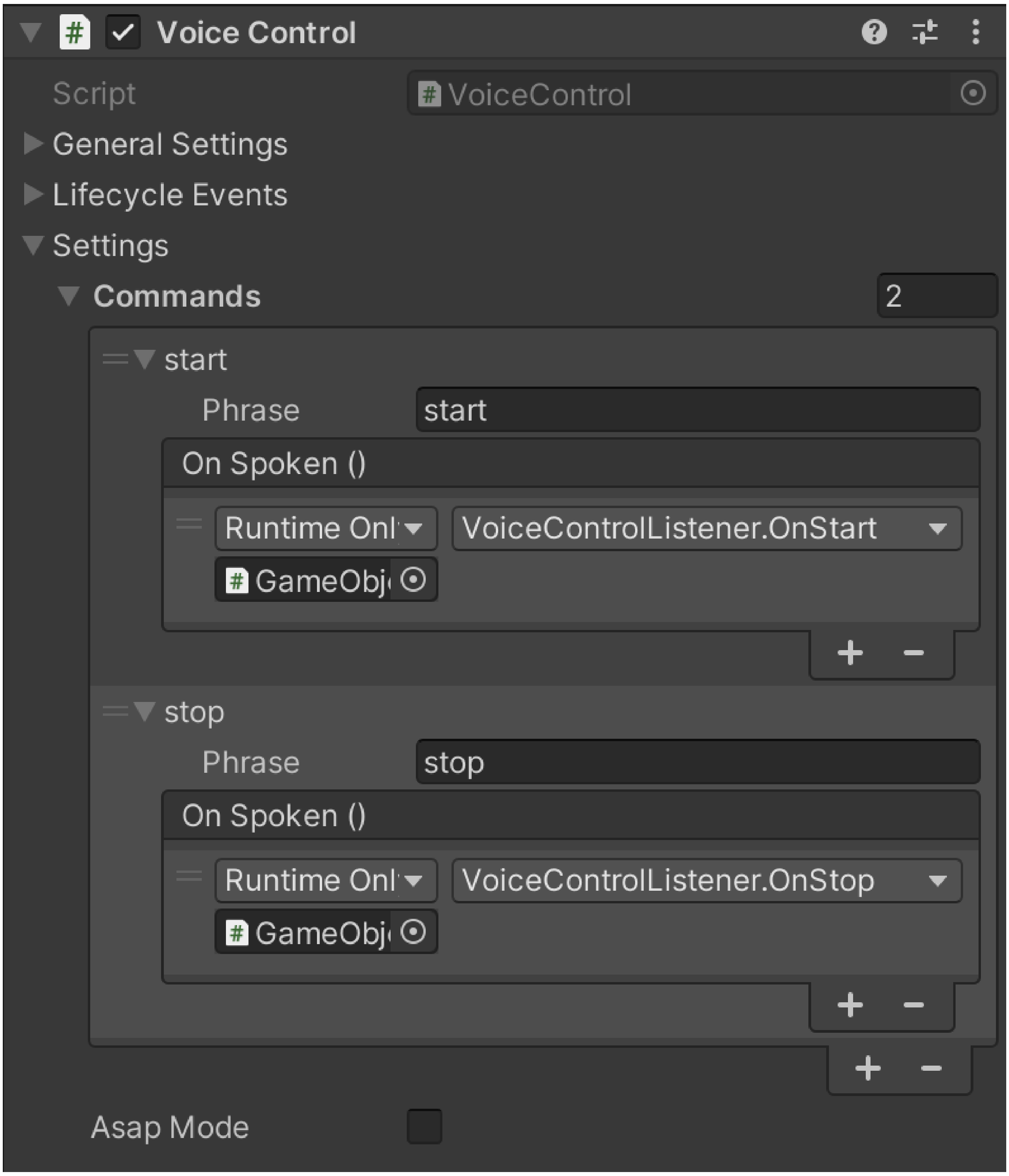
Settings

3. Test voice control

    1. Create a script called **VoiceControlListener.cs**.

    ```
    using UnityEngine;

    public class VoiceControlListener : MonoBehaviour
    {
        public void OnStart()
        {
            Debug.Log("start");
        }

        public void OnStop()
        {
            Debug.Log("stop");
        }
    }
    ```

    2. Add the Voice Control Listener script and connect it to the Voice Control events



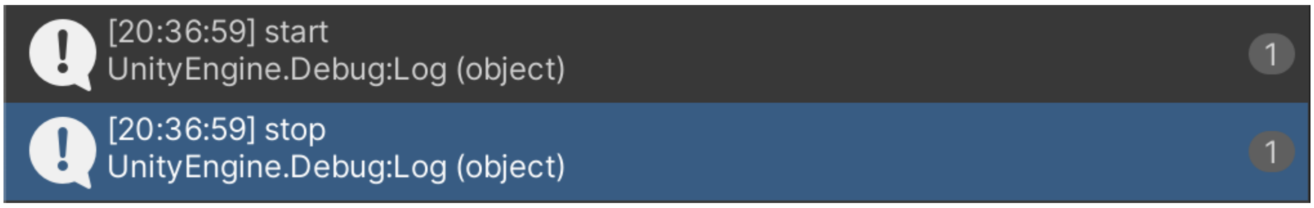
Script: VoiceControl
General Settings
Lifecycle Events
Settings
Commands: 2

start
Phrase: start
On Spoken ()
Runtime Onl | VoiceControlListener.OnStart
GameObj

stop
Phrase: stop
On Spoken ()
Runtime Onl | VoiceControlListener.OnStop
GameObj

Asap Mode: ☐

3. Press **Play**



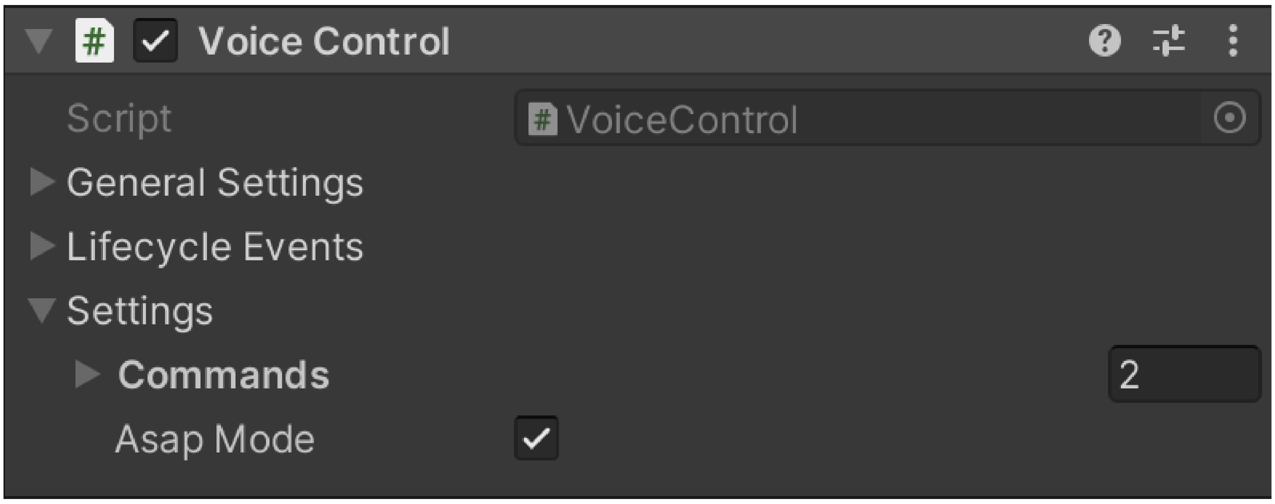
```
[20:36:59] start
UnityEngine.Debug:Log (object)                1
[20:36:59] stop
UnityEngine.Debug:Log (object)                1
```

Scripting

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

public class VoiceControlExample : MonoBehaviour
{
    private void Awake()
    {
        // Create components.
        var voiceControl = gameObject.AddComponent<VoiceControl>();
        var languageModelProvider = gameObject.AddComponent<StreamingAssetsLanguageModelProvider>();
        var speechSource = gameObject.AddComponent<MicrophoneSpeechSource>();

        // Setup speech source and language model provider as in the previous example.
        // ...

        // Bind speech processor dependencies.
        voiceControl.LanguageModelProvider = languageModelProvider;
        voiceControl.SpeechSource = speechSource;

        // Setup voice control
        voiceControl.AsapMode = true;

        voiceControl.Commands = new List<VoiceControlCommand>
        {
            new VoiceControlCommand("start|begin", () => Debug.Log("Start")),
            new VoiceControlCommand("stop", HandleStop)
        };

        voiceControl.StartProcessing();
    }

    private void HandleStop()
    {
        Debug.Log("Stop");
    }
}
```

Asap mode

You can make `Voice Control` component faster by enabling `Asap Mode` flag.



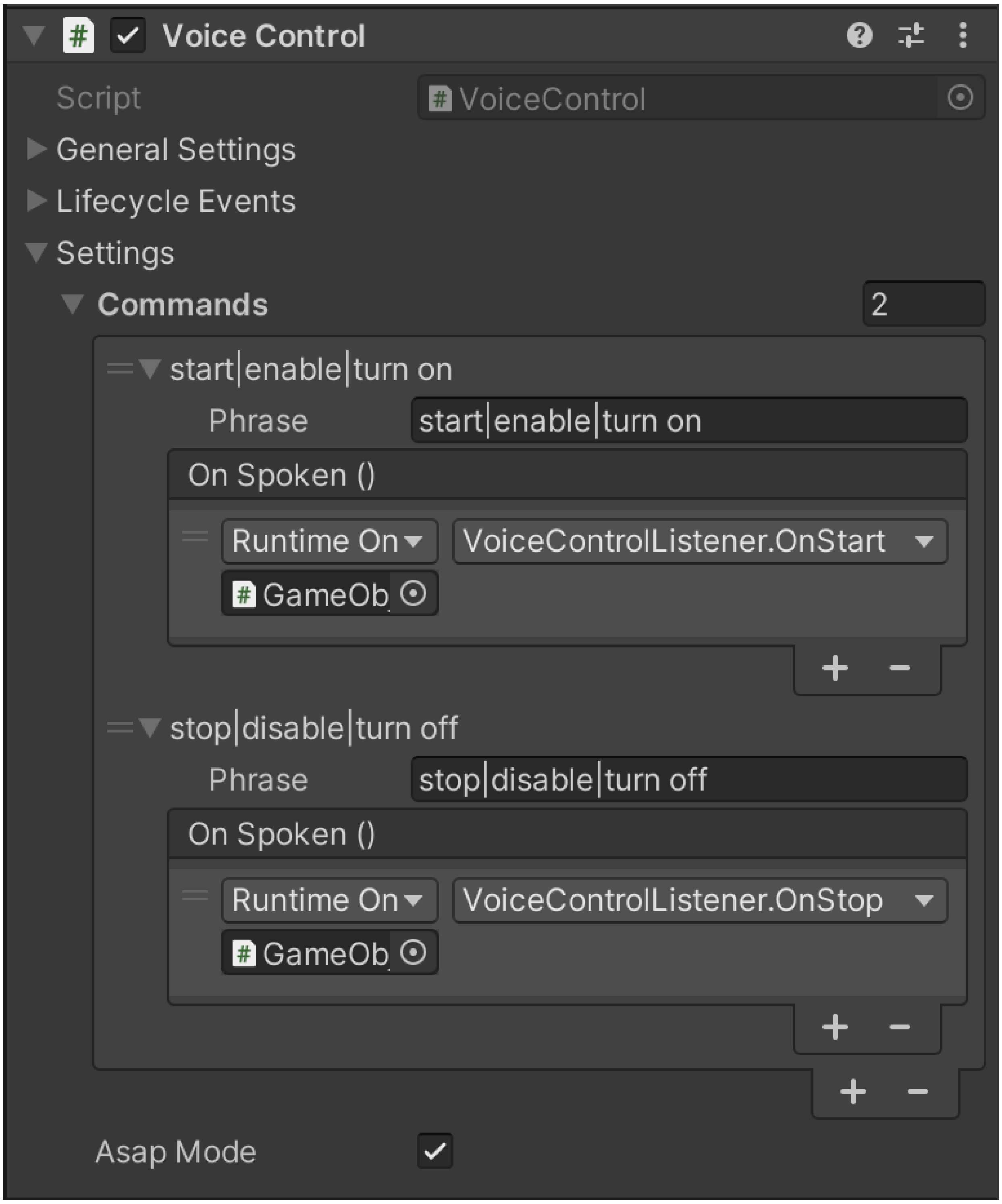
In this mode `Voice Control` component will use the preliminary recognition results to reduce response time.

How to use regular expressions

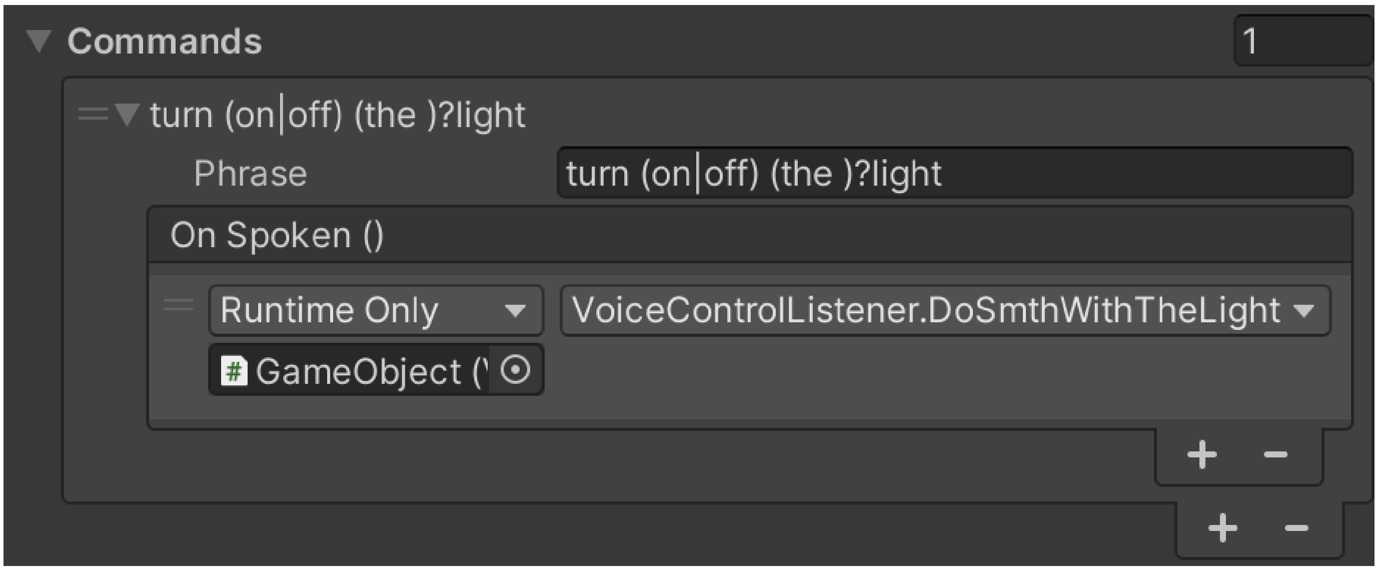
Voice Control supports regex patterns in phrases.

> Note that Recognissimo doesn't support patterns with multiple spaces.

- Use "|" to separate phrases


Voice Control
Script: VoiceControl
General Settings
Lifecycle Events
Settings
Commands: 2
start|enable|turn on
Phrase: start|enable|turn on
On Spoken ()
Runtime On — VoiceControlListener.OnStart — GameOb
stop|disable|turn off
Phrase: stop|disable|turn off
On Spoken ()
Runtime On — VoiceControlListener.OnStop — GameOb
Asap Mode: ✓

- Use more complex syntax. Voice Control will trigger the event when the user says turn on the light, turn off the light, turn on light or turn off light


Commands 1

turn (on|off) (the)?light

Phrase: turn (on|off) (the)?light

On Spoken ()

Runtime Only | VoiceControlListener.DoSmthWithTheLight

GameObject ('

Create your own components

To implement your `Language Model Provider` or `Speech Source`, inherit corresponding base class.

Both `Language Model Provider` and `Speech Source` inherit `SpeechProcessorDependency` class. The base class `SpeechProcessorDependency` provides a mechanism for lazy evaluation of initialization tasks, which are methods registered using the `SpeechProcessorDependency.RegisterInitializationTask` method.

> Initialization task registration should be performed in ***OnEnable()*** event.

Requirements:

- `Language Model Provider` must set `LanguageModelProvider.Model` property during or before the initialization.
- `Speech Source` must set `SpeechSource.SampleRate` property during or before the initialization.
- `Speech Source` must call `SpeechSource.OnSamplesReady()` to push voice data to the recognizer.
- `Speech Source` must call `SpeechSource.OnDried()` if it runs out of samples.
- `Speech Source` must call `SpeechSource.OnRuntimeFailure()` if it cannot continue for exceptional reasons.

Implement LanguageModelProvider

```
using System.IO;
using Recognissimo;
using UnityEngine;

public class LanguageModelProviderExample : LanguageModelProvider
{
    // Path to language model files.
    [SerializeField]
    private string pathToLanguageModel;

    // Use OnEnable() event for subscriptions.
    private void OnEnable()
    {
        // Tasks can be executed explicitly by calling
        // LanguageModelProviderExample.Initialize()
        // or implicitly, when SpeechProcessor.StartProcessing() called.

        RegisterInitializationTask("Execute always", // Name of the task.
            () => Debug.Log("Print every Initialize() call"), // Task.
            CallCondition.Always); // Task call condition.

        RegisterInitializationTask("Execute once",
            () => Debug.Log("Print at first Initialize() call"),
            CallCondition.Once);

        RegisterInitializationTask("Execute when value changed",
            LoadLanguageModel,
            CallCondition.ValueChanged(() => pathToLanguageModel));
    }

    private void LoadLanguageModel()
    {
        if (!Directory.Exists(pathToLanguageModel))
        {
            // Recognissimo will handle the exception and notify SpeechSource.
            throw new DirectoryNotFoundException("Path to language model does not exist");
        }

        // Store loaded model in Model property.
        base.Model = new LanguageModel(pathToLanguageModel);
    }
}
```

Implement SpeechSource

```
using System;
using System.Collections;
using Recognissimo;
using UnityEngine;

public class SpeechSourceExample : SpeechSource
{
    // Audio clip from which the data will be taken.
    public AudioClip clip;

    // Sample rate of audio clip.
    public override int SampleRate => clip.frequency;

    // Use OnEnable() event for subscriptions.
    private void OnEnable()
    {
        RegisterInitializationTask("Load audio clip",
            LoadAudio,
            CallCondition.ValueChanged(() => clip));
    }
    // Called by SpeechProcessor at the start of speech processing.
    public override void StartProducing()
    {
        // Create buffer.
        var buffer = new float[clip.samples];

        // Get audio data.
        clip.GetData(buffer, 0);

        // Send samples to SpeechProcessor.
        OnSamplesReady(new SamplesReadyEventArgs(buffer, buffer.Length));

        // Notify SpeechProcessor that we are out of samples.
        OnDried();
    }

    // Called by SpeechProcessor when finalizing speech processing.
    public override void StopProducing()
    {
        // Do nothing.
    }

    private IEnumerator LoadAudio()
    {
        // Wait while clip is loading.
        clip.LoadAudioData();

        while (clip.loadState == AudioDataLoadState.Loading)
        {
            yield return null;
        }

        if (clip.loadState == AudioDataLoadState.Failed)
        {
            // Use FailInitialization() method to make your code exceptionless.
            FailInitialization(new InvalidOperationException("Clip loading failed"));
        }

        if (clip.channels > 1)
        {
            // Recognissimo will handle exceptions inside the coroutine and notify SpeechSource.
            throw new NotSupportedException("Reading non-mono AudioClip is not supported yet.");
        }
    }
}
```

Speech Processor event handling

```
using System;
using Recognissimo;
using UnityEngine;

public class EventHandlingExample : LanguageModelProvider
{
    [SerializeField]
    private SpeechProcessor processor;

    private void Awake()
    {
        processor.Started.AddListener(WhenStarted);
        processor.Finished.AddListener(WhenFinished);
        processor.InitializationFailed.AddListener(WhenInitializationFailed);
        processor.RuntimeFailed.AddListener(WhenRuntimeFailed);
    }

    private void WhenStarted()
    {
        Debug.Log("Started");
    }

    private void WhenFinished()
    {
        Debug.Log("Finished");
    }

    private void WhenInitializationFailed(InitializationException rawException)
    {
        switch (rawException)
        {
            case InvalidLanguageModelException:
            case InvalidSampleRateException:
            case InvalidAlgorithmInputException:
                // Print exception message.
                Debug.Log(rawException.Message);
                break;
            case InternalInitializationException internalInitializationException:
                Debug.Log($"Recognissimo crashed because of an internal error : " +
                          $"{internalInitializationException.Message}");
                break;
            case DependencyInitializationException dependencyException:
                // Print type of SpeechProcessorDependency which caused the exception.
                Debug.Log(dependencyException.Dependency.GetType());
                // Print name of the initialization task.
                Debug.Log(dependencyException.InitializationTaskName);
                // Print inner exception message.
                if (dependencyException.InnerException != null)
                {
                    Debug.Log(dependencyException.InnerException.Message);
                }
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(rawException));
        }
    }

    private void WhenRuntimeFailed(RuntimeException rawException)
    {
        switch (rawException)
        {
            case InternalRuntimeException internalRuntimeException:
                Debug.Log($"Recognissimo crashed because of an internal error : " +
                Debug.Log($"Recognissimo crashed because of an internal error : " +
                          $"{internalRuntimeException.Message}");
                break;
            case SpeechSourceRuntimeException speechSourceRuntimeException:
                Debug.Log(speechSourceRuntimeException.Message);
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(rawException));
        }
    }
}
```

Namespace Recognissimo

Classes

CallCondition

Helper class for SpeechProcessorDependency that stores the condition of the associated call.

DependencyInitializationException

Thrown when SpeechProcessorDependency initialization fails.

InitializationException

Base class for all SpeechProcessor exceptions during initialization.

InternalInitializationException

Thrown when internal error occurs in SpeechProcessor during initialization. It is recommended that such an exception be reported to the developer.

InternalRuntimeException

Thrown when SpeechSource failed at runtime. It is recommended that such an exception be reported to the developer.

InvalidAlgorithmInputException

Thrown when invalid input is provided to SpeechProcessor implementation. It is recommended that such an exception be reported to the developer.

InvalidLanguageModelException

Thrown when invalid language model provided.

InvalidSampleRateException

Thrown when invalid sample rate provided.

LanguageModel

Stores language model native handle.

LanguageModelProvider

Base class for all model providers.

RuntimeException

Base class for all SpeechProcessor exceptions during runtime.

RuntimeFailureEventArgs

RuntimeFailure event data.

SamplesReadyEventArgs

SamplesReady event data.

SpeechProcessor

Base class for all speech processors.

SpeechProcessorDependency

This class extends UnityEngine.MonoBehaviour by adding lazy evaluation of user-defined initialization tasks.

SpeechProcessorException

Base class for all SpeechProcessor exceptions.

SpeechSource

Base class for all speech sources.

SpeechSourceRuntimeException

Thrown when SpeechSource failed at runtime.

Enums

SpeechProcessorState

SpeechProcessor state.

Delegates

InitializationFailedCallback

Callback raised when initialization failed.

InitializationTaskStartedCallback

Callback raised when a new initialization task is started

Class CallCondition

Helper class for SpeechProcessorDependency that stores the condition of the associated call.

Inheritance

System.Object
CallCondition

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class CallCondition
```

Constructors

CallCondition(Func<Boolean>)

Construct CallCondition from a predicate.

Declaration

```
public CallCondition(Func<bool> predicate)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Func<System.Boolean>	predicate	Predicate function, the return value of which determines whether or not to execute the initialization task to which it is linked.

Exceptions

TYPE	CONDITION
System.ArgumentNullException	If `predicate` is null.

Fields

Always

CallCondition that always allows to execute the associated call.

Declaration

```
public static readonly CallCondition Always
```

Field Value

TYPE	DESCRIPTION
CallCondition	

Once

CallCondition which allows to execute the associated call only once.

Declaration

```
public static readonly CallCondition Once
```

Field Value

TYPE	DESCRIPTION
CallCondition	

Methods

Aggregate(CallCondition[])

Aggregates multiple conditions into one.

Declaration

```
public static CallCondition Aggregate(CallCondition[] conditions)
```

Parameters

TYPE	NAME	DESCRIPTION
CallCondition[]	conditions	Array of conditions.

Returns

TYPE	DESCRIPTION
CallCondition	CallCondition instance.

Check()

Check if the condition is satisfied.

Declaration

```
public bool Check()
```

Returns

TYPE	DESCRIPTION
System.Boolean	Value of underlying condition.

ValueChanged<T>(Func<T>, Func<T, T, Boolean>)

Create CallCondition that allows to execute the associated call only if the return value of `dependencyGetter` changes.

Declaration

```
public static CallCondition ValueChanged<T>(Func<T> dependencyGetter, Func<T, T, bool> equalityComparer = null)
```

Parameters

TYPE	NAME	DESCRIPTION

System.Func<T>	dependencyGetter	Function, a change in the return value of which activates the CallCondition.
System.Func<T, T, System.Boolean>	equalityComparer	Custom equality comparer. If null, System.Collections.Generic.EqualityComparer`1.Equals(`0,`0) is used.

Returns

TYPE	DESCRIPTION
CallCondition	CallCondition instance.

Type Parameters

NAME	DESCRIPTION
T	Generic parameter of `dependencyGetter`.

Class DependencyInitializationException

Thrown when SpeechProcessorDependency initialization fails.

Inheritance

System.Object
System.Exception
SpeechProcessorException
InitializationException
DependencyInitializationException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class DependencyInitializationException : InitializationException, ISerializable
```

Constructors

DependencyInitializationException(SpeechProcessorDependency, String, Exception)

Declaration

```
public DependencyInitializationException(SpeechProcessorDependency dependency, string initializationTaskName, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
SpeechProcessorDependency	dependency	
System.String	initializationTaskName	
System.Exception	innerException	

Properties

Dependency

Declaration

```
public SpeechProcessorDependency Dependency { get; }
```

Property Value

TYPE	DESCRIPTION
SpeechProcessorDependency	

InitializationTaskName

Declaration

```
public string InitializationTaskName { get; }
```

Property Value

TYPE	DESCRIPTION
System.String	

Class InitializationException

Base class for all SpeechProcessor exceptions during initialization.

Inheritance

System.Object
System.Exception
SpeechProcessorException
InitializationException
DependencyInitializationException
InternalInitializationException
InvalidAlgorithmInputException
InvalidLanguageModelException
InvalidSampleRateException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public abstract class InitializationException : SpeechProcessorException, ISerializable
```

Constructors

InitializationException(String)

Declaration

```
protected InitializationException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

InitializationException(String, Exception)

Declaration

```
protected InitializationException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Delegate InitializationFailedCallback

Callback raised when initialization failed.

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public delegate void InitializationFailedCallback(string failedTaskName, Exception exception);
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	failedTaskName	
System.Exception	exception	

Delegate InitializationTaskStartedCallback

Callback raised when a new initialization task is started

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public delegate void InitializationTaskStartedCallback(string taskName, bool isLongRunning);
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	taskName	
System.Boolean	isLongRunning	

Class InternalInitializationException

Thrown when internal error occurs in SpeechProcessor during initialization. It is recommended that such an exception be reported to the developer.

Inheritance

System.Object

System.Exception

SpeechProcessorException

InitializationException

InternalInitializationException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class InternalInitializationException : InitializationException, ISerializable
```

Constructors

InternalInitializationException(String)

Declaration

```
public InternalInitializationException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

Class InternalRuntimeException

Thrown when SpeechSource failed at runtime. It is recommended that such an exception be reported to the developer.

Inheritance

System.Object
System.Exception
SpeechProcessorException
RuntimeException
InternalRuntimeException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class InternalRuntimeException : RuntimeException, ISerializable
```

Constructors

InternalRuntimeException(String)

Declaration

```
public InternalRuntimeException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

Class InvalidAlgorithmInputException

Thrown when invalid input is provided to SpeechProcessor implementation. It is recommended that such an exception be reported to the developer.

Inheritance

System.Object

System.Exception

SpeechProcessorException

InitializationException

InvalidAlgorithmInputException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class InvalidAlgorithmInputException : InitializationException, ISerializable
```

Constructors

InvalidAlgorithmInputException(String)

Declaration

```
public InvalidAlgorithmInputException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

Class InvalidLanguageModelException

Thrown when invalid language model provided.

Inheritance

System.Object
System.Exception
SpeechProcessorException
InitializationException
InvalidLanguageModelException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class InvalidLanguageModelException : InitializationException, ISerializable
```

Constructors

InvalidLanguageModelException(String)

Declaration

```
public InvalidLanguageModelException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

Class InvalidSampleRateException

Thrown when invalid sample rate provided.

Inheritance

System.Object

System.Exception

SpeechProcessorException

InitializationException

InvalidSampleRateException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class InvalidSampleRateException : InitializationException, ISerializable
```

Constructors

InvalidSampleRateException(String)

Declaration

```
public InvalidSampleRateException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

Class LanguageModel

Stores language model native handle.

Inheritance

System.Object
LanguageModel

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class LanguageModel : IDisposable
```

Constructors

LanguageModel(String)

Create new instance.

Declaration

```
public LanguageModel(string path)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	path	Path to the language model files.

Methods

Dispose()

Dispose language model.

Declaration

```
public void Dispose()
```

Class LanguageModelProvider

Base class for all model providers.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
LanguageModelProvider
RemoteLanguageModelProvider
StreamingAssetsLanguageModelProvider

Inherited Members

SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: Recognissimo
Assembly: Recognissimo.dll

Syntax

```
public abstract class LanguageModelProvider : SpeechProcessorDependency
```

Properties

Model

Language model instance. Must be set during initialization.

Declaration

```
public LanguageModel Model { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
LanguageModel	

Class RuntimeException

Base class for all SpeechProcessor exceptions during runtime.

Inheritance

System.Object
System.Exception
SpeechProcessorException
RuntimeException
InternalRuntimeException
SpeechSourceRuntimeException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public abstract class RuntimeException : SpeechProcessorException, ISerializable
```

Constructors

RuntimeException(String)

Declaration

```
protected RuntimeException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

RuntimeException(String, Exception)

Declaration

```
protected RuntimeException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Class RuntimeFailureEventArgs

RuntimeFailure event data.

Inheritance

System.Object
System.EventArgs
RuntimeFailureEventArgs

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class RuntimeFailureEventArgs : EventArgs
```

Constructors

RuntimeFailureEventArgs(SpeechSourceRuntimeException)

Declaration

```
public RuntimeFailureEventArgs(SpeechSourceRuntimeException exception)
```

Parameters

TYPE	NAME	DESCRIPTION
SpeechSourceRuntimeException	exception	

Properties

Exception

Declaration

```
public SpeechSourceRuntimeException Exception { get; }
```

Property Value

TYPE	DESCRIPTION
SpeechSourceRuntimeException	

Class SamplesReadyEventArgs

SamplesReady event data.

Inheritance

System.Object
System.EventArgs
SamplesReadyEventArgs

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class SamplesReadyEventArgs : EventArgs
```

Constructors

SamplesReadyEventArgs(Single[], Int32)

Declaration

```
public SamplesReadyEventArgs(float[] samples, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Single[]	samples	
System.Int32	length	

Properties

Length

Audio samples length.

Declaration

```
public int Length { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Samples

Audio samples in Float32 format.

Declaration

```
public float[] Samples { get; }
```

Property Value

TYPE	DESCRIPTION
System.Single[]	

Class SpeechProcessor

Base class for all speech processors.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessor
SpeechRecognizer
VoiceActivityDetector
VoiceControl

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public abstract class SpeechProcessor : MonoBehaviour
```

Properties

AutoStart

Whether to execute StartProcessing() at start.

Declaration

```
public bool AutoStart { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Finished

SpeechProcessor successfully finished.

Declaration

```
public UnityEvent Finished { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

InitializationFailed

SpeechProcessor or one of its dependencies failed during initialization.

Declaration

```
public UnityEvent<InitializationException> InitializationFailed { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent<InitializationException>	

LanguageModelProvider

Language model provider. This value is read when StartProcessing() called.

Declaration

```
public LanguageModelProvider LanguageModelProvider { get; set; }
```

Property Value

TYPE	DESCRIPTION
LanguageModelProvider	

RuntimeFailed

SpeechProcessor or SpeechSource dependency failed at runtime.

Declaration

```
public UnityEvent<RuntimeException> RuntimeFailed { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent<RuntimeException>	

SpeechSource

Speech source. This value is read when StartProcessing() called.

Declaration

```
public SpeechSource SpeechSource { get; set; }
```

Property Value

TYPE	DESCRIPTION
SpeechSource	

Started

SpeechProcessor successfully started.

Declaration

```
public UnityEvent Started { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

State

Current state of SpeechProcessor

Declaration

```
public SpeechProcessorState State { get; }
```

Property Value

TYPE	DESCRIPTION
SpeechProcessorState	

Methods

StartProcessing()

Start speech processing. SpeechProcessor will setup itself asynchronously, then emit Started. SpeechSource and LanguageModelProvider must be set by the time the method is called.

Declaration

```
public void StartProcessing()
```

Exceptions

TYPE	CONDITION
System.InvalidOperationException	If SpeechSource or LanguageModelProvider is null.

StopProcessing()

Stop speech processing. SpeechProcessor will:

1. stop accepting new samples;
2. process the remaining samples;
3. emit Finished.

Declaration

```
public void StopProcessing()
```

Class SpeechProcessorDependency

This class extends UnityEngine.MonoBehaviour by adding lazy evaluation of user-defined initialization tasks.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
LanguageModelProvider
SpeechSource

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public abstract class SpeechProcessorDependency : MonoBehaviour
```

Properties

IsInitialized

Declaration

```
protected bool IsInitialized { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Methods

FailInitialization(Exception)

Mark current initialization task as failed with specified `exception`.

Declaration

```
protected void FailInitialization(Exception exception)
```

Parameters

TYPE	NAME	DESCRIPTION
System.Exception	exception	Fail reason.

Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)

Execute all initialization tasks registered by RegisterInitializationTask(String, Action, CallCondition) (or any other overload) whose CallCondition is met. At the first call all registered tasks will be executed regardless of their CallCondition

Declaration

```
public IEnumerator Initialize(InitializationTaskStartedCallback initializationTaskStartedCallback, InitializationFailedCallback initializationFailedCallback)
```

Parameters

TYPE	NAME	DESCRIPTION
InitializationTaskStartedCallback	initializationTaskStartedCallback	Callback invoked when a new initialization task is started.
InitializationFailedCallback	initializationFailedCallback	Callback invoked when exception is thrown during initialization.

Returns

TYPE	DESCRIPTION
System.Collections.IEnumerator	Enumerator to run coroutine on.

RegisterInitializationTask(String, Action, CallCondition)

Register initialization task. Task will be executed on the first call to Initialize(InitializationTaskStartedCallback, InitializationFailedCallback) and on subsequent calls if `callCondition` is true. Tasks order is preserved.

Declaration

```
protected void RegisterInitializationTask(string taskName, Action task, CallCondition callCondition)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	taskName	Name of the task.
System.Action	task	Initialization task.
CallCondition	callCondition	Task call condition.

RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)

Register initialization task. Task will be executed on the first call to Initialize(InitializationTaskStartedCallback, InitializationFailedCallback) and on subsequent calls if `callCondition` is true. Tasks order is preserved.

Declaration

```
protected void RegisterInitializationTask(string taskName, Func<IEnumerator> task, CallCondition callCondition)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	taskName	Name of the task.
System.Func<System.Collections.IEnumerator>	task	

CallCondition	callCondition	Task call condition.

Class SpeechProcessorException

Base class for all SpeechProcessor exceptions.

Inheritance

System.Object
System.Exception
SpeechProcessorException
InitializationException
RuntimeException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public abstract class SpeechProcessorException : Exception, ISerializable
```

Constructors

SpeechProcessorException(String)

Declaration

```
protected SpeechProcessorException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

SpeechProcessorException(String, Exception)

Declaration

```
protected SpeechProcessorException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Enum SpeechProcessorState

SpeechProcessor state.

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public enum SpeechProcessorState
```

Fields

NAME	DESCRIPTION
Finalizing	
Inactive	
Initializing	
Processing	

Class SpeechSource

Base class for all speech sources.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
SpeechSource
AudioClipSpeechSource
AudioListenerSpeechSource
MicrophoneSpeechSource

Inherited Members

SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public abstract class SpeechSource : SpeechProcessorDependency
```

Properties

SampleRate

Speech sampling rate. Must be set during initialization.

Declaration

```
public virtual int SampleRate { get; protected set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Methods

OnDried()

Helper method for triggering the event.

Declaration

```
protected void OnDried()
```

OnRuntimeFailure(RuntimeFailureEventArgs)

Helper method for triggering the event.

Declaration

```
protected void OnRuntimeFailure(RuntimeFailureEventArgs eventArgs)
```

Parameters

TYPE	NAME	DESCRIPTION
RuntimeFailureEventArgs	eventArgs	Event argument.

OnSamplesReady(SamplesReadyEventArgs)

Helper method for triggering the event.

Declaration

```
protected void OnSamplesReady(SamplesReadyEventArgs eventArgs)
```

Parameters

TYPE	NAME	DESCRIPTION
SamplesReadyEventArgs	eventArgs	Event argument.

StartProducing()

Called by SpeechProcessor at the start of processing.

Declaration

```
public abstract void StartProducing()
```

StopProducing()

Called when processing stops (e.g. when StopProcessing() called or when RuntimeFailure event emitted).

Declaration

```
public abstract void StopProducing()
```

Events

Dried

Raised when SpeechSource have run out of samples.

Declaration

```
public event EventHandler Dried
```

Event Type

TYPE	DESCRIPTION
System.EventHandler	

RuntimeFailure

Raised when SpeechSource failed during runtime.

Declaration

```
public event EventHandler<RuntimeFailureEventArgs> RuntimeFailure
```

Event Type

TYPE	DESCRIPTION
System.EventHandler<RuntimeFailureEventArgs>	

SamplesReady

Raised when new samples arrive.

Declaration

```
public event EventHandler<SamplesReadyEventArgs> SamplesReady
```

Event Type

TYPE	DESCRIPTION
System.EventHandler<SamplesReadyEventArgs>	

Class SpeechSourceRuntimeException

Thrown when SpeechSource failed at runtime.

Inheritance

System.Object
System.Exception
SpeechProcessorException
RuntimeException
SpeechSourceRuntimeException

Namespace: Recognissimo

Assembly: Recognissimo.dll

Syntax

```
public class SpeechSourceRuntimeException : RuntimeException, ISerializable
```

Constructors

SpeechSourceRuntimeException(String)

Declaration

```
public SpeechSourceRuntimeException(string message)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	

SpeechSourceRuntimeException(String, Exception)

Declaration

```
public SpeechSourceRuntimeException(string message, Exception innerException)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	message	
System.Exception	innerException	

Namespace Recognissimo.Components

Classes

AudioClipSpeechSource

SpeechSource that provides audio data from an AudioClip.

AudioListenerSpeechSource

SpeechSource that provides Unity AudioListener audio data.

MicrophoneSpeechSource

SpeechSource that provides audio data from a microphone.

PartialResultEvent

RemoteLanguageModelProvider

LanguageModelProvider that provides language models located on a remote resource.

ResultEvent

SpeechRecognizer

SpeechProcessor for speech recognition.

StreamingAssetsLanguageModelProvider

LanguageModelProvider that provides language models located in StreamingAssets folder.

VoiceActivityDetector

SpeechProcessor for voice activity detection.

VoiceControl

SpeechProcessor for voice control.

Structs

PartialResult

Partial speech recognition result which may change as recognizer process more data.

RemoteLanguageModelArchive

Remote language model description.

Result

Speech recognition result.

StreamingAssetsLanguageModel

StreamingAssets language model description.

VoiceControlCommand

Phrase/callback pair.

Word

Detailed description of a decoded word.

Class AudioClipSpeechSource

SpeechSource that provides audio data from an AudioClip.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
SpeechSource
AudioClipSpeechSource

Inherited Members

SpeechSource.SamplesReady
SpeechSource.Dried
SpeechSource.RuntimeFailure
SpeechSource.OnSamplesReady(SamplesReadyEventArgs)
SpeechSource.OnDried()
SpeechSource.OnRuntimeFailure(RuntimeFailureEventArgs)
SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: Recognissimo.Components
Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Sources/AudioClip Speech Source")]
public sealed class AudioClipSpeechSource : SpeechSource
```

Fields

clip

Audio clip from which the data will be taken.

Declaration

```
[Tooltip("Audio clip from which the data will be taken")]
public AudioClip clip
```

Field Value

TYPE	DESCRIPTION
UnityEngine.AudioClip	

readSequentially

Whether to read the clip in parts. Setting false will require buffer reallocation for each new clip.

Declaration

```
[Tooltip("Whether to read the clip in parts")]
public bool readSequentially
```

Field Value

TYPE	DESCRIPTION
System.Boolean	

Properties

SampleRate

Speech sampling rate. Must be set during initialization.

Declaration

```
public override int SampleRate { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Overrides

SpeechSource.SampleRate

Methods

StartProducing()

Called by SpeechProcessor at the start of processing.

Declaration

```
public override void StartProducing()
```

Overrides

SpeechSource.StartProducing()

StopProducing()

Called when processing stops (e.g. when StopProcessing() called or when RuntimeFailure event emitted).

Declaration

```
public override void StopProducing()
```

Overrides

SpeechSource.StopProducing()

Class AudioListenerSpeechSource

SpeechSource that provides Unity AudioListener audio data.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
SpeechSource
AudioListenerSpeechSource

Inherited Members

SpeechSource.SamplesReady
SpeechSource.Dried
SpeechSource.RuntimeFailure
SpeechSource.OnSamplesReady(SamplesReadyEventArgs)
SpeechSource.OnDried()
SpeechSource.OnRuntimeFailure(RuntimeFailureEventArgs)
SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: Recognissimo.Components
Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Sources/AudioListener Speech Source")]
public sealed class AudioListenerSpeechSource : SpeechSource
```

Fields

channel

AudioListener channel for receiving data.

Declaration

```
public int channel
```

Field Value

TYPE	DESCRIPTION
System.Int32	

Properties

SampleRate

Speech sampling rate. Must be set during initialization.

Declaration

```
public override int SampleRate { get; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Overrides

SpeechSource.SampleRate

Methods

StartProducing()

Called by SpeechProcessor at the start of processing.

Declaration

```
public override void StartProducing()
```

Overrides

SpeechSource.StartProducing()

StopProducing()

Called when processing stops (e.g. when StopProcessing() called or when RuntimeFailure event emitted).

Declaration

```
public override void StopProducing()
```

Overrides

SpeechSource.StopProducing()

Class MicrophoneSpeechSource

SpeechSource that provides audio data from a microphone.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
SpeechSource
MicrophoneSpeechSource

Inherited Members

SpeechSource.SampleRate
SpeechSource.SamplesReady
SpeechSource.Dried
SpeechSource.RuntimeFailure
SpeechSource.OnSamplesReady(SamplesReadyEventArgs)
SpeechSource.OnDried()
SpeechSource.OnRuntimeFailure(RuntimeFailureEventArgs)
SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: Recognissimo.Components
Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Sources/Microphone Speech Source")]
public sealed class MicrophoneSpeechSource : SpeechSource
```

Properties

DeviceName

Microphone name. Use null or empty string to use default microphone.

Declaration

```
public string DeviceName { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.String	

IsPaused

Whether recording is paused.

Declaration

```
public bool IsPaused { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

IsRecording

Whether recording is active.

Declaration

```
public bool IsRecording { get; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

TimeSensitivity

How often audio frames should be submitted to the recognizer (seconds). Use smaller values to submit audio samples more often. Recommended value is 0.25 seconds.

Declaration

```
public float TimeSensitivity { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Single	

Methods

Devices()

Lists available microphone names.

Declaration

```
public string[] Devices()
```

Returns

TYPE	DESCRIPTION
System.String[]	Available devices.

StartProducing()

Called by SpeechProcessor at the start of processing.

Declaration

```
public override void StartProducing()
```

Overrides

SpeechSource.StartProducing()

StopProducing()

Called when processing stops (e.g. when StopProcessing() called or when RuntimeFailure event emitted).

Declaration

```
public override void StopProducing()
```

Overrides

SpeechSource.StopProducing()

Struct PartialResult

Partial speech recognition result which may change as recognizer process more data.

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public struct PartialResult
```

Fields

partial

Decoded text.

Declaration

```
public string partial
```

Field Value

TYPE	DESCRIPTION
System.String	

result

Detailed description of decoded text.

Declaration

```
public List<Word> result
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<Word>	

Class PartialResultEvent

Inheritance

System.Object

UnityEngine.Events.UnityEventBase

UnityEngine.Events.UnityEvent<PartialResult>

PartialResultEvent

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public class PartialResultEvent : UnityEvent<PartialResult>, ISerializationCallbackReceiver
```

Struct RemoteLanguageModelArchive

Remote language model description.

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public struct RemoteLanguageModelArchive
```

Fields

entry

In-archive path to language model content

Declaration

```
public string entry
```

Field Value

TYPE	DESCRIPTION
System.String	

language

Language of the model.

Declaration

```
public SystemLanguage language
```

Field Value

TYPE	DESCRIPTION
UnityEngine.SystemLanguage	

url

URL to the zipped language model.

Declaration

```
public string url
```

Field Value

TYPE	DESCRIPTION
System.String	

Class RemoteLanguageModelProvider

LanguageModelProvider that provides language models located on a remote resource.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
LanguageModelProvider
RemoteLanguageModelProvider

Inherited Members

LanguageModelProvider.Model
SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: **Recognissimo.Components**

Assembly: **Recognissimo.dll**

Syntax

```
[AddComponentMenu("Recognissimo/Language Model Providers/Remote Language Model Provider")]
public sealed class RemoteLanguageModelProvider : LanguageModelProvider
```

Fields

language

Language for which the language model will be loaded.

Declaration

```
public SystemLanguage language
```

Field Value

TYPE	DESCRIPTION
UnityEngine.SystemLanguage	

languageModels

List of the available language models.

Declaration

```
[Tooltip("List of available language models")]
public List<RemoteLanguageModelArchive> languageModels
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<RemoteLanguageModelArchive>	

Methods

IsDownloaded(SystemLanguage)

Declaration

```
public bool IsDownloaded(SystemLanguage downloadedLanguage)
```

Parameters

TYPE	NAME	DESCRIPTION
UnityEngine.SystemLanguage	downloadedLanguage	

Returns

TYPE	DESCRIPTION
System.Boolean	

Struct Result

Speech recognition result.

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public struct Result
```

Fields

alternatives

List of alternative results.

Declaration

```
public List<string> alternatives
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.String>	

result

Detailed description of decoded text.

Declaration

```
public List<Word> result
```

Field Value

TYPE	DESCRIPTION
System.Collections.Generic.List<Word>	

text

Decoded text.

Declaration

```
public string text
```

Field Value

TYPE	DESCRIPTION
System.String	

Class ResultEvent

Inheritance

System.Object

UnityEngine.Events.UnityEventBase

UnityEngine.Events.UnityEvent<Result>

ResultEvent

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public class ResultEvent : UnityEvent<Result>, ISerializationCallbackReceiver
```

Class SpeechRecognizer

SpeechProcessor for speech recognition.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessor
SpeechRecognizer

Inherited Members

SpeechProcessor.State
SpeechProcessor.LanguageModelProvider
SpeechProcessor.SpeechSource
SpeechProcessor.AutoStart
SpeechProcessor.Started
SpeechProcessor.Finished
SpeechProcessor.InitializationFailed
SpeechProcessor.RuntimeFailed
SpeechProcessor.StartProcessing()
SpeechProcessor.StopProcessing()

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Processors/Speech Recognizer")]
public sealed class SpeechRecognizer : SpeechProcessor
```

Properties

Alternatives

Whether the recognition result should contain a list of alternative results.

Declaration

```
public int Alternatives { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

EnableDetails

Whether the recognition result should include details.

Declaration

```
public bool EnableDetails { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

PartialResultReady

New partial result ready.

Declaration

```
public PartialResultEvent PartialResultReady { get; }
```

Property Value

TYPE	DESCRIPTION
PartialResultEvent	

ResultReady

New result ready.

Declaration

```
public ResultEvent ResultReady { get; }
```

Property Value

TYPE	DESCRIPTION
ResultEvent	

Vocabulary

List of the words to recognize. Speech recognizer will select the result only from the presented words. Use special word "[unk]" (without quotes) to allow unknown words in the output.

Declaration

```
public List<string> Vocabulary { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<System.String>	

Remarks

This feature may not work with some language models.

Examples

```
var vocabulary = new List<string> {"light", "on", "off", "[unk]"};
```

Struct StreamingAssetsLanguageModel

StreamingAssets language model description.

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public struct StreamingAssetsLanguageModel
```

Fields

language

Language of the model.

Declaration

```
[Tooltip("Language of the model")]
public SystemLanguage language
```

Field Value

TYPE	DESCRIPTION
UnityEngine.SystemLanguage	

path

Path relative to StreamingAssets folder.

Declaration

```
[Tooltip("Path relative to StreamingAssets folder")]
public string path
```

Field Value

TYPE	DESCRIPTION
System.String	

Class StreamingAssetsLanguageModelProvider

LanguageModelProvider that provides language models located in StreamingAssets folder.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessorDependency
LanguageModelProvider
StreamingAssetsLanguageModelProvider

Inherited Members

LanguageModelProvider.Model
SpeechProcessorDependency.IsInitialized
SpeechProcessorDependency.RegisterInitializationTask(String, Action, CallCondition)
SpeechProcessorDependency.RegisterInitializationTask(String, Func<IEnumerator>, CallCondition)
SpeechProcessorDependency.Initialize(InitializationTaskStartedCallback, InitializationFailedCallback)
SpeechProcessorDependency.FailInitialization(Exception)

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Language Model Providers/Streaming Assets Language Model Provider")]
public sealed class StreamingAssetsLanguageModelProvider : LanguageModelProvider
```

Fields

language

Language for which the language model will be loaded.

Declaration

```
[Tooltip("Language for which the language model will be loaded")]
public SystemLanguage language
```

Field Value

TYPE	DESCRIPTION
UnityEngine.SystemLanguage	

languageModels

List of the available language models.

Declaration

```
[Tooltip("List of available language models")]
public List<StreamingAssetsLanguageModel> languageModels
```

Field Value

TYPE	DESCRIPTION

System.Collections.Generic.List<StreamingAssetsLanguageModel>	

Class VoiceActivityDetector

SpeechProcessor for voice activity detection.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessor
VoiceActivityDetector

Inherited Members

SpeechProcessor.State
SpeechProcessor.LanguageModelProvider
SpeechProcessor.SpeechSource
SpeechProcessor.AutoStart
SpeechProcessor.Started
SpeechProcessor.Finished
SpeechProcessor.InitializationFailed
SpeechProcessor.RuntimeFailed
SpeechProcessor.StartProcessing()
SpeechProcessor.StopProcessing()

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Processors/Voice Activity Detector")]
public sealed class VoiceActivityDetector : SpeechProcessor
```

Properties

Silenced

Voice became inactive.

Declaration

```
public UnityEvent Silenced { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

Spoke

Voice became active.

Declaration

```
public UnityEvent Spoke { get; }
```

Property Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

TimeoutMs

The number of milliseconds of silence after which the corresponding event should be triggered.

Declaration

```
public int TimeoutMs { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Int32	

Class VoiceControl

SpeechProcessor for voice control.

Inheritance

System.Object
UnityEngine.Object
UnityEngine.Component
UnityEngine.Behaviour
UnityEngine.MonoBehaviour
SpeechProcessor
VoiceControl

Inherited Members

SpeechProcessor.State
SpeechProcessor.LanguageModelProvider
SpeechProcessor.SpeechSource
SpeechProcessor.AutoStart
SpeechProcessor.Started
SpeechProcessor.Finished
SpeechProcessor.InitializationFailed
SpeechProcessor.RuntimeFailed
SpeechProcessor.StartProcessing()
SpeechProcessor.StopProcessing()

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[AddComponentMenu("Recognissimo/Speech Processors/Voice Control")]
public sealed class VoiceControl : SpeechProcessor
```

Properties

AsapMode

Whether to try to recognize voice commands using the preliminary recognition results.

Declaration

```
public bool AsapMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Boolean	

Commands

List of voice commands.

Declaration

```
public List<VoiceControlCommand> Commands { get; set; }
```

Property Value

TYPE	DESCRIPTION
System.Collections.Generic.List<VoiceControlCommand>	

Struct VoiceControlCommand

Phrase/callback pair.

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public struct VoiceControlCommand
```

Constructors

VoiceControlCommand(String, UnityAction)

Create instance and bind `action` to onSpoken.

Declaration

```
public VoiceControlCommand(string phrase, UnityAction action)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	phrase	Phrase to recognize.
UnityEngine.Events.UnityAction	action	Action that will be triggered when the `phrase` is spoken.

VoiceControlCommand(String, UnityEvent)

Create instance.

Declaration

```
public VoiceControlCommand(string phrase, UnityEvent onSpoken)
```

Parameters

TYPE	NAME	DESCRIPTION
System.String	phrase	Phrase to recognize.
UnityEngine.Events.UnityEvent	onSpoken	Unity event that will be triggered when the `phrase` is spoken.

Fields

onSpoken

UnityEvent that will be triggered when the phrase is spoken.

Declaration

```
public UnityEvent onSpoken
```

Field Value

TYPE	DESCRIPTION
UnityEngine.Events.UnityEvent	

phrase

Phrase to recognize. You can use groups "()" and alternations "|" to create options:

```
   "red|green"; // triggered when "red" or "green" is spoken
   "turn (on|off) the light"; // triggered when "turn on the light" or "turn off the light" is spoken
   "turn (on|off) (the )?light"; // optional "the"
```

Declaration

```
[Tooltip("Phrase to recognize")]
public string phrase
```

Field Value

TYPE	DESCRIPTION
System.String	

Struct Word

Detailed description of a decoded word.

Namespace: Recognissimo.Components

Assembly: Recognissimo.dll

Syntax

```
[Serializable]
public struct Word
```

Fields

conf

Confidence.

Declaration

```
public float conf
```

Field Value

TYPE	DESCRIPTION
System.Single	

end

End time of the word in seconds.

Declaration

```
public float end
```

Field Value

TYPE	DESCRIPTION
System.Single	

start

Start time of the word in seconds.

Declaration

```
public float start
```

Field Value

TYPE	DESCRIPTION
System.Single	

word

Decoded word.

Declaration

```
public string word
```

Field Value

TYPE	DESCRIPTION
System.String	